From Disclosure to Exploitation

A Comprehensive Analysis of IoT Vulnerability Targeting and Attacker Decision-Making

Al Alsadi, Arwa

DOI
10.4233/uuid:c918a6a0-72ac-4a24-81bd-217b8234d752

Publication date
2025

Document Version
Final published version

Citation (APA)
Al Alsadi, A. (2025). *From Disclosure to Exploitation: A Comprehensive Analysis of IoT Vulnerability Targeting and Attacker Decision-Making.* [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:c918a6a0-72ac-4a24-81bd-217b8234d752

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

FROM DISCLOSURE TO EXPLOITATION

Arwa Al Alsadi

From Disclosure to Exploitation

A Comprehensive Analysis of IoT Vulnerability Targeting and Attacker Decision-Making

FROM DISCLOSURE TO EXPLOITATION

A COMPREHENSIVE ANALYSIS OF IOT VULNERABILITY TARGETING AND ATTACKER DECISION-MAKING

Dissertation

for the purpose of obtaining the degree of doctor at Delft University of Technology,
by the authority of the Rector Magnificus,
Prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates, to be defended publicly on
Wednesday, 17 December, 2025 at 17:30

by

Arwa ABDULKARIM A AL ALSADI

Master of Science in Cybersecurity and Computer Science, The George Washington University
born in Riyadh, Saudi Arabia.

This dissertation has been approved by the promotors:

Dr.ir. C. Hernández Gañán
Prof. dr. M.J.G. van Eeten

Composition of the doctoral committee:

Rector Magnificus	Chairperson
Dr.ir. C. Hernández Gañán	Delft University of Technology, promotor
Prof. dr. M.J.G. van Eeten	Delft University of Technology, promotor
Independent members:	
Prof.dr. G. Smaragdakis	Delft University of Technology
Prof.dr. W.W. Veeneman	Delft University of Technology
Prof.dr. M. Junger	University of Twente
Dr. M. Vasek	University College London
Dr. S. Zannettou	Delft University of Technology

This research was mainly supported by King Abdulaziz City for Science and Technology (KACST).

Cover by: Reem Alalami

An electronic version of this dissertation is available at
http://repository.tudelft.nl/

To mama, baba, & sisters.

CONTENTS

ACRONYMS

C&C Command and Control.

CMDi Command Injection.

CPE Common Platform Enumeration.

CVE Common Vulnerabilities and Exposures.

CVSS Common Vulnerability Scoring System.

CWE Common Weakness Enumeration.

DDoS Distributed Denial of Service.

ENISA European Union Agency for Cybersecurity.

EPSS Exploit Prediction Scoring System.

HTTPS Hypertext Transfer Protocol Secure.

IoT Internet of Things.

IP Internet Protocol.

ISP Internet Service Provider.

KEV Known Exploited Vulnerabilities.

MITRE MITRE Corporation.

NVD National Vulnerability Database.

PoC Proof-of-concept.

RCE Remote Code Execution.

URL Uniform Resource Locator.

VDB Vulnerability Database.

GLOSSARY

Disclosure The process of reporting a vulnerability to a vendor or to the public, often followed by a patch release..

Dual-use dilemma The ethical and practical challenge where tools or knowledge intended for good (e.g., Proof-of-Concept exploits) can also be misused for malicious purposes..

Exploit A piece of code or technique that takes advantage of a software vulnerability to gain unauthorized control or cause unintended behavior..

Exploit Prediction The process of estimating whether a known vulnerability is likely to be exploited in the wild..

Governance Mechanisms and processes for coordinating actions and responsibilities among stakeholders to manage cybersecurity risks effectively..

Malware Malicious software designed to damage, disrupt, or gain unauthorized access to computer systems..

Patch A software update or fix released by a vendor to correct a vulnerability or improve performance..

Vulnerability A flaw or weakness in software or hardware that can be exploited by attackers to compromise security..

SUMMARY

The rapid growth of Internet-of-Things (IoT) devices, such as smart cameras, home routers, and smart thermostats, has transformed the digital landscape while also introducing new cybersecurity risks. IoT systems are often targeted by attackers due to outdated software, long device lifespans, and fragmented security practices. Although many IoT vulnerabilities are discovered and disclosed, only a small fraction are actually exploited in the wild. This raises important questions about which vulnerabilities are targeted, why attackers choose them, and how long they remain in use.

This dissertation investigates how IoT vulnerabilities are selected for exploitation in practice, with a particular focus on attacker behavior, exploit development, and vulnerability characteristics. It systematically examines the interplay between these factors to understand how they collectively shape exploitation trends in IoT ecosystems. To answer the central research question on *What factors shape the exploitation in IoT vulnerabilities, from target selection to exploit development and prediction?*, this dissertation presents four peer-reviewed studies.

Chapter 2 provides a longitudinal analysis of over 17,000 IoT malware samples, revealing that only a handful of IoT vulnerabilities are targeted and often exploited for years after their disclosure. The average time-to-exploit a vulnerability after disclosure was found to be 29 months, far longer than in traditional IT systems. This temporal persistence highlights the enduring value of certain vulnerabilities within the attacker ecosystem.

Chapter 3 examines factors influencing exploitation frequency in IoT vulnerabilities. It finds that attackers prefer vulnerabilities that are easy to exploit, affect widely deployed devices, and are difficult to patch. Technical severity scores, like CVSS, were less predictive than contextual factors such as device type and patch complexity.

Chapter 4 addresses the limitations of existing prediction systems, such as the Exploit Prediction Scoring System (EPSS), in assessing IoT-specific risk. By incorporating attacker community discussions from underground forums into a new predictive model, the study significantly improves accuracy and highlights the importance of behavioral and vendor-related features in anticipating exploitation for IoT devices.

Finally, Chapter 5 shifts focus to the human element through interviews with 16 Proof-of-Concept (PoC) exploit developers. It finds that disclosure decisions are shaped by individual motivations, ethical considerations, and vendor interactions. PoCs developers play a key role in making vulnerabilities exploitable and often act as gatekeepers in the vulnerability ecosystem. This qualitative study examines the socio-technical dynamics influencing PoC developers' decisions to publish exploits, and how these choices can shape target selection and enable the weaponization of vulnerabilities.

Collectively, these findings show that targeting in IoT is not random but follows strategic patterns driven by cost, opportunity, and long-term exploit value. The dissertation argues that current governance mechanisms—market incentives, disclosure systems, and risk models, are misaligned with real-world exploitation practices and therefore fall short

in addressing the distinct dynamics of IoT security. To address these gaps, it proposes a hybrid governance model that combines regulatory oversight, community collaboration, and market-based tools to more effectively manage the lifecycle of IoT vulnerability and exploitation.

SAMENVATTING

De snelle groei van Internet-of-Things (IoT)-apparaten, zoals slimme camera's, thuisrouters en slimme thermostaten, heeft het digitale landschap getransformeerd, maar brengt ook nieuwe cybersecurityrisico's met zich mee. IoT-systemen worden vaak het doelwit van aanvallen vanwege verouderde software, lange levensduur van apparaten en gefragmenteerde beveiligingspraktijken. Hoewel veel IoT-kwetsbaarheden worden ontdekt en openbaar gemaakt, wordt slechts een klein deel daadwerkelijk misbruikt in de praktijk. Dit roept belangrijke vragen op: welke kwetsbaarheden worden gekozen, waarom maken aanvallers die keuzes, en hoelang blijven ze in gebruik?

Deze dissertatie onderzoekt hoe IoT-kwetsbaarheden in de praktijk worden geselecteerd voor exploitatie, met bijzondere aandacht voor het gedrag van aanvallers, exploitontwikkeling en kenmerken van kwetsbaarheden. Het werk analyseert systematisch de wisselwerking tussen deze factoren om te begrijpen hoe zij gezamenlijk de exploitatiepatronen in IoT-ecosystemen vormgeven. Om de centrale onderzoeksvraag te beantwoorden—*Welke factoren bepalen de exploitatie van IoT-kwetsbaarheden, van doelwitselectie tot exploitontwikkeling en voorspelling?*—presenteert deze dissertatie vier peer-reviewed studies.

Hoofdstuk 2 biedt een longitudinale analyse van meer dan 17.000 IoT-malwaremonsters en laat zien dat slechts een beperkt aantal kwetsbaarheden wordt benut, vaak nog jaren na openbaarmaking. De gemiddelde tijd tussen openbaarmaking en eerste exploitatie bedraagt 29 maanden—aanzienlijk langer dan bij traditionele IT-systemen. Deze langdurige exploitatie benadrukt de aanhoudende waarde van bepaalde kwetsbaarheden voor aanvallers.

Hoofdstuk 3 onderzoekt welke factoren van invloed zijn op de frequentie van exploitatie van IoT-kwetsbaarheden. Aanvallers geven de voorkeur aan kwetsbaarheden die gemakkelijk te misbruiken zijn, voorkomen op grootschalig ingezette apparaten en moeilijk te patchen zijn. Technische ernstscores, zoals CVSS, blijken minder voorspellend dan contextuele factoren zoals apparaattype en patchcomplexiteit.

Hoofdstuk 4 bespreekt de beperkingen van bestaande risicovoorspellingssystemen, zoals het Exploit Prediction Scoring System (EPSS), bij het inschatten van IoT-specifieke risico's. Door discussies uit ondergrondse fora van de aanvallersgemeenschap te integreren in een nieuw voorspellingsmodel, verbetert de studie de nauwkeurigheid aanzienlijk en benadrukt zij het belang van gedragskenmerken en leverancierscontext bij het voorspellen van exploitatie.

Hoofdstuk 5 richt zich op de menselijke kant van exploitontwikkeling, gebaseerd op interviews met 16 ontwikkelaars van Proof-of-Concept (PoC) exploits. De studie toont aan dat beslissingen over openbaarmaking worden beïnvloed door individuele motivaties, ethische overwegingen en interacties met leveranciers. PoC-ontwikkelaars spelen een cruciale rol in het daadwerkelijk misbruikbaar maken van kwetsbaarheden en fungeren vaak als poortwachters binnen het kwetsbaarheidsecosysteem. Deze kwalitatieve studie on-

thult de sociaal-technische dynamiek die bepaalt of kwetsbaarheden uiteindelijk worden bewapend.

Gezamenlijk laten deze bevindingen zien dat de keuze van aanvallers in IoT niet willekeurig is, maar strategisch wordt bepaald door kosten, kansen en langdurige exploitwaarde. De dissertatie stelt dat bestaande governancemechanismen—zoals marktprikkels, meldsystemen en risicomodellen—niet goed aansluiten bij de praktijk van exploitatie en daardoor tekortschieten in het aanpakken van de specifieke dynamiek van IoT-beveiliging. Om deze lacunes te overbruggen, wordt een hybride governancemodel voorgesteld dat regulatoire controle, samenwerking met de gemeenschap en marktgerichte instrumenten combineert om de levenscyclus van IoT-kwetsbaarheden en exploitatie effectiever te beheersen.

1

INTRODUCTION

The widespread deployment of Internet-of-Things (IoT) devices, such as IP cameras, smart home appliances, and smart thermostats, has expanded the attack surface for malicious actors. A turning point came in 2016 with the emergence of the Mirai malware [23], which launched one of the largest distributed denial-of-service (DDoS) attacks ever seen at the time by hijacking over half a million vulnerable IoT devices. Initially leveraging simple default credentials, Mirai's success led to a proliferation of variants following the release of its source code in 2017 [115]. Since then, IoT malware has evolved significantly: campaigns like *VPNFilter*, *Torii*, and *QSnatch* go well beyond simple exploitation, they persist across reboots, evade detection, and often target devices of high value or strategic importance [49, 56, 97].

Unlike desktop or mobile systems [19], IoT malware compromises devices at scale [23], often through automated exploitation rather than the user-dependent vectors [19]. This automated approach allows attackers to rapidly expand their botnets across vulnerable IoT infrastructure.

Although our knowledge about IoT malware families and their capabilities has increased [71, 260], we know much less about the attackers' behavior in terms of the vulnerabilities they target with their exploit code. The growing number of IoT-related vulnerabilities: from a handful in 2010 to more than 3,253 in 2022 [139]. Yet, only a small fraction (15%) are actually exploited in the wild [179]. In traditional computing systems, attackers typically do not exploit all vulnerabilities equally. Instead, they often select just one vulnerability per software version, favor those with low attack complexity, and adopt new vulnerabilities only slowly [14].

Proof-of-concept (PoC) exploit code plays an important but often misunderstood role in IoT vulnerability exploitation. While PoCs are commonly released to support disclosure, aid defenders, or demonstrate impact, their presence alone rarely guarantees that a vulnerability will be patched—or that it leads to exploitation [83, 235]. In IoT ecosystems, this gap is even wider: vendors frequently delay or forgo patches, devices remain deployed for years without updates [27], and PoCs may circulate long before (or long after) attackers choose to operationalize them. At the same time, PoCs can lower the

1

barrier for rapid integration into malware toolchains, accelerating the transition from disclosure to active exploitation when attackers deem a vulnerability worthwhile [19, 136, 179, 241]. As later sections of this work show, the relationship between PoCs and real-world IoT exploitation is neither linear nor predictable by itself [128], but instead shaped by attacker incentives, device heterogeneity, and patching asymmetries. Recognizing this complexity provides necessary context for understanding how attackers ultimately select which IoT vulnerabilities to weaponize.

However, it still remains unclear whether this pattern holds for IoT malware, where device constraints, patch delay, and operational lifespans differ fundamentally from traditional IT systems. Specifically, which IoT vulnerabilities are targeted? Do different malware families go after the same vulnerabilities? How soon do attackers weaponize newly disclosed vulnerabilities? For how long do exploits remain active?

Existing models, derived from desktop and server systems, may not apply to IoT ecosystems, where attackers predominantly target the last-but-one version of the software after a patch is released [9, 227, 250]. This further complicates efforts for defenders to assess risk and prioritize vulnerabilities and allocate resources effectively [156, 212].

This dissertation bridges that gap by analyzing attackers' behavior in IoT exploitation, aiming to advance both the theoretical and practical understanding of vulnerability selection in IoT malware. By addressing these challenges, this work seeks to inform more effective defense strategies and develop IoT-aware security assessment and prioritization tools for the increasingly connected IoT landscape.

1.1. RELATED WORK

Research on attacker target selection has grown significantly across multiple disciplines. This section synthesizes prior work on the technical and behavioral factors influencing exploit development and targeting decisions in malware campaigns, with special attention to IoT ecosystems.

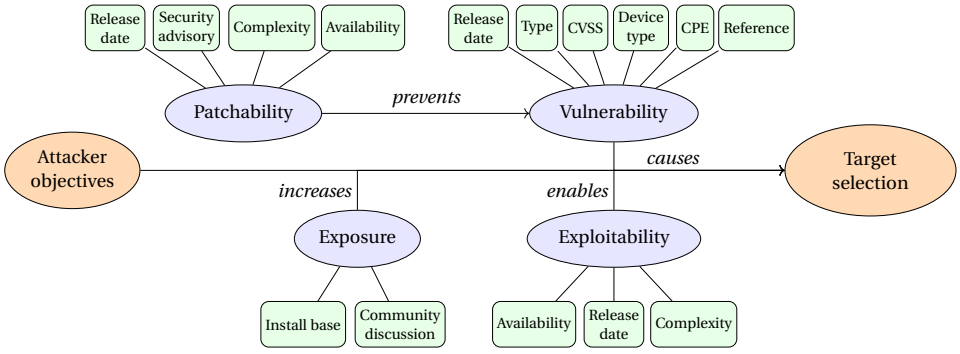


Figure 1.1: Theoretical Model for Target Selection in IoT Vulnerability.

To guide this review, Figure 1.1 presents a theoretical model integrating key findings about IoT vulnerability targeting. The model conceptualizes target selection as shaped

by four interconnected factors: Vulnerability, Patchability, Exploitability, and Exposure. These factors are, in turn, influenced by various sub-features, many of which have been independently studied in past research. By bringing these elements together into a unified model, we aim to provide a comprehensive lens for understanding exploit targeting behavior in IoT malware.

Each of these four factors encompasses a distinct set of features that together describe how vulnerabilities transition from disclosure to exploitation.

Vulnerability captures the intrinsic technical properties of a flaw—its *release date*, *type of weakness* (CWE), *severity* (CVSS), *device type, affected platform—covering systems, software, packages, and vendors* (CPE), and associated *references* in vulnerability databases to measure the vulnerability's level of activity, that may determine its attractiveness to attackers.

Patchability reflects the extent to which mitigation is possible and timely, including whether a *security advisory* or patch is issued, the *release date* of that fix, its *availability* across products, and the *complexity* of deploying it, which can constrain or enable exploitation.

Exploitability represents the feasibility and effort required to weaponize a vulnerability, characterized by proof-of-concept (PoC) exploit *availability*, *release date*, and *complexity* of exploit code as seen in public exploit repositories.

Exposure describes the opportunity side of the threat environment, combining the size or scale of deployment of the *install base* of affected devices connected to the public internet and the level of *community discussion* of such vulnerabilities, signaling visibility and attention.

Together, these dimensions frame the technical, temporal, and contextual mechanisms through which vulnerabilities may evolve from known issues to exploited targets. Their operationalisation and empirical measurement is detailed in Chapter 3 and Chapter 4.

The rationale here is that attackers, like defenders, operate under informational constraints and may rely on publicly available data such as vulnerability disclosure information, patch advisories, or Proof-of-Concept (PoC) exploit code repositories to inform their decisions. For instance, a vulnerability, which is an existing weakness or flaw in an IoT device, can be exploited by an attacker to gain unauthorized access or cause harm. Characteristics of such vulnerabilities, including CVSS severity, disclosure timing, and affected device types, have been shown to correlate with attack likelihood. Patch-related attributes like availability and complexity of applying a patch (fix) can limit the attack surface and thus influence whether attackers invest in developing exploits. Meanwhile, indicators of exploitability, such as PoC exploit availability or complexity, can impact the cost of launching attacks, under the assumption that attackers seek to minimize effort. Finally, exposure factors, such as the size of the vulnerable device install base or the extent of public discussion on social media or underground forums, may increase the perceived benefits of exploitation by signaling a larger or more accessible target population.

1.1.1. ATTACKER OBJECTIVE

Understanding how attackers choose their targets is a central question in cybersecurity research, particularly for vulnerability exploitation in large and diverse ecosystems such as IoT. While thousands of vulnerabilities are disclosed annually, only a small fraction are actively exploited in the wild [8, 19, 179, 235, 264]. This discrepancy has led researchers to explore attacker behavior through the lens of economic theory, especially the assumption that attackers act rationally, seeking to maximize gains while minimizing cost, effort, and risk.

Prior work has shown that cybercrime, much like physical crime, often follows rational decision-making patterns. Studies by Hutchings and Holt [130] and Tajalizadehkhoob et al. [236] demonstrated that attackers, like burglars, tend to select targets that are perceived as easy to access and offer the highest expected return. These findings support a rational choice perspective, in which attackers weigh various factors, such as exploit complexity or availability [12], device exposure or type [19, 114], patch availability [27], and operational payoff [9], when deciding where to direct their efforts.

Recent work by Allodi et al. [14] has further refined this perspective to model cyber attackers as "work-averse". They proposed a theoretical and empirical framework suggesting that attackers avoid vulnerabilities with high development costs or low exploitation payoff. Their model posits that attackers prefer low-complexity exploits and are slow to incorporate new vulnerabilities into their toolkits, especially when older, reliable exploits remain effective. Empirical validation of this model using Symantec's WINE dataset revealed that attackers rarely target more than one vulnerability per software version and tend to reuse known exploits over extended periods.

In IoT malware campaigns, the primary objective is often to build large botnets that can be monetized through various means. Putman et al. [207] demonstrated how botnet operators prioritize scale for profit, emphasizing economic incentives. This economic focus manifests in multiple revenue streams: IoT devices are repurposed at scale for cryptojacking operations [237], while the growth of DDoS-for-hire services commoditizes botnet access for profit [180]. This resource-focused approach drives attackers to prioritize the acquisition of vulnerable devices at scale rather than targeting sophisticated individual systems, which is evident in the long-term persistence with outdated exploits observed in these campaigns. Alrawi et al. [19] found that IoT malware samples target vulnerabilities that had been disclosed over three years prior, reflecting a strong preference for exploit reuse. This aligns with Nayak et al. [179], who reported that only 15% of known vulnerabilities are actively exploited, consistent with earlier studies [8, 135, 235, 264]. Such findings reinforce that IoT attackers prioritize building large-scale compromised device networks for monetization purposes, focusing on reliable exploits that can consistently deliver new bot recruits rather than pursuing sophisticated but risky exploit development that might yield fewer compromised devices [242].

Despite these theoretical advances, relatively little research has empirically tested how these cost-benefit considerations influence target selection in IoT malware specifically.

1.1.2. VULNERABILITY

A central theme in vulnerability management and prioritization research has been the attempt to correlate publicly disclosed vulnerability attributes with their likelihood of

being exploited in the wild. Among the most widely adopted frameworks in this domain is the Common Vulnerability Scoring System (CVSS), introduced in 2003 to provide a standardized measure of severity for software vulnerabilities. Despite its intended purpose as a general severity rating, CVSS scores have often been repurposed as proxies for exploitation risk in vulnerability prioritization and patch management processes. In IoT environments, however, the applicability of CVSS becomes more complex due to the unique characteristics of these devices [22], such as limited update mechanisms, diverse firmware implementations, and prolonged deployment lifecycles that may not align with traditional vulnerability assessment approaches.

Higher CVSS scores are frequently interpreted as indicators of urgent risk, under the assumption that more severe vulnerabilities are also more attractive to attackers. Furthermore, the developers of CVSS explicitly caution against using the score for predicting exploitation likelihood [230]. In practice, a substantial proportion of real-world attacks, nearly 49% by some estimates, occur before a CVSS score has even been assigned or published [60]. However, this assumption has been consistently challenged by empirical research. Studies have shown that CVSS scores have limited predictive power when it comes to identifying which vulnerabilities are actually exploited in the wild [11, 47, 215].

Beyond CVSS, researchers have explored alternative vulnerability features that may better explain attacker behavior such as vulnerability disclosure date, which is used as a proxy to measure a vulnerability age. Alrawi et al. [19] studied over 1.2 million IoT malware samples, found most observed exploits targeted vulnerabilities more than three years old. The *age of vulnerability* reflects attackers' preference for low-effort, high-impact targets, especially when new exploits incur development costs while old ones remain effective.

In addition to vulnerability age, vulnerability type (e.g., buffer overflow, SQL injection) based on the Common Weakness Enumeration Specification (CWE), and vendor or product type leveraging the Common Platform Enumeration (CPE) have been used to model both the timing and frequency of exploitation [27, 137, 175].

These features are often extracted from structured databases such as the National Vulnerability Database (NVD) [177], which includes metadata such as CWE categories, CPE identifiers, patch availability, and references to external advisories.

1.1.3. EXPLOITABILITY

As the limitations of traditional severity metrics such as CVSS became increasingly apparent, researchers began to explore alternative predictors for assessing real-world exploitation risk. One of the most influential of these has been the availability of public proof-of-concept (PoC) exploit code [19, 136, 179, 241]. Unlike CVSS, which abstracts technical severity into a numeric score, PoCs provide tangible, executable demonstrations of how a vulnerability may be leveraged, offering attackers a usable starting point and defenders a concrete signal of threat potential.

Several empirical studies have demonstrated a strong correlation between PoCs disclosure and the subsequent weaponization of vulnerabilities. Nayak et al. [179] found that 90% of real-world exploits emerged within 94 days following the public release of a PoC exploit. More recent cases illustrate how this window continues to shrink. For example, CVE-2024-27198, a vulnerability in JetBrains TeamCity, was exploited just 22 minutes after its PoC exploit code was published online [241]. Vulnerabilities with public PoCs are also

significantly more likely to be exploited as zero-days compared to those without such code: 41% versus 16%, respectively [81].

These observations have led to growing consensus that PoCs are a strong indicator of potential exploitation. Researchers have moved to integrate PoCs-related features into risk assessment models [235]. Indeed, the presence of PoCs has been found to outperform CVSS in several predictive settings [11, 128, 137, 231]. However, the predictive value of PoCs availability is not absolute. Only 4.17% of vulnerabilities are associated with public PoCs within one year of disclosure [128], and many PoC-bearing vulnerabilities are never exploited in the wild [10, 11].

Despite the growing reliance on PoCs as a risk indicator, the use of public PoCs remains controversial. Some researchers and practitioners argue that the publication of PoCs accelerates attacker timelines and increases risk by providing readily usable blueprints. Others contend that responsible disclosure, including PoCs, can improve patch development and prioritization by demonstrating exploitability to vendors and defenders [122, 161]. In many cases, PoCs are developed as part of the vulnerability reporting and validation process, submitted to platforms like GitHub [262], Exploit-db [105], or integrated into testing frameworks and penetration tools [35, 254].

This underscores the need for not only the technical signals PoCs provide, but also the socio-technical processes that govern their development and dissemination.

1.1.4. PATCHABILITY

Patching remains one of the most fundamental defenses against exploitation, yet its effectiveness varies drastically across different environments. While enterprise IT systems often follow semi-automated update schedules and benefit from centralized patch management, IoT ecosystems suffer from long-standing structural barriers to timely remediation. These include manual update processes, firmware fragmentation, vendor abandonment, and lack of user awareness, factors that significantly prolong the window of vulnerability and shape attacker behavior [243].

IoT devices often exhibit considerable delays in patch deployment. Arora et al. [27] reported a median patch latency of 45 days for consumer-grade IoT devices, while many vendors discontinue support shortly after market release, leaving them indefinitely exposed. Unlike desktop or server platforms, most IoT devices lack endpoint protection tools that could mitigate threats even in the absence of patches. As a result, even when vulnerabilities are technically patched, the real-world patch coverage across devices remains low, sustaining systemic risk.

This patching gap has led attackers to adopt opportunistic exploitation strategies that take advantage of known but widely unpatched vulnerabilities. In such environments, the likelihood that a device remains vulnerable for months or even years after disclosure makes it a viable and attractive target for adversaries. While studies of server and desktop environments have similarly revealed *patch gap* exploitation strategies (e.g, attackers target the previous software version shortly after a patch is released [124, 128, 179, 250]), these dynamics are not directly transferable to IoT, given the lack of consistent patching, the long device lifespans, and heterogeneous deployment practices.

Recent work has further examined patching challenges across multiple stakeholders in IoT ecosystems. Becker et al. [208] investigated the behaviors side of vendors and found

that patch deployment practices varied widely across manufacturers, with many failing to provide timely updates or security notifications. On the user side, Oortwijn et al. [194] illustrated the complexity and burden of patching, where users struggled with discovering and applying firmware updates, often relying on insecure workarounds or leaving devices unpatched.

Complementing these findings, Rodríguez et al. have conducted multiple studies on IoT device patching, revealing several key challenges. Their research on persistent IoT malware (e.g., QSnatch) found that while users generally possessed technical competency and did not find remediation difficult, persistent IoT malware takes significantly longer to remediate than Windows or non-persistent IoT malware, with a 30% survival probability after 180 days [211]. Additionally, they found that ISP malware notifications can be effective, with compliance to recommended steps increasing cleanup probability by 32%, though regulatory constraints like GDPR limited the specificity of guidance that could be provided [213]. Furthermore, their analysis of Mirai-infected devices revealed that just nine vendors account for nearly 50% of infections among 70 identified manufacturers, suggesting that targeted intervention with a small number of manufacturers could have major impact on overall IoT security [212].

These studies reinforce that IoT patchability is not solely a technical problem, but a structural one involving coordination between vendors, users, and other intermediaries. Addressing it requires more than patch availability; it demands usability, vendor commitment, and cross-ecosystem incentives.

1.1.5. EXPOSURE

IoT devices require network connectivity to operate effectively, enabling remote monitoring, control, and data collection. However, this connectivity requirement often results in devices being directly exposed to the public Internet, frequently due to misconfigurations, default credentials, or inadequate security controls. This direct exposure creates a fundamentally different attack surface compared to traditional IT systems that typically operate behind firewalls and network segmentation. It is important to distinguish between *device exposure*, how easily a device can be discovered and reached, and *vulnerability exposure*, the awareness and discussion of specific vulnerabilities within attacker communities. This distinction highlights the exposure of the device itself, separate from the vulnerabilities it may host, yet complementary factors in exploitation likelihood. This has led researchers to explore how factors such as device install base and device type matter in attacker target selection.

Several studies have confirmed that the size of the device install base and its public visibility strongly correlate with exploitation likelihood. For instance, devices exposed to internet-wide scans or indexed by search engines like Shodan are significantly more likely to be targeted by attackers [26, 34, 158, 160]. The logic is straightforward: attackers, particularly those using automated scanning tools, are more likely to identify and compromise devices that are readily discoverable and widely deployed. As a result, the mere fact that a device model has a large install base and is publicly accessible increases its risk profile, even in the absence of a known vulnerability.

To better understand these dynamics, multiple studies have sought to quantify IoT exposure and its relationship to real-world targeting. Alrawi et al. [19] and others [34,

166] examined large-scale malware datasets and found that certain IoT device types, such as IP cameras, home routers, and DVRs, are disproportionately represented among exploited devices. Also, studies like those by Blinowski et al. [43] and Alvarez et al. [20] have demonstrated that IoT malware often exploits similar firmware versions or device configurations across models and vendors, enabling exploit reuse at scale. These findings suggest that attackers not only consider technical exploitability but also evaluate potential target reach when designing or selecting malware payloads.

Beyond device exposure, researchers have also investigated vulnerability exposure, the visibility and awareness of specific vulnerabilities within attacker communities. As traditional vulnerability attributes like CVSS scores often fail to capture real-world attacker interest and timing, other research has turned to more dynamic, community-driven signals. These include discussions on social media platforms, underground forums, and bug bounty disclosures, which reflect evolving attacker awareness, intent, and engagement.

Empirical studies have demonstrated that online discussions, particularly on platforms like X (formerly Twitter), can outperform conventional metrics like CVSS or PoCs availability in predicting exploitation. For instance, Sabottke et al. [60] showed that mentions of vulnerabilities on Twitter correlated more strongly with exploitation in the wild than CVSS scores. Subsequent work found that combining tweet frequency and sentiment with metadata significantly improved exploitability estimation, even outperforming models based solely on PoCs availability [215, 235].

However, underground forums and dark web discussions offer another rich source of predictive information. Tavabi et al. [238] introduced a framework that leveraged dark web forum data to predict exploitation more effectively than existing models. Their study found that 14% of vulnerabilities mentioned on underground platforms were eventually exploited in the wild, outperforming other feature sets in predictive accuracy. Similarly, Almukaynizi et al. [16] identified that forum-based mentions yielded the highest exploitation rate among multiple datasets evaluated.

Building on this foundation, Moreno-Vera et al. [168] used the CrimeBB dataset, an archive of 36 underground forums, to identify 1,498 unique CVEs discussed by cybercriminals. Their analysis emphasized that discussions on these platforms are not merely noise; rather, they can provide early warnings of exploit development, interest in high-value targets, and even emerging attack vectors. Additionally, Allodi [15], Asghari et al. [36], and Shakarian [220] have explored how indicators like forum reputation, thread engagement, and vulnerability severity, as interpreted by attackers, affect exploitation probability.

Collectively, these studies highlight the importance of exposure in IoT exploitation, showing the need to complement static vulnerability attributes with contextual environmental indicators. Incorporating these features into predictive models could help defenders better prioritize risks based not only on what could be exploited, but on what is likely to be targeted given its presence, visibility, and distribution across the Internet.

1.1.6. EXPLOIT PREDICTION MODELS

Given the limitations of traditional metrics such as CVSS in predicting real-world attacks, researchers have developed various exploit prediction models that integrate one or multiple features to estimate the likelihood that a disclosed vulnerability will be exploited in the wild [11, 16, 132, 137, 179, 235, 238]. These models aim to improve risk prioriti-

zation by going beyond severity scores and incorporating behavioral, contextual, and community-driven indicators.

Among these models is the state-of-the-art the Exploit Prediction Scoring System (EPSS) [137]. Jacobs et al. [137] developed EPSS, a model that integrates a diverse set of features: CVSS base metrics, vulnerability metadata from the National Vulnerability Database (NVD), PoCs availability, usage in offensive security tools, and signals from social media discussions, to address these gaps. EPSS produces daily updated scores that estimate the probability that a vulnerability will be exploited in the next 30 days. The model has since been adopted by more than 100 vendors [102], including IoT-focused platforms such as Vulners [258] and Shodan [160].

1.2. RESEARCH GAPS

While a growing body of research has examined vulnerability exploitation and target selection, most studies focus on traditional IT environments, leaving the IoT ecosystem underexplored. The unique characteristics of IoT—such as heterogeneous device deployments, inconsistent patching mechanisms, and long-lived vulnerabilities—challenge existing models of attacker behavior and risk prioritization. This dissertation addresses several gaps in our understanding of IoT vulnerability target selection, outlined below.

THE LACK OF LONGITUDINAL ANALYSES OF IOT MALWARE EXPLOITS

Nayak et al. [179] explored key questions such as how many vulnerabilities get exploited, how often, and when exploitation occurs. Their findings challenged the assumption of short-lived attack windows, revealing that only a small fraction of 15% disclosed vulnerabilities are ever exploited in the wild and many vulnerabilities are exploited long after disclosure. However, their study was focused on general-purpose IT systems, which differ fundamentally from IoT ecosystems. IoT environments present distinct structural characteristics, such as firmware homogeneity, infrequent patching, and extended device lifespans, that likely alter attacker behavior and prolong the exploitability of known vulnerabilities. A more focused attempt was made by Alrawi et al. [19], who analyzed 166,000 IoT malware samples to study how vulnerabilities are targeted over time across the IoT malware families. While their work provides valuable initial insights, it did not longitudinally measure exploit lifespans or how long it takes for vulnerabilities to be adopted into malware after disclosure. Moreover, their exploit timestamps were based on industry reports rather than directly observed from malware artifacts. This approach introduces potential inaccuracies in timing measurements, as industry reports may lag behind actual exploitation events or lack the precision needed for detailed temporal analysis.

What remains lacking are empirical, longitudinal studies that trace what and how vulnerabilities are exploited by IoT malware over time, how long attackers persist in using specific exploits, and how exploit adoption evolves across different IoT campaigns.

THE UNEXAMINED INFLUENCE OF IOT-SPECIFIC FEATURES IN TARGET SELECTION

While rational choice frameworks have been proposed to model attacker decision-making, suggesting that actors prioritize low-effort, high-reward opportunities [130, 236], these models have rarely been empirically validated in the context of IoT malware campaigns.

Existing research often describes which vulnerabilities appear in malware binaries but does not explain why certain vulnerabilities are repeatedly selected over others. Key IoT-specific factors, such as device type [19, 34, 166] and device install base [26, 34, 158, 160], remain underexplored in this context. For instance, vulnerabilities affecting widely deployed or easily discoverable devices in public may be more attractive to attackers due to their potential scale and ease of automation. Similarly, features like patch complexity, firmware homogeneity and the exploitability of device-specific services [19] may influence reuse frequency. However, as most studies treat vulnerability selection in aggregate, without disaggregating the role of IoT-specific traits in shaping attacker choices, as a result, we lack a systematic understanding of how environmental and technical characteristics drive exploit frequency across IoT malware.

THE LIMITED EFFECTIVENESS OF EXPLOIT PREDICTION MODELS FOR IOT

State-of-the-art exploit prediction models, such as EPSS [113], represent a significant advancement over static scoring systems like CVSS, offering a more dynamic and data-driven approach to estimating exploitation risk. However, these models exhibit several limitations, particularly in the context of IoT security.

A temporal analysis of EPSS by Damiano et al. [210] highlights key shortcomings: EPSS scores often require weeks or months to reach critical thresholds, lagging behind IoT's rapidly evolving threat landscape. The model also exhibits frequent score fluctuations rather than gradual risk progression, complicating defensive prioritization. Most critically, EPSS rarely prioritized vulnerabilities lacking CVSS or CPE metadata, even when there is evidence of real-world exploitation, indicating a systemic bias rooted in its training data.

In addition, a study conducted by Spring [229] has shown that EPSS struggles to accurately rank actively exploited specifically to IoT vulnerabilities. A notable example is CVE-2017-17215 [186], a long-standing and frequently exploited vulnerability in Huawei routers [19, 77, 78, 134], which received an EPSS score of just 0.05, despite being one of the most exploited IoT CVEs in the wild. This disconnect may stem from EPSS being primarily trained on general-purpose IT vulnerabilities, limiting its applicability to IoT-specific threat landscapes.

In IoT contexts, features such as device type [19, 34, 43, 225] and internet exposure [34, 195] have been shown to drive exploitation more effectively than traditional severity metrics [19, 34, 166]. Nevertheless, collecting these IoT-specific features remains labor-intensive and lacks scalability, often requiring significant manual effort.

However, the most recent iteration of EPSS have started incorporating a new feature derived from community signals, such as social media discussions, to better capture attacker interest and behavior [113]. For instance, they use signals from X (formerly Twitter), which shown to outperform conventional indicators like CVSS scores or PoC availability in predicting exploitation [215, 235]. However, even richer insights may lie in underground forums and dark web discussions, which more directly reflect attacker intent and planning. Tavabi et al. [238] demonstrated that 14% of vulnerabilities mentioned in such forums were later exploited in the wild, outperforming traditional feature sets. Similarly, Almukaynizi et al. [16] found forum-based mentions had the highest exploitation rates across multiple datasets.

This presents a clear opportunity to assess and improve the predictive performance of such models for IoT vulnerabilities, by incorporating attacker-centric features, especially

those derived from hacker community discussions, into the existing framework. Understanding which features improve prediction accuracy for IoT vulnerabilities, specifically, remains a critical and unresolved question.

THE UNDEREXPLORED ROLE OF POCS DEVELOPERS IN EXPLOIT AVAILABILITY

Although public PoC exploits are strongly associated with weaponization [81, 179], we know little about how PoCs developers select which vulnerabilities to target and disclose. Existing research often overlooks the social, ethical, and motivational factors that shape exploit development.

The complexity of this relationship stems in part from the diversity of PoCs development practices. While some developers create minimal exploits to trigger crashes or logs, others publish fully weaponized exploits. The motivations behind PoCs development and publication vary, from improving security posture and transparency to demonstrating technical skill or pressuring vendors into patching [254]. Yet, these behavioral dynamics remain poorly understood in the academic literature. Most studies focus on the *what* and *when* of PoCs publication [63, 254], but little research has investigated the *who* and *why*: the motivations, workflows, and disclosure decisions of the developers creating these PoC exploits proofs.

As PoC publication influences attacker uptake, understanding these human processes is essential for improving coordinated disclosure and responsible vulnerability management.

1.3. RESEARCH QUESTION

This dissertation aims to systematically investigate the key factors that influence the exploitation of IoT vulnerabilities. Specifically, it explores how attacker behaviors, vulnerability characteristics, community influence, and exploit development practices interact and collectively shape exploitation activity in IoT systems. This overarching goal is guided by the following research question:

What factors shape the exploitation in IoT vulnerabilities, from target selection to exploit development and prediction?

To investigate this broad question, the research is divided into four sub-questions, each addressed in a separate study:

1.3.1. STUDY 1: LONGITUDINAL ANALYSIS OF IOT MALWARE EXPLOITS

Despite evidence that IoT devices are attacked within minutes of being online, little is known about how malware exploits evolve and persist over time. This study investigates how IoT malware has evolved over time by analyzing 17,720 malware samples collected from 2015 to 2020. Using static and dynamic analysis, it examines trends in exploit vectors, lifespan of vulnerabilities, exploit functionality, and targeted devices. Key findings show that IoT malware has grown more complex, often reusing old vulnerabilities, with an average time-to-exploit of 29 months after disclosure.

RQ1: *How do attackers select and persist in exploiting IoT vulnerabilities over time?*

1.3.2. STUDY 2: THE INFLUENCING FACTORS IN IOT MALWARE TARGET SELECTION

While prior studies have examined target selection in general cybercrime, there is limited understanding of the factors that influence attackers to focus on specific IoT vulnerabilities. This study analyzes 11,893 IoT malware binaries distributed between 2018 and 2021 to identify key factors influencing exploit frequency in IoT vulnerabilities. Although most binaries, 78%, scan for weak credentials, the remaining subset includes specific exploits. By developing a theoretical model from relevant literature to examine the impact of incorporating indicators derived from four latent variables: exposure, vulnerability, exploitability, and patchability. The study finds that PoC exploit availability, device prevalence, and patch complexity are stronger predictors of targeting than severity scores.

RQ2: *What factors influence attackers' choices of frequently targeting certain IoT vulnerability than others?*

1.3.3. STUDY 3: PREDICTING EXPLOITATION OF IOT VULNERABILITIES

Traditional metrics like CVSS scores and the presence of PoC exploits often fail to fully explain real-world exploitation patterns. While the state-of-the-art exploit prediction model EPSS offers improvements, it remains poorly adapted to IoT environments, overlooking the unique behaviors and threat dynamics associated with IoT attacks. To address this gap, this study extends the EPSS model by incorporating behavioral data extracted from hacker forums. By analyzing 23,373 IoT-related vulnerabilities and 25,000 forum posts, the study identifies 38 features that significantly influence IoT exploitation. Incorporating attacker behavior–related features improves the model's prediction accuracy by 17% and explanatory power by 21%, highlighting the value of integrating attacker discourse alongside technical indicators.

RQ3: *How to improve existing models like EPSS to perform better for IoT vulnerabilities?*

1.3.4. STUDY 4: THE ROLE OF POCS DEVELOPERS IN EXPLOIT AVAILABILITY

Despite ongoing debate over the risks and benefits of publishing Proof-of-Concept (PoC) exploits, little is known about the motivations and decision-making of the developers behind them. Based on interviews with 16 exploit developers, this study explores the creation and disclosure of PoCs. It finds that development and publication practices vary widely, influenced by factors such as personal ethics, past experience, and vendor interactions. Developers range from those who produce basic crash proofs to those crafting fully weaponized exploits. The study offers recommendations to improve coordinated disclosure and foster better dialogue between vendors and the research community.

RQ4: *What are the decision-making processes PoC exploit developers follow that guide the reporting and public disclosure of PoC exploits?*

1.4. DISSERTATION OUTLINE

The outline of the dissertation consists of the four research studies, each addressing their corresponding research question. Table 1.1 provides an overview of the peer-reviewed publications associated with each study and chapter.

Table 1.1: Dissertation outline.

Chapter	Research Question	Publication
Ch. 2	RQ1	**Arwa Abdulkarim Al Alsadi**, Kaichi Sameshima, Jakob Bleier, Katsunari Yoshioka, Martina Lindorfer, Michel van Eeten, and Carlos H. Gañán. 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22). Association for Computing Machinery, New York, NY, USA, 309–321. https://doi.org/10.1145/3488932.3517408* [3]
Ch. 3	RQ2	**Arwa Abdulkarim Al Alsadi**, Kaichi Sameshima, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2023. Bin There, Target That: Analyzing the Target Selection of IoT Vulnerabilities in Malware Binaries. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). Association for Computing Machinery, New York, NY, USA, 513–526. https://doi.org/10.1145/3607199.3607241* [4]
Ch. 4	RQ3	**Arwa Abdulkarim Al Alsadi**, Mathew Vermeer, Takayuki Sasaki, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. Bits and Pieces: Piecing Together Factors of IoT Vulnerability Exploitation. In *Proceedings of the 2025 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Hanoi, Vietnam, https://doi.org/10.1145/3708821.373387* [5]
Ch. 5	RQ4	**Arwa Abdulkarim Al Alsadi**, Lorenz Kustosch, Lamya Alowain, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. "Oh, what people would do with my knife?" Navigating the Dual-Use Dilemma in PoC Exploit Development, Disclosure, and Community Dynamics. *Under submission to USENIX Security '26*

2

Longitudinal Analysis of IoT Malware Exploits

The Internet of things (IoT) is composed by a wide variety of software and hardware components that inherently contain vulnerabilities. Previous research has shown that it takes only a few minutes from the moment an IoT device is connected to the Internet to the first infection attempts. Still, we know little about the evolution of exploit vectors: Which vulnerabilities are being targeted in the wild, how has the functionality changed over time, and for how long are vulnerabilities being targeted? Understanding these questions can help in the secure development, and deployment of IoT networks.

We present the first longitudinal study of IoT malware exploits by analyzing 17,720 samples collected from three different sources from 2015 to 2020. Leveraging static and dynamic analysis, we extract exploits from these binaries to then analyze them along the following four dimensions: (1) evolution of infection vectors over the years, (2) exploit lifespan, vulnerability age, and the time-to-exploit of vulnerabilities, (3) functionality of exploits, and (4) targeted IoT devices and manufacturers. Our descriptive analysis uncovers several patterns: IoT malware keeps evolving, shifting from simply leveraging brute force attacks to including dozens of device-specific exploits. Once exploits are developed, they are rarely abandoned. The most recent binaries still target (very) old vulnerabilities. In some cases, new exploits are developed for a vulnerability that has been known for years. We find that the mean time-to-exploit after vulnerability disclosure is around 29 months, much longer than for malware targeting other environments.

This chapter has been published as: **Arwa Abdulkarim Al Alsadi**, Kaichi Sameshima, Jakob Bleier, Katsunari Yoshioka, Martina Lindorfer, Michel van Eeten, and Carlos H. Gañán. 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22). Association for Computing Machinery, New York, NY, USA, 309–321. https://doi.org/10.1145/3488932.3517408*

2.1. INTRODUCTION

The widespread deployment of Internet-of-Things (IoT) devices, such as IP cameras and smart home appliances, bring us new services but also provides new resources for attackers to compromise. Indeed, devices have been infected at scale [23]. Attackers rely on exploits as infection vectors for IoT [19]—contrary to attacks on desktop and mobile devices, which have increasingly included vectors based on social engineering and user interaction.

While our knowledge has increased about IoT malware families and their capabilities [71, 260], we know much less about the attackers behavior in terms of the vulnerabilities they target with their exploit code. Just as the overall number of newly discovered vulnerabilities keeps growing [188], so does the number of IoT-related vulnerabilities: from a dozen or so reported in 2010 to more than 500 in 2019 [42]. Which of these vulnerabilities are targeted? Do the authors of different malware families go after the same vulnerabilities? How soon do they target a new vulnerability after it has been published? For how long do they keep focusing on a specific vulnerability? We have insights into these behaviors for desktops and servers, where attackers predominantly target the last-but-one version of the software after a patch is released [9, 227, 250].

This pattern is unlikely to hold for IoT, however, since patching is more spotty and difficult in this ecosystem [243].

Earlier research has investigated exploit code used by specific malware families at specific times [23, 73, 124], but we lack a systematic understanding of how vulnerabilities are targeted over time across the IoT malware landscape. The closest related work is a concurrent study by Alrawi et al. [19]. The study conducted a wide-ranging analysis of IoT malware binaries collected in 2019 and identified 25 exploits observed in that year via static analysis. We end up confirming some of the results of this paper. We also go beyond it by directly observing the longitudinal evolution of exploits across five years and across malware families, as well as the timeline of the targeted vulnerabilities. We identify 63 exploits and track them over time, as well as the 68 vulnerabilities they are targeting. This allows us to present previously unobserved patterns in exploit lifetime, vulnerability lifetime, and time-to-exploit.

We base our analysis on IoT malware binaries from three different sources spanning five years (2015–2020). We analyzed 5,855 samples from a hybrid low-high interaction honeypot [197], collected between September 2018–August 2020, and 2,292 samples from the URLhaus malware repository [246], collected between July–October 2020. We clustered all binaries with AVClass2 [217] and Vhash [252] based on their binary similarity. For the URLhaus and honeypot datasets, we identified 156 clusters across 9 different malware families. We conducted dynamic analysis on both datasets to extract exploits. We also manually reverse engineered one sample for each cluster to extract further exploit functions. For all discovered exploits, we created a signature, which we could then trace further back in time using the Genealogy dataset, which was collected from VirusTotal [226] between 2015–2018 [74]. This allowed us to analyze the occurrence of these exploits in 17 additional malware families, beyond the 9 families from the URLhaus and honeypot datasets.

By mapping these exploits to vulnerabilities with and without published Common Vulnerabilities and Exposures (CVE) entries, we observe the evolution of infection vectors

over the past five years. Whereas IoT malware started out exclusively focusing on brute-forcing credentials, infection vectors have since greatly diversified. Beyond brute forcing, we have identified 63 unique exploits across 26 families that target 68 vulnerabilities associated with 49 different manufacturers. The targeted vulnerabilities include recently discovered ones as well older ones, going back up to 12 years. Many vulnerabilities keep being targeted by new binaries for years on end, suggesting that patching is not rendering these attacks ineffective. In summary, our main contributions are:

- We present the first longitudinal study on IoT malware quantifying the exploits, targeted vulnerabilities and devices spanning 5 years. We find that attackers rarely abandon an exploit over time and that the number of targeted vulnerabilities has doubled every year since 2017.
- We create 63 unique signatures for the HTTP requests of each exploit and make these available to the community to facilitate detection.
- The time between a vulnerability publication and the first occurrence of an exploit in a binary is 29 months on average, though this is skewed by some very old vulnerabilities.
- In terms of exploit lifespan, we see that on average exploits are used for 23 months, a number that is increasing as most exploits were still in use in the latest binaries.

We make the datasets, network artifacts, and developed signatures available at:
https://doi.org/10.4121/19248725

2.2. METHODOLOGY

Our goal is to track the evolution of exploit code in IoT malware to gain insights into which vulnerabilities and devices they target over time. Identifying exploit code can be achieved via static and dynamic analysis of binaries. Automated static analysis is prone to code obfuscation and packing,

while manual static analysis—i.e., reverse engineering—is resource intensive, yet more complete and reliable. In contrast, automated dynamic analysis can deal with packing and is more scalable, but comes with the drawback of limitations in the coverage of exploits. We combine manual static analysis and automated dynamic analysis to leverage the strengths of each approach. We then supplement these results by a targeted exploit search in a binary repository that covers three years before our binaries were collected. We present a high-level overview of our methodology in Figure 2.1.

2.2.1. DATA COLLECTION

IOT MALWARE BINARIES

We collect current samples from two different sources (URLhaus and a honeypot) to create exploit signatures, which we then match against an older dataset (Genealogy) to cover an earlier period of time.

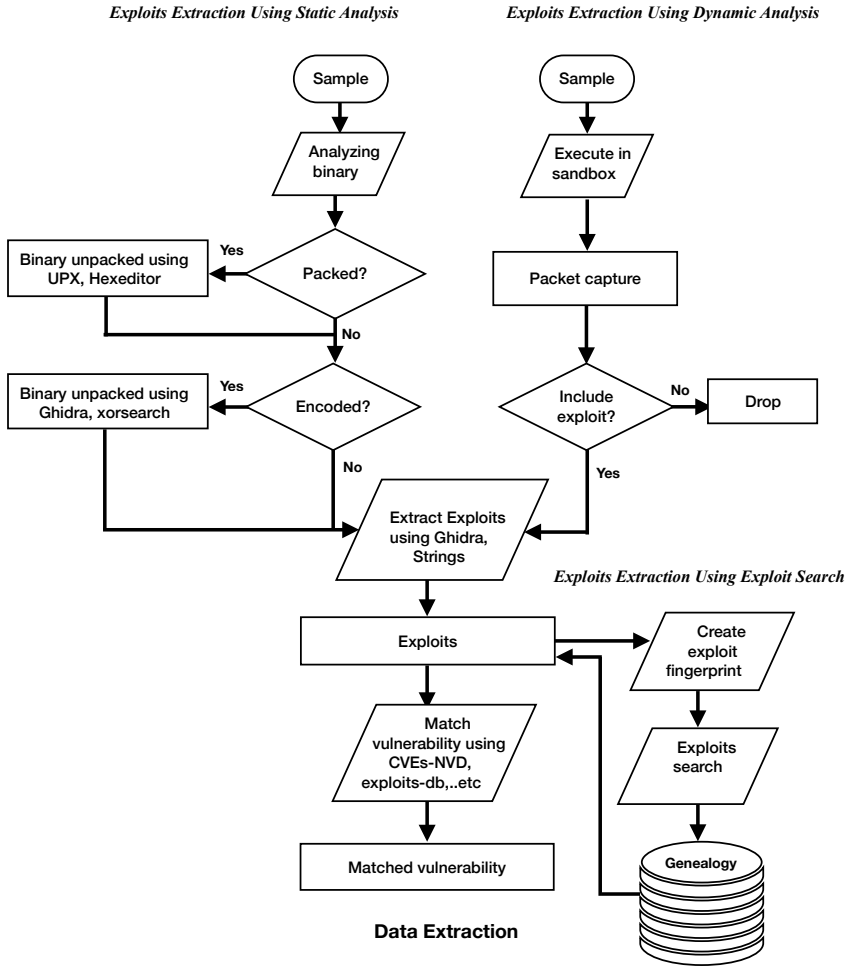


Figure 2.1: Analysis pipeline. We extract exploits using static and dynamic analysis from samples collected from 2018 to 2020 to create signatures, which we then match against a Genealogy dataset of unpacked samples from 2015 to 2018.

URLhaus: This repository collects URLs that are being used for distributing malware [246]. We used this source to collect a dataset of recent binaries as input for our dynamic and static analysis. From July 2020 to October 2020, we retrieved a daily file containing URLs of all captured binaries and other related information, such as file type. Since we focus on IoT malware, we extracted URLs only related to "Executable and Linkable Format" (ELF) files. We ran a daily script to download those files during the 4-month period. In total, we downloaded 2,298 binaries for different architectures including Renesas SH, Motorola 68000, SPARC, Intel 80386, ARM, PowerPC, MIPS, ARC Cores Tangent-A5,

Table 2.1: Number of collected samples per dataset, collection time period, and included malware families.

Dataset	# Binaries	# Shell Scripts	Malware Families (Avclass2)	# Vhash	# Singletons	Collection Period
URLhaus	2,292	6	Mirai, Gafgyt, Tsunami, Xorddos, Hajime, Mozi	108	6	Jul 2020 - Oct 2020
Honeypot	5,855	2,815	Mirai, Gafgyt, Tsunami, Xorddos, Hajime, Mozi, Berbew, Generica, Silex	107	59	Sep 2018 - Aug 2020
Genealogy	6,752	0	Mirai, Gafgyt, Tsunami, Hajime, Generica, Casur, Ddostf, Dnsamp, Dofloo, Gixec, Goram, Iotreaper, Luabot, Minerd, Pnscan, Qysly, Rakos, Ramgo, Remaiten, Satori, Shishiga and Ztorg	277	144	Jan 2015 - Aug 2018

and AMD x86-64. Note that we include x86-based malware as previous work has shown that this is present in IoT devices [182].

While 6 of the 2,298 files were actually shell scripts, we manually verified that their functionality only included binary downloads and no propagation techniques.

Honeypot: We obtained 5,855 MIPS binaries from September 2018 to August 2020 via IoTPOT [197]. IoTPOT is a combination of low-interaction and high-interaction honeypot: the low-interaction honeypot emulates different network services, such as Telnet, HTTP front-ends, CPE WAN Management Protocol (CWMP), a backdoor of Netis routers, and the remote access setup service of several IP cameras. The high-interaction honeypot uses four bare-metal vulnerable IoT devices (a router, an IP camera, and two WiFi storage devices). The honeypot is connected to 130 IP addresses in Japan.

We also obtained a dataset of 2,815 files captured by IoTPOT whose file type were not ELF binaries. We executed them as shell scripts in a closed environment and found that 2,608 had the functionality of downloaders using wget, curl, etc. The remaining 207 files contained 10 Python scripts, two Perl scripts, and the remaining 195 files were recognized as ASCII texts but not script files. We then manually inspected these 10 Python and two Perl scripts and executed them in a sandbox and found that only one Python script includes exploits. Thus, we believe it is reasonable to focus the remainder of our analysis on the binary samples.

Genealogy: We supplemented the first two datasets with samples collected by a research project studying the genealogy of IoT malware [74, 75]. The dataset consists of IoT binaries submitted to VirusTotal [226] between January 2015 and August 2018. The authors selected binaries with file type *ELF* that had been detected by at least five anti-virus (AV) vendors, amounting to a total number of 93,652. For their study, they focus on a subset of 6,752 binaries, which they unpacked with their analysis pipeline. For our analysis, we also use this set of unpacked binaries.

IOT DEVICE VULNERABILITIES

Once we identified exploits, we used the following sources to collect information about the targeted vulnerabilities:

National Vulnerability Database (NVD [177]): This database publishes Common Vulnerabilities and Exposures (CVEs) [239], including their identification number, a description, and public reference(s) for each vulnerability. Only 47 out of the 68 vulnerabilities (67.14%) that we identified have an associated CVE ID. This is a general issue also reported by VulnDB [257], which identified over 83,472 vulnerabilities that never received a CVE ID. We further found one exploit mapped to two CVEs from different years. We also encountered the opposite: two exploits targeting same CVE. Thus, an exploit can be linked to one or multiple vulnerabilities.

Exploit-DB [105]: We used this database of public exploit code and corresponding vulnerabilities as a source for exploits that were not assigned to any CVE.

Other public reports: For 15 exploits, there was no information in the above sources. We used keywords (e.g., strings found in HTTP headers, credentials, function names) to locate reports from AV vendors and researchers describing those vulnerabilities.

2.2.2. MALWARE CLASSIFICATION AND SAMPLING

For all the binaries across the three datasets we used VirusTotal [226] to collect metadata about these files, such as the time a sample was first submitted to VirusTotal, its architecture, and AV labels. We further used AVClass2 [217] to assign malware family labels based on the majority of votes by the individual AV labels. Table 2.1 summarizes the results. If a sample does not have enough votes to assign it to a specific malware family, we label it as a Singleton [218].

Since manual reverse engineering is not feasible for thousands of binaries, we turned to VirusTotal in-house clustering based on Vhash [252]. It is a fuzzy hashing algorithm that clusters files based on their structural similarity.

In total, we obtained 108 different clusters for the URLhaus binaries and 107 clusters for the honeypot binaries. For the manual reverse engineering, we randomly selected one binary from each cluster. The binaries categorized as Singletons all have a unique Vhash, i.e., were assigned their own cluster. Therefore, we included all of them in our subset. Since binaries in each cluster are highly similar, we assume they also share the same exploit code. To double check this assumption, we randomly choose and reversed engineered two additional samples from five clusters. We found that all three binaries from each cluster indeed shared the same exploit code.

2.2.3. EXPLOIT EXTRACTION AND SIGNATURE MATCHING

We perform dynamic analysis on the whole URLhaus and honeypot datasets, manual static analysis on a set of binaries sampled from all clusters, and finally generate and match signatures against the Genealogy dataset.

EXPLOIT EXTRACTION USING DYNAMIC ANALYSIS

We execute each of the 2,298 binaries from URLhaus and 5,855 MIPS binaries from the honeypot in a malware sandbox. The sandbox is constructed as a virtual machine running Linux Debian for the MIPS, ARM, and x86-64 architectures using VirtualBox. we execute each binary for five minutes in an isolated network environment, except for DNS resolutions. This isolated environment helps us distinguish between port scans and C&C communications, especially when they are on the same port, as malware keeps trying

to connect the C&C server in such an environment while port scans on the same host are not usually repeated. Moreover, we determine that a destination port is scanned if the number of destination IP addresses accessed by the binary on the port exceeds our heuristic threshold of 100. After the cleanup of the sandbox, we execute the binary again for five minutes with dummy servers running on the scanned ports. All connection attempts on the scanned port are then redirected to the dummy servers, implemented with PyNetSim, a network simulator for malware analysis. The current version of the dummy servers only establishes TCP sessions and does not respond any further. While this is a clear limitation, we find large portion of the binaries do start scanning and exploiting vulnerabilities right after their execution without interacting with C&C servers or their targets. During the dynamic analysis, 109 binaries from URLhaus and 4,700 binaries from the honeypot dataset were successfully executed and started exploiting.

We then examine the obtained attack payloads to check if they include any exploits. We manually extract the exploits as a characteristic sub-string of the payloads—for example, the target resource of HTTP GET and HTTP POST requests.

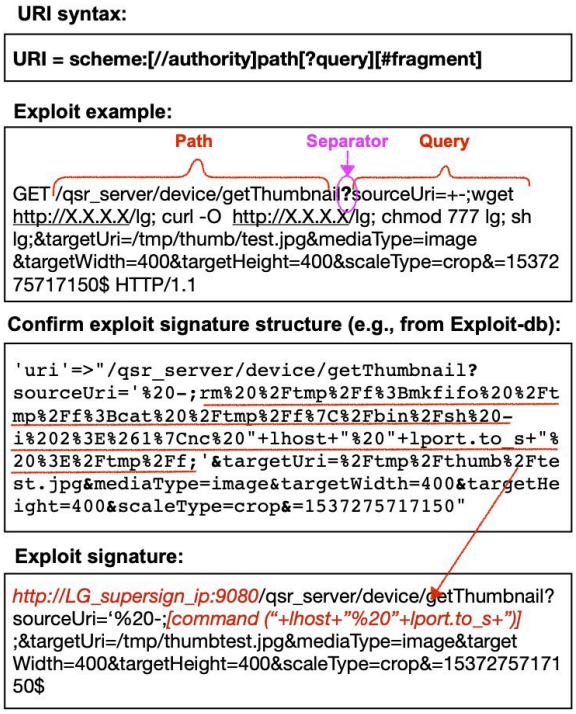


Figure 2.2: Example of a signature we generated for an exploit against CVE-2018-17173 [106, 187].

EXPLOIT EXTRACTION USING STATIC ANALYSIS

We conducted the manual static analysis on the subset of 156 binaries following the steps shown on the left side of Figure 2.1. The first step is to unpack the binary, if necessary. Malware authors use certain tactics to avoid detection and analysis. One of the most

used methods is malware packing. A packer can compress, encrypt and obfuscate a program. Authors can also manipulate headers to corrupt them [159]. We run the binary in DIE [126] to check whether it is packed and, if so, by which packer. If the packer can be identified, we can use it to unpack the binary. For our samples, UPX [190] was the only packer we encountered. We could unpack 12 binaries with a simple `'upx -d filename'`. One sample from the Mozi family used custom UPX packing to hamper automatic unpacking. Customization to UPX can range from simply patching one byte, to more thorough modifications [190, 245]: modifying the UPX! magic headers, ELF magic bytes, the copyright string, section header names, and adding extra junk bytes throughout the binary are common. We unpack binaries packed with UPX manually by using a hexeditor [131] to fix the headers.

Next, we checked whether a binary is obfuscated or further encrypted by using `'strings -a'` [233] to check whether we can read the binary in plain text. If this is the case, we start then to read the whole binary and extract hard-coded credentials as well as exploits in HTTP requests. If this was not possible, we used Ghidra [184] to find the encryption key of 15 binaries and use one of its built-in scripts to decrypt them. If Ghidra could not locate the key, we identified the key ourselves and fed it to the built-in script for the decryption. We then either continued the analysis and exploits extraction in Ghidra or with `strings`. Some binaries still contained gibberish text even after the decryption. Therefore, we used XORSearch in those cases to make sure we extracted all the exploits. From our results from other binaries, we knew which keywords to use to help extract the exploits. For instance, we used `"http"`, `"GET /"`, `"POST /"` and other combinations from the HTTP syntax, as well as hard-coded credentials. If XORSearch found a decoded match, it returned the match with the XOR key used to encode it. Decoding the whole file by using this key allowed us investigate the binary further to find more hidden exploits.

EXPLOIT SIGNATURE GENERATION AND MATCHING

For every exploit we also created an exploit signature as an ordering of strings that can be used to uniquely identify an exploit. For each exploit in HTTP request format, we extracted the header. We used the request target, URI (Uniform Resource Identifier) or absolute path, following the GET or POST method in the request line. We used the standard URI syntax to generate signatures. The first box in Figure 2.2 shows the generic URI syntax that consists of a sequence of components: scheme, authority, path, query and fragment. The most common form is the origin form where an absolute path is followed by a '?' and then a query string. The bottom three boxes in Figure 2.2 show an example of how we generated an exploit signature. We extracted the HTTP request of an exploit and identified the URI syntax of it. Based on the information we found in public databases—e.g., Exploit DB— we confirmed the structure of the exploit signature. For example, in the third box, the parameter that comes after the path and the query was `'sourceUri'` that was triggering a command, then followed by the payload. The colored part might be adapted by attackers, so this could change the signature, while the remainder is stable.

While we mainly focus on exploits, we also looked for brute forcing capabilities as an infection vector. To this end, we used hard-coded credentials to create one signature for all binaries that rely on brute forcing. We used these signatures to extend the timeline of the exploits by tracing back their presence in the binaries from the Genealogy dataset.

These binaries were collected between 2015–2018, so in the three years before the samples collected from URLhaus and the honeypot. We implemented a script to automate the exploit search based on matching the 63 exploit signatures besides binaries that rely on brute forcing credentials with the 6,752 unpacked binaries.

Table 2.2: Observation port scanning in the honeypot dataset during dynamic analysis.

Infection Vector	Protocol	Port
Backdoor	Telnet	443, 9530
Brute Force	Telnet	2223, 23, 23023, 2323, 2332, 26, 5885, 6000, 60000, 9001
Android Debug Bridge Shell CMDi	ADB	5555
Buffer Overflow	HTTP	443
Command Injection	HTTP	80, 88, 8081, 8080, 60001, 8000, 5501, 8082, 81, 8089, 8443, 5500, 49152, 443
Remote Code Execution	HTTP	37215, 1723, 55555, 443, 7547, 8080, 52869, 1024, 1234, 50000, 6666, 8001, 8081, 9080, 80, 8181, 9090, 161, 9000, 5555, 7574, 81

2.3. EXPLOIT LANDSCAPE

We now present the findings on exploits and vulnerabilities in IoT malware. Table 2.3 shows the findings across the three different datasets. Overall, we found 64 infection vectors: brute forcing hard-coded credentials, as well as 63 unique exploits that target 68 vulnerabilities. The occurrence of each vulnerability for each dataset is presented in the rightmost column in Table 2.3. We identify exploits, infection vectors, targeted vulnerabilities and device manufacturers.

Two groups of URLhaus binaries did not contain exploits and are not included in the table. The first group of 27 out of 108 (25%) binaries only included hard-coded credentials for brute forcing. A second group of 11 binaries (10%) did not include any infection vector in the code and only contained functions to receive commands from a command-and-control (C2) server and execute attacks. Prior work [155] has found that these commands implement attack vectors such as: UDP flood, SYN flood, ACK flood, TCP flood, UDP flood, VSE flood, DNS flood, GRE IP flood, GRE Ethernet flood and HTTP flood. These binaries belonged to Tsunami, XORDDoS, Hajime or were Singletons. For the remaining 70 binaries (65%), we found they contained 256 exploits targeting web vulnerabilities—i.e., exploits based on HTTP `GET /` and `POST /` requests.

In Table 2.3, we categorized the vulnerabilities into six groups, based on the vulnerability description in NVD or Exploit-DB: *Remote Code Execution (RCE)*, *Backdoor*, *Command Injection (CMDi)*, *Buffer Overflow*, *Web Application Firewall (WAF) Bypass* and *Brute Force*. More than half of the total vulnerabilities (55.88%) used Remote Code Execution (RCE) as the infection vector. RCE is also the most frequently targeted vulnerability type in both

the URLhaus and the honeypot dataset, (55.9%) and (53.65%) respectively. Among the shared vulnerabilities across all the three datasets, CMDi was the most frequently used infection vector (56.25%).

Mirai used to infect vulnerable devices by using brute force, as was visible in its leaked source code [115] in 2016. More recent binaries have expanded the range of vectors: attackers now scan a number of different ports to exploit different protocols such as: Telnet, Android Debug Bridge (ADB), HTTP. Table 2.2 shows the protocols and ports we observed during the dynamic analysis in respect to each infection vector.

The second column in Table 2.3 shows the 68 unique vulnerabilities. Those that share the same exploit code or are targeted by more than one exploit are marked with '*'. These are the vulnerabilities we extracted from the URLhaus and the honeypot. From the 69 infection vectors (including brute force), 31 are shared between both URLhaus and the Honeypot. The remaining 38 are 27 vulnerabilities found in only the URLhaus dataset and 11 vulnerabilities found in the Honeypot. When matching our exploit signatures against the older Genealogy dataset, we found 16 vulnerabilities matched by 17 exploits signatures. All of them were also present in the URLhaus (i.e., the most recent) dataset except for one: we only found the "SonicWall GMS-XMLRPC CMDi" in the Honeypot and the Genealogy datasets (see the last two columns in Table 2.3).

Not all vulnerabilities are recent ones: The third column in Table 2.3 shows the date each vulnerability was published. The oldest targeted vulnerability, CVE-2009-0545, was published in 2009 and dates back 12 years. However, the same exploit code was used to target another vulnerability, CVE-2019-12725, that was published ten years after that, in 2019. On the other hand, the most recent vulnerability, CVE-2020-17496, was published in August 2020 and exploited a month after that.

Mirai is the most versatile family: it targets all vulnerabilities except for three. This is consistent with the fact that Mirai contained the highest number of binary clusters, so it is basically a family of families. The other malware families contained a subset of the exploits present in the Mirai family. For instance, in the URLhaus dataset, all of its vulnerabilities were present in Mirai samples except for one: CVE-2016-4429, which is a buffer overflow vulnerability in Qualcomm components that was only targeted by one binary categorized as a singleton (see rightmost column in Table 2.3).

Not only is Mirai the most versatile, it is also leading the way. Figure 2.3 shows a timeline of the occurrence of vulnerabilities and exploits in four malware families as well as singletons. Out of 68 vulnerabilities, 64 were observed first in Mirai samples before they were seen in other families. We discuss this figure in more detail in the next section. Mirai was also the only malware family with the most exploits per binary: up to 35 exploits, the maximum number of exploits we observed in a single binary that was part of the most recent samples from the URLhaus dataset (see Figure 2.4).

The only vulnerability that is exploited by all IoT malware families and found across all the three datasets was CVE-2017-17215, which targets Huawei HG532 home routers. In terms of targeted manufacturers and devices, this vulnerability also had the highest frequency across binaries, which is (somewhat) visible in Figure 2.3 from the density of dots, and in last column in Table 2.3. Routers were the most favorable IoT device type for attackers to target over the period of six years. As can be seen in Table 2.4, 2,413 (80%) of targeted IoT devices were routers. In fact, this number is larger than summing up all the

Table 2.3: Summary of infection vectors, vulnerabilities and devices across datasets (U=URLhaus, H=Honeypot, G=Genealogy). We identified 69 vectors (68 exploits + brute forcing default/weak credentials). ● indicates that we found samples exploiting a vulnerability across all three datasets, ◐ that we only found samples in two out of three datasets, and ○ that we only found samples in one dataset, demonstrating the value of samples from different vantage points and across time.

Type	Vulnerability	Vuln. Published	Exploit Published	Families	Manufacturer	Target Device	U H G	# of Samples
RCE	CVE-2009-0545; CVE-2019-12725 *	2009-02-12; 2019-06-04	2009-02-09	Mirai	Zeroshell	Zeroshell Linux Distribution	○	2
	Netgear DGN1000 RCE	2013-06-05	2013-06-05	Mirai, Mozi, Gafgyt	Netgear	DGN1000 Netgear routers	● ● ●	107
	Linksys E-series RCE	2013-07-02	2014-02-16	Mirai, Gafgyt	Cisco	Linksys routers E-series	◐ ◐	150
	Edimax EW-7438RPn-v3 RCE	2015-07-17	2015-07-17	Mirai	Edimax	EW-7438RPn-v3	○	4
	Multi-vendor CCTV/DVR RCE	2016-03-23	2016-03-23	Mirai, Mozi, Gafgyt	Multi-vendor	Multi-vendor CCTV/DVR	● ● ●	79
	NUUO NVRmini RCE	2016-08-06	2016-08-06	Mirai	NUUO	NUUO NVR	◐ ◐	4
	Xfinity Gateway RCE	2016-12-02	2016-12-02	Mirai	Xfinit	Xfinity Gateway	○	3
	CVE-2017-(8221-8225) *	2017-04-25	2017-03-08	Mirai	GoAhead	GoAhead IPcam	○	3
	EnGenius IoT GCS1.4.11 RCE	2017-06-04	2017-06-04	Mirai	EnGenius	EnGenius IoT Cloud Service	◐ ◐	3
	CVE-2017-14135	2017-09-04	2017-07-03	Mirai	Dream Property	Opendreambox	○	1
	CVE-2017-14127; CVE-2019-18396 *	2017-09-04; 2019-10-24	2019-11-13	Mirai	Technicolor	Technicolor TD5336	◐ ◐	5
	Vacron NVR RCE	2017-10-22	2017-10-08	Mirai, Mozi	Vacron	Vacron NVR devices	● ● ●	26
	Shenzhen_TVT RCE	2018-04-03	2018-04-09	Mirai	Shenzhen TVT	Shenzhen TVT DVR/NVR/IPC	○	3
	CVE-2018-10561; CVE-2018-10562 *	2018-04-30	2018-05-03	Mirai, Mozi, Gafgyt	Dasan	GPON Home Routers	● ● ●	259
	CVE-2018-11510	2018-05-28	2018-08-15	Mirai	ASUSTOR	ASUSTOR NAS	○	1
	HomeMatic Zentrale CCU2 RCE	2018-07-18	2018-07-18	Mirai	HomeMatic	HomeMatic Zentrale CCU2	○	3
	CVE-2018-15887	2018-08-26	2018-08-02	Gafgyt	ASUS	ASUS DSL-N12E_C1	○	6
	CVE-2018-17173	2018-09-18	2019-05-06	Mirai	LG	LG Supersign EZ CMS TV	◐ ◐	9
	CVE-2018-20062; CVE-2019-9082 *	2018-12-11; 2019-02-24	2019-01-14; 2020-04-16	Mirai, Singletons	ThinkPHP	v-5.0.23/5.1.31 Server	◐ ◐	21
	CVE-2019-2725	2018-12-14	2019-05-08	Mirai	Oracle	Oracle WebLogic Server	○	1
	CVE-2019-7276	2019-01-31	2019-11-12	Mirai	Optergy	Optergy 2.3.0a	○	3
	CVE-2019-10655	2019-03-30	2019-03-31	Mirai	Grandstream	GAC2500; GVC3202; GXV3275-40; GXP2200 *	○	3
	CVE-2018-20841	2019-06-11	2019-01-14	Mirai	HooToo	HT-TM05&HT-05 routers	○	1
	Sar2HTML 3.2.1 RCE	2019-08-02	2019-08-02	Mirai	Sar2HTML	Sar2html 3.2.1	○	3
	CVE-2020-9054	2020-02-18	2020-02-24	Mirai	Zyxel	Zyxel NAS and Firewall	◐ ◐	6
	Netlink GPON Router 1.0.11 RCE	2020-03-18	2020-03-18	Mirai	Netlink GPON	Netlink GPON Router 1.0.11	○	58
	Symantec SWG 5.0.2.8 RCE	2020-04-09	2020-04-09	Mirai	Symantec	Symantec Web Gateway 5.0.2.8	◐ ◐	34
	Netgear R7000 RCE	2020-06-15	2020-06-15	Mirai	Netgea	Netgear R7000	○	7
	CVE-2019-16759; CVE-2020-17496 *	2019-09-24; 2020-08-12	2020-08-12	Mirai	vBulletin 5.x	Servers using vBulletin 5.x	○	2
Backdoor	CVE-2014-2321	2014-03-10	2014-03-03	Tsunami	ZTE	ZTE F460 and F660	○	2
	Xiaongmai-based DVR/NVR/IPcam	2020-02-04	2020-02-04	Mirai, Gafgyt	Multi-vendor	DVR/NVR/IPcams	○	31
CMDi	CVE-2014-8361 *	2014-10-20	2015-06-01	Mirai, Mozi, Gafgyt	D-Link	D-Link Routers using Realtek SDK	● ● ●	272
	CVE-2014-9094	2014-11-26	2014-07-13	Mirai	WordPress	WordPress Plugin DZS-VideoGallery	◐ ◐	35
	CVE-2015-2051	2015-02-23	2015-06-01	Mirai, Mozi, Gafgyt	D-Link	D-Link DIR-645	● ● ●	93
	AVTECH IPCam/NVR/DVR CMDi	2016-10-11	2016-10-11	Mirai	AVTECH	AVTECH IPcam/NVR/DVR	◐ ◐	69
	CVE-2016-10372	2016-05-16	2016-11-08	Mirai, Mozi, Gafgyt	Zyxel	Eir D1000 Router (rebranded Zyxel)	◐ ◐	78
	CVE-2016-6277	2016-07-22	2017-03-13	Mirai, Mozi, Gafgyt	Netgear	Netgear R7000 and R6400	● ● ●	41
	NUUO OS CMDi	2016-08-06	2016-08-06	Mirai	NUUO	NUUO NVRmini 2 3.0.8	○	3
	MV Power Shell CMDi	2017-02-27	2017-02-27	Mirai, Mozi	MV Power	MVPower DVR TV-7104HE 1.8.4	◐ ◐	168
	CVE-2017-6884	2017-03-14	2017-04-02	Mirai	Zyxel	EMG2926 Router	◐ ◐	39
	CVE-2017-18368	2019-05-02	2016-12-26	Mirai, Singletons, Gafgyt	Zyxel	Zyxel P660HN-T routers	◐ ◐	77
	CVE-2017-17215	2017-12-04	2017-12-25	Mirai, Mozi, Gafgyt, Singletons	Huawei	Huawei home routers HG532	● ● ●	921
	CVE-2018-7841	2018-03-08	2019-05-14	Mirai	U.motion	U.motion software v.1.3.4	○	4
	D-Link DSL-2750B OS CMDi	2018-05-25	2018-05-25	Mirai	D-Link	D-Link DSL-2750B	● ● ●	241
	SonicWall GMS-XMLRPC CMDi	2018-08-01	2018-08-01	Mirai	SonicWall	SonicWall GMS	◐ ◐	1
	CVE-2018-19276	2018-11-14	2019-12-18	Mirai	OpenMRS	OpenMRS before 2.24.0	◐ ◐	5
	CVE-2019-7256	2019-01-31	2019-11-12	Mirai	Linear	Linear eMarge E3 series	○	1
	CVE-2019-12489	2019-05-30	2019-11-13	Mirai	Fastweb	Fastweb Fastgate 0.00.81	○	3
	CVE-2013-7471	2019-06-11	2013-09-17	Mirai, Mozi, Gafgyt	D-Link	D-Link DIR-645	● ● ●	29
	CVE-2019-14931	2019-08-10	2019-08-13	Mirai	Mitsubishi	Mitsubishi smartRTU& INEA ME-RTU	○	7
	CVE-2020-1956	2019-12-02	2020-06-20	Mirai	Apache	Apache Kylin 2.3.0-2.6.5,3.0.1	○	4
	CVE-2019-19824	2019-12-16	2015-07-16	Mirai	TOTOLINK	TOTOLINK Realtek SDK routers	○	7
	CVE-2020-5722	2020-01-06	2020-03-24	Mirai	Grandstream	Grandstream UCM6200 series	◐ ◐	5
	CVE-2020-7209	2020-01-16	2020-05-17	Mirai	HP LinuxKI	HP LinuxKI-v6.01	○	3
	CVE-2020-10173	2020-03-05	2020-02-27	Mirai	Comtrend	Comtrend VR-3033	◐ ◐	5
	CVE-2020-13786	2020-06-03	2020-06-12	Mirai	D-Link	D-Link DIR-865L Ax1.20B01	○	7
Buffer OF	CVE-2016-4429	2016-05-02	2016-05-18	Singletons	Qualcomm	Qualcomm Server	○	5
	CVE-2019-7405	2019-02-05	2019-12-16	Mirai	TP-Link	TP-Link Archer C5-v4 routers	○	4
WAF Bypass	Cloudflare WAF Bypass	2017-04-04	2016-10-25	Mirai, Gafgyt	CloudFlare	CloudFlare WAF	◐ ◐	37
Brute Force	Dictionary Attack	-	-	Mirai, Mozi, Singletons, Tsunami, Gafgyt, xorddos	-	-	● ● ●	5,631
Total							59 4116	

'*' indicates that this entry consists of vulnerabilities that are targeted by the same exploit code or vice versa

other IoT device types together.

Table 2.4: Number of time device types were targeted by binaries, for each dataset, and in total.

Device Category	URLhaus	Honeypot	Genealogy	Total
Router	461	1,342	610	2,413
Home security	93	219	78	390
Web application	36	38	32	106
Web server	22	10	-	32
TV	7	2	-	9
NAS	27	27	-	54
Total	646	1,638	729	3,004

2.4. EXPLOIT LIFESPAN

The number of IoT vulnerabilities, together with the number of exploits targeting these vulnerabilities, has been increasing over the years. Following the methodology explained in Section 2.2.3, we searched for matches for exploit signatures in the Genealogy dataset (2015–2018). Out of 64 exploit signatures, 17 signatures (associated with 16 vulnerabilities) had a match in the Genealogy dataset. Figure 2.3 illustrates one of the reasons for this limited presence of the exploits in older binaries: 32 (47%) of the vulnerabilities were published after August 2018, the end of the collection period of the Genealogy dataset. This still leaves 15 exploits targeting older vulnerabilities absent from the Genealogy dataset. Assuming that the Genealogy dataset is representative for that period, it would mean that these older vulnerabilities were selected by authors of more recent malware, years after they were first published. In total, we found matches in 5,421 samples out of the 6,752 binaries (80%). The lack of matches with the remaining 20% could reflect the use of abandoned exploit code that is no longer present in later binaries, but it is certainly also impacted by the number of packed and encoded samples in this dataset, as acknowledged by the researchers who created the repository. We discuss this limitation in Section 2.7.

With the extended timeline from the Genealogy dataset, we investigated the lifespan of the exploits, i.e., the time from when an exploit was first seen until the last time it was observed. We also investigate the *time-to-exploit*, i.e., the time that passes between the publication of the vulnerability and the first observation of a binary that contains exploit code for that vulnerability. For all binaries, we collected the 'first seen' date from VirusTotal.

Figure 2.3 visualizes the lifespan of malware exploit vectors by mapping the observations of the exploit in binaries (colored dots) and the publication dates of the underlying vulnerabilities (black X) and exploit code (red circle). In some cases, e.g., CVE-2013-7471, the date recorded in the CVE ID was years before the official publish date: 2019-06-11. We noticed in four more CVEs (CVE-2020-1956; CVE-2018-20841; CVE-2019-2725) that the publish date is not consistent with the CVE ID. That might explain why in some cases, the exploit publish date occurred well before the vulnerability publish date. The total

number of exploits that were published before the official vulnerability disclosure date is 20, though in most cases the dates were relatively close together, so this order might reflect inaccuracies in the underlying data rather than the true order of events.

Figure 2.3 also presents the evolution of malware families. All binaries that were first seen in 2015–2016 relied exclusively on hard-coded credentials for brute forcing. In total, these were 4,091 out of the 5,421 binaries. This might have been reinforced by the release of Mirai source code in November 2016 [115]. Mirai was the first IoT botnet able to amass millions of IoT devices and thus considered a leap in the IoT malware realm. Thus, since Mirai code had relied solely on brute forcing hard-coded credentials back then, the other variants and families appeared to follow suit. Since then, brute forcing has remained present in binaries up until the most recent data.

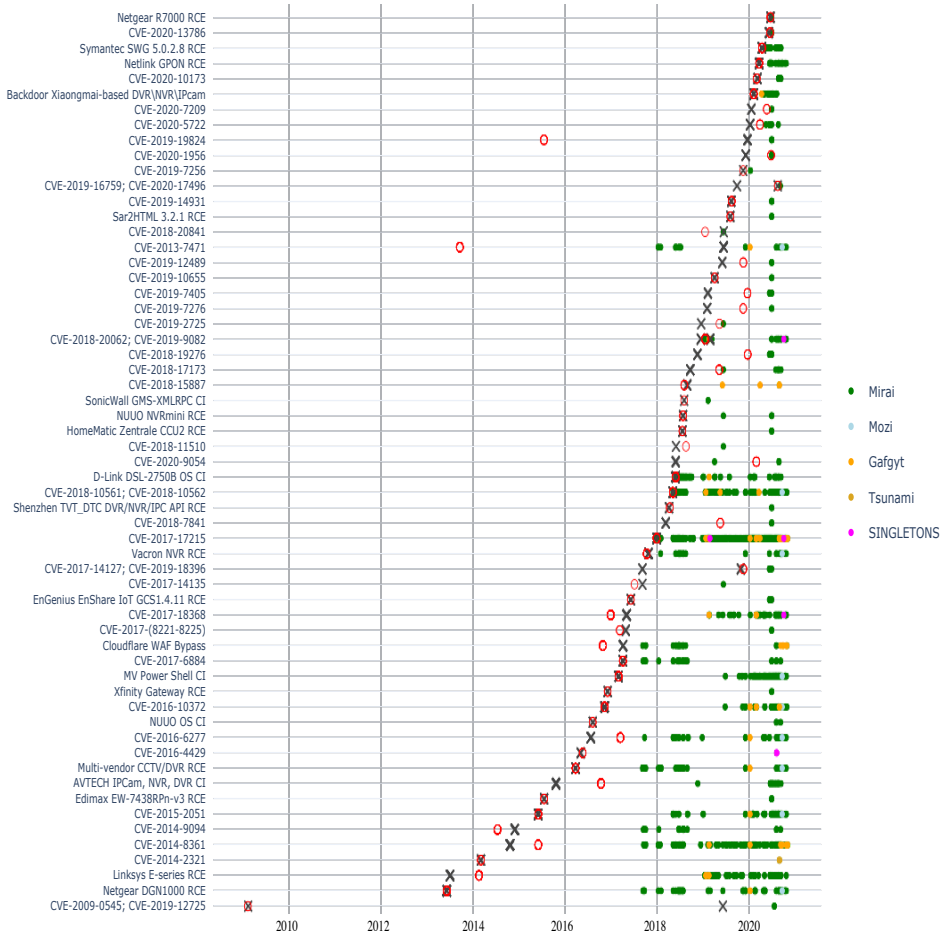


Figure 2.3: Occurrence of binaries that exploit vulnerabilities, per IoT malware family, ordered by vulnerability publication date. Each line represents a unique exploit. It might attack more than one vulnerability. The X symbol highlights the time the vulnerability was first published whereas the O symbol highlights the time the exploit was first discovered.

Figure 2.4 shows that, over time, the number of exploits per binary went up, as viewed by the difference across the datasets. Most binaries contain only one or two exploits, but from one dataset to the next, the upper part of the curve shifts towards the right. We found binaries in 2018 that had nine exploit signatures per individual sample. In binaries captured late 2020, as part of the URLhaus data, we found samples with up to 35 exploits, more than three times the number found in 2018. This is relative to the signatures we had, so it might undercount the exploits in the Genealogy dataset. Still, we see a similar increase compared to the honeypot binaries. Table 2.5 extends this picture by showing that not only individual binaries include more exploits, but that the total number of exploits rapidly increases over time across the malware landscape: from 2017 onwards, the total number of exploits and targeted vulnerabilities roughly doubles each year.

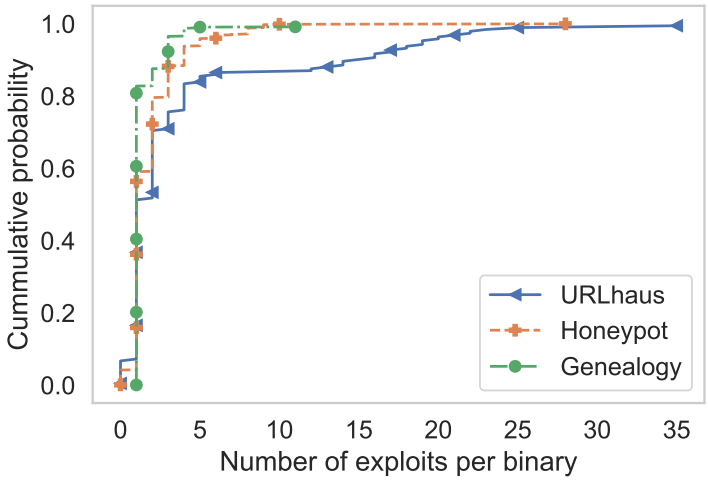


Figure 2.4: CDF of the number of exploits per sample.

Even though the first exploits only emerged in 2017, four of the underlying vulnerabilities had already been published in 2013 or earlier (even before the start of the collection period of the Genealogy dataset). We calculated the time-to-exploit, which is the time between the vulnerability publication and the first time the associated exploit was observed in binaries. On average, the time-to-exploit is 29 months, though the underlying distribution is skewed by very old vulnerabilities. The median is 26 months. When the authors of Mirai and other malware families started branching out from brute forcing to exploits, they apparently selected relatively old vulnerabilities to target first. This might reflect the device types they were interested in, namely routers and home security, as well as the size of install base of the targeted devices. Given that patching is spotty and difficult in the IoT ecosystem, they might have found that many devices were still vulnerable years after the vulnerability was first published.

For a closer inspection of the time-to-exploit for more recently discovered vulnerabilities, we calculated this metric for vulnerabilities published within the period of the honeypot binary collection (September 2018 – August 2020). All vulnerabilities published

Table 2.5: Number of hits (occurrence), exploits, and vulnerabilities per year.

Year	# Occurrences	# Exploits	# Vulnerabilities
2017	46	10	8
2018	727	15	15
2019	376	26	27
2020	1,855	58	63

before September 2018 were excluded. This resulted in 326 exploit occurrences that mapped to 20 unique vulnerabilities (95% in Mirai and the remaining in Gafgyt). The mean time-to-exploit in 2019 was 40 days (median: 30 days). For 2020, when most exploits appeared, the mean was 135 days (median: 96 days). So there does not appear to be a decrease in the time-to-exploit in recent years. The overall average of 2019 and 2020 is 129 days.

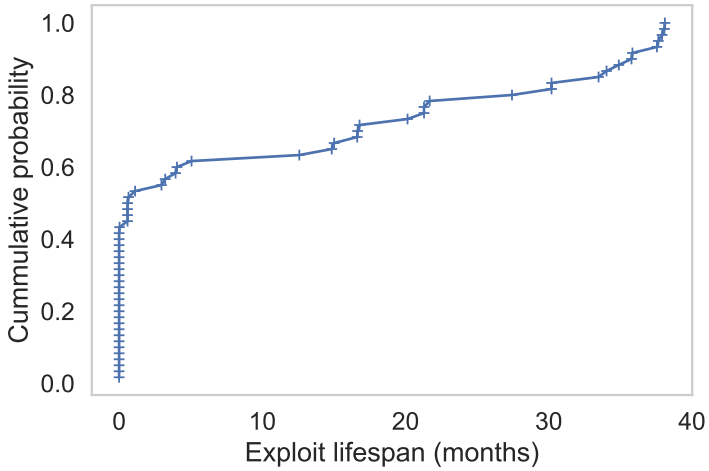


Figure 2.5: CDF of lifespan of each exploit.

The exploit lifespan is the period between when an exploit was first and last seen in binaries. Only in five cases did we observe a short attack window followed by a persistent absence of the exploit in future binaries. Most vulnerabilities are consistently targeted for months or years. On average, they are active for 23 months–and this number would be increasing, since these exploits were still present in the latest binaries we analyzed. Figure 2.5 also shows that the majority of vulnerabilities are exploited for months.

2.5. DISCUSSION

After Mirai's initial infection attempts by brute-forcing default (or weak) credentials, the arsenal of exploits has rapidly increased. Compared to prior work [19], which identified 25 exploits, a more alarming picture is emerging from our findings. Not only did we identify a larger set of exploits and targeted vulnerabilities, namely 68 of each, we also

demonstrated that the evolution of exploits is gaining more and more velocity. Since 2017, the number of exploits and targeted vulnerabilities has roughly doubled every year.

Our findings also provide new insights into the attackers' behavior. For roughly half of all vulnerabilities, the attackers persist in their attacks for two years or more, while for the other half we see a pattern where an exploit is used for a short period and then abandoned. The latter might reflect a pattern of trial and error. If the exploit code is successful enough in recruiting bots, the attacks persist. The longer an attack persists, the more likely it is that the exploit code gets copied across clusters and families. Then, the vulnerability is consistently being attacked for years.

Another interesting finding is the selection of vulnerabilities by the attackers. IoT malware developers select old vulnerabilities, compared to malware developers for desktop operating systems or server software. The latter have been found to focus on reverse engineering the latest vulnerabilities from patch releases and to then attack the one-but-last version of the software [250]. For desktop and server software, the time-to-exploit—i.e., the time between the publication of a vulnerability and the first observation of a binary attacking that vulnerability—typically ranges from a single day (e.g., "Exploit Wednesday" following on Microsoft's "Patch Tuesday") to, at most, a few months for a few high profile attacks as Wannacry and Not-Petya [92, 268]. This strategy makes sense, given that the largest share of vulnerable systems is running the one-but-last version of the software.

We see a dramatically different pattern for IoT malware. The mean time-to-exploit is a stunning 29 months (median is 26 months). This attacker strategy is rational if devices are not really updating their software, since then whatever version was running on the devices when they were originally deployed into the market will remain the same. While new releases might or might not be developed by the manufacturer, our evidence clearly suggests that for the targeted devices, available patches are not being adopted by the device owners. In fact, a recent study by Xie et al. [269] found that, on average, 64.3% of unpatched devices were affected by 176 vulnerabilities due to lack of updates. A recent illustration of this attacker strategy was provided by the so-called Meris botnet, which launched "record-shattering" DDoS attacks against Yandex and Cloudflare [152]. The botnet consisted of 250,000 compromised MikroTik routers that were running different versions of the RouterOS software. By far the largest share was running 6.45.9 [209], which is a whopping 21 versions behind the latest stable release.

How can we improve our defense against the increasing velocity of IoT exploit development? Our data suggests that core of the issue is patching—or rather, the lack of patching. Assuming the observed attacker behavior is rational, then the owners of the devices are not installing updates for years or perhaps ever. This is consistent with other research [248]. The obvious path forward would then be to educate users on the importance of patching these devices.

The problem with relying on users to patch their devices is that manufacturers do not always make patches available and, even if they do, the install path is often far from user-friendly. Many users have incorrect mental models for IoT malware infections. They might see the devices as appliances rather than computers [85, 213]. One study on IoT malware remediation found that many users had no clue on how to interact with their device, even for relatively simple tasks like changing the default password [45].

Another key stakeholder for mitigating this problem is the population of Internet

Service Providers (ISPs). The bulk of all IoT infection reside in the networks of consumer ISPs [56, 183, 212, 273]. Increasingly, ISPs are taking on the task of notifying customers with infected IoT devices, in line with best practices for mitigating malware on personal computers. While remediating malware on, say, Windows machines has become much more user-friendly and effective over the year, the situation for IoT is totally different. There are tens of thousands of different devices in the market [156, 212], all with different remediation paths. Unable to handle this enormous diversity, ISPs and governments have been resigned to providing users with very generic remediation advice that is hardly actionable. Our findings suggest a different way forward: while more and more vulnerabilities are being targeted, the total set is actually finite and somewhat manageable in size: 68 in all. ISPs could be provided with remediation advice for each of these vulnerabilities and their corresponding devices and then point their infected customers to this finite set to see if they have a device on this list.

While it will be hard for manufacturers to repair these intrinsic problems with the devices that are already deployed in the market, our findings do underline the critical importance of user-friendly update mechanisms for new devices. A related issue is the duration of manufacturer support. We observed that attackers selected vulnerabilities that are many years old, in one case even a decade old. No manufacturer is currently supporting IoT devices on those time scales. This plays to the attackers advantage. They now have very attractive proposition: spend some effort on exploit development for an easy-to-exploit vulnerability on a device with a large install base and then recoup this investment by exploiting it for many years, since patching is not happening fast enough and at some point will stop altogether, while the devices remain in the field.

2.6. RELATED WORK

IoT security research has traditionally focused on designing appropriate security controls for resource-constrained devices, yet few studies have looked into the security of already deployed IoT devices and their vulnerabilities. Feng et al. [108] studied IoT vulnerabilities using different sources in the wild such as public vulnerability and exploit databases, forums, mailing lists and blogs to propose using them for more effective defenses. Using similar data sources but leveraging machine learning, Blinowski and Piotrowski [42] proposed a vulnerability classification based on the CVE of IoT systems. Putting aside open data, Alrawi et al. [18] analyzed the vulnerabilities of a subset of home-Based IoT deployments in the first empirical evaluation of the security controls and vulnerabilities present in IoT devices already in the market.

While these previous studies focused on the defensive side of IoT security, other research has also examined attacks by analyzing IoT malware [19, 64, 65, 73, 82]. These studies either leverage honeypots to capture IoT malware (e.g., IoTPOT [197]), collect samples from VirusTotal [226], or use open threat intelligence data (e.g., CyberIOCs [80]). For instance, Khoury et al. [148] investigated which CVEs are more likely to be targeted by IoT malware based on public reports. They showed that IoT malware targets vulnerabilities that can be exploited remotely and do not require user interaction to compromise the device.

In 2021, Alrawi et al. [19] analyzed a set of 166,000 IoT malware samples collected in 2019 with the aim of understanding code reuse and evolution of different IoT malware

families. Similar to our research methodology, the authors used both static and dynamic analyses to identify similarities and differences across the different malware samples. Our work extends their initial effort in four ways that allow us to characterize different types of exploits and their evolution during a larger time period: (1) we provide a much more comprehensive understanding of the targeted vulnerabilities analyzing 68 vulnerabilities (excluding hard-coded credentials) as found in the binaries, to just 25 studied by Alrawi et al. [19], by analyzing 17,720 binaries from three types of data sources; (2) we covered binaries from a much larger time frame, 2015 to 2020, while Alrawi et al. [19] analyzed only one year of binaries from 2019; (3) we extracted exploits via a combination of static analysis, dynamic analysis, and signature matching, rather than only static analysis and YARA signatures; and (4) we longitudinally measured exploit lifetime, vulnerability lifetime, and time-to-exploit, which Alrawi et al. [19] did not measure at all. For the exploits that they discovered, they only identified a single point in time when the first industry report mentioned the exploit. This is not observed from their own data, nor is it measured longitudinally–e.g., the time frame in which binaries use the exploit. Thus, our work presents the first stepping stone to understand how attackers regularly target (very) old vulnerabilities and how they persist in certain attacks for years.

Focusing in vulnerabilities in general, several authors have analyzed the life cycle of vulnerabilities. For instance, Spanos and Angelis [228] argued that vulnerabilities' descriptions can be used to predict vulnerability characteristics. They developed a model that combined analyzing texts and multi-target classification to determine the characteristics, severity, impact and score of a vulnerability. Analogously, Wijayasekara et al. [266] developed a text mining classifier to identify hidden impact vulnerabilities, i.e., vulnerabilities identified only long after their bugs report released in public databases. They found out that hidden impact vulnerabilities has increased from 25% to 36% in Linux and from 59% to 65% in MySQL within two years.

Finally, some authors also looked at the relationship between vulnerabilities and exploits. Nayak et al. [179] found that 85% of known vulnerabilities are never exploited in the wild. They introduced different security metrics such as the count of vulnerabilities exploited and the size of the attack surface. They conducted an empirical study of security in the deployment environment using around 300 million reports of intrusion-protection telemetry that were collected from more than six million hosts. They found no single product within their study that had more than 35% of their disclosed vulnerabilities exploited in the wild. Also, newer products or newer product versions tend to decrease the exploitation ratio and the exercised attack surfaces. Householder et al. [128] focused on answering 'when and how many vulnerabilities get associated public exploits' via analyzing CVE-IDs to find out how they influence exploit publication. They found that around 4% of the published CVEs get a public exploit code associated with them within 365 days. They argued that the exploit publication likelihood increase is influenced by CVSS score, CWE, and the recent mechanism of publishing CVE-ID. They studied 75,807 vulnerabilities for which they found that only 3,164 of them had public exploits during the whole 6 years of study. Those exploits have 2 days as a median time to publication whereas the mean time is 91 days.

2.7. LIMITATIONS AND FUTURE WORK

We rely on a number of external sources and tools, and thus inherit their limitations. First, during the data collection, we found that 17 of the exploits had no CVE ID, which makes it harder to accurately measure the disclosure date and lifespan of a vulnerability. Even for published CVEs, these fields have idiosyncrasies, where publication might not happen until years after the CVE process has been started, as indicated by the year in the CVE's ID.

Second, there is some ambiguity in the malware family classification. We followed best practices by using AVClass2 to normalize labels, but we could not assign 209 of the samples to a specific family and had to treat them as singletons. For further clustering to select samples for our manual analysis, we relied on light-weight fuzzy hashing. Telfhash [163] is widely used by malware analysts, and Pagani et al. [199] showed that it is robust to minor source code changes. In our experiments, Telfhash [163] failed to cluster 48% of samples, and had issues with packed files and corrupted headers. This indicates that Telfhash potentially puts too much emphasis on features from the ELF headers, making it less suitable in our scenario. We achieved better results with the functional similarity as calculated by VirusTotal's Vhash, which unfortunately is not publicly documented. To gain insights into the quality of our classification and clustering based on AVClass2 and Vhash, we compared it against a binary similarity metric using BinDiff [272] on the packed and unpacked URLhaus dataset. The results can be found in Appendix A. Although there are differences, overall there is enough consistency among the clusters identified by BinDiff and those of Vhash (as marked by the different colors) to provide confidence that Vhash is in line with state-of-the-art tools. In general, comparing whole binaries can be very susceptible to obfuscation and it might be better to search for similar functionality instead (e.g. [173]).

Our analysis pipeline faces the same limitations as other studies based on static and dynamic analyses techniques in a malware context: malware writers tend to include anti-analysis and obfuscation techniques to evade detection. Especially code packing is an issue for static analysis, and therefore not only an issue for our clustering, but also for our static exploit signature matching.

During dynamic analysis we faced the issue of samples not executing correctly, as also reported by Alrawi et al. [19]. We investigated this issue by running the same binaries using two different sandbox configurations (vm-sandbox and docker-sandbox). We found that both the number of binaries and amount of traffic captured was 84% higher in the former. Still, improving dynamic analysis coverage is an open and orthogonal research problem.

In terms of exploit coverage, we consider privilege escalation (i.e., exploits after the initial infection) out of scope. Extending our analysis to capture these exploits as well should be straightforward, especially as we expect the Yara signatures created by Alrawi et al. [19] as part of concurrent work in this area can be transformed into our signature format with little effort (and vice versa).

Finally, in terms of malware coverage, it is hard to estimate how representative our dataset is for IoT malware. We made our best effort to collect a dataset that is as diverse as possible. We collected samples from different vantage points, both actively through a honeypot and passively from public repositories and other studies.

2.8. CONCLUSION

In this paper, we performed the first longitudinal measurement study leveraging multiple vantage point to analyze the IoT malware ecosystem and underlying dynamics. Using static and dynamic analysis, as well as signature matching, we extracted the 63 unique exploits from 17,720 binaries belonging to 26 different IoT malware families. Our results show the ecosystem has diversified, from generic brute force attacks to embody a wide variety of device-specific exploits.

Mirai is the family that has evolved the most since its inception in 2016 and it still is the leading innovator. Most exploits were observed in Mirai first. Other malware families followed the same trend increasing the complexity of IoT malware and at the same time targeting more IoT devices and different protocols. The landscape is rapidly evolving: the number of exploits and targeted vulnerabilities has doubled every year since 2017.

Once exploits are developed, they are rarely abandoned. Many still appear in the most recent binaries. In our case, the exploit lifespan is longer than 5 years, though the duration of exploits is 38 months, on average.

Attackers target (very) old vulnerabilities. The mean time-to-exploit between the publication of the vulnerability and the first occurrence of an exploit in a binary is 29 months on average, though this time frame varies wildly across exploits. This is very different from the patterns we observe for malware targeting desktop and server software. Assuming this different attacker strategy for IoT is rational, then our evidence suggests that the targeted IoT devices are rarely, if ever, patched. Thus, windows for exploiting a vulnerability do not decrease rapidly over time. The age of the vulnerability is much less relevant to attackers than the size of the install base of the device and the easy with which exploit vectors can be developed. Once they are developed, they remain in use for years.

Our study clearly shows that attackers are taking advantage of certain weaknesses in the IoT ecosystem, most notably the lack of patching and the diversity of devices and manufacturers–recently estimated to consist of over 14,000 different companies [156]. This is a target-rich environment where devices not only have different vulnerabilities, but also their own paths and dead-ends towards making them more secure against malware. We identified a number of implications of our findings for users, Internet Service Providers and manufacturers.

3

The Influencing Factors in IoT Malware Target Selection

For years, attackers have exploited vulnerabilities in Internet of Things (IoT) devices. Previous research has examined target selection in cybercrime, but there has been little investigation into the factors that influence target selection in attacks on IoT. This study aims to better understand how attackers choose their targets by analyzing the frequency of specific exploits in 11,893 IoT malware binaries that were distributed between 2018–2021. Our findings indicate that 78% of these binary files did not specifically target IoT vulnerabilities but rather scanned the Internet for devices with weak authentication. To understand the usage of exploits in the remaining 2,629 binaries, we develop a theoretical model from relevant literature to examine the impact of four latent variables, i.e. exposure, vulnerability, exploitability, and patchability. We collect indicators to measure these variables and find that they can explain to an extent why some vulnerabilities are more frequently exploited than others. The severity of vulnerabilities does not significantly increase the frequency with which they are targeted, while the presence of Proof-of-Concept exploit code does increase it. We also observe that the availability of a patch reduces the frequency of being targeted, yet that more complex patches are associated with higher frequency. In terms of exposure, more widespread device models are more likely to be targeted by exploits. We end with recommendations to disincentivize attackers from targeting vulnerabilities.

This chapter has been published as: **Arwa Abdulkarim Al Alsadi**, Kaichi Sameshima, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2023. Bin There, Target That: Analyzing the Target Selection of IoT Vulnerabilities in Malware Binaries. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). Association for Computing Machinery, New York, NY, USA, 513–526. https://doi.org/10.1145/3607199.3607241*

3.1. INTRODUCTION

The proliferation of Internet of Things (IoT) devices [46] has made them a prime target for cyber attackers looking to exploit devices and build up attack infrastructure, as analyzed in Chapter 2. The poor security features of many IoT devices are music to the attackers' ears, as they can easily compromise these devices that were never designed to repel complex attacks. In this abundance of devices, what drives an attacker's decision to target a particular device type?

Previous research on attacker target selection [130, 236] has shown that a *rational choice* approach is often used, with attackers selecting targets that are perceived to be easy to exploit, have known vulnerabilities, and offer a high reward. This has been observed in both physical burglaries and cybercrime, with offenders selecting accessible targets with known vulnerabilities. A new and more realistic attacker model [14] suggests that cyber attackers are not equally likely to exploit all possible vulnerabilities. Instead, attackers will choose to exploit only one vulnerability per software version, include only vulnerabilities with low attack complexity, and be slow at introducing new vulnerabilities into their arsenal. No research has yet tested empirically whether these factors also hold for target selection in IoT malware development. It is unclear whether attackers base their decisions on specific characteristics of the devices, their exposure, or other considerations.

In this paper, we aim to improve our understanding of how attackers choose their targets by studying the vulnerabilities and exploits used in IoT malware that was distributed between 2018–2021. Although 78% of these binaries only scanned the Internet for devices with weak authentication, the remaining 2,629 binaries were targeting specific vulnerabilities. We first develop a theoretical model based on relevant literature to understand the factors that influence the choice of exploits in IoT malware. We then relate these factors to four latent variables (exposure, vulnerability, exploitability, and patchability) and examine the impact on the usage of particular exploits, as measured by the frequency of specific exploit code in binaries observed in the wild.

By understanding these relationships, we aim to provide answers to this overarching question: *How can we explain the attackers' choices of IoT vulnerabilities as targets in malware binaries?*

To answer this question, we gather information about various aspects of IoT vulnerabilities, such as their publication date, the availability of proof-of-concept (PoC) exploit code, and the release dates of patches. Our findings indicate that certain types of vulnerabilities are more frequently exploited, yet this is *not* explained by the severity of the vulnerabilities. We also find that the availability of a patch decreases the exploitation of a particular vulnerability, while the existence of an PoC exploit increases it. Additionally, we find that more complex patches are related with higher exploit frequency. Our results also show that the more widespread a given device model is, the more likely it is to be targeted by exploits.

Our main contributions are as follows:

- We develop a theoretical model that posits that exploit usage is influenced by latent variables such as vulnerability, patchability, exploitability, and exposure, which can be inferred from observable indicators.
- We characterize the targeted IoT vulnerabilities in malware binaries by collect-

ing and analyzing data on vulnerabilities, exploits, patching and device exposure information.

- We estimate a generalized linear model to understand how different factors, such as vulnerability disclosure, patch release dates, patch complexity, and the number of exposed devices on the Internet, influence target selection.
- We provide recommendations to decrease the exploitability of IoT devices in the wild.

3.2. THEORETICAL MODEL

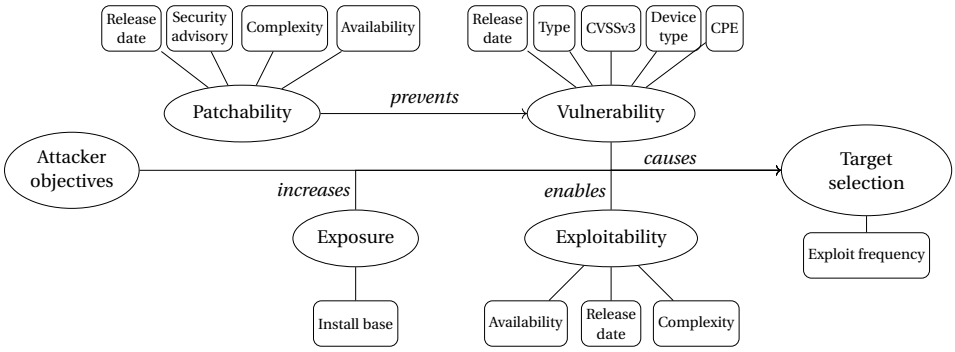


Figure 3.1: Theoretical Model for Exploit frequency in IoT Binaries.

When analyzing the exploit code malware binaries, it became apparent that some vulnerabilities are more often targeted than others. Some exploit code is only seen once in the wild, while other code has been in use for years. To help explain our main question of how attackers choose IoT vulnerabilities as targets in malware binaries, we will use the exploit frequency in binaries as our dependent variable.

While research in this area is limited, previous studies have investigated various factors and indicators that influence target selection in IoT vulnerabilities. For example, some studies have examined the relationship between attack frequency, vulnerability disclosure, and patching [27], while others studied the relationship between attacks and exposure [34]. We aim to bring together these indicators, and some new ones, in a comprehensive theoretical model.

While we can't always know the true intentions of attackers, we can build on work that assumes attackers maximize their benefit against the lowest possible cost. Allodi et al. [14] formalized this theoretical starting point in their "work-averse attacker" model. They posit that not all vulnerabilities will be attacked equally. The initial fixed costs of exploit development induce attackers to delay implementation and deployment of exploits of vulnerabilities. They found empirical validation for this model in analyzing Symantec's WINE attack signatures.

The key question then becomes: what factors impact the attackers' benefits and costs for exploiting IoT vulnerabilities? Extending the earlier theoretical work, we develop

a model that explains the frequency with which IoT vulnerabilities are targeted from four factors that impact the cost and benefits to the attacker. First, the features of the *vulnerabilities* themselves, such as their severity, may make them more attractive to attackers. Second, *patchability*, may reduce the benefits. If a patch is deployed for a vulnerable IoT device, this may reduce the available attack surface that the exploit can successfully compromise. Third, *exploitability*, considers the availability and complexity of the Proof-of-Concept exploit code, which reduces the cost to the attacker. Fourth, *exposure*, considers the size of the install base of devices with the vulnerability. A larger install base increases the benefits to attackers of targeting that vulnerability.

These factors allow us to synthesize indicators from previous studies and study their relative influence. For example, the characteristics of vulnerabilities such as disclosure date, severity and type can be used as indicators to predict whether or how soon vulnerabilities are likely to be selected as a target by attackers [175] and how frequent [27]. Similarly, the availability of Proof-of-Concept (PoC) exploit code is correlated to the likelihood that attackers select the vulnerability for their attacks [136, 175], while the availability and the time of the release of patches [27] as well as the effort needed to deploy them [47] can all impact the attack surface [27].

Moreover, when exposed devices can be discovered via search engines, it increases the likelihood of these devices being targeted for vulnerabilities. [26, 34, 158]. In addition to the indicators from earlier work, we also collect new ones, most notably patch complexity and exploit complexity.

In sum, we developed a theoretical model, shown in Figure 3.1, that interprets these indicators as signals for the four underlying factors. This allows us to theoretically interpret the relationships. To the best of our knowledge, there is not yet a study that systematically measures the impact of these factors – and all of the associated indicators – on the target selection of vulnerabilities in IoT malware binaries. We now briefly explore each of the four factors in a bit more detail. (In Section 3.3 we will discuss the data sources we have used to measure the indicators for each factor, as summarized in Table 3.1.)

Vulnerability. Vulnerability refers to a weakness or flaw to exist in an IoT device that can be exploited by an attacker to gain unauthorized access or cause harm. As each vulnerability has its own characteristics, we want to understand which ones have the most impact on the attackers' choice to select as a target. We used five indicators of vulnerability to measure the impact on IoT exploit frequency in binaries. One of these indicators is the *vulnerability release date* in public, which can provide important information about the number of exploit frequency. In another words, the older a vulnerability is, the higher the potential number of frequency is as it can accumulate over time. We will be using it to measure the lifetime a vulnerability by counting the days since the vulnerability release date until last day of binaries collection.

For vulnerabilities with CVE-ID, we collected the *severity* as measured by CVSSv3 metric. The CVSSv3 scoring system is used to assign a numerical score to a vulnerability based on its characteristics and the potential impact if exploited. We only collected CVSSv3 since it is the most recent revision of CVSS scoring system for vulnerabilities published 2016 onwards, which corrected the shortcomings in older versions. The CVSSv3 score ranges from 0 to 10, with higher scores indicating a more severe vulnerability. Additionally, we also counted the number of affected systems using their corresponding

Common Platform Enumeration (CPE) for each vulnerability. CPE is a naming scheme that structures the information about affected systems, software, and packages. Thus, it can help to identify which systems are affected by a vulnerability and which are not.

Next, we collected data on *device type* that is affected by the vulnerability, as some types of devices are more often targeted than others. We categorized the devices according to their use. Finally, we examined the *vulnerability type* to determine if some vulnerability types such as: Remote Code Execution, Command Injection and Password Overflow are more targeted than others.

Patchability. Patchability refers to the ease or difficulty of patching (fixing) a vulnerability in an IoT device in order to reduce the likelihood of a successful attack. Our hypothesis is that the patching of IoT devices can affect the frequency of exploits.

The availability of patches will lower the chances an attack succeeds, therefore, patchable devices will be targeted less in IoT binaries.

To test this, we use a range of four indicators to measure relevant patching properties. One such indicator is the *patch availability*. We assess this by determining whether patches are available on the vendor's website or on security blogs, which leads to our second indicator, the existence of a vendor's *security advisory*. We hypothesize that the existence of security advisory can lead to more patched devices and less vulnerable devices. This is because if manufacturers or vendors do not have effective public source dedicated to issue security advisories in place, users may be less likely to apply patches to their devices. Then, if patches are available, we evaluate their *complexity*, as this could be a factor in users not updating their devices. We do this by counting the number of steps required to apply the patch listed in the security advisory, as we believe that the more steps involved, the less likely a user is successful or even willing to install the update. Finally, is the *patch release date* provided by vendors, which can serve as a proxy for the amount number of exploit number to measure whether the release time of a patch has an impact on the number of exploits showing in binaries. We use the publication date to measure the patch lifetime by counting the days between the patch publish date and last day of binaries collection.

Exploitability. Assuming that attackers minimize effort, available and complexity of PoC exploit code would increase the odds of exploit code showing up in binaries. We use three simple indicators to measure the exploitability of a particular vulnerability. First, *exploit availability*, where we determine whether the exploit we found in binaries has PoC exploit available in public repositories. If so, we collect information of our second indicator, *exploit complexity*. We crudely approximate this by counting the number of lines of the PoC exploit code, assuming that an exploit with more code is most costly to use successfully by attackers. Lastly, *exploit publish date*, which refers to the publish date where the PoC exploit code was released. We also use the publication date to measure the exploit lifetime by counting the days between the exploit publish date and last day of binaries collection.

Exposure. Exposure refers to the size of the discoverable attack surface that an attacker might target with an exploit. We hypothesize that the larger the *install base* is of devices exposed to Internet-wide scans or public search engines (e.g., Shodan [160]), the higher the chances an attacker would target them.

Table 3.1: Description of variables.

Variable	Description	Data sources
Installed base	Number of IoT devices exposed in public	Shodan
Vulnerability type (CI)	A dummy variable that denotes if the vulnerability type is command injection	NVD
Vulnerability type (RCE)	A dummy variable that denotes if the vulnerability type is remote code execution	NVD
Vulnerability type (other)	A dummy variable that denotes for other the vulnerability type	NVD
Device type (router)	A dummy variable that denotes if the device type is router	NVD
Device type (surveillance)	A dummy variable that denotes if the device type is surveillance	NVD
Device type (Other)	A dummy variable that denotes for other device type	NVD
Severity (CVSSv3)	A value between 0 and 10 that measures the impact of exploiting the vulnerability	NVD
CPE	Number of affected systems, software and packages	NVD
Vulnerability lifetime	Number of elapsed days from the vulnerability published date until the last capture day of binaries	NVD
Patching lifetime	Number of elapsed days from the patching release date until the last capture day of binaries	Security advisory
Patch complexity	Number of steps a user needs to patch a device	Security advisory
Patch availability	A dummy variable that denotes whether a patch is available or not	NVD, Security advisory
Security advisory	A dummy variable that denotes whether a vendor has released a security note for a vulnerability or not	Vendors' website
Exploit availability	A dummy variable that denotes whether an exploit code is available or not	Exploit-db, Github, Security reports
Exploit complexity	Number of lines for exploit code	Exploit-db, Github, Security reports
Exploit lifetime	Number of elapsed days from the PoC exploit code publish date until the last capture day of binaries	Exploit-db, Github, Security reports

3.3. METHODOLOGY

Our methodology involves two main steps in order to explain the target selection of IoT vulnerabilities using exploit frequency in binaries as our dependant variable. First, we gather the indicators, including information on the exploit code in the malware binaries, the vulnerabilities they exploit, the patch status of the affected devices, and the level of exposure of the devices (see Table 3.2 for data collection work process). Second, we use a count regression model to analyze and quantify the significance of each indicator. The output of the model provides insights into the factors that influence the decisions of attackers in selecting what exploit code to include in IoT binaries, allowing us to better understand the threat landscape and make recommendations for improving the security of IoT systems.

3.3.1. DATA COLLECTION

In this section, we will discuss the data collection work process we used to measure the indicators for each factor, as summarized in Table 3.2.

IOT EXPLOITS IN BINARIES.

In our study, we collected IoT malware samples from an IoT-specific honeypot over a period of four years (2018 – 2021). We used dynamic analysis to extract exploits from these samples and map them to vulnerabilities.

Collecting IoT malware binaries. We obtained 11,893 binaries from September 2018 to September 2021 via IoTPOT [196].

The present version of IoTPOT is a high-interaction honeypot using bare-metal vul-

Table 3.2: Overview of data collection work process.

Automated	Semi-automated	Manual
Binary collection	Vulnerability release date	Exploit signature generation
Exploit extraction	Patch availability	Mapping exploit to vulnerability
Severity (CVSSv3)	Security advisory	Installed base
CPE	Exploit availability	Device type
	Exploit release date	Vulnerability type
		Patch release date
		Patch complexity
		Exploit complexity

nerable IoT devices running Telnet and/or HTTP as Web User Interface. For the entire observation period, the honeypot utilized eight devices (three routers, two WiFi storage devices, an IP camera, a printer, and a satellite decoder) connected to 64 IP addresses in Japan. Additionally, from September 2018 to February 2020, the honeypot utilized seven more devices (five IP cameras, a router, and a WiFi storage device) connected to 32 IP addresses in Japan. Although the devices connected to our high-interaction honeypot may not represent the overall population of IoT devices on the internet, the honeypot can still serve our research objectives and generate insights into the population of IoT exploits used in the wild. Therefore, we believe that the set of binaries collected over a period of fours years captures the essence of exploits targeting vulnerable IoT devices, not only in Japan but globally.

Extracting exploits. We perform dynamic analysis on all the 11,893 collected honeypot binaries. In a sandbox, we executed each of the binaries, which is built as a virtual machine running Linux Debian for the MIPS and ARM architectures. With the exception of DNS resolutions, we run each binary for five minutes in a closed network environment. As malware repeatedly tries to connect to the C&C server in such an environment whereas port scans on the same host are typically not repeated, this isolated environment help us in differentiating between port scans and C&C communications, especially when they are on the same port. In addition, if the malware on a destination port accesses more than 100 destination IP addresses, we conclude that the port is being scanned. Then, we execute all binaries again for five minutes with dummy servers that pretend to be the target on the scanned ports once the sandbox has been cleaned up. With the implementation of PyNetSim, we redirect all connection attempts on the examined port to the dummy servers, which merely establishes TCP sessions with no further response. We found that a significant number of binaries actually begin scanning for and exploiting vulnerabilities immediately after execution, without contacting C&C servers or their intended targets.

During the dynamic analysis of the 11,893 binaries, 2,256 of them did not show any scanning (propagating) behavior. There can be different reasons why they did not show propagation: they simply did not have any propagating capabilities; they might have needed a trigger, such as a command from C&C server; or they might not have been executed successfully. As discussed in Chapter 2, similar patterns have been noted in earlier research [19].

Given that we could not detect exploits, we dropped these 2,256 binaries from further

analysis. In the end, we had dynamic analysis results for 9,637 samples with propagating behavior. Of this set, 7,008 binaries conducted scans on only Telnet ports, such as 23, 2323, and 2223/TCP, to try brute-forcing with known credentials. Since this general attack vector is not tied to a specific CVE or known vulnerability, we did not include these binaries in the model. For the remaining 2,629 binaries, we found evidence of targeting specific CVE or known vulnerability via 48 different ports and various protocols. All of the detected protocols, such as CPE WAN Management Protocol (CWMP) and Simple Object Access Protocol (SOAP), are HTTP-based in terms of the transport layer, which is consistent with the findings in [24, 142, 202].

Exploit signature generation. For each payload captured from the dynamic analysis, we manually extract HTTP-based request lines and HTTP headers to create a signature of a distinctive sub-string, such as a destination resource of HTTP requests, which can be used to identify an exploit. We use the URI (Uniform Resource Identifier) syntax or absolute path of the `GET` or `POST` in the request line to generate the unique signature. In Figure 3.2, we present an example of an exploit payload. As can be seen in this figure, the URI syntax consists of four components (colored in red): scheme, authority, path, query and fragment. The scheme part is followed by a colon and two forward slashes, if specified. Then comes the path name that begins with a single forward slash. Next, the query string preceded by a question mark. Finally, the fragment, which is a set of characters for resource that is subordinate to another.

However, the most common format is: the path name, question mark and the query string. We use this URI syntax during the extraction of each request. In this example, the final exploit signature is `/backupmgt/localJob.php?session=fail`.

```
GET /backupmgt/localJob.php?session=fail;cd+/tmp;wget
+http://212.192.241.72/lolol.sh;curl+O+http://212.192.241.72/lolol.sh; sh+lolol.sh
HTTP/1.1
Connection: close
Accept-Encoding: gzip, deflate
Accept: /
User-Agent: Dark
```

Figure 3.2: Example exploit payload

Using this signature generation method, we created 59 unique signatures from the captured HTTP requests. To identify the vulnerabilities that each one of these exploit targeted, we search them in public exploit databases such as Exploit-db [105] or Github[262]. From these two sources, we were able to identify 45 out of the 59 signatures. The remaining 14 signatures with no information in the above sources, we use other public sources such as reports from AV vendors and researchers that describe them. For every signatures that were confirmed to be used as an exploit code to target vulnerability, we tag them as *Known exploit*. Otherwise, we would tag them as *Unknown payloads*. However, all the signatures that we generated were found in the public sources meaning that they were all tagged as known exploit.

Apart from using signatures to identify exploits, we also used them to allow us to count how often each exploit appears across the dataset. We call this 'exploit frequency', our

dependent variable. It is meant to capture the more prevalent exploits among attackers, hence the vulnerabilities they prefer to target.

CHARACTERIZING IOT EXPLOITS

After we confirmed that these signatures belonged to known exploits according to the above public sources, we collect information about their corresponding vulnerabilities and their proof of concept (PoC) exploit codes. Next, we characterized these vulnerabilities using the National Vulnerability Database (NVD [177]). We then collected data on the patch information using the vendor's security advisories.

Mapping exploit to vulnerabilities. We manually mapped exploits to their corresponding vulnerabilities using the signatures to search in Exploit-db, Github or Google Search. For example, for each exploit entry in Exploit-db, there is a CVE field we can retrieve the vulnerability information, if available. Otherwise, we used the vulnerability description title in Exploit-db, e.g., "OptiLink ONT1GEW GPON version 2.1.11_X101 RCE" [219]. We applied the same approach for the other two sources. We found that the 59 unique exploits were mapped to 64 vulnerabilities. Next, we checked whether a proof of concept (PoC) exploit code was available for them. If so, we retrieved the date when the PoC exploit code was released and measure its complexity through counting the lines of code.

Exploit coverage. At the heart of our analysis is frequency data: how often is a vulnerability attacked in the wild? For this, we needed to capture binaries in live attacks on our honeypot infrastructure. The number of times that an exploit was present in these captured binaries tells us how often a vulnerability was targeted. This means we can't supplement the data with binaries from an existing repositories of binaries, since those data sources lack the frequency with which the binaries were observed in live attacks. The honeypot infrastructure was running in Japan. This location might impact the observations. That said, we did not observe any binaries targeting specific network ranges. Rather, scanning was traversing randomly through the IPv4 space. To assess the coverage of our dataset with previous work, we compared the overlap of exploits with datasets used in Chapter 2 and [19]. As shown in Chapter 2, the exploits were extracted using both static and dynamic analysis. Therefore, it does not have the limitations of relying exclusively on dynamic analysis to identify exploit code, so it provides a good point of comparison. We observe that our set of exploits is consistent with their datasets, namely UrlHaus (Jul 2020 - Oct 2020), IoTPoT (Sep 2018 - Aug 2020), and the Genealogy (Jan 2015 - Aug 2018). Our results showed that we covered 55.17% (32 out of 58) of the exploits in UrlHaus, all exploits in IoTPoT, and 80% (12 out of 15) in the Genealogy datasets. To compare against Alrawi et al. [19], we looked at the top 25 IoT exploits reported in their study. Out of the 25 exploits, 17 list a CVE-ID. Of these 17, 14 are also in our set, 3 are not. The remaining 8 exploits have no CVE-ID or affected product listed, so we cannot compare them to our dataset. So although we can only conduct a limited comparison, for the vulnerabilities with a CVE-ID, we have a high overlap with their dataset.

Characterizing IoT vulnerabilities. To collect information on the 64 vulnerabilities targeted by the captured exploits, we used the National Vulnerability Database (NVD). It publishes Common Vulnerabilities and Exposures (CVEs), their associated identification number, description, public references, and severity score. In case, a vulnerability is

reserved in the NVD, e.g., due to pending CVE status, we used the MITRE website [239] to collect the information. However, 15 of the 64 vulnerabilities were not registered in NVD, so they had no CVE-ID, consistent with the findings in Chapter 2 and earlier research [19]. For those, we used the vulnerability description title in the exploit databases.

Identifying patches. For the identified IoT vulnerabilities, we collect patch information that has been made available by the device manufacturer through security advisories. Security advisories are documents that provide information about vulnerabilities in products, including Internet of Things (IoT) devices. They are typically issued by the manufacturer or vendor of the device in question, or by independent organizations or government agencies that have discovered a vulnerability. These advisories normally include brief headers that list the vulnerability title, advisory release date, affected products, advisory ID, and any relevant CVE ID. We found them on the website of the manufacturer or vendor, or on the website of the organization or agency issuing the advisory. In this study, we used the security advisory pages of vendors. Specifically, we gathered patch information for each vulnerability, including existence of advisory notes by vendors, patch release dates, availability and patch complexity.

MEASURING DEVICE EXPOSURE

Vulnerabilities are associated with certain device models. To measure the exposure of these targeted IoT device models, we use search engines to identify the number of Internet-connected devices and findable devices of each model. This can help us understand the potential scale of exploiting a certain vulnerability. By searching for the specific model numbers of vulnerable devices, we can approximate the size of the install base for those devices. Shodan is unlikely to provide us with a very accurate absolute number of publicly reachable devices of a certain type, but what we need is a proxy of the install base of each device relative to the other targeted devices.

Estimating device installed base. We use Shodan [160] to determine the quantity of Internet-connected devices for a particular vulnerable model. Shodan allows us to search for specific types of devices, such as servers, routers, and other Internet of Things (IoT) devices, and retrieve information about them. By querying Shodan, we gather information about devices that are publicly available on the Internet, including their metadata and any web interfaces they may have.

To measure the install base of vulnerable IoT devices, we developed a systematic querying approach (see Figure 3.3) that involves using the vendor or device name as a starting point, then manually checking the search results and inspecting the HTML field to confirm if the search results match the actual targeted device. We also looked for other signatures that contain the model name or the affected software version a given device is running. We then filtered the results based on the device's installed software version, if available.

For example, some banners include a server name that shows the device-specific, and then we search again in Shodan using these signatures. Through experimentation and testing, we found that approximately 57% of the devices can be found using the device name approach, while the remaining cases can be found using the server name approach.

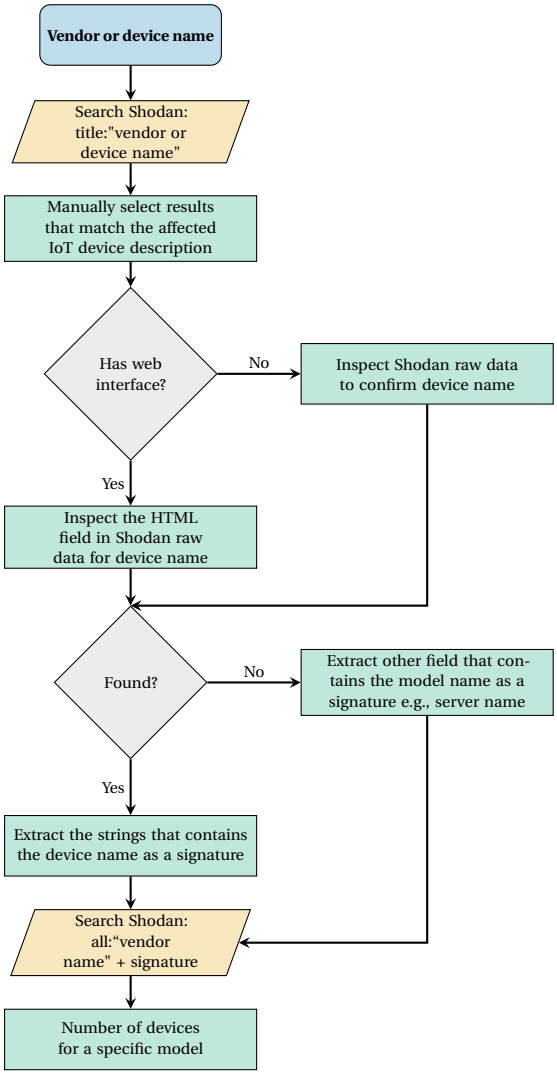


Figure 3.3: Flowchart to determine the number of publicly accessible IoT devices using Shodan.

3.3.2. DATA PROCESSING

To prepare for analysis and modelling, we clean up the data, transforming it in a way to make it suitable for the analysis. We used the CVE-ID or the vulnerability name as the unit of analysis to count the frequency for each exploit. The relationship between exploits and vulnerabilities is often one-to-one, but not always. It was one-to-one for 36 out of 49 observations. Where it is not one-to-one, this creates an issue of creating duplicate data in the observations for the model.

We encountered three scenarios that were not one-to-one. First, sometimes we encountered multiple exploits targeting the same vulnerability. If we would count each

of these combinations as a separate observation, we would get observations where the vulnerability properties are copied, hence feeding the model with artificially duplicate data while the model assumes these are all independent datapoints. This is obviously not allowed. Therefore, for this first scenario, we merge the frequency counts for each exploit and then connect those with the one vulnerability. We investigated this further and found that in all these cases, the different exploit codes were coming from the same PoC. So they were already linked, if not basically the same exploit to begin with. So merging their counts makes sense. This scenario holds for six of our observations.

The second scenario is where we have one exploit targeting multiple vulnerabilities. Here we merge the vulnerabilities into one observation. Since they have different publish dates, we picked the oldest date amongst them to cover the whole life cycle of such case. This scenario holds for four of our 49 observations. Since multiple vulnerabilities also have different CVSS3 base scores, we use the highest severity score.

The third scenario is where we found multiple exploits targeting multiple vulnerabilities. This case is basically a mix of previous two scenarios: some exploits stem from the same PoC and that PoC actually targets multiple vulnerabilities. In those cases, we merged both the exploit frequency counts as well as the vulnerabilities. Again, we picked the oldest vulnerability publish date and the highest CVSS3 score for the observation. This scenario holds for three of our 49 observations.

A final bit of data processing concerns the size of the install base. Since the size of the observed install bases spans multiple orders of magnitude, we log-transformed the indicator to approximate a normal distribution that the model can work with.

3.3.3. MODELING APPROACH

We used a general linear model (GLM) as it is a commonly used statistical tool for studying the relationship between a dependent outcome variable and one or more independent explanatory variables. In the context of this study, where the dependent variable is a count of the number of times a certain exploit code was observed in a binary, using a GLM allows us to investigate the influence of multiple factors on this outcome. The dependent variable (exploit frequency) will be a non-negative integer. Previous research suggests that the Poisson regression model is a suitable option for modeling this type of data (i.e., discrete, non-negative, and sporadic) [251]. However, a major limitation of the Poisson regression model is that the variance of the dependent variable must be equal to its mean. This is not our case as the variance is more than hundred times larger than the mean. When the mean and variance of the data are not equal, the estimated Poisson model coefficients tend to be underestimated and biased. This limitation can be addressed by using the negative binomial distribution, which is well-suited for describing discrete, non-negative events, and does not have the requirement that the mean must be equal to the variance. We estimate the negative binomial dispersion parameter using maximum likelihood, which is the technique that is most often used, and usually provides a slightly better estimate and smaller standard deviation than other commonly used estimators. The density function of the negative binomial model is given by:

$$f(y|x,\beta) = \frac{\Gamma(y+r)}{\Gamma(y+1)\Gamma(r)} \left(\frac{1}{1+\frac{1}{r}\cdot x^{\top}\beta}\right)^{y} \left(\frac{1}{1+\frac{1}{r}\cdot x^{\top}\beta}\right)^{r}$$

whereby in our case, y is the count of exploit frequency, x is the vector of explanatory variables, β is the vector of model coefficients, and r is a dispersion parameter for the

negative binomial family. The function Γ is the gamma function, which is defined as $\Gamma(n) = (n-1)!$ for positive integers n and has a more general definition for other values of n.

The j_{th} explanatory variable can be tested for its significance in explaining the frequency of an exploit by examining the respective coefficient β_j. However, it is necessary to check if the variable is already represented by other factors, as redundant variables can result from high collinearity with another variable. The variance inflation factor (VIF) can be used to quantify this redundancy and is defined as:

$$VIF_j = \frac{1}{1 - R_j^2}$$

where R_j^2 is the R^2 in a model containing x_j as the explanatory variable only. O'Brien proposed [191] to indicate multicollinearity $VIF(\beta_j) > 5$ or $VIF(\beta_j) > 10$. In this paper, we chose a threshold of 5 as VIF values exceeding this level indicate the presence of multicollinearity issues. [116, 143, 162]. There are some approaches for designing a measurement for the fitness of the GLMs. The R^2 can only be computed in a linear regression model, so most coefficients of determination work with the likelihood function. McFadden's R^2 is chosen for the present analysis and defined by:

$$R^2 = 1 - \frac{LL_{full}}{LL_{null}}$$

where LL_{full} is the log-likelihood of the full model and LL_{null} is the log-likelihood of the null model (a model with no explanatory variables).

3.4. DESCRIPTIVE FINDINGS

We present the factors and their indicators, which we hypothesized will explain the vulnerability target selection in IoT malware binaries. Table 3.3 presents a summary of descriptive statistics for the indicators.

3.4.1. DISTRIBUTION OF EXPLOIT FREQUENCY IN BINARIES

The exploit frequency distribution, as shown in Figure 3.4, ranges from a minimum of 1 to a maximum of 2,177 times the same exploit was seen in the 2,629 binaries. This results in a highly skewed distribution. The average frequency is approximately 169, while the median is 23. We encountered 12 exploits (24.5%) appearing only once in the binaries. All but one of these 12 exploits targeted a unique vulnerability that was not targeted ever again in another binary.

Figure 3.5 presents the time series of the exploit frequency of the top five most frequently exploited vulnerabilities. The vulnerability with the highest frequency (2,177 occurrences) was CVE-2017-17215 [186]. Despite the release of a patch around the time of the vulnerability's discovery, it continued to be targeted in binaries. The second highest was CVE-2014-8361 [185], which was targeted by three different exploits that all came from the same PoC exploit code. This exploit was released eight months after the publication of the vulnerability in October 2014. The devices affected by CVE-2014-8361 have reached their End-of-Life, meaning that the vendor will no longer provide software upgrades or address any new vulnerabilities discovered in them.

Table 3.3: Descriptive statistics per indicator.

Indicator	Stats / Values	Freqs (%)
Installed_Base [numeric]	Mean (sd) : 37098.1 (217960.2) min < med < max: 2 < 485 < 1527377 IQR (CV) : 2819 (5.9)	≤ 35: 13 (26.5%) ≤ 572: 26 (51.0%) ≤5108: 37 (75.5%)
Vulnerability_Published [logical]	1. FALSE 2. TRUE	15 (30.6%) 34 (69.4%)
Vulnerability_type [factor]	1. A Password Overflow Issue 2. Backdoor 3. Command Injection 4. RCE	1 (2.0%) 2 (4.1%) 19 (38.8%) 27 (55.1%)
Device_Type [factor]	1. NAS 2. Router 3. Surveillance 4. TV 5. Web app 6. Web server	3 (6.1%) 28 (57.1%) 7 (14.3%) 1 (2.0%) 7 (14.3%) 3 (6.1%)
Severity_CVSS3 [numeric]	Mean (sd) : 6.1 (4.5) min < med < max: 0 < 8.8 < 10 IQR (CV) : 9.8 (0.7)	0.0 : 17 (34.7%) 7.2 : 2 (4.1%) 7.5 : 1 (2.0%) 8.8 : 7 (14.3%) 9.8 : 21 (42.9%) 10.0 : 1 (2.0%)
CPE [numeric]	Mean (sd) : 1.9 (4.4) min < med < max: 0 < 1 < 27 IQR (CV) : 1 (2.3)	0 : 16 (32.7%) 1 : 21 (42.9%) 2 : 5 (10.2%) 3 : 3 (6.1%) 5 : 1 (2.0%) 6 : 1 (2.0%) 16 : 1 (2.0%) 27 : 1 (2.0%)
Vulnerability Lifetime [numeric]	Mean (sd) : 1373 (971.6) min < med < max: 49 < 1222 < 4614 IQR (CV) : 1217 (0.7)	≤ 660: 13 (26.5%) ≤1222: 26 (51.0%) ≤1877: 37 (75.5%)
Patch_Availability [logical]	1. FALSE 2. TRUE	25 (51.0%) 24 (49.0%)
Security_Advisory [logical]	1. FALSE 2. TRUE	24 (49.0%) 25 (51.0%)
Patch_Complexity [numeric]	Mean (sd) : 2.1 (3.7) min < med < max: 0 < 1 < 13 IQR (CV) : 3 (1.8)	≤ 1: 35 (71.4%) ≤ 4: 41 (83.7%) ≤ 8: 48 (98.0%)
Patch_Lifetime [numeric]	Mean (sd) : 1010.9 (983.3) min < med < max: 17 < 907 < 3045 IQR (CV) : 753 (0.7)	≤ 66: 3 (12.0%) ≤554: 6 (24.0%) ≤907: 13 (52.0%)
Exploit_Availability [logical]	1. FALSE 2. TRUE	1 (2.0%) 48 (97.9%)
Exploit_Complexity [numeric]	Mean (sd) : 76.4 (67.7) min < med < max: 0 < 69 < 317 IQR (CV) : 81 (0.9)	≤ 21: 13 (26.5%) ≤ 69: 26 (51.0%) ≤101: 37 (75.5%)
Exploit_Lifetime [numeric]	Mean (sd) : 1308 (983.3) min < med < max: -27 < 1165 < 4412 IQR (CV) : 1161 (0.7)	≤ 556: 13 (26.5%) ≤1165: 26 (51.0%) ≤1717: 37 (75.5%)

The next vulnerability in terms of exploit frequency is MV Power Shell CI, which was published in August 2015. Unlike the previous two, it did not have a patch available and its PoC exploit was not released until a year and a half after its publication. It took another 46 months for the first exploit to show up in a binary. The fourth most exploited vulnerability

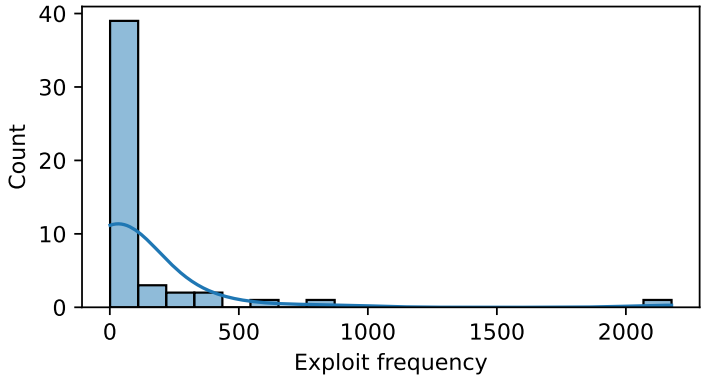


Figure 3.4: Distribution of exploits frequency.

in binaries was CVE-2017-18368. Its PoC exploit was made public six months before the vulnerability was disclosed, but a patch is still not available to the public. The final case in the top five is the combined CVE-2018-20062 and CVE-2019-90082, which were exploited in binaries using the same exploit a few times after their first disclosure in November 2018, but stopped being seen in binaries until after the PoC exploit was released in April 2020. From then on, these vulnerabilities were exploited in binaries for several months. To contrast, in some other cases (such as CVE-2013-7471 and Xiaogmai Backdoor) PoC exploits had been available for years, but attacks only started to appear more frequently in binaries after the vulnerabilities were publicly disclosed.

3.4.2. DISTRIBUTION OF VULNERABILITY FEATURES

We found all targeted vulnerabilities were published in 2013 onwards except for one that was published in 2009 (CVE-2009-0545). However, the 2009 vulnerability was found to share the same PoC exploit with a vulnerability that was published 10 years later, CVE-2019-12725. At the other extreme, the most recent vulnerability, CVE-2021-38647, was published in August 2021 and was seen in binaries a month after that. Most targeted vulnerabilities were published in 2018 and 2021.

Looking at the vulnerability type in Table 3.3, Remote Code Execution (RCE) represents the majority (55%) in the IoT binaries, followed by Command Injection around (39%). The other vulnerability types, Backdoor and Password Overflow, both represent 4% and 2%, respectively. That being said, CVE-2017-17215, which is the vulnerability with the highest exploit frequency of 2,177 times and was seen continuously in binaries used command injection for exploitation.

In terms of the six device types we encountered, routers are the primary type attackers target, which covers 28 out of the 49 observations. Surveillance and Web application were each seen in seven cases, NAS and Web servers supporting IoT devices in only three. The one remaining device type is TV, which was only seen once. This result ties well with the findings in Chapter 2, and with previous studies showing that TVs are among the least targeted device types [19].

Also, Table 3.3 shows that the maximum CVSSv3 score is 10 out of 10 (critical), which

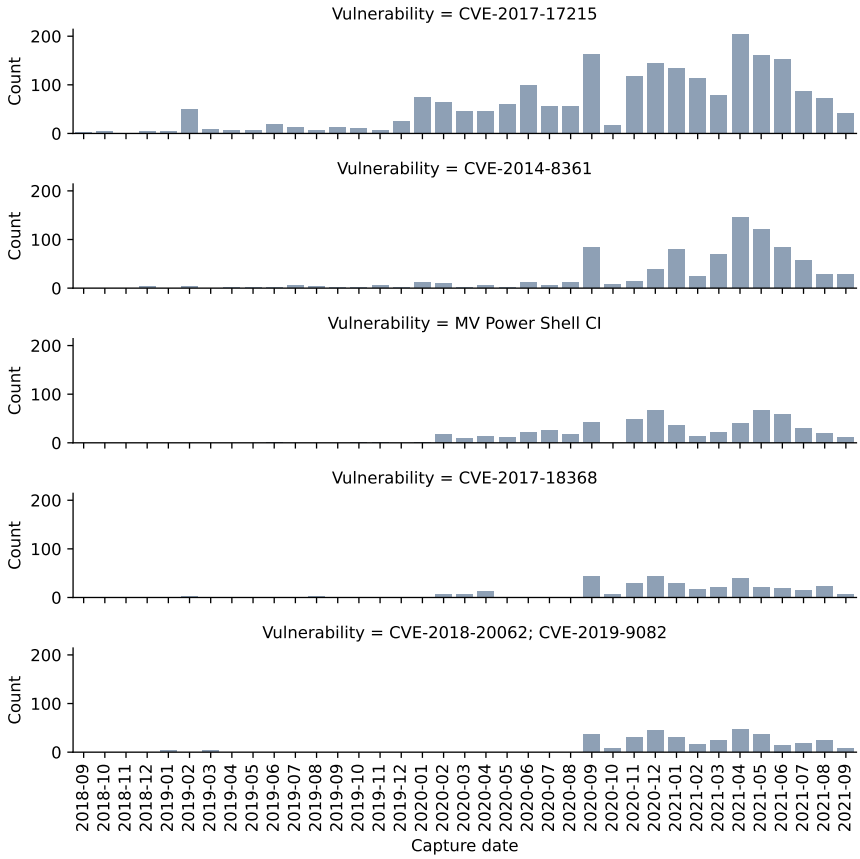


Figure 3.5: Top five targeted vulnerabilities based on their monthly exploit frequency in binaries.

was seen in one case: CVE-2019-7256. Remarkably, this vulnerability was exploited only once in binaries. Overall, the average score base is 6.1 ("Medium") and the median is 8.8 ("High").

Next indicator is CPE. The average affected versions is 1.9. This means that 85.7% of the total cases had two or less versions affected for a given vulnerability. On the other hand, the maximum CPE number is 27. These 27 affected systems belonged to one vulnerability (CVE-2020-9054), of which the corresponding exploit seen only once in our malware binaries.

Finally, the vulnerability lifetime. The shortest exploit window we observed since the publication of vulnerability was 49 days for CVE-2021-38647 that was published in August 2021, less than two months before the end of our binary collection. On the contrary, the largest is 4,614 days for a vulnerability that was published in 2009 (CVE-2009-0545). However, the average exploit window for is 1,373 days.

3.4.3. DISTRIBUTION OF PATCHING FEATURES

Although patching is critical to mitigate any security flaw in a vulnerable device, Table 3.3 shows that only 49% of vendors have actually released patches. That means almost half of the IoT vulnerable devices we observed remained vulnerable, unless users found another way to secure them in the absence of a patch.

The percentage is a bit higher (51%) when we measure if a vendor has issued a security advisory. In one case a vendor did provide a security advisory, but the patch was not available.

Another indicator is patch complexity. When patching only needs from a user to upgrade a firmware, we count this as one step unless it states otherwise in the security notes. For example, the maximum number of steps to patch was seen in D-Link DSL-2750B device, the user needs 13 steps, whereas for CVE-2016-6277, users needs to perform six steps according to the instructions on their security advisory. In average, a user needs around 2 steps to patch a vulnerable device, and one step in median.

In regards to the patch lifetime, the patches for the targeted vulnerabilities were available and released for around 1,010 days, on average. The vulnerability with the maximum patch lifetime of 3,045 was Netgear DGN1000 RCE, and CVE-2021-38647 had the minimum window of 17 days for patching.

3.4.4. DISTRIBUTION OF EXPLOIT FEATURES

We assume the availability of public PoC exploit can influence the exploit frequency in binaries. Table 3.3 shows that all exploits have PoC exploit available in public except for one exploit. It was for a vulnerability we mentioned earlier that was reserved by NVD (CVE-2019-7405), and it was only seen twice in our binaries. So while PoC was available almost always, and this might influence whether a vulnerability is targeted at all, it cannot help us explain how often a vulnerability is targeted.

We approximate complexity via the number of lines of PoC exploit. The average PoC exploit is around 76 lines, with the minimum being one line and the maximum 317 lines. The minimum was seen in exploitation of Vacron NVR RCE vulnerability. Only fewer cases were seen when the lines of PoC code getting higher than the average. In addition, the vulnerability with the highest frequency of being targeted had an exploit complexity of less than the average (25).

Lastly, the exploit lifetime. The maximum time a PoC exploit was available is 4,412 that was targeting CVE-2009-0545; CVE-2019-12725, yet only seen once in binaries. On the other hands, the most recent PoC exploit that was targeting the most recent vulnerability (CVE-2021-38647) for 20 times, it was available 27 days *after* the last day of binary collection. On average, PoC exploits were available for 1,308 days.

3.4.5. DISTRIBUTION OF EXPOSURE FEATURES

The installed base for the targeted devices of the 49 observations can bee seen in Table 3.3. The mean and median install base for the targeted devices are 37,098 and 485, respectively. However, the standard deviation is 217,960. This explains how spread out the data are from the mean. It can be explained when looking at the maximum number of installed base of 1,527,377 (outlier) while the minimum number of exposed devices in public is two. It is worth discussing these interesting findings of the highest install base device. It was

for Huawei HG532 router that is affected by CVE-2017-17215, which is the vulnerability with the highest exploit frequency and were targeted continuously in binaries. Yet, the least number of exposed devices was ZTE ZXV10 H108L gateway, which was seen targeted five times in binaries. An interesting finding in the understanding of the install base in regards to device types, we found that the largest top four exposed devices were routers. On the other hand, the smallest top four install base were all web application except for one, which was a router.

3.5. QUANTIFYING THE FACTORS THAT DRIVE IOT VULNERABILITY EXPLOITATION

To quantify to what extent some factors explain why some IoT vulnerabilities are more frequently targeted than others, we create a statistical model based on our theoretical model. By doing so, we can develop a more comprehensive understanding of the factors that drive IoT vulnerability exploitation.

3.5.1. INDICATORS SELECTION

The first step is to select indicators for inclusion in the model. This selection process is based on the principle of non-redundancy, whereby highly correlated indicators are removed from the model. This reduced set of indicators is then used as explanatory factors in the GLM, and the significance of each indicator is assessed in order to further refine the model. The process of checking for redundancy is performed using the variance inflation factor (VIF), and indicators with VIF values greater than 5 are iteratively omitted. This leaves us with ten indicators. Based on the the guideline from O'Brien et al. [191] we exclude any of the indicators whose generalized variance inflation factor (GVIF) exceeds five (see Table 3.4).

Table 3.4: Variance inflation factors

Variable	GVIF	Df	GVIF^(1/(2*Df))
Installed_Base	1.19	1	1.09
Vulnerability_Lifetime	1.15	1	1.07
Severity_CVSSv3	1.63	1	1.28
Vulnerability_type	1.53	2	1.11
Device_Type	1.38	2	1.08
Patch_Complexity	1.56	1	1.25
Patch_Availability	3.46	1	1.86
Security_Advisory	3.01	1	1.73
Exploit_Complexity	1.43	1	1.20
Exploit_Availability	1.32	1	1.15

3.5.2. MODEL FIT

We explore the model fit by adding variables to the model using a stepwise forward procedure. The models are fitted using the log link function, and the Type III test is used

to evaluate the significance of the variables [234]. This is because the results of the test indicate the significance of the complete variable, which may not be the same as the significance of the individual estimates of the different categories for categorical variables. Even if the estimates of some categories do not significantly differ from zero, the variable can still be significant to the model because the categories can differ significantly from each other.

We add variables to the model step-wise in the order they are presented in table Table 3.5. First, we start having the intercept only model (Model0) and then adding *exposure* variables, followed by *vulnerability*, *patching* and *exploit* variables respectively. This lead to the five different models as shown in Table 3.5.

Table 3.5: GLM regression results.

	Model0	**Model1**	**Model2**	**Model3**	**Model4**
(Intercept)	4.831***	2.475***	0.194	0.369	−1.881
	(0.389)	(0.735)	(1.833)	(1.161)	(1.276)
Exposure					
Install Base		0.186*	0.148	0.202**	0.225***
		(0.091)	(0.112)	(0.073)	(0.063)
Vulnerability					
Vulnerability Lifetime		0.0006+	0.0006	0.0005*	0.0007***
		(0.0003)	(0.0004)	(0.0002)	(0.0002)
Severity (CVSSv3)			0.065	0.005	−0.028
			(0.091)	(0.062)	(0.052)
Command Injection			2.469	1.399	2.783**
			(1.710)	(1.066)	(0.999)
Remote Code Execution			1.440	−0.242	0.794
			(1.669)	(1.040)	(0.944)
Router			0.094	1.137*	1.334**
			(0.855)	(0.560)	(0.480)
Surveillance			0.703	1.771*	1.480*
			(1.232)	(0.810)	(0.705)
Patching					
Complexity				0.294***	0.262***
				(0.068)	(0.059)
Availability				−1.122	−0.886
				(0.772)	(0.684)
Security Advisory				0.029	0.108
				(0.712)	(0.642)
Exploit					
Complexity					−0.008*
					(0.003)
Availability					1.382
					(0.974)
Num.Obs.	49	49	49	49	49
R^2 McFadden	-	0.151	0.204	0.334	0.377
AIC	830.6	709.7	675.6	573.6	542.5
BIC	832.5	715.3	690.8	594.4	567.1
Log.Lik.	−414.280	−351.836	−329.814	−275.805	−258.274
F	-	3.968	1.448	5.265	6.257
RMSE	337.69	299.13	268.71	499.84	479.02

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001

To evaluate the goodness of fit, we employed selection criteria based on established practices in the literature [2, 50, 147, 149, 261]. First, we used the Akaike Information Criterion (AIC) to compare model scores and determine the best-fitting model for the data [8], which also tends to perform better with relatively smaller sample sizes. Second, the Bayesian Information Criterion (BIC) places a stronger penalty on model complexity, favoring simpler models. This criterion is especially useful for identifying the most parsimonious model. A lower AIC or BIC value indicates a better fit [50, 147, 261]. Next, we calculated the Log-likelihood [127] to compare different models and assess their relative fit to the data. Then, the F statistic [127] to evaluate and compare the variance explained by the model against the unexplained variance, providing a measure of the improvement in model fit compared to a null model. Unlike AIC or BIC value, The higher the log-likelihood and the F statistic value, the better a model fits a dataset. Furthermore, we used the Root Mean Square Error (RMSE), a widely used statistical metric for comparing regression models [50]. RMSE offers a measure of overall accuracy for the model's predictions, aiding in the quantification of average prediction error. Even though our focus is to explain not to predict, we only reported the RMSE scores for completeness.

Looking at the scores provided by these criteria, Model 4 had the best fit among the models, with a lower AIC, BIC, and log likelihood, and a higher F statistic and lower RMSE. Even though the prediction score of RMSE in model 4 doesn't reflect the lowest, we used it to only report the observed value. Additionally, several of the variables in Model 4 were statistically significant, including the "Install Base", "Vulnerability Lifetime", "Vulnerability type Command Injection", "Device Type Router", "Device Type Surveillance", "Patch Complexity", and "Exploit Complexity" variables. This suggests that these factors play a significant role in determining exploit frequency in IoT binaries.

Looking at McFadden's R^2 value, we can get insights about how well a model explains the variance in the data. Note that in a GLM with a negative binomial distribution, the R^2 value is often lower than in other types of regression, such as linear regression, because the negative binomial distribution is often a better fit for overdispersed data (data with higher variance than expected under the model). This means that even though the model may have a good fit to the data, there will still be some remaining variation that is not explained by the model, resulting in a lower R^2 value.

Additionally, our primary goal is explanatory rather than predictive. The negative binomial model captures the underlying process that generates the observed distribution of exploit frequencies, rather than aiming for point-level prediction. Accordingly, the R^2 values should be interpreted as indicators of how much of the variation in exploit frequency can be explained by the modeled factors—values that are consistent with expectations for real-world behavioral and ecological data where high variability is inherent.

3.5.3. INTERPRETATION OF THE VARIABLE EFFECTS

The coefficients of the GLM model represent the relationship between the latent variables (vulnerability, patching, exploit, and exposure) as measured by the corresponding indicators and the frequency of exploits. If the coefficient for a particular indicator of vulnerability is positive, this suggests that an increase in that indicator is associated with an increase in exploit frequency in binaries. On the other hand, if the coefficient is negative, this suggests that an increase in that indicator is associated with a decrease in exploit

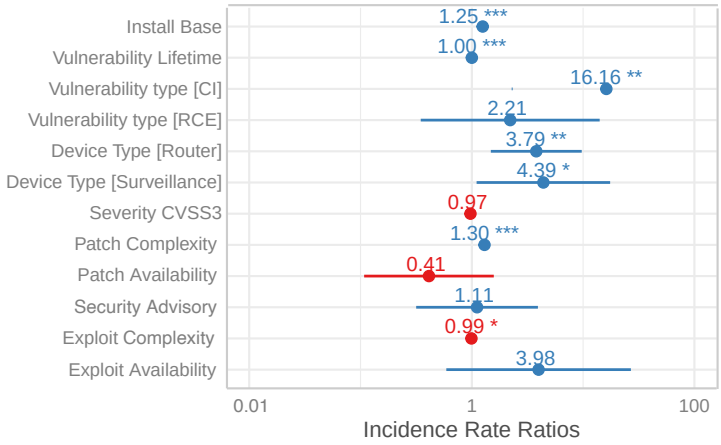


Figure 3.6: Forest plot of the GLM exponentiated coefficients

frequency in binaries. The coefficients can be used to understand how the different latent variables and their indicators influence the target selection of vulnerabilities in IoT binaries.

All the coefficients are positive except for *Patch Availability, Severity (CVSSv3)* and *Exploit Complexity*. The exponentiated coefficient and the 95% confidence interval are plotted in Figure 3.6.

The forest plot shows the exploits frequency is affected in a different manner by several factors. The variable with the largest impact is the vulnerability type, with a 16.16-fold increase in the exploits frequency when the vulnerability is a command injection compared to other vulnerability types. The install base also has a significant effect, with a 1.25-fold increase in the frequency of exploits per unit increase in the installed base. For example, if the installed base is 1,000, a unit increase in the installed base (i.e, 10,000) would correspond to a 25% increase in the frequency of exploits when all other variables in the model are held constant. Other significant factors include the device type, with a 3.79-fold increase in the frequency of exploits when the device is a router compared to other types of devices, and patch complexity, with a 1.3-fold increase in the exploits frequency per unit increase in patch complexity. This confirms that vulnerability severity (CVSS) rating has no impact, which was studied in previous work in [47].

On the other hand, some variables have a negative impact on the frequency of exploits. These include patch availability, with a 0.41-fold decrease in the frequency of exploits when patches are available compared to when they are not, and exploit complexity, with a 0.99-fold decrease in the frequency of exploits per unit increase in exploit complexity.

To understand the particular impact of a change in the explanatory variables, we compute a set of contrasts at the mean. In Table 3.6, the contrast estimates show the difference in the mean of the dependent variable (exploits) between each group and a reference group. For example, the contrast estimate for “Vulnerability type: Command Injection - Other" indicates that, compared to a hypothetical exploit with average characteristics, the number of exploits is 287 higher for vulnerabilities of type “Command Injection" compared to vulnerabilities of other types.

Table 3.6: Contrasts at the mean

Variable	Contrast	Estimate	Std.Error
Install_Base	+1	9.43	7.26
Vulnerability_Lifetime	+1	0.03	0.02
Vulnerability_type	CI - Other	287.00	197.00
Vulnerability_type	RCE - Other	22.90	24.10
Device_Type	Router - Other	30.80	23.70
Device_Type	Surveillance - Other	37.40	32.10
Severity_CVSSv3	+1	-1.17	2.36
Patch_Complexity	+1	11.00	8.33
Patch_Availability	TRUE - FALSE	-24.60	28.00
Security_Advisory	TRUE - FALSE	4.30	26.70
Exploit_Complexity	+1	-0.33	0.26
Exploit_Availability	TRUE - FALSE	31.30	27.20

Looking at Table 3.6, we can see that some of the contrast estimates have negative values. This indicates that, on average, the number of exploits is lower for the group being compared to the reference group. For example, the contrast estimate for “Patch Availability: TRUE - FALSE" shows that, on average, the number of exploits is 24.60 lower when a patch is available to remediate the vulnerability targeted by a specific exploit. Similarly, the contrast estimate for “Exploit Complexity: +1" indicates that, on average, the number of exploits is 0.33 lower for exploits with higher complexity compared to those with lower exploit complexity.

3.6. DISCUSSION

Our results have shown that several factors drive the exploitation of IoT vulnerabilities, including the exposure of the IoT devices and their vulnerability, the existence and complexity of patches and exploits, and the type of affected devices. These factors are aligned with traditional criminology theories, where rational attackers are interested in maximizing success while minimizing their effort and the chances of being caught. This suggest that attackers are limited by their resources when deciding which vulnerabilities to exploit, and they tend to focus on exploiting vulnerabilities that have a high impact and low complexity, as it is more cost-effective for them. This is in contrast to the commonly held belief that attackers will exploit any vulnerability they can find. The evidence from actual exploits confirms this theory indicates that attackers are not keen on additional exploitation work, and rather tend to stick to the same tactics as long as they are successful against a significant number of IoT devices. This aligns with data from underground markets [14], and it suggests that attackers are work-averse in the sense that they are not inclined to change their approach if it is still effective. From this perspective, IoT malware is likely to target vulnerabilities with a high potential for financial gain and a low risk of detection. This means that IoT malware is likely to focus on vulnerabilities that can be easily exploited, have a large install base, and provide significant benefits to attackers.

Attackers target vulnerabilities in commonly used devices, such as routers and security cameras, allowing them to conduct distributed denial-of-service attacks. These

vulnerabilities provide attackers with a large number of potential targets and can be exploited with relatively little effort. Additionally, the potential for disruption makes these vulnerabilities highly lucrative for attackers. On the other hand, IoT malware is less likely to target vulnerabilities with low potential for financial gain. For example, vulnerabilities in niche devices or those that only allow for limited access may not be worth the effort for attackers, as they may not provide significant benefits. Similarly, vulnerabilities that are difficult to exploit may also be less attractive to attackers, as they may not be able to exploit them successfully without being detected.

Our findings suggest that the CVSS score of a vulnerability did not significantly impact the exploit frequency. This means attackers may not necessarily prioritize exploiting vulnerabilities with higher CVSS scores over those with lower scores. This could be because attackers may have different priorities and objectives than those used to determine the CVSS scores. For example, a vulnerability with a lower CVSS score may be more attractive to an attacker if it is in a commonly used IoT device or if it has the potential for significant disruption. Additionally, we also find that the availability of patching did not significantly reduce the frequency of exploits. This suggests that even when a patch is available, attackers may still exploit the vulnerability successfully. This could be because not all users may apply the patch, or because attackers may find a way to bypass the patch. This highlights the importance of not only patching vulnerabilities but also of continuous monitoring, to detect and respond to any attempted exploitation.

Based on the results of our analysis, we recommend the following measures to decrease the usage of exploits and consequently improve the security of IoT devices:

- Prioritizing efforts on device types that are targeted more frequently. Large manufacturers with large install bases for their products should leverage their scale to improve defenses, as they are more likely to be targeted.

- Investing in better updating mechanisms that take into account patching and patch complexity to disincentivize IoT malware developers.

- Relying solely on CVSS scores on prioritizing vulnerability can lead to the risk of being targeted by cyberattacks. Instead, taking into consideration other characteristics of vulnerability such as vulnerability type and device types can help mitigating attacks.

- Providing a mechanism for consumers to obtain information on vulnerabilities, patches, and device support status, as well as considering the adoption of frameworks for secure software development and security labeling, can improve overall security and help consumers make informed decisions about the devices they use as most of the targeted devices are consumer-grade IoT.

- Enforcing strong authentication practices—such as unique, randomly generated passwords per device instance and mandatory password changes during initial setup—combined with encrypted access protocols (e.g., HTTPS, SSH instead of Telnet), can significantly reduce unauthorized access. Eliminating default or hard-coded credentials removes low-effort attack vectors and forces adversaries to invest

in more complex and costly exploit development. Where possible, password requirements should be managed on the vendor side—through firmware design or provisioning processes—rather than relying solely on end-user behavior.

- Disabling or uninstalling unused services and debug hooks, modes, and interfaces can help reduce the device's attack surface. The most frequent vulnerability types we observed in the wild are linked to services that are not necessary for the proper functioning of the devices. Hence, removing these would directly reduce the attack surface.

3.7. LIMITATIONS

Due to the fact that we used a variety of external sources, it was inevitable to inherit their limitations as well.

First, the exploit coverage. It is hard to estimate how representative our dataset is for exploits frequency in IoT malware. During the dynamic analysis of binaries collected over the last four years, we were able to capture 59 unique exploits during. This number could be limited due to some IoT malware techniques used by malware authors such as anti-analysis and obfuscation to evade detection. This limitation of dynamic analysis methods has been documented before [19] and is consistent with our findings. As shown in Chapter 2, the set of exploits identified here closely match those observed in prior studies employing both static and dynamic analysis approaches [19]. However, to cover a wider range of exploits in IoT binaries, we recommend collecting data from diverse and comprehensive sources and conducting longitudinal studies over an extended period. This will enable the capturing of the evolving nature of IoT exploits. It is also important to involve cross-referencing and validating exploit information using multiple sources, such as Virustotal [252], to ensure a more representative set of IoT exploits.

Moreover, during the phase of collecting information on vulnerabilities, we had to look for different sources to measure publication date for the 15 vulnerabilities with no CVE-ID. We used sources we believe a given vulnerability was first discovered and use the date of the publication as the disclose date. This issue of vulnerability disclosure date was in some published CVEs. Their publication date didn't match the year in the CVE-ID as some CVEs where published years before the assigned date.

Second, in terms of device exposure, we used Shodan to measure the size of install base of IoT devices in public. Even though it is open for public, navigating through to find the correct device has certain limitations [26, 34]. It's not a single specific query that can be carried out for all IoT devices. To maximize our chances of finding the correct IoT devices, we manually checked the raw data produced by the banner grabbing for all the devices in order to find the most matching device name, services or software version to build our query syntax. Despite manually verifying the available information for all checked devices, the software version data is not always accessible due to reliance on Shodan's collected banners. Therefore, we can't be sure that those devices are running the vulnerable software version. They might not be vulnerable. That said, this same limitation is faced by an attacker who is estimating the potential victim population before deciding which vulnerability to target. Only banner would be available to estimate the install base of a vulnerable device, until the attacker can run actual exploit code against potential

victims. In this sense, this noisy data would still provide a good predictor, since we are trying to infer the reasoning of the attacker.

Finally, the patch availability. We couldn't find information for half of the cases even though we search their vendors website for any updates. Although patching information can be found easier for published vulnerabilities in the NVD, some of these CVEs are not updated to include their security advisory in the resources. For those case, we did our best to look on the internet, as well as for the vulnerabilities that didn't have CVE-ID.

3.8. RELATED WORK

Attackers select specific IoT vulnerabilities to target in IoT binaries, yet only few studies that focus on the attacker decision process [72]. Some studies looked at the relationship between vulnerabilities and exploits. Householder et al. [128] looked into 'when and how many vulnerabilities get associated public exploits' using CVE-IDs.

As a result of 6 years study of 75,807 vulnerabilities, only 3,164 of them had public exploits within 91 days, on average. On the other hand, as discussed in Chapter 2, a study examining which vulnerabilities are targeted in the wild and for how long analyzed 17,720 samples statically and dynamically using three different datasets from 2015 to 2020. The findings show that only 63 unique exploits were identified, and the average time it takes for an attacker to exploit a vulnerability after disclosure is around 29 months. Similarly, Alrawi et al.[19] used both static and dynamic analyses of 166k samples collected in 2019 to find a list of targeted IoT vulnerabilities found in Chapter 2, which is also aligned with our findings. In addition, Nayak et al. [179] found only 15% of known vulnerabilities are exploited in the wild using around 300 million reports of intrusion-protection collected from more than six million hosts. This finding of attackers targeting only a few vulnerabilities out of thousands in the wild was also addressed in Chapter 2 and previous studies [8, 11, 19, 264].

While these previous studies focused on vulnerabilities and exploits, others have looked at their relationship with respect to patching in IoT and non-IoT devices. Nakajima et al. [174] found some vendors don't work to improve their patch release delay over the years of study; and vulnerabilities with sever impact had no prioritize over the others. Yet, other factors can also impact the duration of patch deployment. For example, Nappa et al. [175] found that hosts belonging to security analysts and applications with an automated updating mechanism have significantly lower median times to patch, and Kotzias et al. [150] found enterprise hosts are patching faster than consumers of 90% of vulnerable systems take more than six months on average to patch, while others found that attackers prefer to spend their effort on known vulnerable devices that remained unpatched [8, 192, 264] or on ones that needs less effort to compromise [72].

Finally, some authors looked at the relationship between attacks and Internet-exposed devices. Bada and Pete [34] they analyzed forums that discuss the IoT search engine "Shodan" using CrimeBB dataset from the Cambridge Cybercrime Centre. They investigated the main use cases of Shodan for attackers. They found that Shodan is actively used by attackers to collect information about vulnerable devices. Others reported that using Shodan and other search engines for exposed vulnerable devices made them susceptible for being used by attackers [26, 158], yet these search engines were used in research to help security defenders automate their detection for vulnerable devices [26, 38].

3.9. CONCLUSION

We found that several factors have an impact on the frequency of exploits. The type of vulnerability had the largest impact, with a 16.16-fold increase in the frequency of exploits when the vulnerability is a command injection compared to other types of vulnerabilities. The installed base and the type of device were also found to have a significant effect, as well as patch complexity. On the other hand, variables such as patch availability and exploit complexity were found to have a negative impact on the frequency of exploits.

Our findings have important implications for cybersecurity efforts, as they can inform the design of more robust countermeasures and controls against IoT attacks and help protect against the growing threat of IoT botnets targeting connected devices. Additionally, our study highlights the need for proactive efforts to address vulnerabilities in IoT devices, as the proliferation of these devices is likely to continue and create new opportunities for attackers to exploit.

4

PREDICTING EXPLOITATION OF IOT VULNERABILITIES

The proliferation of Internet of Things (IoT) devices has led to a surge in vulnerabilities, with traditional metrics like CVSS and PoC exploits failing to fully explain exploitation patterns. To address this, we leverage features from the-state-of-the-art prediction model EPSS – such as CVSS, CWE, vendors, external references, vulnerability age, and PoCs – and combine it with new features derived from hacking communities. Our study of 23,373 IoT-related CVEs and 25k posts from 25 hacking forums highlights the importance of including insights on attacker behavior from discussions involving vulnerabilities. We identified 38 features with a p-value < 0.05 that impact attackers' selection of IoT vulnerabilities. We use two metrics to evaluate our model with features from hacking forums: McFadden's pseudo R^2, which showed a 21% improvement in explaining variance, and the Brier score for prediction accuracy, with a 17% improvement over EPSS. These results emphasize that current state-of-the-art methods struggle to capture the distinct nuances and complexity of IoT threats, and incorporating available information such as insights into attacker behavior can enhance the factors influencing the targeting of IoT vulnerability better.

This chapter has been published as: **Arwa Abdulkarim Al Alsadi**, Mathew Vermeer, Takayuki Sasaki, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. Bits and Pieces: Piecing Together Factors of IoT Vulnerability Exploitation. In *Proceedings of the 2025 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Hanoi, Vietnam, https://doi.org/10.1145/3708821.373387*

4.1. INTRODUCTION

In recent years, the proliferation of Internet of Things (IoT) devices has brought about a corresponding surge in vulnerabilities, amplifying the potential risk of exploitation. For example, accordingly to VARIoT [139], the number of reported IoT vulnerabilities increased from 772 in 2010 to over 3,253 by 2022. This escalation in volume of IoT vulnerabilities necessitates a critical examination of the current state-of-the-art mechanisms for predicting and understanding the factors that lead to exploitation. Since it is unlikely that vendors and users have the time and resources to mitigate all discovered vulnerabilities, it helps to focus on those that actually will get exploited.

Previous research, including our earlier analyses presented in Chapter 2 and Chapter 3, has shown the importance of using IoT-specific features for understanding IoT targeting in the wild [19, 34, 166].

However, the collection of these IoT-specific features such as device types [19, 34, 43, 225] and number of exposed devices in the internet [34, 195] is labor-intensive and currently not scalable and requiring manual work. Therefore, we want to support vendors and users by improving our understanding of what factors make certain IoT vulnerabilities targeted by attackers.

Since 2003, security practitioners have commonly relied on the Common Vulnerability Scoring System (CVSS) public information as a metric for predicting which vulnerabilities are more likely to be targeted by attackers – even though the developers of CVSS actually state it should not be used for this purpose [230]. Higher CVSS scores are interpreted to mean that the vulnerability is more likely to be exploited, hence more urgently needs to be patched. Yet, empirical research has indicated CVSS is unable to predict attacks in the wild [11, 47, 215]. Additionally, on average, 49% of real-world exploits occur before CVSS scores are published [60].

Recognizing the limitations of vulnerability severity scores in exploit prediction, others have explored alternative predictors such as the availability of Proof of Concept (PoC) exploits in public. Unfortunately, this feature has also been questioned as a reliable predictor, as instances most vulnerabilities with PoC exploits never manifest as real-world targets of attacks [10, 11]. Only 4.17% vulnerabilities get associated public exploits within 365 days [128]. Do most attacks occur within that set of vulnerabilities with a PoC?

To fill this gap in understanding which vulnerabilities get attacked in the wild within 30 days, Jacobs et al. introduced the Exploit Prediction Scoring System (EPSS) [137]. It integrated information from various sources such as CVSS scores, vulnerability features from the National Vulnerability Database (NVD), availability of PoC exploits, vulnerabilities used in offensive security tools, and features derived from social media. At the time of writing, 111 vendors have integrated EPSS in their products, including IoT vendors [102] and platforms such as Vulners [258] and Shodan [160].

While EPSS has shown promise in predicting exploitation for the total set of CVEs, this set consists mostly of CVEs for general-purpose IT systems. EPSS often assigns low scores to IoT vulnerabilities, even to those known to be exploited in the wild, such as the ones included in CISA's Known Exploited Vulnerabilities (KEV) catalog [229]. For instance, CVE-2017-17215 was assigned an EPSS score indicating a near-zero probability (0.05) of being exploited in the wild within 30 days, starting from May 2021, which is the earliest available score in EPSS. It took over three years for the score to increase significantly,

reaching 0.96 by April 2023. However, this does not align with real-world findings, as this IoT vulnerability was identified as one of the most frequently and consistently targeted vulnerability over an extended period in multiple studies and security reports conducted as early as January 2015, all the way up to February 2024. aligns with the findings in Chapter 2 and Chapter 3, as weel as in prior work[19, 77, 78, 134].

This indicates potential limitations in accurately assessing the severity of IoT vulnerabilities. To raise awareness among vendors and users of EPSS, we aim to reassess the effectiveness of features like CVSS, PoC, and others used in EPSS, specifically for IoT-related vulnerabilities. Any evaluation of EPSS is complicated, however, because the EPSS is unavailable and not fully disclosed, even in its peer-reviewed publications. Without access to the underlying data, model, and source code, independent verification or explanation of its findings is not possible.

We aim to better explain the targeting of IoT vulnerabilities over EPSS by testing whether the inclusion of features from hacking forums can improve the model performance. This extension builds on work that showed black market or underground hacking forums provide information for predicting exploitation [16, 36, 220, 238]. This brings us to the central question for our paper: *What factors determine whether an IoT vulnerability is targeted?*

We create a dataset of IoT vulnerabilities published between January 2016 to June 2023 using VARIoT [139]. This results in a set of 23,373 IoT-related CVEs affecting hardware, software, and applications. Within this set, we need to distinguish between vulnerabilities that were observed to be targeted by attackers and those that were not. We meticulously collected instances of vulnerabilities that were observed to be targeted in the wild from seven data sources: VirusTotal [226], CISA's Known Exploited Vulnerabilities (KEV) Catalog [66], VulnCheck's Known Exploited Vulnerabilities (KEV) [256], AttackerKb [31], Google Project Zero [117], IoTPOT [196], and X-Pot [145] based honeypots.

Next, we set out to explore which features can help us distinguish between targeted and non-targeted vulnerabilities. We identified a lower-bound set of 848 targeted IoT-related CVEs. Subsequently, we extracted features for explaining the targeted set. We extended EPSS with features from the CrimeBB database [201], which contains scrapes of 36 different hacker forums. We found 699 IoT-related CVEs discussed in over 25k posts across 25 different forums, with targeted IoT vulnerabilities being more frequently discussed than non-targeted ones.

Based on these features, we create a model and evaluate its performance using the Brier score and McFadden's pseudo R^2 score. The Brier score measures prediction accuracy, while the R^2 score assesses the overall fit of the model. We incorporate features from hacker forum data, consolidated into a single 'Engagement' feature using PCA. Our model's McFadden's R^2 score increased by 21% with the inclusion of this feature, highlighting the value of hacker forum data in explaining the variance in targeted IoT vulnerabilities. Using the Brier score, our model, with 38 features (fewer than EPSS), improves prediction accuracy by 17% over the EPSS model.

The main contributions of the paper are:

- We develop a theoretical model explaining why certain IoT vulnerabilities are exploited, based on vulnerability, exploitability, and hacking community factors.

- We analyze discussions across 36 hacker forums, identifying 699 IoT-related CVEs mentioned in over 25,000 posts. We find that targeted IoT vulnerabilities are discussed more frequently than non-targeted ones, highlighting the influence of hacking forums on exploitation patterns and risk assessment.
- We develop a logistic regression model that improves EPSS prediction for IoT vulnerability exploitation by 17% by incorporating hacking forum features, which enhance the model's ability to explain variance in targeting by 21% and significantly impact pre-exploitation risk assessment.
- We publicly release the code for our exploit prediction model, ensuring transparency and reproducibility [1].

4.2. RELATED WORK

Several studies have examined the relationship between CVSS scores and exploit-related factors. Allodi et al. [10] found that using CVSS alone to predict attacks is unreliable, comparable to random selection. This inconsistencies in severity ratings is supported by Wunder et al. [267]. Although the high CVSS scores do not reliably correlate with exploits in the wild, and excluding CVSS had minimal impact on results [47], only 9% of CVSS scores are available at disclosure [215], and nearly half of exploits occurring before CVSS publication [60].

On the other hand, the existence of PoC exploits was found to be a better risk factor [11], but it is not always a reliable indication of exploitation in the wild [10]. A 2020 study [135] found that a low rate, only 5%, of known vulnerabilities are exploited in the wild, consistent with the findings in Chapter 2 prior research [8, 11, 19, 179, 235, 264]. Yet, features within PoC exploits can be valuable for estimating exploitation frequency shown in Chapter 3, predicting functional exploits [235], early exploitability [132], or predicting exploits in the wild [135].

In the realm of exploit prediction in the wild, various methodologies have been explored. For example, Twitter discussions can predict exploitation more accurately than CVSS [60] or PoC-based approaches [215], and help estimate exploitability scores [235]. Other studies have used vulnerability descriptions and online discussions to predict exploitation over time [16, 137, 235] or early exploitability of just disclosed vulnerabilities[132].

Previous studies have explored IoT-specific features to understand IoT targeting in the wild [19, 34, 166]. However, collecting these features, such as device types [19, 20, 34, 43, 225] and the number of exposed devices online [34, 195], is labor-intensive, not scalable, and requires manual effort. Since these features are not readily available online to scale for our dataset of over 23k CVEs and would require similar manual work from vendors and users, we do not include them. As discussed in Chapter 2 and Chapter 3, these approaches face significant practical limitations despite their analytical value.

The most comprehensive approach is the Exploit Prediction Scoring System (EPSS), a data-driven framework predicting if a vulnerability will be exploited [136]. EPSS incorporates various data sources including CVSS, vendor information, published exploit code, and references to external websites discussing the vulnerability. While achieving accurate

[1] https://doi.org/10.4121/69d111e7-30e6-4206-b27d-22a5f4e33722

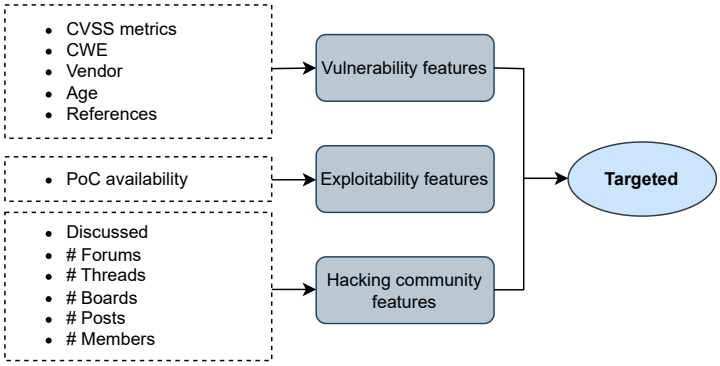


Figure 4.1: Theoretical Model.

exploitation estimates with a ROC AUC of 0.838, they aimed to improve by adding data like social media and vulnerability scans. In their subsequent version, EPSSv3, they incorporated those additional variables besides more vulnerability features such as CWE and age of vulnerability resulting in 82% performance improvement over previous models, which enriched the predictive capability of the model [137]. However, the lack of full disclosure of the EPSS model hinders its replication and use by others, preventing independent verification of its findings [229].

Our study advances beyond the state of the art by including data from underground hacking forums, specifically for IoT vulnerabilities. While using vulnerability-related features and PoC exploits as in [137], we expanded our approach by examining discussions within underground hacking forums, specifically targeting IoT vulnerabilities. For example, Tavabi et al. [238] found that dark web discussions outperforms state-of-the-art methods in exploit prediction and improved exploit prediction, with 14% of vulnerabilities mentioned being exploited. Data from dark web also had the highest exploitation rate among four datasets used to predict exploitation in the wild in [16]. Leveraging the CrimeBB dataset, which comprises 36 different underground forums, Moreno-Vera et al. [168] identified 1,498 unique CVEs discussed in these platforms, highlighting the value of incorporating hacker-used platforms for more accurate predictions of vulnerability targeting. This comprehensive approach, integrating insights from online discussions, and underground forums, enhances our understanding and prediction of cyber threats and exploitation activities [15, 36, 132, 215, 220].

4.3. THEORETICAL MODEL

To help answer what determines the targeting by attackers of IoT vulnerabilities, we draw upon previous studies that have investigated various factors and features influencing the selection of a vulnerability as a target. For instance, some studies have explored vulnerability features such as disclosure date [91, 137, 175], severity metrics [11, 91, 137], software and hardware weaknesses, vendors, and label references to advisories, solutions, and tools [91, 137], while others have examined the correlation between exploitation in the wild and the presence of PoC exploits [11, 137, 175, 215]. Additionally, the relationship

between discussions of vulnerabilities in social media or underground forums and their exploitation in the wild has also been explored [15, 36, 201, 215, 220, 235]. Our aim is to consolidate these features from prior and recent studies into a comprehensive theoretical model. We develop a model to explain which IoT vulnerabilities will be targeted (Figure 4.1). Our model shares similarities with EPSS in integrating various factors that influence the targeting of certain vulnerabilities. However, while we focus exclusively on vulnerabilities within the IoT realm, EPSS adopts a broader approach, including vulnerabilities across all domains. Though our model draws inspiration from EPSS's approach to integrate factors derived from online sources to enhance vulnerability analysis and prediction, it extends this by incorporating features extracted from underground forums. Overall, our model consists of three factors: vulnerability features, exploitability features and hacking community features (see Figure 4.1).

Vulnerability features. Vulnerability features such as disclosure date, CVSS metrics, software and hardware weaknesses, vendors, and label references to advisories and other external websites might predict which vulnerability is likely to be targeted by attackers. Jacobs et al. [137] identified 30 influential features, including count of references to external websites, vulnerability age, specific vendors, and CVSS submetrics, as significant predictors of exploit likelihood. Thus, we integrate vulnerability features into our model.

Exploitability features. The relationship between the existence of PoC exploit code and exploitation in the wild has been studied in previous work [11, 137, 175, 215]. Thus, we include it as feature.

Hacking community features. Discussions of vulnerabilities on social media [60, 215, 235] and underground forums [15, 36, 220] have been found to provide more accurate predictions of vulnerability exploitation than vulnerability features or PoC exploits alone. However, using Twitter feeds was not listed among the top 30 most influential features in EPSS [137]. In fact, Tavabi et al. [238] found that utilizing discussions from the dark web for exploit prediction outperforms state-of-the-art methods, including social media feeds. Thus, we incorporate hacking community features using underground hacking forums to explain whether discussions of IoT vulnerabilities on such platforms can increase the likelihood of these vulnerabilities being targeted. For example, Moreno-Vera et al. [169] identified 1,498 unique CVEs discussed within these underground hacking forums, and Almukaynizi et al. [16] found that 14% of vulnerabilities mentioned on the dark web and deepweb were observed being exploited in the wild. This dataset had the highest rate of exploitation among the four datasets used in the study, highlighting the value of incorporating hacker-used platforms for discussions of vulnerabilities on underground forums.

4.4. METHODOLOGY

The focus, first, lays on gathering data on all IoT vulnerabilities from January 2016 to June 2023, driven by the notable surge in attacks on IoT devices following the 2016 release of the Mirai botnet source code [24]. Next, we collect data on which subset of the vulnerabilities were targeted in the wild. We then distinguish between targeted and non-targeted vulnerabilities based on observed attack data. Subsequently, we gather features categorized into vulnerability, exploitability, and hacking community, all listed in Figure 4.1. Finally, we employ a regression model to assess the significance of each

Table 4.1: Description of variables in the theoretical model.

Variable	Description	Count	Type	Data sources
Targeted IoT vuln.	List of IoT exploits in the wild	1	Binary	CISA, Virustotal, IoTPOT X-Pot, VulnCheck KEV Google Project Zero AttackerKB
CVSSv3 metrics	Measuring the impact and exploitability of a vulnerability	23	Multi-level	NVD
CWE	List of software and hardware weaknesses	111	Binary	NVD
Vendor	List names of affected vendors	212	Binary	NVD
Age of vulnerability	Number of days from the CVE publication date until the day of the analysis	1	Numeric	MITRE CVE list
References with labels	Number of references with each of the reference labels	19	Numeric	MITRE CVE list, NVD
PoC exploits	Denote whether the PoC exploit is available or not	1	Binary	Exploit-DB
Discussed	Denote whether a certain IoT CVEs was discussed or not	1	Binary	CrimeBB
#Forums	Number of forums in which a particular IoT CVE was discussed	1	Numeric	CrimeBB
#Boards	Number of boards in which a particular IoT CVE was discussed	1	Numeric	CrimeBB
#Threads	Number of threads in which a particular IoT CVE was discussed	1	Numeric	CrimeBB
#Posts	Number of posts in threads in which a particular IoT CVE was discussed	1	Numeric	CrimeBB
#Members	Number of members involved in threads in which a particular IoT CVE was discussed	1	Numeric	CrimeBB

feature in influencing IoT vulnerability targeting. This allows for a deeper understanding of the threat landscape and facilitates recommendations for enhancing the security of IoT systems.

4.4.1. COLLECTING IOT VULNERABILITIES

We identify targeted and non-targeted IoT vulnerabilities through a two-step process. First, we collect IoT vulnerabilities from the VARIoT dataset [249], covering vulnerabilities from January 2016 to June 2023. Second, we determine which were targeted using seven sources: (i) VirusTotal [226], (ii) CISA KEV [66], (iii) VulnCheck KEV [256], (iv) AttackerKB [31], (v) Google Project Zero [117], and honeypots such as (vi) IoTPOT [196] and (vii) X-Pot [145].

We then classify vulnerabilities by subtracting the targeted IoT vulnerabilities from the full list, generating a separate set of non-targeted vulnerabilities.

IoT vulnerabilities. We focus exclusively on IoT-specific CVEs using VARIoT [139], which identifies IoT-related vulnerabilities based on its broad definition: “an item (except a phone, PC, tablet, or data center hardware) with network connectivity and data exchange capabilities”[138]. While acknowledging VARIoT’s limitations in accurately classifying IoT-related vulnerabilities, we find its comprehensive repository valuable for our research objectives, as demonstrated in various studies [85, 208, 222].

From VARIoT, we collect all CVE-IDs published between January 2016 and June 2023 to align with the spread of the Mirai source code in 2016 [24]. Some IoT vulnerabilities in VARIoT lack CVE-IDs from NVD, often originating from other sources like CNVD. In line with EPSS, we only include vulnerabilities with CVE-IDs. We identified 23,373 CVEs classified as IoT vulnerabilities in VARIoT.

Targeted IoT vulnerabilities. To identify targeted IoT vulnerabilities, we use VARIoT's comprehensive CVE list and cross-reference it with seven different data sources. Of course, not all attacks are observed in our datasets, or in any datasets, meaning our data provides a lower bound set of targeted CVEs. Additionally, the time frame covered by some datasets does not encompass the entire period from January 2016 to December 2024, during which we collected IoT vulnerabilities.

First, we employed a VirusTotal search query for binaries tagged with a CVE-ID in our VARIoT dataset (*tag:cve-**) [253]. We found 526 instances out of 23,373 total CVEs. This count, however, could be impacted by VirusTotal's search constraints, limited to vulnerabilities targeted within 90 days from our search date. Second, we use the KEV Catalog. This list includes 1,026 CVEs. There is an overlap of 218 CVEs with the 23,373 CVEs that were classified as IoT by VARIoT. The third source was VulnCheck's Known Exploited Vulnerabilities (KEV) [256]. This list includes 3,089 with 426 overlapping CVEs that were classified as IoT by VARIoT. This makes VulnCheck KEV the source with the highest number of IoT CVEs exploited in the wild among our datasets. Next, we used the 23K CVEs to fetch data via cURL and then scraped AttackerKB [31] to collect CVEs tagged as "Exploited in the Wild". We identified 312 out of 23,373 total IoT CVEs. For the fifth source, we leverage Google Project Zero [117], which provides a public list of 325 zero-day CVEs exploited in the wild. Of these, 45 IoT CVEs overlapped with the 23K CVEs in VARIoT. The last two datasets are both from honeypots, IoTPOT [196] and X-Pot [145]. We gathered a list of targeted vulnerabilities and then identified which of these vulnerabilities overlapped with out set of VARIoT CVEs. In IoTPOT logs, collected from September 2018 to September 2021, we observed 21 targeted IoT CVEs. In X-Pot logs, collected from July 2019 - October 2023, we identified 114 targeted IoT vulnerabilities.

We identify 848 unique targeted IoT vulnerabilities within these seven datasets. Thus, we treat the remaining 22,525 CVEs out of the total set of 23,373 CVEs as non-targeted IoT vulnerabilities.

4.4.2. COLLECTING EXPLANATORY FACTORS

To assess the influence of the explanatory factors in the theoretical model (section 4.3), we identify and gather features that can serve as proxies for those factors. In this section, we outline the process of collecting these features, which are summarized in Table 4.1.

Vulnerability features. We use the NVD [177] dataset to collect five vulnerability features following the methodology used in [137].

CVSSv3 metrics: Given our dataset's focus on IoT vulnerabilities from 2016 onward, we use CVSSv3 [111] to derive base metrics, split into exploitability and impact. Exploitability includes five components: attack vector, attack complexity, privileges required, user interaction, and scope. Impact covers confidentiality, integrity, and availability. Except for 33 CVEs lacking NVD scores, we applied one-hot encoding to all sub-metrics, resulting in 22 variables. We also include the overall CVSSv3 base score as a single binary variable, representing the combined score of these sub-metrics.

CWE: We collected the Common Weakness Enumeration (CWE), a list of software and hardware weakness types that can become vulnerabilities [189]. We collect 240 different CWE for 23,317 CVEs. The remaining 95 CVEs lacked CWE assignments due to being unavailable or private. We created binary variables for the 111 CWE categories associated

with 10 or more vulnerabilities for modeling (see section 4.7).

Vendor: As some vendors might be more attractive to attackers than others, we use the Common Platform Enumeration (CPE) from NVD to extract only the vendor name(s). We collect vendors for all but 88 CVEs due to the CVEs being marked as "RESERVED" or "REJECT". We did not alter or fix any typos or misspellings within the records. Out of 1,423 unique vendors we found, only 212 vendors with 10 or more CVEs are included in our model, as including vendors with fewer CVEs did not improve performance, according to [136]. Then, we created a binary variable for each vendor and added a feature to measure the total number of vendors per CVE to evaluate whether CVEs affecting more vendors are more likely to be targeted.

Age: The age of a vulnerability may affect its likelihood of exploitation. To test whether older vulnerabilities might be less appealing to attackers due to reduced vulnerable population [137] within the IoT realm, we used the publish date from MITRE [239] to calculate the age — the number of days between the publish date and our feature extraction date. We collected the publish dates for all vulnerabilities but 56 that were marked "RESERVED" or "REJECT".

References: We measure level of activity and analysis related to vulnerabilities in [137] by quantifying the number of references linked to each CVE in MITRE CVE list [239], excluding 423 CVEs marked as “RESERVED,” “REJECT,” or those without an assigned label to the hyperlink. Furthermore, also collect the unique reference labels assigned to these CVEs listed by NVD, such as Third Party Advisory, Vendor Advisory, and VDB Entry. We identify 18 distinct labels, which we use as binary features.

Exploitability features. We use the exploit database Exploit-DB [105] to collect PoC availability. While other sources such as Metasploit [164] and GitHub [262] also host PoC exploits, Exploit-DB remains one of the best coverage source for PoC exploits [135].

PoC availability: We evaluate the exploitability risk of vulnerabilities based on their availability. We only identify PoC exploits for 890 (3.8%) out of the 23,373 CVEs, indicating a relatively limited presence of publicly available exploits for these vulnerabilities.

Hacking community features. To analyze IoT vulnerability discussions on underground forums, we use the 2023 CrimeBB dataset [201], provided by the Cambridge Cybercrime Centre [51]. It includes 36 forums in English, Russian, and Spanish; however, “Hackers Armies” was unavailable, leaving 35 forums in our analysis. These forums serve as platforms for exchanging ideas and participating in various activities, some of which are illegal. They follow a structured format with boards covering topics from hacking to marketplaces (see Appendix Table B.1). Each board contains threads on specific subjects, composed of posts by members. We extract six features from these discussions:

Discussed: To measure whether an IoT vulnerability was discussed within the underground forums, we match post content with CVE-IDs from our dataset. For this matching, we allow for variations in capitalization.

Number of forums: We assess the popularity of a vulnerability across the forums by counting the number of forums in which the IoT vulnerability was discussed.

Number of boards: We count how many boards discussed specific IoT CVEs to understand the range of topics and interests they span. In total, these vulnerabilities appeared on 122 boards across 25 forums.

Number of threads: Counting distinct threads discussing each IoT vulnerability helps

quantify its activity level. We identified 735 unique threads discussing IoT-related vulnerabilities.

Number of posts: Assuming more active discussion indicates higher risk of real-world targeting, we count posts in threads mentioning IoT CVEs. We found 25,782 posts, including both specific and general discussions involving IoT CVEs.

Number of members: To measure engagement level around IoT vulnerabilities, we counted distinct members involved in related discussions, using the same method as for post counts. A total of 5,129 users participated in threads specifically about or generally on IoT CVEs.

Number of members: To measure engagement level around IoT vulnerabilities, we counted distinct members involved in related discussions, using the same method as for post counts. A total of 5,129 users participated in threads specifically about or generally on IoT CVEs.

We acknowledge that the member count may represent an upper bound on actual unique individuals. As noted by the CrimeBB dataset creators that various accounts might belong to the same actor [200], meaning that Sybil accounts (i.e., multiple accounts controlled by single users) cannot be definitively ruled out in underground forums. However, several factors mitigate this concern for our predictive modeling. First, the structural barriers in these forums, including account verification processes [67], requirements for technical skill demonstration [21], and reputation systems [146, 172], make creating and maintaining multiple accounts significantly more costly than on open social media platforms, naturally limiting the extent of this phenomenon. Second, our predictive models incorporate multiple correlated features (number of posts, threads, boards, and forums), which together capture discussion intensity; machine learning models with redundant features have demonstrated robustness to noisy data [125, 214], such that if Sybil accounts inflate member counts, the genuine activity signals from other features would still dominate the prediction. Third, even if member counts were systematically inflated, this would affect all CVEs proportionally, preserving the relative ordering used for risk prioritization. Finally, the primary risk of overcounting members would be to slightly increase false positives (predicting exploitation for CVEs with inflated engagement metrics), which does not fundamentally undermine defensive utility [151, 157], as practitioners still benefit from a reduced prioritization set compared to reviewing all CVEs.

Similar to EPSS, we did not perform experiments to add or filter out intentional misinformation or automated posts in underground forum discussions. However, prior research [16, 215] shows that the introduction of misinformation had minimal impact on exploit prediction accuracy. The work in [16] demonstrated that exploit prediction models remain robust against adversarial noise in discussions from underground forums, where the aforementioned structural barriers such as account verification, skill demonstrations, and reputation systems make large-scale data poisoning far more difficult, compared to social media platforms like Twitter/X. Even if substantial noise would be added (e.g., 20% noise in testing data), this leads to minimal degradation in performance (AUC scores above 0.87). In [215], the authors indicate that adversaries can manipulate public platforms like Twitter/X to poison classifiers, but the impact on precision is limited, with a drop to only around 20% under the most sophisticated attack scenarios. In sum the

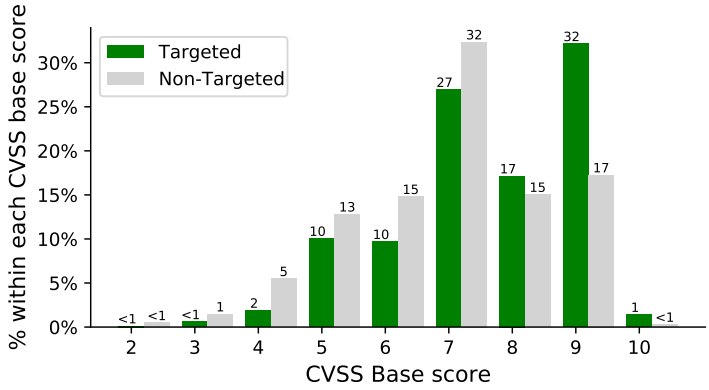


Figure 4.2: Proportion of IoT vulnerabilities within each CVSS base scores.

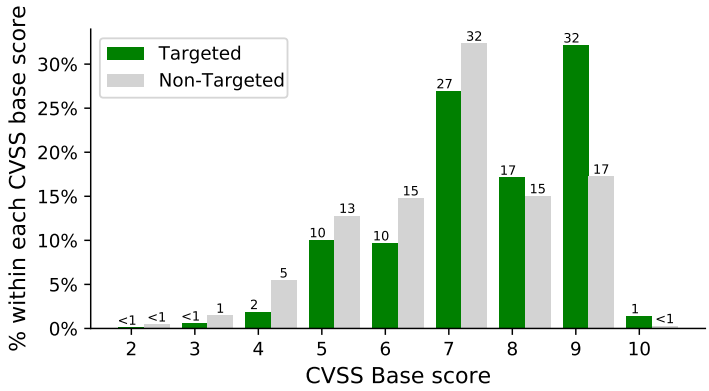


Figure 4.3: Targeted and non-Targeted IoT vulnerabilities within each CVSS base score.

impact is limited. We would also like to note that the main impact of misinformation would be to increase the false positive rate: more CVEs are being discussed on the forums, and hence some might be predicted as higher risk, yet in reality they will not actually be attacked. A somewhat higher false positive rate does not directly undermine the model's utility, as prioritization remains valuable for defenders. Out of thousands of CVEs, the model predicts a small subset as likely to be exploited. Even if the latter set would double in size because of false positives, it still helps defenders to prioritize a small fraction of a total set of thousands.

4.5. VULNERABILITY AND EXPLOITABILITY

In this section, we present the findings of vulnerability characteristics and their PoC exploit code among both targeted and non-targeted IoT vulnerabilities.

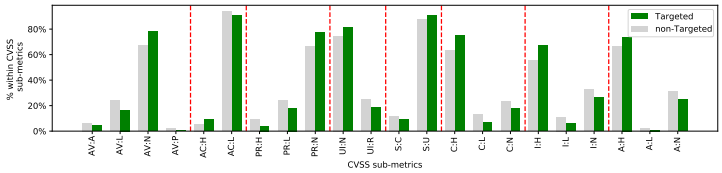


Figure 4.4: Targeted vs non-Targeted IoT vulnerabilities for each CVSS sub-metrics.

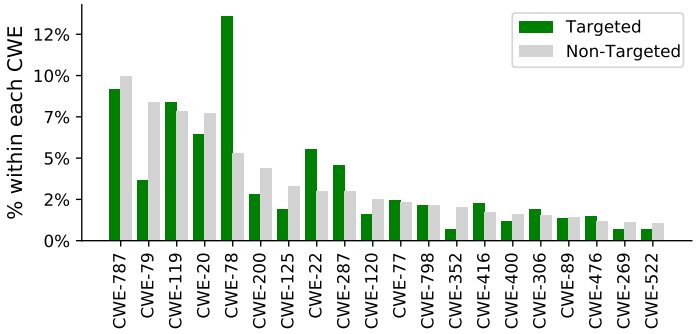


Figure 4.5: The top 20 CWE categories among targeted and non-targeted IoT vulnerabilities, ordered by CWE the with highest number of vulnerabilities.

4.5.1. VULNERABILITY FEATURES

CVSSv3 metrics. The CVSS base score, ranging from 0 to 10, is derived from Exploitability and Impact metrics, assessing how easily a vulnerability can be exploited and its potential consequences [111].

Analyzing targeting CVEs rates per CVSS score (Figure 4.2), 15.38% of CVEs with a score of 10 were targeted, but lower scores also saw targeting, indicating a weak correlation between score and targeting. Attackers often choose lower-scored vulnerabilities, despite many high-severity, with scores 9 and 10, remaining unused.

In fact, in absolute numbers, the attackers have chosen many more vulnerabilities with low scores, as we can see in Figure 4.3. Figure 4.3 shows the proportion of targeted and non-targeted IoT vulnerabilities for each CVSS base score. While higher CVSS scores (e.g., 9 and 7) account for a significant share of targeted vulnerabilities, lower-scored vulnerabilities (e.g., 5 and 6) are also targeted. For example, 20% of the total targeted vulnerabilities have a CVSS base score of 5 or 6. Also, many high-severity vulnerabilities remain untargeted. Only about 1% of total targeted vulnerabilities have a base score of 10, while the largest subset (32%) has a base score of 9. The average base score for targeted CVEs is 7.5, only slightly higher than 6.9 for nontargeted CVEs. This indicates that adversaries do not solely focus on high-severity vulnerabilities but may also target medium-severity ones, likely considering other factors such as ease of exploitation, availability of PoC exploits, and real-world impact. In short, this suggests that CVSS has weak predictive power, in line with earlier research [13].

Examining the distribution of IoT vulnerabilities within CVSS sub-metrics (Figure 4.4), we begin with exploitation metrics (left). The majority of targeted CVEs can be remotely

exploited, with 67.15% using the network as the attack vector (AV:N). This potentially impacts attack complexity, as nearly 94% exhibit low attack complexity (AC:L), while 66.2% require no privileges (PR:N), and 74.69% do not require user interaction (UI:N). This confirms previous findings that attackers tend to target vulnerabilities with low attack complexity [14].

That said, we do see that non-targeted CVEs have almost the same distribution. In other words, many more CVEs also possess these features and, yet, they are not targeted. We see the same thing for the impact metrics (right side of Figure 4.4). This all aligns with previous findings in Chapter 3 and [10] that CVSS alone is not an exhaustive predictor on its own for exploitation in the wild. We now see why: these features do not effectively differentiate targeted from non-targeted vulnerabilities. It is unclear whether this is because experts are not reliable in rating these features, or the features themselves are not that relevant to attacker's targeting decision.

Common Weakness Enumeration (CWE). The hardware and software weaknesses in vulnerable systems are categorized using the CWE framework. Among our 23k IoT-related vulnerabilities, we identified 240 CWE categories, with four—CWE-178, CWE-782, CWE-917, and CWE-1336—appearing only in targeted CVEs. All were observed once except CWE-917, which appeared in three vulnerabilities. This may reflect their rarity rather than attacker preference.

We found 88 CWE categories in both targeted and non-targeted CVEs, with the top 20 distributions shown in Figure 4.5. Among these, CWE-78 (OS Command Injection) was the most common in targeted CVEs (13.6%), This result is consistent with the findings in Chapter 2, Chapter 3, and related work [19], which similarly highlight CWE-78 as a primary infection vector. CWE-787 (Out-of-Bounds Write) ranked second (9.16%) and was also the most frequent in non-targeted CVEs for nearly 10%.

In sum, like CVSS scores, CWE categories provide only a weak signal of attacker preference. To put it differently, a random selection of vulnerabilities would yield a similar CWE distribution to that for the observations of targeted vulnerabilities.

Vendors. Examining the associations between vendors and CVEs, we found only 280 out of 1,423 vendors were linked to targeted CVEs, while 1,143 remained untargeted. Figure 4.6 shows that vendors with more CVEs tend to have more targeted vulnerabilities. For instance, Apple and Cisco lead with 145 and 110 targeted CVEs, respectively, while 45 vendors have only one targeted CVE.

On average, these 280 vendors have 5.9 targeted CVEs (SD = 15.5). Among untargeted vendors, Qualcomm has the most with 507 CVEs, followed by IBM with 254. In short, we see a linear relationship – more CVEs overall will result in more targeted CVEs – with quite some variance.

In addition, targeted CVEs also tend to be associated with more vendors—1.9 per CVE on average, compared to 1.2 for non-targeted ones. This suggests vendor involvement could influence attacker selection. Perhaps more vendors per CVE indicates a larger install base of vulnerable systems.

Age. The age of a vulnerability could potentially influence its likelihood of being exploited. We measure the age by the number of days between the vulnerability published date and the time of our feature extraction. The overall average age of all IoT vulnerabilities included in this study is 1,430.5 days.

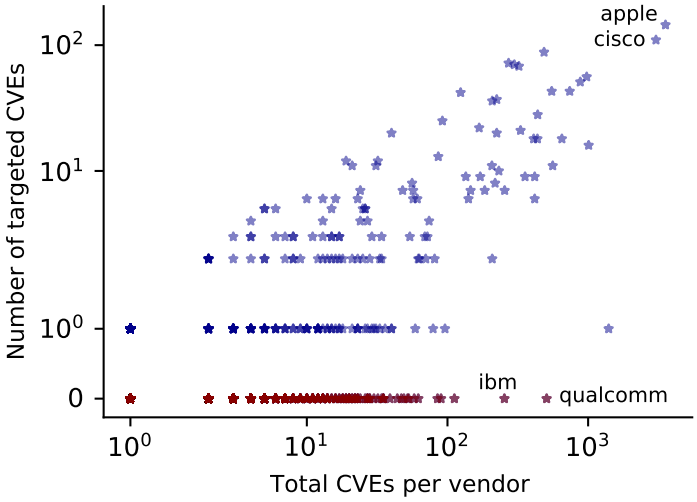


Figure 4.6: Relationship between targeted and non-targeted CVEs vs. total CVEs per vendor.

The age distribution shows targeted vulnerabilities are slightly younger, with a mean of 1,393 days and median of 1,358 days, compared to non-targeted vulnerabilities with a mean of 1,432 days and median of 1,410 days.

While the age of a vulnerability may influence its likelihood of being targeted, there is no significant difference in the average age between targeted and non-targeted CVEs. Despite slight variations, both categories exhibit a wide range of ages, indicating that age alone may is not clear factor for targeting within the IoT vulnerability landscape.

References. The results presented in Table 4.2 shows the distribution of all 18 reference labels across IoT vulnerabilities. Both targeted and non-targeted CVEs share the top five labels, but these labels appear more frequently in targeted CVEs, except for "Vendor Advisory", which shows only a slight increase. This suggests that targeted vulnerabilities receive more vendor attention.

This observation is further emphasized by the fact that the "Patch" label appears twice as often in targeted CVEs, and "Exploit" label is present in 38.06% of targeted CVEs compared to 18.65% in non-targeted CVEs. Causality might be reversed here: because these CVEs are targeted in the wild, vendors might feel more pressure to develop patches and the attacks might trigger the release of public PoC exploit code available. This can only be resolved by looking at whether a label predates the attacks or the other way around.

Other labels, like "US Gov. Resource", "Mailing List", "Mitigation", "Release Notes", and "Issue Tracking", are more common in targeted CVEs, though many labels show little difference compared to non-targeted ones. These commonalities suggest many similarities across both targeted and non-targeted CVEs within the IoT landscape. A final difference to observe the average count of labels per CVE for targeted versus non-targeted vulnerabilities. For targeted CVEs, this stands at 3, compared to 2.1 for non-targeted CVEs. Both groups share the same range, with a maximum of 10 labels per CVE and a minimum

of 1. This supports the idea that the number of references plays a role in exploitation likelihood, aligning with EPSS research [137].

4.5.2. EXPLOITABILITY FEATURES

We looked at the presence of PoC exploits to see if it can explain attackers' decision on targeting certain vulnerabilities. We found public PoC exploits for only 890 IoT vulnerabilities out of the total 23,373. This aligns with earlier work which found that around 4% of vulnerabilities get associated public exploits within a year [128].

Only 169 (18.9%) of the PoC exploits were associated with targeted CVEs, while the majority 721 (81%) pertained to non-targeted CVEs. This means that out of the 848 targeted vulnerabilities, only 19.9% have public PoC exploits. This reinforces findings from prior research indicating that the existence of PoC exploits is not the main driver of attacker's selection of vulnerabilities. It seems that most develop their own exploit code or at least get it from another source than PoC exploits being published [10, 11].

Table 4.2: Summary of total reference labels in targeted and non-targeted vulnerability.

Reference label	Non-targeted	Targeted
Vendor Advisory	15,842 (71.63%)	554 (66.51%)
Third Party Advisory	11,822 (53.45%)	608 (72.99%)
VDB Entry	5,566 (25.17%)	299 (35.89%)
Exploit	4,124 (18.65%)	317 (38.06%)
Patch	2,907 (13.14%)	214 (25.69%)
US Gov. Resource	1,693 (7.65%)	81 (9.72%)
Mailing List	1,265 (5.72%)	146 (17.53%)
Mitigation	878 (3.97%)	65 (7.80%)
Release Notes	756 (3.42%)	67 (8.04%)
Broken Link	670 (3.03%)	39 (4.68%)
Technical Description	528 (2.39%)	21 (2.52%)
Issue Tracking	421 (1.90%)	74 (8.88%)
Product	251 (1.13%)	13 (1.56%)
Permissions Required	232 (1.05%)	8 (0.96%)
Not Applicable	154 (0.70%)	19 (2.28%)
Press/Media Coverage	21 (0.09%)	7 (0.84%)
URL Repurposed	4 (0.02%)	2 (0.24%)
Tool Signature	2 (0.01%)	1 (0.12%)

4.6. HACKING COMMUNITY

Analyzing posts from the CrimeBB dataset, we identified discussions about IoT vulnerabilities and their potential exploitation within the hacking forums. Our analysis uncovered over 25k posts discussing 699 unique IoT vulnerabilities, as summarized in Table 4.3.

We observed that the discussions that mention IoT vulnerabilities within these underground hacking forums primarily focused on a wide range of exploit-related subjects. Notably, there was a substantial presence of inquiries regarding PoC exploits for certain

Table 4.3: Summary of forums discussed IoT vulnerabilities.

Forum	#Boards	#Threads	#Posts			#Members			#CVEs	
			Specific	General	Total	Specific	General	Total	Targeted	Total
antichat	4	67	0	19202	19202	0	2287	2287	60	135
blackhatworld	2	3	0	11	11	0	9	9	4	5
breached	10	21	158	817	975	151	596	747	13	16
cracked	1	1	47	0	47	47	0	47	1	1
dread	1	1	0	2	2	0	2	2	4	4
elhacker	7	71	10	120	130	6	35	41	45	139
forum-team	3	4	0	9	9	0	4	4	3	9
freehacks	1	1	0	2	2	0	2	2	4	5
garage-for-hackers	1	2	1	1	2	1	1	2	2	2
greysec	2	4	3	41	44	2	20	22	2	4
hack-forums	25	73	55	1021	1076	37	472	509	38	78
ifud	1	1	0	9	9	0	8	8	1	2
kernelmode	1	2	0	13	13	0	6	6	0	2
lolzteam	5	8	0	59	59	0	41	41	7	15
offensive-community	2	2	1	1	2	1	1	2	2	3
probiv	6	59	0	342	342	0	26	26	27	99
raidforums	8	20	11	788	799	10	529	539	18	31
runion	2	2	0	224	224	0	93	93	3	4
safe-sky-hacks	1	1	0	1	1	0	1	1	1	1
torum	4	5	0	16	16	0	11	11	9	12
underc0de	6	163	8	172	180	5	23	28	121	289
unknowncheats	3	6	5	100	105	4	61	65	4	5
v3rmillion	1	2	0	5	5	0	5	5	0	2
xss-forum	24	215	154	2340	2494	81	599	680	103	255
zismo	1	1	0	33	33	0	18	18	1	1
Total (unique)	93	735	453	25308	25760	345	4824	5101	699	222

IoT vulnerabilities, as well as requests for help or hiring to exploit such vulnerabilities with marketplace sections dedicated to buying, selling, or exchanging hacking-related products and services. Other discussions involved shared exploit kits, scripts, and tutorials. The forums also featured discussions on security advisories and attack news, including vulnerabilities utilized in zero-day exploits and ransomware attacks. It is evident that these forums serve as significant hubs for discussing IoT vulnerabilities and potential exploits.

Table 4.4: Distribution of hacking community features per IoT CVE, split between targeted CVEs/non-targeted CVEs.

	#Forum	#Boards	#Threads	#Posts			#Members		
				Total	Specific	General	Total	Specific	General
Average	20.57/32.3	5.59/10.75	1.5/2.25	706/660	16/3	690/658	153.24/93.48	12/2.08	143.24/91.8
Min.	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
Max.	121/168	117/158	12/26	12,828/14,519	140/25	12,828/14,519	1,646/1,444	133/18	1,646/1,444

4.6.1. IOT VULNERABILITIES IN HACKING FORUMS

We used post content to identify IoT vulnerabilities discussed in CrimeBB using their CVE-ID (see Section 4.4.2). These discussions covered 699 unique IoT vulnerabilities, of which 222 were targeted. These vulnerabilities were discussed in 25 different forums, as

detailed in Table 4.3. In total, 93 boards, 735 threads, 25,760 posts, and 5,101 members were involved in these discussions.

We found 93 different boards discussing IoT CVEs. Interestingly, while “hack-forums” was not the forum with the highest number of discussed IoT CVEs, it was the forum with the highest number of 25 different boards, suggesting a significant diversity of IoT vulnerabilities discussed across various boards compared to other forums. Following closely behind is the “xss-forum” with nearly a similar number of boards, totaling 24.

The predominant IoT vulnerability identified across 16 boards was CVE-2016-5195, a targeted vulnerability also known as Dirty COW, a Linux kernel vulnerability allowing attackers to gain write access to read-only memory mappings, potentially leading to privilege escalation.

The variation in thread activity highlights the dynamic nature of discussions surrounding IoT vulnerabilities. For instance, the “xss-forum” forum exhibited the highest number with 215 unique threads, indicating a significant focus on IoT vulnerabilities within that community.

Forums with the highest volume of posts do not always feature the most discussions on IoT vulnerabilities, e.g., despite “hack-forums" including over 42 million posts, only 1,021 were related to IoT vulnerabilities, with just 55 focusing on specific CVE(s) (Table B.1). In fact, the “antichat” forum led with 19,202 posts on IoT-related CVEs, with a total volume of posts exceeding 2.6 million. “safe-sky-hacks” had only one post discussing an IoT CVE. CVE-2018-10561 (a vulnerability in the DSL-2640B router) garnered the highest total posts at 8,078, in both specific and general discussions, while CVE-2022-40684 had the most specific posts at 115, noted as targeted.

But when looking at forums discussing specific IoT CVEs in Table 4.3, only 11 out of the 25 forums exhibited such discussions. Among these, the “breached” forum showed the highest number of posts, with a total of 158 posts. Interestingly, while general discussions were prevalent across all forums except for one, namely “cracked” where all 47 posts were exclusively linked to specific IoT CVE, it’s noteworthy that “antichat” forum recorded the highest number of posts addressing IoT CVEs within general threads. Nevertheless, despite that, “antichat” forum did not feature any posts within threads focusing on specific IoT vulnerabilities.

We analyzed forum posts mentioning IoT vulnerability using their CVE-IDs to shed light on the extent of discussions. The “xss-forum” had the most posts, with CVE-2020-1472 (critical vulnerability in Microsoft’s Netlogon) and CVE-2018-13379 (critical vulnerability found in Fortinet FortiOS SSL VPN) being mentioned 27 times each. Both were observed as targeted in our dataset.

A total of 5,101 individuals participated in threads discussing IoT vulnerabilities using their CVE-IDs. Of these, 345 were involved in threads dedicated to specific IoT CVEs, while the remaining 4,756 contributed to general discussions. This correlates the volume of posts with the number of participants, with forums having more posts also attracting more members. For instance, “xss-forum” led in both post volume and member engagement.

4.6.2. TARGETED VS NON-TARGETED HACKING DISCUSSIONS

The distribution across forums where IoT vulnerability discussions took place made it apparent that among the 25 forums, 13 of them predominantly focused on targeted

CVEs over non-targeted ones (see Table 4.3). Conversely, 10 forums showcased a greater inclination toward non-targeted vulnerabilities while the remaining two forums exhibited an equal number of posts discussing both targeted and non-targeted vulnerabilities. Among these forums, "underc0de" emerges as the most active in discussing IoT vulnerabilities, with a total of 289 IoT vulnerabilities, including the highest number of targeted vulnerabilities at 121. That was also reflected in the number of threads. Regarding the most observed IoT vulnerabilities across forums, CVE-2017-5754 (`Meltdown`, hardware vulnerability affecting modern microprocessors [84, 198]) and CVE-2017-5715 (`Spectre Variant 2`, vulnerability that exploits speculative execution in modern microprocessors [84, 198]) were the most prevalent, appearing in 9 forums and were also identified as targeted vulnerabilities.

We measured the number of forums, boards, threads, posts and members per IoT vulnerability in Table 4.4. Overall, targeted vulnerabilities show higher averages compared to non-targeted ones across all features in the hacking community. For instance, the average number of forums per targeted IoT vulnerability is 20.57, which decreases to 5.59 at the board level and further drops to 1.5 at the thread level. Conversely, the average number of forums, boards and threads per non-targeted CVE are 32.3, 10.75, and 2.25, respectively. Similarly, targeted vulnerabilities received more posts on average, 706 compared to 660 for non-targeted ones, with specific targeted vulnerabilities garnering more discussion than general ones with 16 posts compared to 3 for non-targeted ones. Moreover, discussions on targeted vulnerabilities involved 153.24 members on average, which drops to 93.48 for non-targeted ones, with specific targeted vulnerabilities attracting more participants than general discussions, involving 12 members, which is around three times the number of participants engaged in non-targeted discussions that stands at 2.08 members.

4.6.3. TIME TO EXPLOIT IN THE WILD

We measure "Time to Exploit" as the number of days between the first discussion of an IoT-related CVE in hacking forums and its first observed exploitation in the wild. To reduce false positives, we exclude posts that did not explicitly mention the CVE-ID in the same thread. However, Figure 4.7 presents both cases for the Top 10 IoT CVEs with the longest and shortest time-to-exploit, where ● highlights posts mentioning CVE-IDs, and ● highlights posts in the same thread. Due to missing or unclear exploit dates in some sources, we relied on five datasets: AttackerKB, VulnCheck KEV, Google Zero Project, IoTPOT, and X-Pot. Among 222 discussed and targeted IoT CVEs, 159 had available exploit dates. Of these, 78 (49.05%) were discussed before or on the same day as their first exploitation, while 81 were discussed afterward. The time gap between discussion and exploitation varies, with an average time-to-exploit of 482.6 days and median of 383.5 days. The shortest time-to-exploit is zero days, exploitation occurred on the same day as forum discussion, were observed in seven cases all for CVEs published between 2020 and 2022. The longest time-to-exploit is 2,240 days (CVE-2016-8735, an Apache Tomcat vulnerability). Notably, CVEs with the shortest time-to-exploit had just 1-5 posts mentioning their CVE-ID, while those with the longest delays were mentioned only once, except for three cases appearing up to four times. These findings underscore the importance of monitoring hacking forums for early warnings on IoT vulnerabilities, allowing organizations to identify and mitigate threats before widespread exploitation.

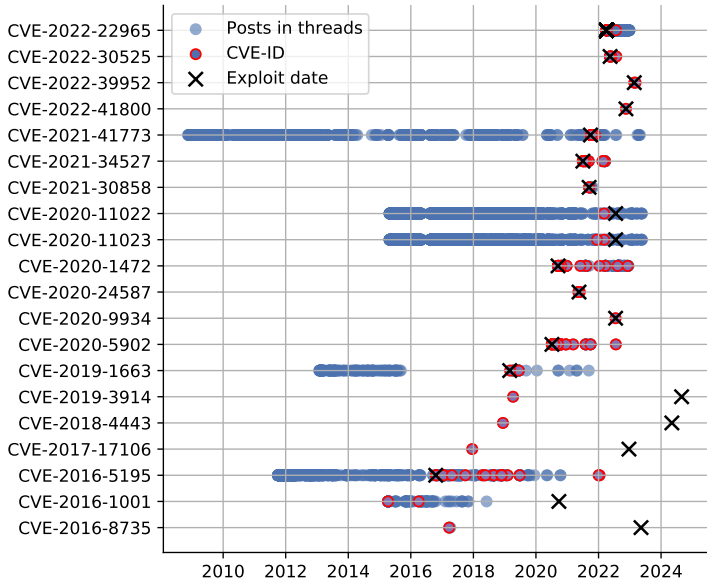


Figure 4.7: Top 10 IoT CVEs for both the longest and shortest time-to-exploit after post discussing CVE-ID, number of days between the first discussion mentioning the CVE-ID and its first observed exploitation in the wild, ordered by CVE publication year. The ● highlights posts mentioning CVE-IDs.

4.7. EXPLAINING EXPLOITATION

4.7.1. MODEL AND FEATURE SELECTION

In the first version of EPSS, Jacobs et al. use logistic regression to build an exploit prediction system [136], showing that the features used share a somewhat linear relationship. EPSSv2 and EPSSv3 adopt the more complex XGBoost model [61] to capture non-linear relationships [137], sacrificing interpretability for improved predictive accuracy.

Our goal is to analyze and explain the factors influencing whether an IoT vulnerability is targeted by attackers, without relying on time-based patterns. To avoid added complexities, we use cross-sectional data, capturing targeting information at any given point. By choosing a logistic regression model, we can easily make predictions while also clearly identifying which features contribute to those predictions, allowing us to both predict vulnerability targeting and explain the significance of the influencing factors.

Upon further investigation of our potential feature set, we find a collection of linearly dependent features. Including such features in a model can introduce multicollinearity into the eventual model, undermining the statistical significance of the other features [7], leading to an overfitted model. We exclude the eight linearly dependent features from the dataset. These dependencies arose structurally from how NVD organizes vulnerability information. For example, CVSS v3.1 includes both an overall base score and its component metrics (Confidentiality, Integrity, Availability impacts), where the base score is mathematically derived from these components, creating perfect collinearity. Including such redundant features would artificially inflate their apparent importance and violate the independence assumptions of logistic regression. That leaves a set of 365 features

with which to fit the model.

Jacobs et al. [136] use elastic net regularization (among other methods) to condense their collection of 3,587 features down to 16 though. We aim to reproduce as much as the original methodology to maintain a level of comparability, and to be able to accurately judge the value of the additional hacking community features. Thus, we similarly use elastic net regularization for our feature selection.

We first split the dataset features into two groups: (1) hacking community features and (2) all other features. We use two feature selection methods, both applying elastic net regularization to select the most important features in (2). The first method trains a model using only the regularized features from (2), serving as a control. The second method adds these steps: (i) perform PCA on the hacking features, (ii) select the principal component (PC) with the highest variance, (iii) add this PC to the dataset, and (iv) train the model with the updated dataset.

PCA helps summarize many correlated features [40] while retaining key information, making it useful for our analysis. We select the most significant PC, labeled **Engagement**, to simplify further analysis. This PC accounts for 64% of the variance in hacking community features. As in [136], we tune the hyperparameters by performing a grid search for the α and λ that minimize the Bayesian Information Criteria (BIC). We find the following optimal values: $\alpha = 0.3$ and $\lambda = 0.005$. The regularization selects 93 features (see Figure B.1).

4.7.2. MODEL EVALUATION

We split our dataset into a training and testing set using a split of 0.33. After regularization and PCA processing, we fit two logistic regression models using the training set according to the different feature selection approaches described in subsection 4.7.1. For each approach, we then use each model to estimate the probability that an IoT vulnerability in the test set will be targeted by malicious actors. To evaluate model performance, we compute several metrics: the Brier score and McFadden's pseudo R^2 (R^2_{MF}) score (see Table 4.5 for the full list of metrics). We also calculate the models' Brier score andR^2_{MF} score performance improvement (PI) as a percentage increase (see [271].)

The Brier score measures the mean square error of probabilistic predictions, i.e., how close the predictions are to the actual value. The R^2_{MF} score—being a normalized version of the MSE—is scale-independent and is more concerned with the general quality of the fit of the model. The closer the Brier score is to 0, the better. Contrarily, the closer the R^2_{MF} score is to 1, the better.

We compare the metrics with the EPSS scores we collected using EPSS API [112] for all vulnerabilities in our dataset except for 88, of which two were observed as targeted. This is because these CVEs are stored in the NVD but labeled as "REJECT" or "RESERVED" in the CVE List and do not appear in search results. EPSS predicts the likelihood of exploitation within a 30-day window [137], so we collect EPSS scores for our set of vulnerabilities for 1 September 2023, 30 days before the last day of data collection (1 October 2023). This allows for a consistent evaluation and comparison with EPSS.

The poor performance of EPSS reiterates its unsuitability for IoT-specific vulnerabilities. Both the control and Engagement models, despite using fewer features, outperformed EPSS on the same set of vulnerabilities. For example, the Brier score for both

Table 4.5: Brier score and R^2_{MF} scores for all models.

Model	Brier score	R^2_{MF}
No Engagement feature	0.031	0.183
Include Engagement feature	0.028	0.222
EPSS scores	0.035	-0.157

of our models, performed similarly, with scores of 0.031 and 0.029, which represent an improvement over EPSS's score of 0.035, though the difference is small in absolute terms. Furthermore, when comparing the prediction accuracy of our second model with the Engagement feature to EPSS using their Brier scores, we obtained a 17% performance improvement (PI) over EPSS.

Not only did the model with the Engagement feature perform best in Brier score, but also had a 21% PI in the R^2_{MF} score when comparing it with the control model. This highlights the value of extracting insights from the hacking community (see Figure B.2).

While Brier scores offer insight into prediction accuracy, the R^2_{MF} scores more clearly differentiate the models. The control model had an R^2_{MF} score of 0.183, whereas the model with the Engagement feature achieved 0.222, reflecting the 21% PI. This suggests that while both models performed similarly in terms of individual prediction errors, the Engagement model was better at explaining the variance in targeted IoT vulnerabilities. Incorporating hacking community data greatly improves the model's predictive power, emphasizing the value of monitoring such activity. In contrast, the EPSS model yielded an R^2_{MF} score of -0.157, indicating it performed worse than a mean-based prediction. This further demonstrates EPSS's limitations for IoT vulnerabilities across both metrics, as it struggles to capture the unique nuances of IoT-related threats.

4.7.3. RISK ASSESSMENT USING HACKING FORUM

To evaluate the impact of hacking forum discussions on the risk assessment of vulnerabilities before their first exploitation, we conducted an experiment comparing model predictions in two scenarios. First, with the first-exploitation date in hand for the 78 CVEs that were both discussed in hacking forums and targeted before or on the same day as their first observed exploitation (see subsection 4.6.3), we applied the trained model from subsection 4.7.1 to this subset. To specifically assess the impact of discussions prior to first exploitation, we recalculated and aggregated forum activity data only up to their first exploitation date. Since all CVEs in this subset were targeted, the predicted probability of targeting should ideally be close to 1. The mean prediction for this subset was 0.57.

To assess whether hacking forum features significantly influence targeting predictions, we compared these results to a case where the same 78 CVEs were assumed to be targeted but had no prior forum discussions. So, we excluded all hacking forum-related features while keeping all other characteristics unchanged, assuming no prior forum discussions. After applying the same data processing steps, we predicted the targeted label for this second scenario. The mean prediction dropped to 0.26.

The observed reduction in predicted targeting probability of the second scenario confirms that removing forum discussions significantly reduced the model's confidence

in identifying these vulnerabilities as targeted and reinforces the role of hacking forum discussions in enhancing pre-exploitation risk assessment. In addition, we computed "Thiel's U" [41] for these features to help identify their predictive power. Since "Thiel's U" closer to 1 indicates a feature almost perfectly explains the target, we found that BaseScore with a Thiel's U of 0.995, followed by count of references at 0.984, Engagement at 0.978, and number of vendors per CVE at 0.977, have the highest predictive power.

4.7.4. MODEL INTERPRETATION

FEATURES SIGNIFICANCE.

The fitted model includes 38 features with < 0.05, as shown in Figure B.2. Each coefficient indicates the feature's influence on the likelihood of an IoT vulnerability being targeted. Among the 38 features linked to vulnerability, exploitability, and hacking community factors, all but five increase the likelihood of an IoT vulnerability being targeted.

Significance of vulnerability features. Of the 38 features, 36 were derived from vulnerability data, including 1 CVSS metric, 22 vendors, 8 CWE categories, and 5 from references features.

CVSSv3 metrics: The only CVSS metric with a significant yet negative impact on targeting is "Privileges Required: Low" (PR:L); a one-unit increase reduces the likelihood of an IoT vulnerability being targeted by 0.56 times, perhaps because attackers prioritize vulnerabilities requiring no authentication for broader impact or favor those with higher privileges for deeper system access.

Vendor: This represent the most influential feature, with 22 out of 38 features linked to targeting likelihood. All vendors, except "Huawei" and "FreeBSD," increase the likelihood of an IoT vulnerability being targeted by 1.8 to 34.6 times, with "Microchip" having the highest impact. In contrast, "Huawei" and "FreeBSD" decrease targeting likelihood by 0.02 and 0.03 times, respectively. The sharp decline associated with "Huawei" may be due to only one targeted vulnerability, CVE-2017-17215, compared to 1,400 non-targeted ones in our dataset.

While this vulnerability, allowing authenticated attackers to remotely exploit Huawei HG532 routers, was the most frequently and consistently targeted IoT vulnerability, this pattern aligns with findings in both Chapter 2 and Chapter 3, as well as prior research [19, 77, 78, 134]. Additionally, the high standard error suggests some uncertainty in the estimate (see Table B.2 in the Appendix).

CWE: All eight CWE categories increase the likelihood of an IoT vulnerability being targeted, except for CWE-352 (Cross-Site Request Forgery), which decreases it by 0.23 times. The remaining CWEs raise targeting likelihood between 1.75 and 12.29 times, with CWE-116 (Improper Encoding or Escaping of Output) having the highest impact, increasing it by 12.29 times per unit increase.

References: We identified five reference-related features assigned to CVEs by NVD, all positively correlated with a 1.43 to 2.11-fold increase in the likelihood of an IoT vulnerability being targeted, except for "Third Party Advisory", which decreases it by 0.68 times. "Issue Tracking" has the highest impact, increasing the likelihood of targeting by 2.11 times.

Significance of exploitability features. *PoC availability:* Although the availability of PoC exploits alone do not fully explain IoT targeting, the presence of PoC exploits

alongside other features make IoT vulnerabilities 3.57 times more likely to be targeted compared to the ones without PoC exploits.

Significance of Engagement. This feature captures whether an IoT vulnerability was discussed in underground hacking forums, including details like the number of forums, boards, threads, posts, and members involved. For example, when community members actively engage in discussions about specific vulnerabilities, the likelihood of those vulnerabilities being targeted in the wild increases by 1.27 times. This indicates that increased forum discussions lead to a higher probability of malicious targeting. These findings align with previous results in subsection 4.7.2, where including forums data improved the model's variance explanation by 21% and its targeting prediction of IoT vulnerabilities by 17% over EPSS.

Table 4.6: Top 10 SHAP features: Our model with Engagement vs. EPSS

Model with Eng.	EPSS rank	EPSS model	Model with Eng. rank
CVSS:3.1/Scored	#28	CVE: Count of Ref.	#8
CVE: Age of CVE	#5	Tag: Remote	-
Engagement	-	Exploit: Exploit DB	#5
Ref: VDB Entry	#14	Tag: Code Exec.	-
CVSS:3.1/I:H	#24	Vendor: Microsoft	#19
Exploit: Exploit DB	#4	CVSS:3.1/PR:N	-
CVSS:3.1/A:H	#7	CVE: Age of CVE	#2
CVE: Count of Ref.	#1	CVSS:3.1/AV:N	#9
CVSS:3.1/AV:N	#7	CVE: Age of CVE	#2
Vendor: Intel	-	CVSS:3.1/C:H	#14

FEATURES IMPORTANCE.

The SHAP value distribution highlights feature influence, with the top 10 in our model ranging from 0.017 to 0.002. The highest SHAP value is for the CVSS base score, followed by CVE Age (0.013) and Engagement (0.008), while the lowest is for the vendor "Intel" (see Figure B.3). Notably, the Engagement feature underscores the role of attacker-driven insights. In contrast, EPSS top features have SHAP values between 0.42–0.09, led by the count of references, Remote tag (0.34), and Code Execution (0.29), with CVSS metric (C:H) ranking lowest. We compare the feature importance derived from our model including the Engagement feature, with that from EPSS using SHAP values, as it's the only public information provided by EPSS [137]. This comparison allows us to assess the relative significance of various features in predicting exploitation, contrasting general vulnerabilities in EPSS with our model's focus on IoT vulnerabilities. Our model and EPSS exhibit considerable overlap in their top 10 influential features, with 8 of our top 10 features appearing in EPSS and 7 of EPSS's features present in our model (see Table 4.6). This alignment underscores shared key predictors, such as Age of CVE, exploit availability (Exploit-DB), CVSS metrics, reference label "VDB Entry", and vendor "Intel". For example, the top predictor in our model is the CVSS base score but it ranks lower in EPSS (#28), while the count of references is EPSS top feature yet ranks lower for IoT in our model (#8). Notably, hacker forum engagement ranks as the 3rd most influential feature in our

model but is absent in EPSS, emphasizing the model with Engagement's stronger focus on attacker community discussions in risk assessment.

4.8. DISCUSSION

EPSS was trained on a dataset dominated by general-purpose computing vulnerabilities, as they make up the bulk of the total vulnerability population. However, IoT vulnerabilities differ in important ways that make them harder to predict using these same general features.

For instance, while our model and EPSS share similarities in their top 10 most influential features (see Table 4.6), only 11 of the top 30 most influential features values overlapped when considering the full list in Figure B.3. In terms of the differences, we find more vendor-related features dominating our model, with 10 appearing in the top 30, compared to only two in EPSS. This underscores the role vendors play in IoT exploitability, identifying Zyxel and Sonicwall among the most frequently exploited vendors, consistent with Chapter 2, Chapter 3, and [19]. Similarly, we found that two categories of CWE weaknesses correlate with attacker targeting for IoT vulnerabilities, yet EPSS does not rank any CWE-related features in its top 30.

These discrepancies highlight the unique and complex nature of IoT vulnerabilities. Spring [229] argued that EPSS often assigns low scores to IoT vulnerabilities that are actively exploited, such as those listed in CISA's (KEV) catalog. For example, CVE-2017-17215 (Huawei HG532 routers) remained a frequent target yet received a near-zero EPSS score (0.05) for exploitation within 30 days as of May 2021. It took three years for the score to rise to 0.96 by April 2023—delayed recognition that does not align with real-world attacker behavior [19, 77, 78, 134]. Prior work emphasizes the importance of IoT-specific features, such as device types [19, 34, 43, 225] and internet exposure levels [34, 195], which significantly influence attacker targeting [19, 34, 166], which was also shown in the previous chapters. While we acknowledge that collecting such data is labor-intensive and difficult to scale for integration into automated models like EPSS, these findings suggest IoT vulnerabilities may be exploited differently from general ones.

Additionally, EPSS relies on social media discussions, such as Twitter/X, to detect exploitation, whereas research suggests that dark web discussions are more predictive [16]. To address this gap of improving the prediction for IoT vulnerabilities, we incorporated hacking community discussions into our model, enhancing the model's ability to explain variance by 21%. Not only did this inclusion improve the model's explanatory power, but it also increased its predictive accuracy by 17% over EPSS [137]. To further assess the impact of hacking forum discussions on pre-exploitation risk assessment, we found that including forum data increased the mean predicted targeting probability for these CVEs to 0.57, whereas excluding forum-related features reduced it to 0.26.

These cases accounted for 49% of discussed CVEs with known exploit dates, averaging a 482.6-day 'time-to-exploit". Notably, 9% were exploited on the same day they were first discussed. In contrast, EPSS did not rank social media feeds among its top 30 predictors, highlighting the importance of hacking forum discussions in identifying targeted IoT vulnerabilities.

This suggests that IoT vulnerabilities should likely be treated separately in prediction models, as the factors influencing their exploitation differ significantly from those of

general vulnerabilities. This provides an area for future research and model development, particularly in the realm of IoT security, that might account for IoT-specific features.

4.9. LIMITATIONS

Several limitations exist within our methodology. Our approach solely relied on CVE identifiers for identifying IoT vulnerabilities, both those targeted in the wild and those discussed in underground forums. This may exclude vulnerabilities not assigned CVE-IDs, potentially overlooking significant threats.

Additionally, the automated search methods employed using VARIoT dataset to identify IoT-related vulnerabilities might have resulted in missed or irrelevant entries. In underground forums, there is a risk of input data manipulation by malicious actors, who may discuss random IoT vulnerabilities to disrupt classification.

Furthermore, the lack or ambiguity of exploit capture dates for targeted IoT vulnerabilities in some datasets, such as CISA and VirusTotal, poses challenges in understanding the timing of targeting activities.

Finally, as seen in Chapter 2 and Chapter 3 that IoT-specific features like device type [19, 20, 43] or install base [34, 195] requires manual data collection, which is impractical given the scale of our dataset of over 23k CVEs.

Instead, this study focuses on supporting vendors and users in evaluating IoT vulnerabilities by leveraging readily available online information.

4.10. CONCLUSION

We analyzed factors influencing IoT vulnerability targeting using features like CVSS severity, CWE categories, vendors, external references, vulnerability age, and PoC exploits derived from EPSS, along with new insights from hacking forums. For over 23k IoT-related CVEs, we leveraged vulnerability features, PoC availability, and discussions from more than 25k posts across 25 hacking forums. We identified 38 features with p-value < 0.05 that affected attackers' targeting choices of IoT vulnerabilities. Our model, utilizing hacking forum data, showed a 21% performance improvement in McFadden's pseudo R^2 score. Using fewer features than EPSS, our Brier score prediction accuracy increased by 17% compared to EPSS, which showed that existing state-of-the-art methods often failed to capture the complexities of IoT threats, and integrating insights into attacker behavior from online sources enhanced the prediction of IoT vulnerability targeting.

5

The Role of PoC Exploit Developers in Exploit Availability

The cybersecurity landscape faces an escalating challenge as proof-of-concept (PoC) exploits transition from theoretical demonstrations to weaponized attacks within mere minutes of public disclosure. While existing research has extensively documented the temporal dynamics and malicious deployment of PoC exploits, a critical gap remains in understanding the human factors underlying their creation. Through semi-structured interviews with 16 PoC developers across diverse global regions, we apply Expectancy-Value Theory to reveal that PoC development operates as a complex motivational ecosystem where technical confidence, value assessments, and risk calculations intersect within competing dual-use tensions. Our findings demonstrate that PoC development exists on a sophisticated continuum from simple crash demonstrations to fully weaponized exploits, with developers' choices shaped by multifaceted calculus rather than binary ethical frameworks. We identify three novel theoretical extensions: dual-use moral reasoning enabling responsibility externalization, dynamic value assessment where vendor behavior continuously reshapes disclosure decisions, and identity navigation requiring ongoing negotiation between ethical research commitments and technical mastery identities. Vendor responsiveness, community dynamics, and cultural-legal constraints create contextual forces that significantly influence disclosure strategies. We show that developers adopt risk-mitigation approaches when navigating tensions between security improvement and potential misuse, challenging prevailing binary conceptualizations of "responsible" versus "irresponsible" disclosure.

This chapter has been published as: **Arwa Abdulkarim Al Alsadi**, Lorenz Kustosch, Lamya Alowain, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. "Oh, what people would do with my knife?" Navigating the Dual-Use Dilemma in PoC Exploit Development, Disclosure, and Community Dynamics. *Under submission to USENIX Security '26*

5.1. INTRODUCTION

The rapid weaponization of vulnerabilities poses a critical challenge for cybersecurity. Proof-of-Concept (PoC) exploits, initially requiring hours or even days to translate into active threats, are now being leveraged in alarmingly short timeframes. Early reports indicated an average of one hour and seven minutes for PoC exploits to be weaponized after public disclosure [81], but recent evidence demonstrates a dramatic reduction, with instances of exploitation occurring within a mere 22 minutes of disclosure [241].

While existing research has examined the malicious use of public PoC exploits [81, 241, 270] and the rapid adoption of exploits by attackers, these studies primarily rely on observational data, focusing on the "what" and "when" of deployment. Less attention has been paid to the "who" and "why"—the motivations, processes, and decision-making of PoC developers creating and disseminating PoC. This leaves a significant gap in our understanding of the PoC lifecycle, specifically the factors influencing PoC creation, reporting, and publication decisions. Understanding this phenomenon requires examining not just the technical artifacts themselves, but the complex motivational landscape that drives their creation. To address this challenge, we apply Expectancy-Value Theory (EVT) as our theoretical lens, which provides a systematic framework for understanding how PoC developers navigate the inherent tensions between beneficial security research and potential harmful applications.

Existing research reveals three important gaps that our study addresses. First, no previous study has applied motivational frameworks to the decision-making processes of PoC developers. Second, there is a lack of comprehensive studies examining the full lifecycle of PoC including creation, validation, reporting, and publication as well as the definitions, practices, and contextual factors that shape each stage. Third, the informal ecosystems in which PoC circulate, often characterized by pseudonymity and legal uncertainty, have been largely ignored because most research focuses on formal institutional disclosure channels or observational studies.

We fill these gaps by extending EVT with three key contributions: (i) dual-use moral reasoning, which enables PoC developers to externalize responsibility for potential misuse; (ii) dynamic value assessment, where vendor behavior, disclosure outcomes, and community feedback continuously reshape PoC developers' subjective task values in real time; and (iii) identity navigation, emphasizing the ongoing negotiation between ethical research commitments and technical mastery identities. Additionally, we incorporate contextual factors outside traditional EVT, including vendor relationship dynamics, community trust and anonymity patterns, and cultural and legal constraints, which critically influence decision-making but fall beyond standard motivational constructs. We then map these extended constructs to three key research questions aligned with EVT components:

RQ1: What definitions and perceptions do PoC developers hold regarding PoC exploits?

RQ2: Which technical, ethical, and economic factors influence PoC creation, validation, and reporting?

RQ3: How do PoC developers navigate dual-use decision points in disclosure and publication strategies?

To address these research questions, we conducted semi-structured interviews with 16 PoC exploit developers from diverse regions and backgrounds. This qualitative approach yielded rich insights into their motivations, target selection, tooling decisions, reporting practices, and concerns about misuse. Our EVT-informed analysis reveals PoC development as a dynamic negotiation among technical confidence, multifaceted value assessments, and cost considerations, all situated within complex dual-use and contextual environments. This framework enabled us to make the following contributions:

- We deliver the first comprehensive analysis of the PoC development lifecycle from developers' perspectives, showing it spans a spectrum from simple crashes to fully weaponized exploits, guided by a complex motivational calculus beyond mere technical skill or ethics.
- We extend EVT to dual-use technology contexts, introducing novel constructs of dual-use moral reasoning, dynamic value assessment, and identity navigation that explain how PoC developers navigate competing beneficial and harmful applications of their work.
- We identify and categorize the key factors that influence the PoC development process across five stages (vulnerability identification, PoC development, validation, reporting, and publication), demonstrating that personal motivation, vendor response, and past experience play crucial roles in shaping dual-use decisions.
- We identify that differences between offensive-minded and defensive-minded practitioners significantly influence PoC development and disclosure practices, revealing distinct motivational profiles that shape the security landscape through different risk tolerance and value prioritization.

5.2. RELATED WORK

Studies on Security Practices and Hacker Motivations. A substantial body of research uses qualitative and interview-based methods to understand the practices, motivations, and challenges of security professionals, ethical hackers, and testers. Votipka et al. [254] compared white-hat hackers and software testers, highlighting differences in knowledge, access, and motivation, and showing how PoC exploits are used to validate vulnerabilities when communication with PoC developers is absent. Chng et al. [63] provided a comprehensive framework of hacker types and strategies, while Piao et al. [204] revealed how bug bounty participants collaborate informally to enhance discovery. Alomar et al. [17] examined real-world disclosure practices, underscoring persistent challenges in vendor–researcher coordination.

Building on this, research has also investigated bug bounty programs and vulnerability disclosure channels. Studies highlight the economic incentives and dilemmas hackers face in navigating institutional programs [1, 29, 171]. Walshe and Simpson [259] assessed coordinated vulnerability disclosure (CVD) programs and proposed improvements to foster more trustful interactions. The *Psychology of Hackers* study [140] further examined the psychological drivers behind vulnerability discovery. Recently, Chaliasos et al. [57] evaluated automated tools in the DeFi ecosystem, finding that state-of-the-art analyzers

would have prevented only a small fraction of high-profile attacks, highlighting gaps between tool capabilities and practitioner needs.

On the maintainer side, a more recent work by Ayala et al. [33] investigated OSS maintainers listed in the GitHub Advisory Database, identifying supply chain mistrust and lack of automation as major challenges. They found that barriers to adopting platform security features include lack of awareness and perceptions of non-necessity. Notably, some maintainers continue to allow public vulnerability reporting despite previous vulnerabilities, highlighting gaps between security best practices and actual behavior.

Complementing this literature, ethnographic work has examined hacker communities as sites of identity and moral reasoning. Coleman's foundational study [68] and recent historical analyses of vulnerability practices [255] underscore how political, ethical, and social values shape hacking. These perspectives are essential for situating PoC exploit development as not only a technical task but also one embedded in contested moral economies and dual-use dilemmas.

PoC Exploits and Exploitation Timelines. Another line of work examines the lifecycle of PoC exploits and their weaponization in the wild. Nayak et al. [179] showed that some vulnerabilities are rapidly targeted while others remain dormant. Alrawi et al. [19] documented how IoT malware campaigns quickly integrate public exploits. More recent measurements confirm that adversaries may weaponize PoC within minutes of publication [81], underscoring the urgency of timely patching. Chapter 2 and Chapter 3 provided large-scale analyses of exploit timelines, showing that while many exploits appear quickly, others surface years later or even prior to CVE publication date, complicating assumptions about disclosure. Related historical analyses [128, 231] emphasize how exploit availability reshapes patching priorities and attacker decision-making.

Economics and Dual-Use Dilemmas. Economic perspectives on vulnerability exploitation highlight adversaries' cost–benefit reasoning. Allodi's work [9, 14] shows that attackers prioritize low-cost, high-impact vulnerabilities and adopt new exploits selectively. At the same time, dual-use dilemmas emerge as tools circulate across legitimate and malicious domains. Kang et al. [144] demonstrate how instruction-following LLMs amplify dual-use risks by enabling standard security attacks at scale. Silic [221] similarly analyzed the organizational dilemmas of adopting dual-use open source security tools, showing both defensive benefits and heightened risks. Ethical scholarship on dual-use has a long history in the biological sciences, where debates on governance emphasize the tension between advancing knowledge and mitigating misuse [167]. Our study extends this dual-use lens to the socio-technical practices of PoC exploit developers, who often operate under pseudonymity and legal uncertainty, in contrast to more institutionalized bug bounty ecosystems.

Motivational Theories in Security Behavior. Motivation theories from psychology provide additional explanatory power for security behavior. Expectancy-Value Theory (EVT) [90, 265] posits that task engagement depends on expectations of success ("Can I succeed at this task?") and the subjective value of the task ("Is the task worth doing?"), which includes intrinsic, attainment, and utility values, as well as costs. EVT has been applied in security contexts such as password adoption [141] and mobile identity protection [6]. Chen et al. [62] further demonstrated its usefulness in phishing interventions, revealing how expectancy and value factors shape employee engagement. Other domains

have also adopted EVT, such as studies on gender-related academic and occupational goals [203].

While EVT has not yet been applied to offensive security practices, our work extends the framework to PoC exploit development. We argue that EVT is well-suited to capture PoC developers' decision-making, as it directly addresses *why* individuals take on tasks, *what* values they assign to them, and *how* costs shape engagement. This complements economic and ethical accounts of dual-use by foregrounding the situated motivations of actors.

Finally, prior analyses of dual-use dilemmas at policy and organizational levels—such as work on economic analysis of dual-use research governance [87] and organizational study [221]—demonstrate the importance of expectancy–value reasoning beyond individual actors. Our study, by focusing on PoC exploit developers, fills a gap in connecting motivational theory to the dual-use challenges of PoC exploit circulation.

5.3. METHODOLOGY

We employed a qualitative research design using semi-structured interviews to explore the practices, motivations, and decision-making processes of individuals involved in PoC exploit development. We recruited 16 participants from diverse geographic and professional backgrounds through social media, professional networks, and snowball sampling. After piloting the interview protocol, we conducted remote interviews, transcribed and anonymized the data, and analyzed it using a hybrid inductive-deductive coding approach supported by qualitative analysis software. Inter-coder reliability was assessed to ensure consistency, and participants were invited to review their contributions for accuracy.

5.3.1. PARTICIPANT SCREENING

We began by distributing a registration form through various channels, including social media platforms, relevant email lists, and direct outreach to individuals known in the security community. This form collected information about potential participants' backgrounds and experience with PoC exploit development. A critical element of this form was to determine whether interested individuals met our pre-defined inclusion criteria. Specifically, we required that all participants have prior experience developing at least one proof-of-concept exploit from start to finish. Responses were carefully reviewed to filter out individuals who did not meet this essential criterion. In cases where clarity was needed, we contacted registrants directly to discuss their experience and confirm their suitability for the study before proceeding further. In total, 16 participants were selected and interviewed from a pool of 33 responses, with thematic saturation reached well within this sample size [59].

5.3.2. PARTICIPANT RECRUITMENT

Following the initial screening, we focused our recruitment efforts on the pool of individuals who had been deemed eligible based on their registration form responses and, where necessary, follow-up communications. This targeted recruitment involved sending personalized invitations to participate in the study.

The invitations provided detailed information about the study's purpose, scope, and requirements, as well as assurances of anonymity and confidentiality. Our outreach emphasized that participation was entirely voluntary and that participants could withdraw at any time. Through these focused efforts, we successfully recruited a total of 16 participants from the pool of screened candidates. These participants were based across a range of global regions, including North America, South America, Europe, Africa, and Asia, providing a diverse set of perspectives and experiences.

5.3.3. PILOT INTERVIEWS

Before the main study, three pilot interviews (one in-person, two via video) were conducted to test and refine the interview protocol. The lead researcher, with a second author—who has a background in psychology—present during the pilot sessions. The role of this second author was to observe the interview flow, evaluate the clarity of questions, ask follow-up questions when needed, and ensure the overall interview environment encouraged open and unbiased responses. Based on this feedback, the protocol was refined by clarifying ambiguous terms, reordering thematic sections for smoother flow, and rephrasing prompts to encourage deeper responses.

5.3.4. INTERVIEW PROCEDURE

We conducted semi-structured interviews via video or audio conferencing, lasting 45-60 minutes (two extended to approximately two hours). The lead researcher conducted all interviews except one, which was conducted in Spanish by a native-speaking co-author to accommodate the participant's language preference. This transcript was translated using the Argos Translate Python library [25], following established methodology [37]. The offline neural machine translation tool executed entirely on local resources to maintain GDPR compliance and data protection requirements.

Two interviews included third-party interpreters at participants' request to facilitate communication when participants struggled with English expression. Interpreters were strictly limited to translation and clarification roles, with clear instructions ensuring only participant perspectives were captured.

All participants provided informed consent prior to sessions, with study details reiterated at interview commencement. Interviews followed a structured format: warm-up questions covering background, experience, and motivations (Questions 1-8), followed by core questions on PoC development processes (Questions 10-16). The complete interview protocol is available in section C.1.

5.3.5. DATA PROCESSING AND ANALYSIS

All interviews were transcribed and anonymized by the lead author, which then were imported into `Atlas.ai` [30] for qualitative analysis. We employed iterative open-coding to analyze the interviews data [109]. We used a descriptive analysis approach to identify patterns in participants' responses and summarize common themes [216]. The initial round of coding was performed independently by two authors. We then compared codes to assess inter-coder reliability, using a Python implementation of Krippendorff's alpha (α). We used the `krippendorff` Python package [54], as our standard reliability metric. The resulting score was $\alpha = 0.857$, indicating strong agreement between coders and

a high level of coding consistency. This high level of reliability was facilitated by the use of a shared codebook and consistent application of predefined coding themes (see section C.2).

5.3.6. PARTICIPANT DEMOGRAPHICS

We recruited sixteen participants spanned North and South America, Europe, Asia, and Africa [118]. Roles included penetration testers, red and blue teamers, independent researchers, consultants, and in-house product security professionals; several held leadership positions (e.g., founders, CEOs).

Table 5.1 provides a detailed overview of the individuals who contributed to our study on PoC exploit development. It summarizes key characteristics such as geographic distribution, professional roles, experience levels, and frequency of PoC development among the 16 participants. The PoC counts in Table 5.1 include only those exploits participants explicitly identified as PoC.

This demographic context is essential for understanding the diversity and representativeness of our qualitative sample, highlighting factors that may influence perspectives on PoC creation, disclosure, and the broader dual-use dilemma explored in the main study.

Table 5.1: Summary of participant roles and background.

ID	Role	Sec. Role	Owner	Yrs Job	Yrs Sec.	Yrs PoC	#Vuln.	#PoC (public)	Platform	Freq.	Skill	Software Types	Recru.
P1	Pentester	F, V, BB	-	4.6	-	9	6–7	10–12 (2–3)	-	unclear	3.5	Web, Desktop, Electron, IoT	F, S
P2	Cybersec R&D Eng.	F, V, BB	-	1	4	-	-	300 (0)	-	unclear	3	Web, Network mgmt, IoT, PC, HD	C
P3	Owner		✓	9	16	11	-	15–20 (>3)	GitHub	3–4/mo	3–4	Web, Mobile, band &IoT devices, Network & ICS protocols	S
P4	Blue teamer	V, BB	✓	3	4–5	-	-	7–8	-	2–3/yr	3.0	PDF, Network mgmt	M
P5	Pentester (red teamer)	V	-	4	-	2	-	7 (1)	-	Monthly	2.5	mgmt System	S
P6	CTO	-	✓	8	-	17	-	30 – 40 (5–10)	-	5–10/yr	2	Industrial switches, SSH vuln, Ransomware exfiltration tool vuln	C
P7	Cybersec consultant– Pentester (red teamer)	-	-	2	-	2–3/mo	12	12 (0)	-	2–3/we	2–3	Hypervisor software	F
P8	Pentester	V, BB	✓	10		6–7	> 40	50–60	YouTube, others	10–20/yr	3	Any SW or Sys but mainly Web	S
P9	Hacker (Pentester)	Company PT	-	3	3	11	unclear	10 (0)	-	Monthly	3	Web, Mobile apps	C
P10	Hacker (Red Team)	V	✓	1	3	10	6–7	10–12 (2–3)	-	2–3/mo	3.0	Web, Local apps	F, C
P11	Lead Pentester	V, BB	-	2	6	8	35 (only in 2024)	100s (40–50)	LinkedIn	-	4.0	Web, AI apps	F, C
P12	Pentester	BB	-	-	4	5	3	<50 (36)	ExploitDB	Monthly	Web: 3, Bin: 4	Web, Binaries, Mobile	C
P13	Sec. Researcher	-	-	4	4	19	20	20 (0)	-	3/yr	4	IoT, Web UI	C
P14	Hacker (Pentester)	V	-	18	50	-	unclear	> 100 (>1)	Conferences, Youtube, others	unclear	5	Systems (Mainly Windows)	S
P15	Pentester	-	-	6–7	6–7	6–7	2	2,000 (0)	-	2–3/mo	4-5	Embedded sys, IoT (consumer&industrial)	C
P16	Security analyst& reverse engineer	-	-	12	12	12	10	100s (0)	-	Daily	4-5	Web, Embedded sys, IoT	C

Roles – F: Freelancer, V: Voluntary security researcher/tutor, BB: Bug bounty.
Skill scale: 1 = Beginner, 5 = Expert.
Recruitment method – C: Personal Contact, M: Social Media, F: Form, S: Snowball.

5.3.7. LIMITATIONS

Our study has several limitations common to exploratory qualitative research, as in similar related work [1, 17, 254]. First, participant recall may be incomplete or biased, as they were asked to retrospectively describe complex, expert-level tasks [254]. To mitigate this, we used hierarchical structuring in our interview protocol to prompt comprehensive responses [232]. Second, participants may have adjusted their responses to influence how they were perceived, either positively or negatively [254]. While we took steps to create a non-judgmental atmosphere, social desirability bias cannot be ruled out. Third, because the study explicitly focused on PoC development, it may have attracted participants with particular experiences or interests, introducing selection bias. Additionally, some individuals, especially those concerned about privacy or intellectual property, may have opted not to participate. To partially mitigate this, we used diverse recruitment channels and screened participants to ensure variation in experience and geography. We also acknowledge the gender imbalance in our sample, all participants identified as male. Despite targeted outreach efforts, we were unable to recruit women PoC developers. This reflects broader gender disparities in cybersecurity communities [118]. Lastly, while our findings offer valuable insights, they may not generalize beyond our sample. However, the number of participants 16 aligns with qualitative best practices, which suggest 12–20 participants for thematic saturation [120], and our focused recruitment provides strong direction for future work in this niche practitioner group [254].

5.4. UNDERSTANDING POC DEVELOPER PERSPECTIVES AND DECISION-MAKING

We apply EVT to analyze our semi-structured interview data and reveal how PoC exploit developers navigate the inherent dual-use dilemma in their work. PoC development spans a continuum from simple crash proofs to fully automated, weaponized exploits. Higher sophistication—automation, weaponization, code-centricity—amplifies both security research value and misuse risk. Figure 5.1 presents our adapted EVT framework, highlighting dual-use decision points where PoC developers balance competing motivations, integrating *Traditional EVT constructs* (yellow), *Extended EVT contributions* (pink), and *Contextual factors outside EVT* (blue).

We first examine how PoC developers define and perceive PoC exploits (RQ1), then identify the technical, ethical, economic, and contextual factors that shape PoC creation, validation, and reporting (RQ2), and finally explore how PoC developers negotiate dual-use decision points in their disclosure and publication strategies (RQ3). Notably, our EVT-based analysis reveals a phenomenon we term *motivational ambivalence*, where the very factors—such as intrinsic satisfaction from technical mastery, that drive thorough PoC exploit development also heighten the potential for misuse (see Table 5.2).

5.4.1. HOW POC DEVELOPERS DEFINE AND PERCEIVE POC EXPLOITS (RQ1)

PoC developers vary widely in what they consider a valid PoC, ranging from simple crash demonstrations to fully weaponized exploit code. Their definitions reflect not only technical benchmarks but also personal standards for originality, sophistication, and intended use. Underlying these definitions are key motivational factors: beliefs

Table 5.2: Motivational Factors in PoC Development Dual-Use Decisions Based on EVT Analysis

EVT Component	Encouraging Dual-Use Risk	Discouraging Dual-Use Risk
Expectancy Beliefs	High technical confidence, Patch diffing success, AI-assisted development	Task difficulty uncertainty, Limited system knowledge, Tool/resource constraints
Intrinsic Value	Intellectual satisfaction, Technical mastery, Weaponization challenge	Preference for defensive applications, Lack of interest in harmful applications
Utility Value	Personal advancement, Financial gain, Skill demonstration, Security improvement	Low perceived benefit, Vendor assistance, Ecosystem benefits
Attainment Value	Technical mastery identity, Community recognition	Core ethical values, Professional identity
Cost Factors	Low perceived legal risk, Anonymity protection, Vendor unresponsiveness	High legal/reputational risk, Moral responsibility burden, Potential misuse guilt
Extended EVT:		
Moral Reasoning	Tool neutrality beliefs, Responsibility deflection	Strong moral framework, Harm prevention
Dynamic Value Assessment	Vendor unresponsiveness, disclosure/publication outcomes, and community feedback	Vendor responsiveness, Coordinated disclosure benefits
Identity Navigation	Personal gain, Financial gain	Ethical identity, Professional reputation
Outside EVT:		
Vendor Relationships	Unresponsive vendors, Poor communication	Responsive vendors, Strong CVD frameworks, Clear communication and timelines
Community Dynamics	Anonymous collaboration benefits, Knowledge sharing pressure	Trust erosion concerns, Community accountability
Cultural/Legal Context	Legal gray areas, Weak enforcement, Cultural acceptance	Strong legal frameworks, Cultural disclosure norms, Clear consequences

about their own technical capabilities and likelihood of success, the intrinsic satisfaction of demonstrating mastery, the utility of PoC for security improvement, the role of PoC work in their professional identity, and the costs they associate with development and disclosure. Together, these factors shape how PoC developers define, prioritize, and engage with PoC exploit creation (highlighted in yellow in Figure 5.1).

THE POC EXPLOIT DEFINITION SPECTRUM

PoC developers' beliefs about their technical abilities and expectations of success shape their position on what a PoC exploit is and should do, influencing their PoC development strategies and publication choices.

Perceived Technical Ability: Several PoC developers explicitly connected PoC creation to demonstrating technical understanding and validating their expertise. For instance, P12 mentioned *"because it proves that you understand the vulnerability. Ego plays a role here."* This reflects how a self-concept of ability drives behavior choices where PoC developers with high technical confidence pursue challenging targets to validate and display expertise.

PoC developers demonstrated varying perspectives on legitimate PoC targets, with some insisting that valid PoC should exclusively target self-discovered or zero-day vulnerabilities. As P10 explained: *"I'd say it's when you find a vulnerability on your own, like a zero day,"* while P13 similarly stated: *"I think its for the zero day vulnerability."* This exclusivity preference reflects high a self-concept of technical ability, where PoC developers prioritize original discovery over adaptation of known vulnerabilities.

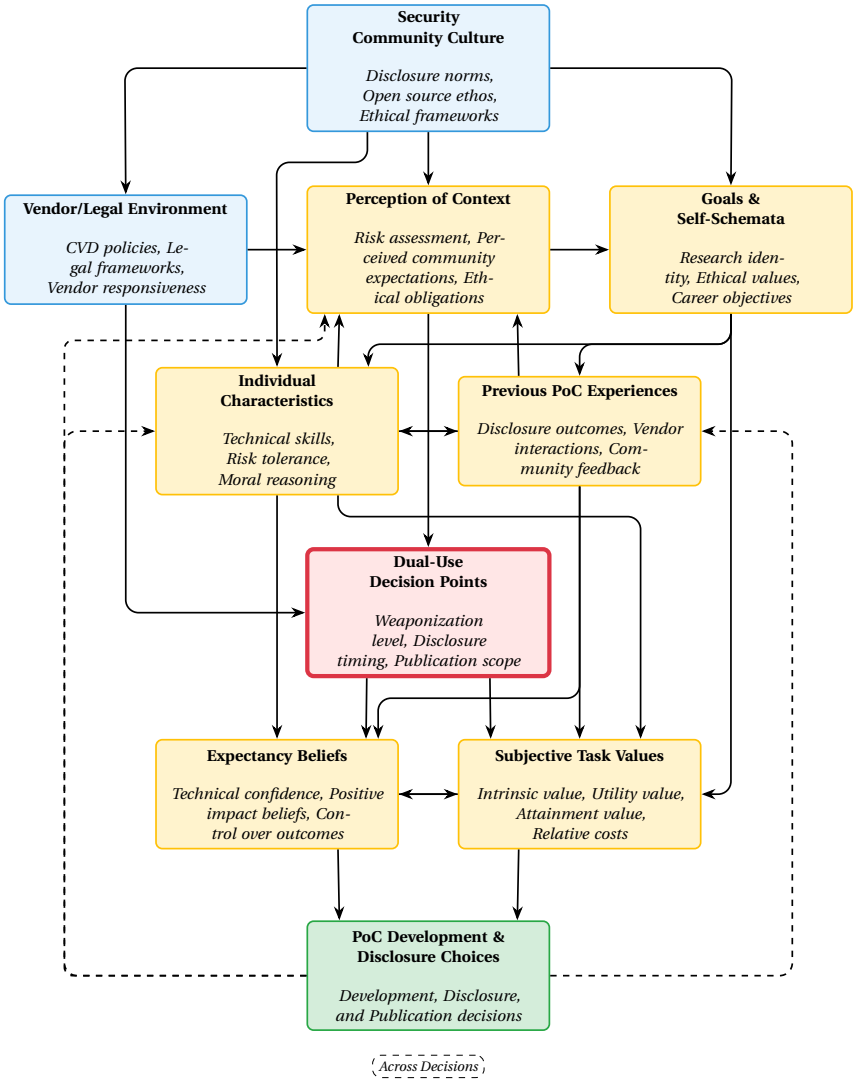


Figure 5.1: The Expectancy-Value Model for PoC Development Dual-Use Decisions. *(Adapted from [90])*

On the other hand, other PoC developers maintained that PoC can target known vulnerabilities discovered by others and still be considered valid as long as they demonstrate technical effectiveness, indicating different self-concept thresholds for meaningful technical contribution.

P8 similarly described preferences for *weaponized proof of concept* creation, indicating how high a self-concept of ability leads to more sophisticated development goals. The intrinsic satisfaction derived from technical mastery serves as a powerful motivator independent of external rewards.

PoC developers systematically adapt their approaches based on perceived expertise

in different vulnerability domains. P12 explained their systematic approach: *"If [a vulnerability] is publicly published, I use Shodan to get the CVE. If that CVE has no PoC, I develop one."* This demonstrates how domain knowledge confidence (expectancy belief) drives efficient targeting strategies.

Task Difficulty Perceptions and Success Expectations: PoC developers demonstrated preferences for *known vulnerabilities* with or without existing PoC, viewing these as optimal targets balancing challenge with success probability. This approach was described as:

> *"So we look at the differences between the old version and the new version. And that often requires some reverse engineering skills. Based on what the vendor changed in the new version, we can deduce what the vulnerability was in the old version and write code to verify that those vulnerabilities still persist. In the new version, there might be a single line change that reflects the patch."* (P2)

This "patch diffing" approach used by reverse engineering silent patching reflects sophisticated task difficulty assessment—leveraging vendor patches as roadmaps reduces perceived difficulty while maintaining intellectual challenge. Such approaches maximize expectation of success while preserving intrinsic value through technical problem-solving.

In addition, pursuing existing vulnerabilities often leads to discovering new vulnerabilities. P4 explained:

> *"For example, in the case of a PDF reader, I noticed several vulnerabilities published recently, so I examined if one product or multiple products are affected. The larger the user base, the more likely you will find an easy target."* (P4)

For *unknown vulnerabilities*, PoC developers often adopt a trial-and-error approach. P12 mentions that he *"downloads, installs, and fuzzes the software,"*, while P15 tailors his methods based on access to physical devices, firmware, or mere specifications. Access to necessary tools and knowledge significantly affects PoC developers' willingness to pursue specific vulnerabilities, with deep system knowledge reducing perceived task difficulty and increasing expectation of success.

Prior experience guides both target selection and the choice of tools and programming languages for PoC development. A strong preference for Python is common due to its simplicity, extensive libraries, and compatibility with AI-driven methods. P11 noted: *"AI agents can work with Python extremely efficiently."* Similarly, P1 stated: *"Just use AI to make a good starting point."* PoC developers also utilize languages such as .NET, C#, bash, and Go, selecting the appropriate tools based on target specifics and operational requirements.

AI integration enhances expectancy beliefs by boosting PoC developers' technical confidence and perceived capability to handle complex vulnerabilities. The availability of AI-assisted PoC development tools strengthens the self-concept of ability, as PoC developers feel more capable of creating sophisticated PoC regardless of their initial expertise level, which reduces effort costs. This technological augmentation of expectancy beliefs creates a feedback loop where increased confidence leads to more ambitious

targeting and development approaches, while simultaneously reducing the perceived difficulty of moving toward more weaponized implementations.

PoC developer expectations for PoC sophistication varied significantly, with some requiring full automation and weaponization capabilities. P2 emphasized: *"Only consider it as a proof concept exploit if it's an automated code that anyone can run and get the same results."* This automation standard reflects high expectancy beliefs about technical outcomes— PoC developers confident in creating reliable, weaponized tools set higher success thresholds. P8 described the ideal as a "0-click" solution where "typing in an IP address" automatically yields complete system control. Such weaponization expectations demonstrate how high a self-concept of ability leads to more sophisticated PoC development goals, potentially increasing dual-use risks through enhanced exploit capabilities. P6 described the evolution from basic demonstration to automated tool: initially demonstrating a vulnerability, but a robust PoC should *"eventually automate all steps so it can be reliably handed over to, for example, a red team or the vendor for remediation."*

MOTIVATIONAL DYNAMICS IN POC DEVELOPMENT

PoC exploit development sits at a crossroad in cybersecurity, embodying a dual-use dilemma where the same tools that drive security research and remediation can also be weaponized for malicious purposes. This creates a complex motivational landscape for PoC developers, who must constantly balance technical ambition, values, ethical considerations, and risk management.

Intrinsic Value and Craftsmanship: PoC developers derive deep intrinsic satisfaction from crafting PoC exploits, emphasizing the paramount importance of executable code over abstract descriptions. As one participant (P8) succinctly put it, *"the most fun is to write a weaponized Proof of Concept."* This code-centric philosophy positions functional code as the ultimate demonstration of mastery. Reflecting this view, P12 stated: *"For me only the code counts... the perfect PoC is a code base."* Similarly, P7 highlighted that a PoC is *"any piece of code... tailored to exploit a vulnerability... especially if it is novel."*

This orientation motivates PoC developers to build PoC exploit from scratch, particularly for new vulnerabilities, ensuring technical rigor and security integrity. P12 described their workflow for web exploits: *"I start by exploiting the vulnerability manually, then I replicate the process in Python code. After that, I target the vulnerable SQL field and escalate until I achieve remote code execution."* The drive for ethical craftsmanship also manifests in efforts to minimize disruption: P5 embedded sleep commands into an SQL injection PoC to avoid crashing the system, and P3 praised *"thinking outside the box"* to achieve elegant yet non-disruptive demonstrations.

Utility Focus: Beyond personal fulfillment, many PoC developers prioritize utility — the benefit their work brings to cybersecurity. P2 framed this value as, *"...cybersecurity should be a fundamental right... And by writing proof of concepts, because the name kind of says it. It's a proof of concept, you can prove to people that there is a risk, and motivate them to mitigate that risk."* PoC developers often favor responsible disclosure to vendors or authorities. For instance, P13 supports full disclosure to national authorities, while

P11 advocates delaying public release: *"...don't release... instantly... but eventually it's valuable."*

PoC are seen as tools for remediation. P1 emphasized, *"PoC are purely demonstrations to get things fixed,"* and P15 noted they help others *"reproduce the problem."* This approach balances knowledge sharing with reducing potential misuse risks.

Identity and Recognition: Disclosure decisions are strongly tied to complex professional identities and ethical considerations. Many PoC developers see themselves as responsible contributors rather than as seekers of notoriety. P11 expressed support for openness tempered by caution: *"I'm for openness, but... wait for fixes... to help defenders."* Cultural and regional contexts influence these decisions; for example, P16 noted that some prefer limited disclosure to avoid reputational damage.

Cost and Moral Challenges: PoC developers contend with significant costs, including legal risks, reputational concerns, and psychological burdens. P14 abstains from working on zero-day PoC due to legal restrictions. For example, P1 recounted severe consequences when a PoC he was part of its development, which was disclosed through one of the responsible disclosing channels, ended up in a sobering incident:

> *"We made a full exploit chain... and it somehow ended up in the wrong hands... a lot of companies got affected by it. We had the FBI, Department of Homelands and some other big organizations come to us and talk about it... and they had to shut it down. So, it's like I kinda distanced myself in the background when that... was happening, because I don't want anything of that and you get nothing from it. You put your neck on the block, you're just waiting for it to be cut off. For what? For nothing. So, yeah. No! So, that's why I'm kinda reluctant with publishing proof concepts like publishing a limited disclosure with explaining that there's a vulnerability, what to do and stuff, sure. But saying like, here's a weaponized proof of concept, if you press on enter, you can take over the server. Yeah, I don't see any use for that."* (P1)

Concerns about irresponsible disclosure are common. P2 warned: *"...giving weapons to others who can't handle them responsibly is irresponsible."* Many take precautions to safeguard PoC, such as maintaining them locally with strict security measures. Economic incentives complicate decisions: P12 admitted to selling PoC in the past but now pursues responsible disclosure.

Takeaway 1

PoC development spans a continuum from simple crash demos to fully weaponized exploits, driven by complex motivations including technical mastery, ethical reflection, and risk negotiation. PoC developers' decisions are understood as dynamic calculations balancing their confidence (expectancy), perceived value (utility, intrinsic, attainment), and costs (legal, moral).

5.4.2. FACTORS DRIVING POC DEVELOPMENT DECISIONS (RQ2)

This section examines the key factors shaping PoC development decisions, extending EVT to capture the unique challenges of dual-use technologies. Central to this extension is the concept of *dual-use decision points* (highlighted in red in Figure 5.1), where developers weigh competing technical, ethical, and contextual considerations.

Our extended EVT framework comprises three components: *Moral Reasoning*, where PoC developers navigate ethical reflections on tool neutrality and responsibility; *Dynamic Value Assessment*, reflecting how PoC developers continuously adjust their valuation of tasks based on vendor responsiveness, disclosure outcomes, and community feedback; and *Identity Navigation*, describing ongoing negotiation between competing professional identities in the face of dual-use tensions. Together, these elements illuminate the complex, context-sensitive calculus guiding PoC creation, validation, and disclosure decisions..

MORAL REASONING ABOUT TOOL NEUTRALITY

PoC developers grapple with moral responsibility stemming from the dual-use nature of their work, which introduces psychological costs beyond typical effort or time constraints. They often adopt a moral reasoning stance that views PoC as neutral tools—akin to knives—where the ethical implications depend on how others use them. This perspective helps PoC developers manage ethical tensions and sustain engagement despite potential misuse.

The philosophical reasoning about *tool neutrality* represents a distinct form of ethical reasoning that while a PoC exploit can be adapted for harmful purposes, the tool itself is inherently neutral, its effect determined by how it is employed, which was analogized by one of the PoC developers as:

> *"Well, releasing an exploit have their risk. It's like a knife. It's like it's not good or bad; and a knife is just a tool. You can use the knife to feed your family, or you can use the knife to kill somebody else... So you the person who crafted the knife... oh, what people would do with my knife."* (P9)

POC DYNAMIC VALUE ASSESSMENT

PoC development involves ongoing recalibration of risk and benefit evaluations, with PoC developers' subjective task values shifting dynamically in response to external influences such as vendor responsiveness, disclosure outcomes, and community feedback.

Several participants highlighted a preference for withholding public disclosure of PoC until vulnerabilities have been patched. For example, participant P10 expressed reluctance to share PoC prematurely:

> *"I'm not really in favor of publicly sharing the PoC unless I know the vulnerability has been patched. The people running the software were given a few months to patch, so I only publish after they have had ample time to update."* (P10)

On the other hand, participant P1 raised concerns about the limits of timing in mitigating risk, noting:

> *"If you look at the timelines when a proof of concept is published and when it gets abused, those timelines are pretty interesting. A patch does not mean everyone is protected immediately."* (P1)

Some PoC developers practice controlled disclosure, delaying publication of weaponized PoC for extended periods, explained as:

> *"If I have a weaponized proof of concept, I do not share it with the public right away. Maybe after a year or two, when everything is fixed, then I release it, often via my website."* (P8)

Participant P12 provided a critical view of how vendor transparency impacts risk assessment:

> *"If there's a public exploit code... some vendors don't admit what the patch does. But if the vendor is fast and transparent, having public PoC balances the outcome — attackers' odds are about 50/50."* (P12)

This dynamic process reflects how PoC developers continuously update their value calculations based on vendor behavior and disclosure context. Moreover, participants described a maturation from personal or opportunistic motives to more collaborative and career-oriented ones. This evolution was reflected by a PoC developer as:

> *"Now I don't do that [selling PoC] anymore; I've grown and consider reporting publicly. Maybe it could even change my career path — like working for Facebook."* (P12)

BALANCING PROFESSIONAL IDENTITY WITH ETHICAL RESPONSIBILITIES

While traditional EVT's attainment value assumes straightforward alignment between activities and identity, dual-use contexts create identity conflicts between roles as ethical researchers, technical experts, and security contributors, requiring active negotiation rather than simple alignment. The decision between supporting or discouraging PoC publication reflects different security philosophies and identity positions, noted as:

> *"Depends on your point of view. If you care about security, we should have a private place to disclose PoC. If you don't care about securing clients, every PoC should be public."* (P12)

Some PoC developers shifted their disclosure practices from *full to partial* when experiencing real or potential misuse of their PoC, representing identity adaptation in response to dual-use consequences. Even under responsible disclosure, PoC developers must decide whether to proceed with full or partial disclosure processes. Some opt for full disclosure:

> *"I often always submit full disclosure. Yeah, because it is useful for the [National CVD], and also useful for the vendors to identify the vulnerabilities." "Yeah, I basically trust the [National CVD].. So yeah, I would like to share my PoC only with them, but I don't want to make it public."* (P13)

Conversely, others choose partial disclosure:

> *"My real proof of concept code is something that I rarely share. And that's because my experience with a lot of responsible disclosure teams, within vendors, are not to be trusted completely. Like I've heard also from people around me that say, well, I submitted my code to just show them like the vulnerability, and then, out of the blue, a few days later, you see it online."* (P4)

Personal motivations can also play a critical role in shaping disclosure decisions. P12 recounted an early experience involving a cross-site scripting vulnerability in Facebook's chat feature, explaining that he deliberately chose not to disclose this vulnerability because it served his personal interests, using the exploit for about two months until Facebook eventually patched it. This example shows how personal gain in PoC development can influence a PoC developer's disclosure strategy.

Takeaway 2

Beyond individual skills and motivations, PoC developers engage in ongoing, complex negotiations shaped by ethical reasoning, external feedback, and identity tensions. Moral responsibility is often externalized via tool neutrality, while dynamic value assessments fluctuate with vendor responsiveness and community reactions. PoC developers continuously balance their professional identities—as innovators, collaborators, and guardians—within a shifting ecosystem marked by legal ambiguities and cultural complexities.

5.4.3. CONTEXTUAL FORCES SHAPING POC DEVELOPER CHOICES (RQ3)

Similar to [62], our analysis reveals important contextual factors that operate outside the conventional framework but significantly influence dual-use in PoC development decisions that don't fit neatly within traditional framework boundaries. These include security community culture with its disclosure norms and ethical frameworks, and vendor/legal environments encompassing CVD policies and regulatory constraints. These external factors shape the context within which individual motivational processes unfold (colored in blue in Figure 5.1).

VENDOR RESPONSIVENESS & COMMUNICATION

Vendor responsiveness creates dynamic feedback loops that modify future disclosure decisions in ways not captured by static EVT value assessments. A crucial aspect of the disclosure process is the tone of communication and responsiveness between PoC developers and vendors. Most PoC developers believe that adopting a friendly and respectful tone not only fosters cooperative dialogue but also helps mitigate legal risks:

> *"I [think] definitely in your first messaging, saying something, giving them consequences for not replying is coming off very, you know, evil. To them it sounds like you're an enemy, whereas I more wanna make myself sound like a friend to them. So they can ask me questions, come back to me, or even order a real pen test if they want. But if you start saying, after 90 days, I will disclose this then, you kind of put the whole conversation in a negative tone, whereas*

> *it should be positive. And usually, I honestly couldn't care too much about actually disclosing stuff. What am I gonna do? I'm just gonna make it public. Threatening companies increases the chance of getting lawyers..., which is far worse than just being nice."* (P11)

However, the situation often changes when vendors fail to mirror that same level of communication, forcing PoC developers to adjust their disclosure to direct disclosure models. One PoC developer recalled a case involving vulnerabilities in smart lights:

> *"I found the vulnerability in these [smart] lights. I tried to let them know and they said no, we're not interested. Go away."* (P11)

In another instance, following an unresponsive disclosure to Microsoft, P12 described:

> *"the first in my life I published to Microsoft. They didn't care. They didn't even disclose it. They didn't give me some reward or something. Nothing happened so I throw that script on the web. And 3 days later, we receive a patch through Windows update."* (P12)

In both cases, PoC developers transitioned from responsible disclosure approach to direct publication to pressure vendors into remediation.

PoC developers increasingly rely on national frameworks—such as CERT programs—to report vulnerabilities. These delegations not only facilitate responsible disclosure but also reduce the administrative burden of direct vendor communication. Many participants report that these systems provide a structured and predictable timeline for vendor responses.

COMMUNITY TRUST AND ANONYMITY DYNAMICS

PoC developer communities are characterized by anonymity and low trust, which shape unique social dynamics around information sharing and collaboration. Despite limited trust, collaboration remains vital, with PoC developers exchanging knowledge and jointly solving problems through platforms such as Discord, Telegram, bug bounty programs, and online forums. P11 exemplified this saying:

> *"Yeah, in the bug bounty community, we do collaborate a lot and it's really fun to see. Definitely. If you're working on the same target, somebody will go. Hey, I have found this small issue here. I need this behavior to kind of chain it together and make it more impactful and then somebody else will go. I found that behavior on this on this website. Does that work together and then so yeah, that's quite fun to collaborate together in that way."* (P11)

Within this collaborative environment, a layered dual-use dilemma emerges. On one level, shared motives for knowledge exchange drive the community, as P12 explains:

> *"There are many collaboration channels... to share knowledge because sometimes they share the same goals you never know how many people hate Microsoft."* (P12)

However, P12 also stated the open sharing of early-stage PoC can inadvertently enable unauthorized manipulation:

> *"Well, issue was where in the community I discussed that. I said I have command execution, not remote code execution. There is a command execution in Microsoft website, so one of the guys develop and had remote code execution on that server and he calls denial of service for Microsoft. It didn't last so much like two or three minutes, all servers were down."* (P12)

The dilemma deepened when the collaborator chose not to contribute back:

> *"He succeeded in getting some remote code execution, and when I asked him to share the code and so he didn't share it back, the guy just logged out. Happens, yeah. You know, you just move on because everyone is anonymous."* (P12)

The anonymous collaboration model doesn't fit traditional social influence categories within EVT. It creates different dynamics than conventional social contexts where EVT has been applied. This community paradox was described by P12 as:

> *"Zero trust you. Don't trust anyone in that community, but, as I said, you go to certain servers, certain channels where people share the same motives and goals. So trust is not a factor."* (P12)

CULTURAL & LEGAL CONSTRAINTS ON POC PUBLICATION

Cultural and legal factors create constraints that operate outside traditional EVT motivational processes while significantly influencing decision-making outcomes. P16 highlighted that cultural and legal pressures may discourage public PoC publication, except where the impact is minimal and remediation does not require end-user action.

Other reasons PoC developers opt not to disclose vulnerabilities or PoC at all range between: issues deemed insignificant (P7), experiencing burnout (P1), someone else having published it (P7), wish to avoid reputational risk (P16), due to cultural or language communication challenges (P11). According to P2, social media can rapidly amplify risks:

> *"looks for what's discussed on platforms like Twitter. And because people on Twitter can't shut up, so even though they have an NDA, they will post it on Twitter."* (P2)

These cultural dynamics affect information flow and knowledge PoC development in ways that indirectly influence EVT components while representing external constraints on PoC developer decision-making.

The PoC developers often bypass intermediary channels of disclosure by publicly, yet often anonymously, sharing both vulnerability details and PoC, often via online or underground platforms, as P12 noted: *"other PoC are like just thrown in the web, or ExploitDB, [or in] GitHub somewhere."* These platform choices reflect cultural norms and legal constraints that operate independently of individual motivational processes.

Takeaway 3

PoC exploit developers' decisions balance intrinsic satisfaction and utility with risks of misuse and legal costs. Vendor responsiveness critically influences disclosure strategies, fostering cooperation or prompting direct public release. Anonymous, low-trust communities enable collaboration but increase misuse risks. Cultural and legal contexts further complicate disclosure decisions. This dynamic interplay transforms PoC disclosure into a continuous negotiation between advancing security and managing dual-use risks, far beyond a simple responsible versus irresponsible dichotomy.

5.5. DISCUSSION

Our findings fundamentally challenge the prevailing binary conceptualization of responsible versus irresponsible disclosure that dominates current cybersecurity discourse [17, 259]. Rather than operating within fixed ethical frameworks, PoC developers engage in sophisticated motivational negotiations that extend far beyond the technical-ethical dichotomy assumed by much prior work [63, 254].

Where previous studies have primarily focused on observable behaviors and temporal patterns of exploit deployment [19, 179], our EVT-informed analysis reveals the underlying psychological mechanisms driving PoC development decisions. This represents a significant departure from economic models that assume rational cost-benefit calculations [9, 14] by demonstrating how expectancy beliefs, value assessments, and identity considerations create complex motivational landscapes that resist simple optimization frameworks.

Our extension of EVT through dual-use moral reasoning, dynamic value assessment, and identity navigation addresses a critical theoretical gap in cybersecurity research. While motivational theories have been successfully applied to defensive security behaviors [62, 141], this study provides the first systematic application to offensive security practices. The concept of motivational ambivalence—where the same factors that drive thorough exploit development also heighten misuse potential—offers a novel framework for understanding dual-use technology development that extends beyond existing governance models [87, 167].

Dual-Use Moral Reasoning as Responsibility Externalization. The insight of tool neutrality beliefs as a systematic form of moral reasoning highlighted by PoC developers in this study provides new theoretical insight into how practitioners manage psychological costs in dual-use contexts. Unlike previous work that has treated ethical considerations as static constraints [68], we demonstrate that PoC developers actively construct moral frameworks that enable continued engagement despite potential harmful consequences. This process of responsibility externalization represents a previously undocumented psychological mechanism that may apply broadly across dual-use technology domains.

Dynamic Value Assessment and Contextual Adaptation. Traditional EVT assumes relatively stable value assessments, but our findings reveal that PoC developers continuously

recalibrate their risk-benefit evaluations based on vendor responsiveness, disclosure outcomes, and community feedback. This dynamic process challenges static models of responsible disclosure [231] and suggests that governance approaches must accommodate ongoing negotiation rather than fixed rules. The identification of vendor behavior as the primary driver of this recalibration process provides empirical support for relationship-based rather than rule-based disclosure frameworks.

Identity Navigation in Professional Contexts. Our analysis reveals that dual-use decision-making involves continuous negotiation between competing professional identities—ethical researcher, technical expert, security contributor—rather than simple alignment with pre-existing values. This extends beyond existing work on hacker identity formation [68] by showing how professional contexts create identity conflicts that require active management rather than passive expression of fixed ethical commitments.

CHALLENGING ECONOMIC AND RATIONAL CHOICE MODELS

Our findings directly contradict the assumptions underlying current economic models of vulnerability disclosure and exploitation. Allodi's work on attacker economics [9, 14] demonstrates that malicious actors prioritize low-effort, high-impact vulnerabilities through rational cost-benefit analysis. However, our PoC developer interviews reveal that PoC creation involves fundamentally different decision-making processes that cannot be reduced to simple economic optimization.

The concept of motivational ambivalence illustrates this divergence most clearly. Economic models would predict that rational actors would minimize dual-use risks to avoid potential costs, but our participants consistently described how intrinsic satisfaction from technical mastery drove them toward more sophisticated—and potentially more dangerous—exploit development. This pattern suggests that non-economic motivational factors may be more influential than previously recognized in shaping the dual-use technology landscape.

Similarly, our finding that vendor responsiveness is the primary determinant of disclosure decisions challenges economic models that assume stable preferences and consistent cost-benefit calculations. The dynamic nature of value assessment revealed in our data suggests that PoC developer behavior is fundamentally contextual and relationship-dependent rather than driven by fixed utility functions.

IMPLICATIONS FOR VULNERABILITY GOVERNANCE AND POLICY

Our results have implications for current approaches to vulnerability governance and coordinated disclosure policy. Existing frameworks assume that clear rules and structured processes will produce predictable outcomes [259], but our findings suggest that individual motivational factors may be more influential than institutional constraints in determining actual behavior.

The Recognition Paradox and Incentive Misalignment. We identify a previously undocumented asymmetry in community recognition patterns: vulnerability discoverers receive celebration while PoC developers face stigmatization, despite both activities representing dual-use research. This recognition paradox creates perverse incentives that may actually increase security risks by pushing PoC development underground or toward less

responsible disclosure practices. Current bug bounty research has focused on economic incentives [1, 29], but our findings reveal that social recognition dynamics may be equally important in shaping behavior.

This misalignment suggests that governance frameworks focused solely on constraining harmful behavior may be counterproductive. Instead, policies that acknowledge and reward responsible PoC development alongside vulnerability discovery may be more effective in channeling dual-use activities toward beneficial outcomes.

Dynamic Governance for Dynamic Decisions. The finding that PoC developers continuously recalibrate their value assessments based on external feedback suggests that static governance approaches may be fundamentally inadequate for dual-use contexts. Unlike traditional risk management frameworks that assume stable risk profiles, dual-use technologies require governance systems that can adapt to changing motivational landscapes and contextual factors.

Our identification of vendor responsiveness as the primary driver of disclosure decisions provides specific guidance for improving governance outcomes. Rather than focusing primarily on rule development and enforcement, effective governance may require greater attention to relationship management and communication quality between stakeholders.

5.6. RECOMMENDATIONS

Based on our empirical findings, we propose specific interventions derived directly from our EVT-informed analysis of PoC developer decision-making. Each recommendation addresses specific motivational factors, contextual constraints, or community dynamics identified in our results.

FOR VENDORS AND ORGANIZATIONS

R1: Implement Structured Response Protocols: Our finding that vendor responsiveness is the primary determinant of disclosure decisions (exemplified by P12's Microsoft case where unresponsiveness led to immediate public disclosure) requires systematic changes to vendor communication practices. Organizations should establish 48-hour acknowledgment protocols, assign dedicated technical liaisons, and provide regular progress updates to PoC developers throughout the disclosure process.

R2: Deploy Recognition Programs: Our discovery that vulnerability discoverers receive celebration while PoC developers face stigmatization creates perverse incentives. Based on participants' descriptions of wanting professional recognition (P12's career advancement motivation), vendors should create formal acknowledgment programs that recognize responsible PoC development through technical advisory roles, security team consultation opportunities, and public recognition equivalent to vulnerability discovery programs.

R3: Establish Graduated Response Systems: Our identification of PoC development existing on a sophistication continuum from crash demonstrations to weaponized exploits requires differentiated organizational responses. Vendors should implement classification systems that distinguish between demonstration-level PoC (requiring standard disclosure timelines) and weaponized exploits (requiring expedited response protocols

and enhanced PoC developer support).

FOR POLICYMAKERS AND STANDARDS BODIES

R4: Create Safe Harbor Provisions: Multiple participants (P14, P16) described avoiding PoC development due to legal uncertainty, directly impacting the cost component of our EVT analysis. Policymakers should establish explicit safe harbor protections for good-faith security research including PoC development, with clear criteria based on disclosure intent and target selection rather than tool sophistication.

R5: Support Dynamic Disclosure Frameworks: Our discovery that PoC developers continuously recalibrate risk-benefit evaluations based on vendor behavior requires policy frameworks that accommodate this dynamic process. Standards bodies should develop graduated disclosure models supporting time-delayed publishing (as practiced by P8), partial disclosure options (preferred by P4), and context-specific evaluation criteria rather than uniform disclosure timelines.

R6: Fund National CVD Infrastructure: Our finding that PoC developers increasingly rely on national CERT programs as trusted intermediaries (reported by multiple participants as reducing vendor communication burden) indicates the value of institutional infrastructure. Policymakers should expand funding for national vulnerability coordination centers that can serve as neutral intermediaries between PoC developers and vendors.

FOR PLATFORMS AND TECHNICAL INFRASTRUCTURE

R7: Develop Sophisticated Content Management: Our identification of anonymous, low-trust community dynamics that enable valuable collaboration while increasing misuse risks requires technical infrastructure that can manage these tensions. Platforms should implement graduated sharing systems that allow vulnerability details sharing while controlling weaponized component access through community membership verification, temporal restrictions, and reputation-based access controls.

R8: Create Verified Researcher Networks: The "zero trust but shared goals" phenomenon identified in our community analysis suggests the need for reputation systems that preserve beneficial anonymity while providing accountability mechanisms. Platforms should develop verified researcher programs that enable higher-trust collaboration through cryptographic identity systems and peer verification processes.

R9: Implement AI-Aware Risk Assessment: Our discovery that AI integration enhances PoC developers' expectancy beliefs and enables more sophisticated exploit development requires automated risk assessment systems. Platforms should invest in machine learning systems that can evaluate relative risk levels of PoC submissions and apply appropriate sharing restrictions based on sophistication metrics rather than requiring manual review.

FOR THE SECURITY RESEARCH COMMUNITY

R10: Establish Dual-Use Decision Training: Our identification of ongoing identity negotiation between ethical research and technical mastery roles requires structured support for navigating these tensions. The community should develop training programs that teach dual-use decision-making frameworks, impact assessment methodologies, and ethical reflection processes rather than relying on informal trial-and-error learning described by

participants.

R11: Create Collaborative Disclosure Support Networks: Our finding that vendor behavior creates dynamic feedback loops affecting future disclosure decisions suggests the need for community infrastructure that can buffer these relationships. Professional associations should establish intermediary organizations that can facilitate vendor-PoC developer communication while providing accountability and support mechanisms for responsible disclosure processes.

R12: Develop Community Standards: Our documentation of differential stigmatization of PoC developers versus vulnerability discoverers requires systematic community norm changes. Security conferences, professional organizations, and community platforms should establish recognition standards that celebrate responsible exploit development as security contribution equal to vulnerability discovery, including dedicated presentation tracks and award categories.

FOR ACADEMIC AND INDUSTRY RESEARCH

R13: Expand Motivational Research Applications: Our successful application of EVT to offensive security practices demonstrates the value of psychological frameworks in cybersecurity research. Researchers should explore additional motivational theories (self-determination theory, social cognitive theory) and their applicability to other security domains including malware analysis, threat intelligence, and security tool development.

R14: Conduct Longitudinal Studies of Dynamic Value Assessment: Our finding that vendor behavior shapes future disclosure decisions through dynamic value assessment requires longitudinal research designs that can capture these temporal effects. Researchers should design multi-wave studies that track how disclosure experiences influence subsequent PoC developer decision-making across different vendor types and disclosure contexts.

R15: Investigate Cross-Cultural Variations in Dual-Use Decision-Making: Our identification of cultural constraints as operating outside traditional EVT suggests the need for systematic cross-cultural research. International research collaborations should examine how different legal frameworks, cultural norms, and institutional contexts influence dual-use decision-making to inform more effective global cooperation on vulnerability governance.

5.7. CONCLUSION

In this study, we examined how PoC exploit developers navigate the dual-use dilemma inherent in their work around PoC exploit development. By applying Expectancy-Value Theory (EVT), we showed how PoC exploit developers' decisions are guided not merely by capability or ideology, but through dynamic assessments of their technical capabilities, expected outcomes, and multifaceted value considerations.

Our analysis shows that the dual-use dilemma is not simply a binary choice between responsible and irresponsible disclosure, but rather a complex navigation of competing EVT components influenced by contextual factors specific to dual-use technologies. Expectancy beliefs about technical success and security impact, combined with subjective assessments of utility value, intrinsic satisfaction, identity alignment, and various cost

considerations, systematically predict PoC developer behavior across the disclosure spectrum.

The theoretical extensions we proposed—dual-use identity navigation, contextual cost amplification, and dynamic value assessment—expand EVT's applicability to cybersecurity contexts while identifying areas for future theoretical development. These insights offered practical guidance for improving coordinated vulnerability disclosure processes through EVT-informed interventions that address PoC developer motivations rather than simply imposing policy constraints.

Our findings suggested that supporting responsible dual-use technology development requires addressing the full spectrum of PoC developer motivations—technical confidence, value perceptions, identity considerations, and cost concerns—rather than focusing solely on regulatory or technical controls. By understanding and supporting the complex motivational landscape revealed through our analysis, we can better harness the security benefits of dual-use technologies while minimizing their potential for harm.

ETHICAL CONSIDERATIONS

This research received approval from our institution's human research ethics board prior to any data collection activities. The sensitive nature of cybersecurity research and the specialized practitioner community we studied required careful attention to ethical protocols throughout the research process.

Our informed consent procedures were designed to ensure participant autonomy and full understanding of research implications. Before any interviews commenced, the lead researcher provided detailed explanations of the study's objectives, methodological approach, data handling procedures, and participant rights to all potential participants. We emphasized that involvement was entirely voluntary, with participants maintaining the right to withdraw at any point without consequence or explanation.

No financial incentives or compensation were provided to participants, eliminating potential coercion and ensuring that participation decisions were based purely on voluntary interest in contributing to cybersecurity research. This approach was particularly important given the professional nature of our participant community and the potential for economic incentives to influence disclosure behaviors that were central to our research questions.

Data protection protocols exceeded standard requirements given the sensitive technical content and potential identification risks inherent in specialized cybersecurity discussions. All audio recordings were immediately transferred to secure, institutional servers with restricted access limited exclusively to the core research team. Transcription processes included comprehensive anonymization procedures that removed not only direct personal identifiers but also contextual details that could enable re-identification within the cybersecurity community. This included careful redaction of specific technical implementations, organizational affiliations, geographic indicators, and temporal markers that could serve as identifying fingerprints.

Storage and access controls followed institutional data governance policies, with encrypted storage, regular access audits, and automatic deletion schedules aligned with ethical approval requirements. Only authorized research team members with appropriate training in sensitive data handling were granted access to interview materials, and all

access activities were logged and monitored.

To ensure accuracy and maintain participant agency over their contributions, we implemented a member-checking process whereby participants were invited to review relevant sections of the final manuscript. This process allowed participants to verify that direct quotes and personal statements were appropriately represented, request modifications where necessary, and confirm that their perspectives had been accurately captured within our analytical framework. Several participants provided valuable feedback during this review process, leading to clarifications and refinements that improved the accuracy of our representations.

Furthermore, we recognized the potential legal and professional risks that participants faced by discussing their work in PoC development, particularly given varying legal frameworks across different jurisdictions. Our anonymization procedures were specifically designed to prevent any potential negative consequences for participants while preserving the analytical value of their contributions to cybersecurity research.

6

CONCLUSION

This dissertation has investigated the factors that shape the target selection in IoT vulnerabilities, with a particular focus on how attacker behavior, vulnerability characteristics, exploit development practices, and community dynamics interact to influence target selection and exploitation timelines.

A total of four peer-reviewed studies were presented across Chapters 2 to 5, each contributing to the overarching goal of answering the following research question:

What factors shape the exploitation in IoT vulnerabilities, from target selection to exploit development and prediction?

In this final chapter, we synthesize the empirical findings from Chapters 2 to 5 to address the main research question. We then consider the broader implications of these findings for vulnerability management and IoT security policy. Lastly, we propose directions for future research that build upon the insights developed in this dissertation.

6.1. SUMMARY OF THE FINDINGS

6.1.1. CHAPTER 2: LONGITUDINAL ANALYSIS OF IOT MALWARE EXPLOITS

This chapter examines how IoT malware evolves over time and how long specific vulnerabilities continue to be targeted after public disclosure. By analyzing over 17,000 malware samples collected between 2015 and 2020 using both static and dynamic methods, this study found that only a handful number of IoT vulnerabilities are targeted by attackers and exploited for significantly longer periods than those in traditional computing environments. On average, the time between vulnerability disclosure and observed in malware use was 29 months. We learned that malware authors continuously reuse old exploits, adapting them to work with newer versions of malware. This behavior is reinforced by the fragmented nature of the IoT update ecosystem, where many devices remain unpatched or unsupported long after their release. These findings demonstrate that traditional vulnerability lifecycle models underestimate the persistence of risk in IoT

contexts and highlight the need for security strategies that reflect the long operational life of IoT devices.

6.1.2. CHAPTER 3: INFLUENCING FACTORS IN IOT MALWARE TARGET SELECTION

This study examines the technical and contextual factors that influence how frequently specific IoT vulnerabilities are exploited. The analysis shows that attackers do not target vulnerabilities randomly, but instead focus on those that offer high impact with minimal effort. Command injection vulnerabilities, for example, were found to be exploited over 16 times more frequently than other types. Other key predictors of exploitation include device type, installed base size, and patch complexity, devices that are widely deployed and harder to patch were more likely to be targeted. In contrast, vulnerabilities with available patches or high exploit complexity were less likely to be exploited, suggesting that attackers weigh both opportunity and cost. These findings highlight the strategic nature of attacker behavior and underscore the need for proactive mitigation efforts. Prioritizing attention on vulnerabilities that combine technical severity with broad exposure can help reduce the risk of widespread exploitation as IoT adoption continues to grow.

6.1.3. CHAPTER 4: PREDICTING EXPLOITATION OF IOT VULNERABILITIES

Chapter 4 addresses the challenge of predicting which IoT vulnerabilities are likely to be exploited. It evaluates the performance of current prediction tools like the Exploit Prediction Scoring System (EPSS) and compares them with models that incorporate community signals from underground forums. By analyzing over 23,000 IoT-related CVEs and correlating them with discussions from 25 hacking forums, the study found that EPSS consistently underestimates the exploitation risk of IoT vulnerabilities. For example, CVE-2017-17215, widely exploited in the wild, received a near-zero EPSS score despite years of attacker use. The enhanced model developed in this research improves predictive accuracy by incorporating community data, particularly forum discussions, which often precede real-world exploitation. Vulnerabilities discussed in forums were not only more likely to be exploited but were also targeted more quickly. The model identified 38 statistically significant features, with vendor-specific traits being especially predictive. These findings suggest that effective prediction for IoT vulnerabilities requires different inputs and assumptions than those used for general IT systems, and that attacker community signals are valuable early indicators of risk.

6.1.4. CHAPTER 5: THE ROLE OF POC DEVELOPERS IN EXPLOIT AVAILABILITY

Chapter 5 shifts focus to the people behind exploit development. Through interviews with 16 proof-of-concept (PoC) exploit developers from various regions, this study shows that decisions to publish PoCs are shaped as much by human motivations and ethical reasoning as by technical challenges. PoCs development exists on a spectrum, from simple crash demonstrations to fully weaponized code, and developers often choose what to publish based on their goals, values, and experiences with vendors. Hostile or unresponsive vendors tend to push developers toward full public disclosure, while

collaborative vendors encourage responsible coordination. PoCs developers also prefer working on low-effort, high-impact vulnerabilities, especially when no existing PoCs is available. Cultural and regional factors also shape disclosure behavior. In some regions, public sharing of security research, specifically PoCs, is discouraged, while in others it is seen as a form of recognition and knowledge contribution. The study concludes that PoC developers act as key gatekeepers in the exploit ecosystem, and their decisions directly affect how quickly vulnerabilities are weaponized and whether defenders have the opportunity to respond in time.

6.2. IMPLICATIONS FOR GOVERNANCE AND POLICY MAKING

The governance of IoT security challenges can be examined through three classic coordination mechanisms— **hierarchy**, **market**, and **network**— originally conceptualized by Powell (1990) [206] in organizational theory as distinct modes of coordination, and later adapted to public governance and metagovernance frameworks by Meuleman (2008) [165] and others. Each mechanism reflects different assumptions about control, accountability, and collaboration: hierarchical governance relies on authority and regulation; market governance on competition and incentives; and network governance on trust, reciprocity, and interdependence among actors.

In the context of IoT security, these models offer complementary perspectives for understanding how responsibilities are distributed among states, vendors, and security communities.

6.2.1. HIERARCHICAL GOVERNANCE

Hierarchical governance operates through top-down regulatory systems spanning international, national, and local levels. In IoT security, this structure aims to enforce minimum standards while accommodating national variations in implementation and enforcement priorities. However, the effectiveness of such governance models depends critically on their alignment with the actual patterns of IoT vulnerability exploitation.

Recent years have witnessed unprecedented regulatory activity around IoT security [44]. The European Union's Cyber Resilience Act (CRA) [69] establishes comprehensive requirements for manufacturers, including mandatory notification of designated Computer Security Incident Response Teams (CSIRTs) when vulnerabilities are actively exploited, patching vulnerabilities "without undue delay" [103], and prohibiting insecure products from entering the EU market [52]. Similarly, the UK's Product Security and Telecommunications Infrastructure (PSTI) Act [244] bans default passwords and mandates structured vulnerability disclosure processes. Australia's Cyber Security Act [32] requires compliance statements and prohibits the supply of non-compliant devices, while the United States IoT Cybersecurity Improvement Act focuses primarily on federal procurement standards [247]. The US Cyber Trust Mark program [263] provides voluntary certification requiring adherence to specific cybersecurity criteria. Countries including Singapore, Finland, and Germany have introduced voluntary labeling schemes to help consumers identify secure devices [96].

Coordinating these national efforts, the European Union Agency for Cybersecurity (ENISA) [98] develops certification frameworks under the Cybersecurity Act and supports

consistent IoT security baseline requirements across member states [99, 101]. The guidelines by ENISA address both individual device security [99] and supply chain security [100], recognizing the multi-jurisdictional risks inherent in globally distributed manufacturing. National Cybersecurity Certification Authorities (NCCAs) [104] represent progress toward independent enforcement capabilities, establishing baseline security expectations such as prohibiting default credentials and requiring secure update mechanisms.

However, our empirical analysis reveals fundamental misalignments between current regulatory approaches and actual IoT exploitation patterns. The enforcement challenges facing hierarchical governance stem from several structural issues. First, national regulations often diverge significantly, creating regulatory complexity for global manufacturers who must navigate multiple compliance frameworks simultaneously [123]. Second, online platforms frequently bypass jurisdictional boundaries, making enforcement difficult when devices are sold across borders [176]. Third, global supply chains complicate enforcement due to the mismatch between where devices are manufactured and where they are ultimately deployed [107, 123, 176, 205].

Most critically, our findings demonstrate that IoT vulnerability exploitation follows extended timelines that are poorly aligned with policies designed for rapid response. The typical progression from vulnerability disclosure to active exploitation spans years rather than months, with devices remaining vulnerable long after their official end-of-life periods. This temporal mismatch suggests that current regulatory frameworks, which often assume relatively quick exploitation cycles, may be fundamentally misaligned with the threat landscape they aim to address.

Our analysis also reveals that IoT exploitation concentrates among few and specific manufacturers, with vendors such as Huawei and D-Link featuring prominently in exploitation patterns. As demonstrated in chapter 4, 10 of the top 30 predictive features for IoT exploitation were vendor-specific, indicating consistent patterns of security neglect among certain manufacturers. This concentration suggests that vendor-focused oversight may be more effective than general-purpose vulnerability scoring systems like the Exploit Prediction Scoring System (EPSS).

These findings have direct implications for regulatory design. Governments should mandate extended support periods that align with the 29-month average IoT exploitation timeline identified in our research, as chapter 3 confirmed that end-of-life devices remain vulnerable for years after manufacturer support ends. Regulatory frameworks should also prioritize enforcement toward high-risk vendors rather than applying uniform standards across all manufacturers, given the clear concentration of exploitation among specific companies.

Some jurisdictions are already experimenting with alternative enforcement approaches. Australia's extraterritorial provisions require manufacturer compliance regardless of production location [32], while the Netherlands has explored retailer responsibility for security updates [88]. However, international retailers like MediaMarkt and Coolblue face significant challenges in maintaining direct manufacturer relationships necessary for such approaches to succeed.

Future governance models must evolve beyond reactive policies that respond to incidents after exploitation has occurred. Instead, they should support proactive monitoring and enforcement mechanisms that reflect the extended timelines and evolving nature

of IoT threats. This includes developing governance frameworks that can adapt to new technologies and ensuring that regulatory evolution keeps pace with innovation in the IoT ecosystem. The integration of empirical threat intelligence, such as the patterns identified in this dissertation, represents a crucial step toward evidence-based IoT security governance that can effectively address the challenges of a globally connected device landscape.

6.2.2. NETWORK GOVERNANCE

Network governance, based on collaboration and shared processes, is well-suited for addressing the fragmented nature of IoT security. The prolonged delay between IoT vulnerability disclosure and exploitation signals coordination problems where vulnerabilities remain exposed for years due to inadequate governance structures. Existing Coordinated Vulnerability Disclosure (CVD) programs [129] often fall short, operating as vendor-driven processes excluding security researchers and PoC exploit developers, rather than inclusive multi-stakeholder mechanisms [178, 259].

Network governance, based on collaboration and shared processes, is well-suited for addressing the fragmented nature of IoT security. The prolonged delay between IoT vulnerability disclosure and exploitation signals coordination problems where vulnerabilities remain exposed for years due to inadequate governance structures. Existing Coordinated Vulnerability Disclosure (CVD) programs [129] often fall short, operating as vendor-driven processes excluding security researchers and PoC exploit developers, rather than inclusive multi-stakeholder mechanisms [178, 259]. Effective CVD programs require defined response timelines reflecting the extended lifespans of IoT vulnerabilities, communication protocols discouraging hostile vendor reactions that push researchers toward full disclosure, persistent coordination channels supporting multi-year vulnerability lifecycles, and accountability mechanisms for unresponsive vendors, as observed in this dissertation.

Understanding the human dimension of these coordination failures is crucial for developing effective governance responses. The interviews conducted in chapter 5 revealed that PoC exploit developers choose public disclosure of vulnerabilities and weaponized PoC exploits when facing silent patching, unresponsiveness, or hostility from vendors [70]. Disclosure decisions are shaped by vendor behavior, peer norms, and cultural expectations, requiring governance frameworks that build trust and ensure fair treatment [193]. Analysis in chapter 4 further demonstrated that underground forum discussions strongly predict exploitation, highlighting how security researcher frustration with unresponsive vendors can contribute to weaponization timelines. These behavioral insights indicate that network governance must integrate security researchers and PoC exploit developers alongside manufacturers, ISPs, and public sector stakeholders through multi-stakeholder working groups including academics, NGOs, and industry associations [193].

Addressing these coordination challenges requires robust institutional infrastructure. Critical to network governance success are established cybersecurity coordination organizations including Computer Emergency Response Teams (CERTs) [224], Computer Security Incident Response Teams (CSIRTs) [76], and the Forum of Incident Response and Security Teams (FIRST) [110]. Organizations like the Dutch Institute for Vulnerability Disclosure (DIVD) [89] and Shadowserver Foundation [240] demonstrate how non-profit

organizations fill coordination gaps through vulnerability research, notification services, and threat intelligence sharing complementing traditional CVD processes. The findings in chapter 5 also highlighted that researchers prefer reporting to national vulnerability disclosure programs to avoid vendor confrontation, legal complications, and overwhelming communication demands. Expanding neutral intermediary roles could reduce hostile disclosure scenarios leading to weaponized PoC releases. These organizations can systematically track vendor behavior patterns regarding disclosure responsiveness, patching timelines, and security advisory quality, providing evidence-based assessments to regulatory bodies like NCCAs for enforcement decisions including oversight, penalties, or market bans for persistently non-compliant vendors.

Building on these institutional foundations, public-private partnerships represent core network governance components [53], allowing governments to monitor technical threats while offering businesses predictable compliance paths. These partnerships require clear accountability mechanisms and performance metrics aligning commercial goals with security outcomes [53, 58]. The predictive insights from underground forum analysis suggest that governance models should ethically incorporate these signals for early warnings while respecting privacy and legal constraints. However, regional differences in disclosure handling require flexible frameworks supporting global coordination while adapting to local legal and cultural constraints.

Ultimately, effective network governance depends on international coordination that builds upon these national and regional efforts. International coordination through the Internet Engineering Task Force (IETF) [223] and global industry alliances [110, 133, 274] must incorporate predictive threat models and longitudinal intelligence sharing crucial for IoT vulnerability coordination [174]. These frameworks should leverage the behavioral insights and institutional capabilities developed at national levels while addressing the cross-border nature of IoT threats and the global scope of vulnerability disclosure challenges.

6.2.3. MARKET GOVERNANCE

Market governance mechanisms attempt to address IoT security challenges through economic incentives rather than regulatory mandates. Our research identifies three fundamental market failures that undermine IoT security: manufacturers externalize long-term security costs onto users and society, severe information asymmetries prevent consumers from evaluating device security, and temporal mismatches exist between devices' economic lifecycles and their security maintenance requirements [94, 170]. The 29-month vulnerability-to-exploitation timeline exemplifies these failures, as manufacturers typically cease support while users continue depending on devices for years, creating situations where manufacturer incentives fundamentally misalign with user security needs.

The core challenge lies in information asymmetries that prevent consumers from making informed security decisions. Most buyers cannot evaluate device security before purchase, leading to decisions based primarily on price and features rather than security considerations [94, 95]. This creates conditions where insecure devices can compete effectively against secure alternatives, as the security differences remain invisible to consumers until after a compromise occurs. Secure devices generate benefits for the

broader network by reducing compromise risks, but manufacturers cannot capture these positive externalities in their pricing [170]. Conversely, insecure devices create negative externalities as end-of-life devices continue posing risks years after vendor support ends, costs that manufacturers do not bear directly.

Information-based mechanisms represent the most developed approach to addressing these market failures. Singapore's Cybersecurity Labelling Scheme (CLS) [79] addresses knowledge gaps by assigning security ratings to smart devices, enabling consumers to incorporate cybersecurity considerations into purchasing decisions. The CLS employs a hybrid model with mandatory certification for critical devices like WiFi routers while maintaining voluntary participation for other device categories. This creates flexible structures that encourage broader industry participation while establishing minimum security baselines for high-risk devices.

However, the effectiveness of information-based mechanisms depends critically on consumer responsiveness to security information. Even with clear labeling, buyers may prioritize price or convenience over security protection, particularly when security benefits remain abstract until a compromise occurs. This rational ignorance presents a fundamental challenge to market-based approaches, as perfect information does not guarantee optimal security outcomes if consumers systematically undervalue protection [123].

Incentive-based mechanisms, including insurance and liability policies, could potentially align long-term security risks with economic responsibility. Cyber insurance could theoretically internalize the risks associated with our observed extended exploitation window by requiring strong security practices for lower premiums, providing ongoing financial incentives for device hardening and vulnerability management beyond the point of sale [119]. Such mechanisms would create continuous economic pressure for security maintenance throughout device lifecycles rather than just at the design phase [170]. However, the effectiveness of insurance mechanisms depends critically on their interaction with liability frameworks. Insurance functions most effectively when paired with clear liability standards that define manufacturer responsibility, as insurers use liability exposure to price premiums and structure coverage. Without well-defined liability, insurers struggle to assess risks accurately, leading to either prohibitively expensive premiums or overly broad coverage that encourages moral hazard.

Current cyber insurance markets, however, lack IoT-specific risk assessment capabilities due to the complexity and diversity of IoT technologies, creating significant information gaps between insurers and manufacturers [93]. The interconnected nature of IoT systems introduces systemic risks where single vulnerabilities could trigger widespread claims that insurers may struggle to handle [93]. Additionally, moral hazard concerns arise when insurance coverage potentially reduces manufacturer motivation to invest in security measures, as the financial consequences of security failures become externalized to insurers. This moral hazard problem becomes particularly acute in the absence of complementary liability standards, if manufacturers face minimal legal consequences for security failures while maintaining insurance coverage, the economic incentive to invest in security diminishes substantially.

Liability-based approaches that hold manufacturers legally responsible for security failures face substantial obstacles in the IoT context. Proving causation between manu-

facturer actions and downstream harm remains difficult for intangible consequences like data breaches or privacy violations, unlike traditional physical product liability cases [170]. The exploitation delay identified in our research further complicates efforts to link manufacturer decisions to eventual security incidents [181]. Strict liability standards may also create unintended consequences by pushing smaller manufacturers from markets or encouraging relocation to jurisdictions with weaker regulatory oversight, potentially widening security gaps across the industry [93]. Yet these liability challenges directly impact insurance market development. Insurers require predictable liability frameworks to price policies accurately and determine coverage limits. The current uncertainty around IoT liability, particularly regarding who bears responsibility across complex supply chains and what constitutes reasonable security practices, prevents the emergence of mature insurance markets. Conversely, well-functioning insurance markets could provide valuable data to inform liability standards, as insurers collect extensive information about security failures, their causes, and associated costs [93].

The interdependence between liability and insurance creates both challenges and opportunities for IoT security governance. A coordinated approach could leverage their complementary strengths: liability frameworks establish baseline security expectations and accountability, while insurance mechanisms provide continuous economic incentives for security investment and generate empirical data about risk patterns. For example, tiered liability standards could exempt manufacturers from certain liabilities if they maintain specified insurance coverage and adhere to defined security practices, creating a feedback loop where insurance requirements operationalize liability standards while liability frameworks provide the structure insurers need for viable markets [170]. However, the temporal characteristics revealed in our research, particularly the 29-month vulnerability-to-exploitation timeline and extended exploitation periods, complicate this relationship, as both liability claims and insurance payouts may occur years after the initial security failure, making causation and risk assessment more challenging for both mechanisms.

Platform-based mechanisms offer more immediate and scalable approaches to improving IoT security. Bug bounty platforms like HackerOne, Bugcrowd, and Synack provide structured marketplaces for vulnerability discovery and responsible disclosure, offering economic incentives that compete with malicious motivations for exploit development [28, 86]. Our analysis in chapter 5 identified proof-of-concept exploit developers as key decision-makers in exploit availability, highlighting the importance of these mechanisms in shaping vulnerability trajectories before they reach malicious actors.

Traditional coordinated vulnerability disclosure (CVD) models, however, fail to scale effectively in IoT ecosystems due to unresponsive vendors, lack of dedicated security teams, and operational coordination challenges across thousands of fragmented manufacturers [86]. Platform-based approaches may inadvertently create security disparities by concentrating researcher attention on well-funded organizations while neglecting devices from smaller manufacturers, potentially reinforcing systemic asymmetries in exploitability patterns [28].

Our findings demonstrate that IoT security cannot rely solely on market mechanisms to address the fundamental challenges identified in this research. The extended exploitation timelines, vendor concentration patterns, and information asymmetries require

complementary governance interventions. Market mechanisms work best when combined with regulatory mandates, vulnerability coordination hubs, and industry standards that address disclosure gaps and promote more equitable security outcomes across the diverse IoT ecosystem. The challenge lies in designing market incentives that complement rather than substitute for other governance approaches, recognizing that economic mechanisms alone cannot overcome the structural characteristics that make IoT security a persistent challenge.

6.3. FUTURE WORK

This dissertation lays the groundwork for a richer understanding of IoT vulnerability exploitation, yet, it also reveals several promising directions for future research that could further advance our understanding of IoT vulnerability exploitation and attackers behavior to improve defensive strategies.

DEVELOPING IOT-SPECIFIC PREDICTION MODELS

Chapter 4 demonstrated that existing prediction systems like EPSS often underestimate the risk of IoT vulnerabilities. This highlights the need for models specifically designed for IoT, rather than adapting tools built for general-purpose IT systems. Future work should focus on building prediction models that reflect the unique characteristics of IoT environments from the ground up. Chapters 2 and 3 showed that factors such as device type and install base are strong predictors of exploitation, but these features currently require manual collection. Future research should explore automated methods for gathering this information at scale, such as internet-wide scanning, device fingerprinting, or integration with manufacturer databases. Once collected, these features can be combined with real-time attacker signals to improve predictive performance. Machine learning techniques also offer promise. Natural language processing, for instance, could be used to monitor underground forums and identify early indicators of attacker interest in new vulnerabilities. This would give defenders more time to respond before exploitation begins.

MODELING REALISTIC ATTACKER BEHAVIOR

While this dissertation draws on the notion of "work-averse" attacker behavior, introduced in prior work by [14], to contextualize the role of PoC exploit developers, future research could further formalize attacker utility models using behavioral game theory, attacker personas, or reinforcement learning frameworks. Current models often assume rational, profit-maximizing attackers, but real attacker behavior may be influenced by cognitive biases, social factors, and bounded rationality. These more realistic behavioral models could improve predictions of exploit adoption timelines and simulate how attackers respond to new governance or market interventions, such as increased bug bounty rewards, stricter liability laws, or labeling schemes. Additionally, such models could help predict attacker responses to specific IoT security measures, including automatic update mechanisms, device isolation strategies, or network segmentation approaches.

INVESTIGATING REGIONAL AND CULTURAL VARIATIONS

Chapter 5 revealed that cultural norms and legal environments influence how PoC exploit developers approach disclosure. However, this area remains underexplored. Future work

should include broader studies across regions to better understand how cultural, legal, and economic factors shape exploit development and disclosure decisions. Comparative research could also examine whether attacker behavior varies by geography. For example, attackers in different regions may focus on different device types or target vulnerabilities based on regional infrastructure or market penetration. These insights could inform location-specific defense strategies and improve our understanding of how IoT threats evolve in different parts of the world.

BIBLIOGRAPHY

[1] Omer Akgul, Taha Eghtesad, Amit Elazari, Omprakash Gnawali, Jens Grossklags, Michelle L. Mazurek, Daniel Votipka, and Aron Laszka. 2023. Bug Hunters' Perspectives on the Challenges and Benefits of the Bug Bounty Ecosystem. In *32nd USENIX Security Symposium*. USENIX Association, Anaheim, CA, 2275–2291. https://www.usenix.org/conference/usenixsecurity23/presentation/akgul

[2] Shanjida Akhter. 2015. *Generalized Linear Modeling for Cottage Insurance Data*. Master's thesis. University of Oslo.

[3] Arwa Abdulkarim Al Alsadi, Kaichi Sameshima, Jakob Bleier, Katsunari Yoshioka, Martina Lindorfer, Michel van Eeten, and Carlos H. Gañán. 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22)*. Association for Computing Machinery, New York, NY, USA, 309–321. https://doi.org/10.1145/3488932.3517408

[4] Arwa Abdulkarim Al Alsadi, Kaichi Sameshima, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2023. Bin There, Target That: Analyzing the Target Selection of IoT Vulnerabilities in Malware Binaries. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23)*. Association for Computing Machinery, New York, NY, USA, 513–526. https://doi.org/10.1145/3607199.3607241

[5] Arwa Abdulkarim Al Alsadi, Mathew Vermeer, Takayuki Sasaki, Katsunari Yoshioka, Michel Van Eeten, and Carlos Gañán. 2025. Bits and Pieces: Piecing Together Factors of IoT Vulnerability Exploitation. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security (ASIA CCS '25)*. Association for Computing Machinery, New York, NY, USA, 1032–1049. https://doi.org/10.1145/3708821.3733875

[6] Yasser Alhelaly, Gurpreet Dhillon, and Tiago Oliveira. 2023. When expectation fails and motivation prevails: the mediating role of awareness in bridging the expectancy-capability gap in mobile identity protection. *Computers & Security* 134 (2023), 103470. https://doi.org/10.1016/j.cose.2023.103470

[7] Michael Patrick Allen. 1997. The problem of multicollinearity. In *Understanding Regression Analysis*. Springer US, Boston, MA, Chapter 37, 176–180. https://doi.org/10.1007/978-0-585-25657-3_37

[8] Global Cyber Alliance. 2022. *GCA Internet Integrity Papers: IoT Policy and Attack Report II*. Technical Report. Global Cyber Alliance.

https://www.globalcyberalliance.org/reports_publications/iot-policy-and-attack-report-ii/.

[9] Luca Allodi. 2017. Economic Factors of Vulnerability Trade and Exploitation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1483–1499. https://doi.org/10.1145/3133956.3133960

[10] Luca Allodi and Fabio Massacci. 2012. A Preliminary Analysis of Vulnerability Scores for Attacks in Wild: The Ekits and Sym Datasets *(BADGERS '12)*. Association for Computing Machinery, New York, NY, USA, 17–24. https://doi.org/10.1145/2382416.2382427

[11] Luca Allodi and Fabio Massacci. 2014. Comparing Vulnerability Severity and Exploits Using Case-Control Studies. *ACM Trans. Inf. Syst. Secur.* 17, 1, Article 1 (aug 2014), 20 pages. https://doi.org/10.1145/2630069

[12] Luca Allodi and Fabio Massacci. 2017. Attack Potential in Impact and Complexity. In *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES '17)*. ACM, 1–6. https://doi.org/10.1145/3098954.3098965

[13] Luca Allodi and Fabio Massacci. 2017. Security Events and Vulnerability Data for Cybersecurity Risk Estimation. *Risk Analysis* 37 (2017). https://api.semanticscholar.org/CorpusID:22919745

[14] Luca Allodi, Fabio Massacci, and Julian Williams. 2022. The work-averse cyberattacker model: theory and evidence from two million attack signatures. *Risk Analysis* 42, 8 (2022), 1623–1642. https://doi.org/10.1111/risa.13732

[15] Luca Allodi, Woohyun Shim, and Fabio Massacci. 2013. Quantitative Assessment of Risk Reduction with Cybercrime Black Market Monitoring. In *2013 IEEE Security and Privacy Workshops*. 165–172. https://doi.org/10.1109/SPW.2013.16

[16] Mohammed Almukaynizi, Eric Nunes, Krishna Dharaiya, Manoj Senguttuvan, Jana Shakarian, and Paulo Shakarian. 2019. *Patch before exploited: An approach to identify targeted software vulnerabilities*. Springer Science and Business Media Deutschland GmbH, 81–113. https://doi.org/10.1007/978-3-319-98842-9_4

[17] Noura Alomar, Primal Wijesekera, Edward Qiu, and Serge Egelman. 2020. "You've Got Your Nice List of Bugs, Now What?" Vulnerability Discovery and Management Processes in the Wild. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, 319–339. https://www.usenix.org/conference/soups2020/presentation/alomar

[18] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP.2019.00013

[19] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Ryan Court, Kevin Snow, Fabian Monrose, and Manos Antonakakis. 2021. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle. In *30th USENIX Security Symposium*. USENIX Association, 3505–3522. https://www.usenix.org/conference/usenixsecurity21/presentation/alrawi-circle

[20] Carlos A. Rivera Alvarez, Arash Shaghaghi, David D. Nguyen, and Salil S. Kanhere. 2021. Is this IoT Device Likely to be Secure? Risk Score Prediction for IoT Devices Using Gradient Boosting Machines. arXiv:cs.CR/2111.11874 https://arxiv.org/abs/2111.11874

[21] Aman Mishra. 2025. Researchers Uncover Hacking Tools and Techniques Shared on Russian-Speaking Cybercrime Forums. GBHackers. https://gbhackers.com/researchers-uncover-hacking-tools-and-techniques/#google_vignette

[22] P. Anand, Y. Singh, A. Selwal, P. K. Singh, and K. Z. Ghafoor. 2021. ivqfiot: an intelligent vulnerability quantification framework for scoring internet of things vulnerabilities. *Expert Systems* 39 (2021). Issue 5. https://doi.org/10.1111/exsy.12829

[23] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium*. USENIX Association, Vancouver, BC, 1093–1110. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[24] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium*. USENIX Association, Vancouver, BC, 1093–1110. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[25] Argosopentech. Accessed: 2025. Argos Translate Package Index. https://www.argosopentech.com/argospm/index/

[26] Marc Arnaert, Yoann Bertrand, and Karima Boudaoud. 2016. Modeling Vulnerable Internet of Things on SHODAN and CENSYS : An Ontology for Cyber Security. In *International Conference on Emerging Security Information, Systems and Technologies*. https://personales.upv.es/thinkmind/SECURWARE/SECURWARE_2016/securware_2016_14_40_30129.html

[27] Ashish Arora, Anand Nandkumar, and Rahul Telang. 2006. Does information security attack frequency increase with vulnerability disclosure? An empirical analysis.

Information Systems Frontiers 8 (01 2006), 350–362. https://doi.org/10.1007/s10796-006-9012-5

[28] Junaid Arshad, Muhammad Talha, Bilal Saleem, Zoha Shah, Huzaifa Zaman, and Zia Muhammad. 2024. A Survey of Bug Bounty Programs in Strengthening Cybersecurity and Privacy in the Blockchain Industry. *Blockchains* 2, 3 (2024), 195–216. https://doi.org/10.3390/blockchains2030010

[29] Soodeh Atefi, Amutheezan Sivagnanam, Afiya Ayman, Jens Grossklags, and Aron Laszka. 2023. The Benefits of Vulnerability Discovery and Bug Bounty Programs: Case Studies of Chromium and Firefox. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*. Association for Computing Machinery, New York, NY, USA, 2209–2219. https://doi.org/10.1145/3543507.3583352

[30] ATLAS.ti. 2025. ATLAS.ti: The Qualitative Data Analysis & Research Software. https://atlasti.com/.

[31] AttackerKB. 2024. AttackerKB. https://attackerkb.com.

[32] Australian Government. 2024. Cyber Security Act 2024. https://www.legislation.gov.au/C2024A00098

[33] Jessy Ayala, Yu-Jye Tung, and Joshua Garcia. 2025. *A mixed-methods study of open-source software maintainers on vulnerability management and platform security features.* USENIX Association, USA.

[34] Maria Bada and Ildiko Pete. 2020. An exploration of the cybercrime ecosystem around Shodan. In *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. 1–8. https://doi.org/10.1109/IOTSMS52051.2020.9340224

[35] Balbix. 2024. White Paper - Essential Guide to Prioritized Patching. https://info.balbix.com/prioritized-patching

[36] Randa Basheer, Bassel Alkhatib, and Zhiyong Xu. 2021. Threats from the Dark: A Review over Dark Web Investigation Research for Cyber Threat Intelligence. *J. Comput. Netw. Commun.* 2021 (jan 2021), 21. https://doi.org/10.1155/2021/1302999

[37] Pierpaolo Basile, Elio Musacchio, Marco Polignano, Lucia Siciliani, Giuseppe Fiameni, and Giovanni Semeraro. 2023. LLaMAntino: LLaMA 2 Models for Effective Text Generation in Italian Language. arXiv:cs.CL/2312.09993 https://arxiv.org/abs/2312.09993

[38] Genge Bela and Calin Enachescu. 2015. ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services. *Security and Communication Networks* 9 (04 2015). https://doi.org/10.1002/sec.1262

[39] H Russell Bernard, Amber Wutich, and Gery W Ryan. 2016. *Analyzing qualitative data: Systematic approaches.* SAGE publications.

[40] Mats Björklund. 2019. Be careful with your principal components. *Evolution* 73, 10 (2019), 2151–2158. https://doi.org/10.1111/evo.13835

[41] Friedhelm Bliemel. 1973. Theil's Forecast Accuracy Coefficient: A Clarification. *Journal of Marketing Research* 10, 4 (1973), 444–446. https://doi.org/10.2307/3149394 Accessed: 2025-02-27.

[42] Grzegorz Blinowski and Paweł Piotrowski. 2020. *CVE Based Classification of Vulnerable IoT Systems.* Springer Nature, 82–93. https://doi.org/10.1007/978-3-030-48256-5_9

[43] Grzegorz Blinowski, Paweł Piotrowski, and Michał Wiśniewski. 2021. Comparing Support Vector Machine and Neural Network Classifiers of CVE Vulnerabilities. (01 2021), 734–740. https://doi.org/10.5220/0010574807340740

[44] Tayssir Bouraffa and Kai-Lung Hui. 2025. Regulating Information and Network Security: Review and Challenges. *ACM Comput. Surv.* 57, 5, Article 126 (Jan. 2025), 38 pages. https://doi.org/10.1145/3711124

[45] Brennen Bouwmeester, Elsa Rodríguez, Carlos Gañán, Michel van Eeten, and Simon Parkin. 2021. "The Thing Doesn't Have a Name": Learning from Emergent Real-World Interventions in Smart Home Security. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. USENIX Association, 493–512. https://www.usenix.org/conference/soups2021/presentation/bouwmeester

[46] Laura J. Bowman. 2022. Statista. *Journal of Business & Finance Librarianship* 27, 4 (2022), 304–309. https://doi.org/10.1080/08963568.2022.2087018

[47] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2010. Beyond heuristics: learning to classify vulnerabilities and predict exploits *(KDD '10)*. Association for Computing Machinery, New York, NY, USA, 105–114. https://doi.org/10.1145/1835804.1835821

[48] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[49] Calvin Brierley, Jamie Pont, Budi Arief, David J. Barnes, and Julio Hernandez-Castro. 2020. Persistence in Linux-Based IoT Malware. In *Secure IT Systems: 25th Nordic Conference, NordSec 2020, Virtual Event, November 23–24, 2020, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 3–19. https://doi.org/10.1007/978-3-030-70852-8_1

[50] Peter Bruce and Andrew Bruce. 2017. *Practical Statistics for Data Scientists*. O'Reilly Media. https://datapot.vn/wp-content/uploads/2023/12/datapot.vn-Practical-Statistics-for-Data-Scientists.pdf?srsltid=AfmBOornMxQWM_k6V0idDF657VazYSQjhIVf7Iu3qsR1xqfpiAbpK4_P

[51] Cambridge. 2024. Department of Computer Science and Technology: Cambridge Cybercrime Centre. https://www.cambridgecybercrime.uk/.

[52] Polona Car and Stefano De Luca. 2024. EU Cyber Resilience Act. https://www.europarl.europa.eu/RegData/etudes/BRIE/2022/739259/EPRS_BRI(2022)739259_EN.pdf Fourth edition.

[53] Madeline Carr. 2016. Public-private partnerships in national cyber-security strategies. *International Affairs* 92 (01 2016), 43–62. https://doi.org/10.1111/1468-2346.12504

[54] Santiago Castro. 2017. Fast Krippendorff: Fast computation of Krippendorff's alpha agreement measure. https://github.com/pln-fing-udelar/fast-krippendorff.

[55] Santiago Castro. 2017. Fast Krippendorff: Fast computation of Krippendorff's alpha agreement measure. https://github.com/pln-fing-udelar/fast-krippendorff.

[56] Orçun Çetin, Carlos Gañán, Lisette Altena, Takahiro Kasama, Daisuke Inoue, Kazuki Tamiya, Ying Tie, Katsunari Yoshioka, and Michel van Eeten. 2019. Cleaning Up the Internet of Evil Things: Real-World Evidence on ISP and Consumer Efforts to Remove Mirai. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/cleaning-up-the-internet-of-evil-things-real-world-evidence-on-isp-and-consumer-efforts-to-remove-mirai/

[57] Stefanos Chaliasos, Marcos Antonios Charalambous, Liyi Zhou, Rafaila Galanopoulou, Arthur Gervais, Dimitris Mitropoulos, and Benjamin Livshits. 2024. Smart Contract and DeFi Security Tools: Do They Meet the Needs of Practitioners?. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 60, 13 pages. https://doi.org/10.1145/3597503.3623302

[58] Alexa Charles. Accessed June 1, 2025. Public–Private Partnerships in Cybersecurity. https://www.prevention-cybercrime.ca/_files/ugd/9d4ef1_4a135253f5dd4974bcd693c15989bf98.pdf

[59] Kathy Charmaz. 2006. *Constructing grounded theory : a practical guide through qualitative analysis*. Sage Publications, London; Thousand Oaks, Calif. http://www.amazon.com/Constructing-Grounded-Theory-Qualitative-Introducing/dp/0761973532

[60] Haipeng Chen, Rui Liu, Noseong Park, and V.S. Subrahmanian. 2019. Using Twitter to Predict When Vulnerabilities will be Exploited. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. Association for Computing Machinery, New York, NY, USA, 3143–3152. https://doi.org/10.1145/3292500.3330742

[61] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[62] Xiaowei Chen, Sophie Doublet, Anastasia Sergeeva, Gabriele Lenzini, Vincent Koenig, and Verena Distler. 2024. What Motivates and Discourages Employees in Phishing Interventions: An Exploration of Expectancy-Value Theory. In *Twentieth Symposium on Usable Privacy and Security (SOUPS 2024)*. USENIX Association, Philadelphia, PA, 487–506. https://www.usenix.org/conference/soups2024/presentation/chen

[63] Samuel Chng, Han Yu Lu, Ayush Kumar, and David Yau. 2022. Hacker types, motivations and strategies: A comprehensive framework. *Computers in Human Behavior Reports* 5 (2022), 100167. https://doi.org/10.1016/j.chbr.2022.100167

[64] Jinchun Choi, Afsah Anwar, Abdulrahman Alabduljabbar, Hisham Alasmary, Jeffrey Spaulding, An Wang, Songqing Chen, DaeHun Nyang, Amro Awad, and David Mohaisen. 2022. Understanding Internet of Things malware by analyzing endpoints in their static artifacts. *Computer Networks* 206 (2022), 108768. https://doi.org/10.1016/j.comnet.2022.108768

[65] Jinchun Choi, Afsah Anwar, Hisham Alasmary, Jeffrey Spaulding, DaeHun Nyang, and Aziz Mohaisen. 2019. IoT malware ecosystem in the wild: a glimpse into analysis and exposures. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (SEC '19)*. Association for Computing Machinery, New York, NY, USA, 413–418. https://doi.org/10.1145/3318216.3363379

[66] CISA. 2023. Known Exploited Vulnerabilities Catalog | CISA. https://www.cisa.gov/known-exploited-vulnerabilities-catalog.

[67] CloudSEK. 2023. How Cybercriminals Utilize Dark Web Forums for Collaboration and Trade. https://www.cloudsek.com/blog/how-cybercriminals-utilize-dark-web-forums-for-collaboration-and-trade

[68] Gabriella Coleman. 2012. *Coding Freedom: The Ethics and Aesthetics of Hacking*. Princeton University Press. 268 pages. https://press.princeton.edu/books/paperback/9780691144610/coding-freedom

[69] European Commission. 2021. The European Union's Cyber Resilience Act (CRA). https://www.european-cyber-resilience-act.com/.

[70] Copper Horse. 2021. *The Contemporary Use of Vulnerability Disclosure in IoT*. Technical Report. IoT Security Foundation (IoTSF). https://iotsecurityfoundation.org/wp-content/uploads/2021/11/The-Contemporary-Use-of-Vulnerability-Disclosure-in-IoT-IoTSF-Report-4-November-2021.pdf

[71] Andrei Costin and Jonas Zaddach. 2018. IoT Malware: Comprehensive Survey, Analysis Framework and Case Studies. *BlackHat USA* (2018). https://api.semanticscholar.org/CorpusID:198973256

[72] Louis Anthony Tony Cox. 2008. Some limitations of "Risk = Threat x Vulnerability x Consequence" for risk analysis of terrorist attacks. *Risk analysis : an official publication of the Society for Risk Analysis* 28, 6 (December 2008), 1749—1761. https://doi.org/10.1111/j.1539-6924.2008.01142.x

[73] Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. 2018. Understanding Linux Malware. In *2018 IEEE Symposium on Security and Privacy (SP)*. 161–175. https://doi.org/10.1109/SP.2018.00054

[74] Emanuele Cozzi, Pierre-Antoine Vervier, Matteo Dell'Amico, Yun Shen, Leyla Bilge, and Davide Balzarotti. 2020. The Tangled Genealogy of IoT Malware. In *Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC '20)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3427228.3427256

[75] Emanuele Cozzi, Pierre-Antoine Vervier, Matteo Dell'Amico, Yun Shen, Leyla Bilge, and Davide Balzarotti. 2021. The Tangled Genealogy of IoT Malware Dataset. https://github.com/eurecom-s3/tangled_iot/tree/master/dataset

[76] CSIRT, Computer Security Incident Response Team. 2001. https://www.csirt.org/.

[77] CUJO AI. 2023. The Zerobot Botnet: Vulnerabilities Targeted and Exploits Used in Detail. https://cujo.com/blog/the-zerobot-botnet-vulnerabilities-targeted-and-exploits-used-in-detail/.

[78] CUJO AI. 2024. IoT Botnet Report 2021: Malware and Vulnerabilities Targeted. https://cujo.com/blog/iot-botnet-report-2021-malware-and-vulnerabilities-targeted/.

[79] Cyber Security Agency of Singapore (CSA). 2025. About Cybersecurity Labelling Scheme for IoT. https://www.csa.gov.sg/our-programmes/certification-and-labelling-schemes/cybersecurity-labelling-scheme/about

[80] CyberIOCs. 2020. CyberIOCs Daily malware pack. https://freeiocs.cyberiocs.pro

[81] CyberPeace. 2024. Rapid Exploitation of Proof-of-Concept Exploits: A Growing Cybersecurity Threat. https://www.cyberpeace.org/resources/blogs/rapid-exploitation-of-proof-of-concept-exploits-a-growing-cybersecurity-threat

[82] Fan Dang, Zhenhua Li, Yunhao Liu, Ennan Zhai, Qi Alfred Chen, Tianyin Xu, Yan Chen, and Jingyu Yang. 2019. Understanding Fileless Attacks on Linux-based IoT Devices with HoneyCloud. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 482–493. https://doi.org/10.1145/3307334.3326083

[83] Wenjing Dang, Kaixuan Li, Sen Chen, Zhenwei Zhuo, Lyuye Zhang, and Zheli Liu. 2025. Real-World Usability of Vulnerability Proof-of-Concepts: A Comprehensive Study. arXiv:cs.SE/2510.18448 https://arxiv.org/abs/2510.18448

[84] Dell. 2018. Meltdown/Spectre (CVE-2017-5715, CVE-2017-5753, CVE-2017-5754): Impact on Dell Products. https://www.dell.com/support/kbdoc/nl-nl/000177783/meltdown-spectre-cve-2017-5715-cve-2017-5753-cve-2017-5754-effect-op-dell-producten.

[85] Antoine d'Estalenx and Carlos Gañán. 2022. NURSE: eNd-UseR IoT malware detection tool for Smart homEs. In *Proceedings of the 11th International Conference on the Internet of Things (IoT '21)*. Association for Computing Machinery, New York, NY, USA, 134–142. https://doi.org/10.1145/3494322.3494340

[86] Aaron Yi Ding, Gianluca Limon De Jesus, and Marijn Janssen. 2019. Ethical hacking for boosting IoT vulnerability management: a first look into bug bounty programs and responsible disclosure. In *Proceedings of the Eighth International Conference on Telecommunications and Remote Sensing (ICTRS 2019)*. ACM, 49–55. https://doi.org/10.1145/3357767.3357774

[87] Thomas Douglas. 2013. *An Expected-Value Approach to the Dual-Use Problem*. ANU Press, 133–152. http://www.jstor.org/stable/j.ctt5hgz15.14

[88] Dutch Government / Business.gov.nl. 2022. Experiment assigning security update responsibility to retailers in the Netherlands. https://business.gov.nl/products-services-and-innovation/product-safety/rules-for-smart-devices-and-digital-products/

[89] Dutch Institute for Vulnerability Disclosure(DIVD). 2019. https://www.divd.nl/.

[90] Jacquelynne S. Eccles and Allan Wigfield. 2020. From expectancy-value theory to situated expectancy-value theory: A developmental, social cognitive, and sociocultural perspective on motivation. *Contemporary Educational Psychology* 61 (2020), 101859. https://doi.org/10.1016/j.cedpsych.2020.101859

[91] Michel Edkrantz. 2015. Predicting Exploit Likelihood for Cyber Vulnerabilities with Machine Learning. https://api.semanticscholar.org/CorpusID:60325485

[92] Michel Edkrantz, Staffan Truve, and Alan Said. 2015. Predicting Vulnerability Exploits in the Wild. In *Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud) (CSCLOUD '15)*. IEEE Computer Society, USA, 513–514. https://doi.org/10.1109/CSCloud.2015.56

[93] Stacy-Ann Elvy. 2021. *Products Liability in the Internet of Things Age*. Cambridge University Press, 160–195. https://doi.org/10.1017/9781108699235.006

[94] Pardis Emami-Naeini, Yuvraj Agarwal, Lorrie Faith Cranor, and Hanan Hibshi. 2020. Ask the Experts: What Should Be on an IoT Privacy and Security Label? arXiv:cs.CY/2002.04631 https://arxiv.org/abs/2002.04631

[95] Pardis Emami-Naeini, Janarth Dheenadhayalan, Yuvraj Agarwal, and Lorrie Faith Cranor. 2021. Which Privacy and Security Attributes Most Impact Consumers' Risk Perception and Willingness to Purchase IoT Devices?. In *2021 IEEE Symposium on Security and Privacy (SP)*. 519–536. https://doi.org/10.1109/SP40001.2021.00112

[96] Pardis Emami-Naeini, Janarth Dheenadhayalan, Yuvraj Agarwal, and Lorrie Faith Cranor. 2023. Are Consumers Willing to Pay for Security and Privacy of IoT Devices?. In *32nd USENIX Security Symposium*. USENIX Association, Anaheim, CA, 1505–1522. https://www.usenix.org/conference/usenixsecurity23/presentation/emami-naeini

[97] Emma Davis. 2023. QNAP Tells How To Deal With QSnatch. https://howtofix.guide/qnap-tells-how-to-deal-with-qsnatch/.

[98] ENISA. 2004. European Union Agency for Cybersecurity. https://certification.enisa.europa.eu/index_en

[99] ENISA. 2017. Baseline Security Recommendations for IoT. https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot

[100] ENISA. 2020. Guidelines for Securing the Internet of Things. https://www.enisa.europa.eu/publications/guidelines-for-securing-the-internet-of-things

[101] ENISA. 2024. Supporting NIS2 implementation through actionable guidance. https://www.enisa.europa.eu/news/supporting-nis2-implementation-through-actionable-guidance

[102] EPSS. 2024. Who is using EPSS? https://www.first.org/epss/who_is_using/.

[103] European Commission. 2024. Cyber Resilience Act (CRA) - Article 14: Reporting obligations of manufacturers. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R2847

[104] European Union Agency for Cybersecurity (ENISA). 2025. National Cybersecurity Certification Authorities. https://certification.enisa.europa.eu/take-action/national-cybersecurity-certification-authorities_en

[105] ExploitDB. 2022. Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers. https://www.exploit-db.com/.

[106] Alejandro Fanjul. 2018. LG SuperSign EZ CMS 2.5. Retrieved May 01, 2021 from https://www.exploit-db.com/exploits/45448

[107] Muhammad Junaid Farooq and Quanyan Zhu. 2019. IoT Supply Chain Security: Overview, Challenges, and the Road Ahead. arXiv:cs.CR/1908.07828 https://arxiv.org/abs/1908.07828

[108] Xuan Feng, Xiaojing Liao, XiaoFeng Wang, Haining Wang, Qiang Li, Kai Yang, Hongsong Zhu, and Limin Sun. 2019. Understanding and Securing Device Vulnerabilities through Automated Bug Report Analysis. In *28th USENIX Security Symposium*. USENIX Association, Santa Clara, CA, 887–903. https://www.usenix.org/conference/usenixsecurity19/presentation/feng

[109] Jennifer Fereday and Eimear Muir-Cochrane. 2006. Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development. *International Journal of Qualitative Methods* 5, 1 (2006), 80–92. https://doi.org/10.1177/160940690600500107

[110] FIRST. 1990. FIRST is the global Forum of Incident Response and Security Teams. https://www.first.org/

[111] FIRST Inc. 2023. Common Vulnerability Scoring System v3.0: Specification Document. https://www.first.org/cvss/specification-document

[112] Forum of Incident Response and Security Teams, Inc. 2024. Exploit Prediction Scoring System (EPSS). https://www.first.org/epss/api.

[113] Forum of Incident Response and Security Teams, Inc. 2025. Exploit Prediction Scoring System (EPSS) Version 4 . https://www.first.org/epss/.

[114] Erick Galinkin, John Carter, and Spiros Mancoridis. 2021. Evaluating Attacker Risk Behavior in an Internet of Things Ecosystem. In *Decision and Game Theory for Security: 12th International Conference, GameSec 2021, Virtual Event, October 25–27, 2021, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 354–364. https://doi.org/10.1007/978-3-030-90370-1_19

[115] Jerry Gamblin. 2016. Mirai Source Code. https://github.com/jgamblin/Mirai-Source-Code

[116] James Gareth, Witten Daniela, Hastie Trevor, and Tibshirani Robert. 2013. *An introduction to statistical learning: with applications in R*. Spinger. https://doi.org/10.1007/978-1-0716-1418-1

[117] Google. 2024. Google Project Zero. https://googleprojectzero.blogspot.com/p/0day.html.

[118] Stuart Graham and Shelagh Kennedy. 2020. Women in Cybersecurity: A Manifesto for Today. *Journal of Cybersecurity Education, Research and Practice* 2020, 1 (2020), 1–15. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7158845/

[119] ENISA ADVISORY GROUP. 2019. Opinion Consumers and IoT security. https://www.enisa.europa.eu/sites/default/files/all_files/Final%20Opinion%20ENISA%20AG%20Consumer%20IoT%20Perspective%20-09.2019.pdf

[120] Greg Guest, Arwen Bunce, and Laura Johnson. 2006. How Many Interviews Are Enough? An Experiment with Data Saturation and Variability. *Field Methods* 18, 1 (2006), 59–82. https://doi.org/10.1177/1525822X05279903

[121] Greg Guest, Arwen Bunce, and Laura Johnson. 2006. How many interviews are enough? An experiment with data saturation and variability. *Field methods* 18, 1 (2006), 59–82. https://doi.org/10.1177/1525822X05279903

[122] Munawar Hafiz and Ming Fang. 2016. Game of detections: how are security vulnerabilities discovered in the wild? *Empirical Softw. Engg.* 21, 5 (Oct. 2016), 1920–1959. https://doi.org/10.1007/s10664-015-9403-7

[123] Hollie Hennessy. 2025. *Consumer IoT Device Cybersecurity Standards, Policies, and Certification Schemes 2025*. Technical Report. Connectivity Standards Alliance. https://csa-iot.org/wp-content/uploads/2025/06/Consumer-IoT-Device-Cybersecurity-Standards-Policies-and-Certification-Schemes-2025-_FINAL.pdf

[124] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. 2019. Measurement and Analysis of Hajime, a Peer-to-peer IoT Botnet. *Proceedings 2019 Network and Distributed System Security Symposium* (2019). https://api.semanticscholar.org/CorpusID:142503559

[125] Christoph-Alexander Holst and Volker Lohweg. 2019. Feature fusion to increase the robustness of machine learners in industrial environments. *at - Automatisierungstechnik* 67 (2019), 853–865. Issue 10. https://doi.org/10.1515/auto-2019-0028

[126] Hors. 2021. Detect-It-Easy: Program for determining types of files for Windows, Linux and MacOS. https://github.com/horsicq/Detect-It-Easy

[127] David W. Hosmer, Trina A. Hosmer, Saskia le Cessie, and Stanley Lemeshow. 1997. A comparison of goodness-of-fit tests for the logistic regression model. *Statistics in medicine* 16 9 (1997), 965–80. https://doi.org/10.1002/(sici)1097-0258(19970515)16:9<965::aid-sim509>3.0.co;2-o

[128] Allen D. Householder, Jeff Chrabaszcz, Trent Novelly, David Warren, and Jonathan M. Spring. 2020. Historical Analysis of Exploit Availability Timelines. In *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)*. USENIX Association. https://www.usenix.org/conference/cset20/presentation/householder

[129] Allen D Householder, Garret Wassermann, Art Manion, and Chris King. 2017. *The CERT Guide to Coordinated Vulnerability Disclosure*. Technical Report CMU/SEI-2017-TN-002. Carnegie Mellon University Software Engineering Institute. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=503330

[130] Alice Hutchings and Richard Clayton. 2017. Configuring Zeus: A case study of online crime target selection and knowledge transmission. In *2017 APWG Symposium on Electronic Crime Research (eCrime)*. 33–40. https://doi.org/10.1109/ECRIME.2017.7945052

[131] Maël Hörz. 2020. HxD - Freeware Hex Editor and Disk Editor | mh-nexus. https://mh-nexus.de/en/hxd/

[132] Emanuele Iannone, Giulia Sellitto, Emanuele Iaccarino, Filomena Ferrucci, Andrea De Lucia, and Fabio Palomba. 2024. Early and Realistic Exploitability Prediction of Just-Disclosed Software Vulnerabilities: How Reliable Can It Be? *ACM Trans. Softw. Eng. Methodol.* (mar 2024). https://doi.org/10.1145/3654443 Just Accepted.

[133] Industry IoT Consortium (IIC. 2014. Integrating the two consortia leverages the best minds from each and expands our collaboration with industry, academia, and government. https://iiconsortium.org/

[134] J. Salvio, R. Tay. 2022. Fresh TOTOLINK Vulnerabilities Picked Up by Beastmode Mirai Campaign. https://www.fortinet.com/blog/threat-research/totolink-vulnerabilities-beastmode-mirai-campaign.

[135] Jay Jacobs, Sasha Romanosky, Idris Adjerid, and Wade Baker. 2020. Improving vulnerability remediation through better exploit prediction. *Journal of Cybersecurity* 6 (01 2020). https://doi.org/10.1093/cybsec/tyaa015

[136] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. 2021. Exploit Prediction Scoring System (EPSS). *Digital Threats* 2, 3, Article 20 (July 2021), 17 pages. https://doi.org/10.1145/3436242

[137] J. Jacobs, S. Romanosky, O. Suciu, B. Edwards, and A. Sarabi. 2023. Enhancing Vulnerability Prioritization: Data-Driven Exploit Predictions with Community-Driven Insights. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE Computer Society, Los Alamitos, CA, USA, 194–206. https://doi.org/10.1109/EuroSPW59978.2023.00027

[138] Marek Janiszewski, Anna Felkner, Piotr Lewandowski, Marcin Rytel, and Hubert Romanowski. 2021. Automatic Actionable Information Processing and Trust Management towards Safer Internet of Things. *Sensors* 21, 13 (2021). https://doi.org/10.3390/s21134359

[139] Marek Janiszewski, Marcin Rytel, Piotr Lewandowski, and Hubert Romanowski. 2022. VARIoT - Vulnerability and Attack Repository for the Internet of Things. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 752–755. https://doi.org/10.1109/CCGrid54584.2022.00085

[140] Adeel Javaid. 2013. Psychology of Hackers. *SSRN Electronic Journal* (01 2013). https://doi.org/10.2139/ssrn.2342620

[141] Jeffrey L Jenkins, Alexandra Durcikova, Grayson Ross, and Jay F Nunamaker Jr. 2010. Encouraging Users to Behave Securely: Examining the Influence of Technical, Managerial, and Educational Controls on Users' Secure Behavior. https://www.scopus.com/pages/publications/84870959774

[142] John P. John, Alexander Moshchuk, Steven D. Gribble, and Arvind Krishnamurthy. 2009. Studying Spamming Botnets Using Botlab. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*. USENIX Association, USA, 291–306. https://www.usenix.org/conference/nsdi-09/studying-spamming-botnets-using-botlab

[143] Ron Johnston, Kelvyn Jones, and David Manley. 2018. Confounding and collinearity in regression analysis: a cautionary tale and an alternative procedure, illustrated by studies of British voting behaviour. *Quality & quantity* 52 (2018), 1957–1976. https://doi.org/10.1007/s11135-017-0584-6

[144] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. 2023. Exploiting Programmatic Behavior of LLMs: Dual-Use Through Standard Security Attacks. arXiv:cs.CR/2302.05733 https://arxiv.org/abs/2302.05733

[145] Seiya Kato, Rui Tanabe, Katsunari Yoshioka, and Tsutomu Matsumoto. 2021. Adaptive Observation of Emerging Cyber Attacks targeting Various IoT Devices. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 143–151. https://ieeexplore.ieee.org/document/9464004

[146] KELA Cyber. 2023. Top 8 Deep Web and Dark Web Forums. https://www.kelacyber.com/blog/top-deep-web-and-dark-web-forums/

[147] David R. Anderson Kenneth P. Burnham. 2002. *Model Selection and Inference: A Practical Information-Theoretic Approach*. Springer New York, NY. https://doi.org/10.1007/978-1-4757-2917-7

[148] Raphaël Khoury, Benjamin Vignau, Sylvain Hallé, Abdelwahab Hamou-Lhadj, and Asma Razgallah. 2020. An Analysis of the Use of CVEs by IoT Malware. In *Foundations and Practice of Security: 13th International Symposium, FPS 2020, Montreal, QC, Canada, December 1–3, 2020, Revised Selected Papers*. Springer-Verlag, Berlin, Heidelberg, 47–62. https://doi.org/10.1007/978-3-030-70881-8_4

[149] Maciej Korczynski, Maarten Wullink, Samaneh Tajalizadehkhoob, Giovane C. M. Moura, Arman Noroozian, Drew Bagley, and Cristian Hesselman. 2018. Cybercrime After the Sunrise: A Statistical Analysis of DNS Abuse in New GTLDs. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18)*. Association for Computing Machinery, New York, NY, USA, 609–623. https://doi.org/10.1145/3196494.3196548

[150] Platon Kotzias, Leyla Bilge, Pierre-Antoine Vervier, and Juan Caballero. 2019. Mind Your Own Business: A Longitudinal Study of Threats and Vulnerabilities in Enterprises. https://doi.org/10.14722/ndss.2019.23522

[151] KPMG. 2023. The False Positive Conundrum: Effectively Prioritizing and Triaging Alerts. https://kpmg.com/us/en/articles/2023/false-positive-conundrum.html

[152] Brian Krebs. 2021. KrebsOnSecurity Hit By Huge New IoT Botnet "Meris". https://krebsonsecurity.com/2021/09/krebsonsecurity-hit-by-huge-new-iot-botnet-meris/

[153] Klaus Krippendorff. 2019. *Content analysis: An introduction to its methodology*. Sage publications. https://doi.org/10.4135/9781071878781

[154] Richard A Krueger. 2014. *Focus groups: A practical guide for applied research*. Sage publications.

[155] Ayush Kumar and Teng Lim. 2019. A Secure Contained Testbed for Analyzing IoT Botnets. https://doi.org/10.48550/arXiv.1906.07175

[156] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All Things Considered: An Analysis of IoT Devices on Home Networks. In *28th USENIX Security Symposium*. USENIX Association, Santa Clara, CA, 1169–1185. https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-deepak

[157] Legit Security. 2025. Alert Prioritization and Risk-Based Triage. https://www.legitsecurity.com/aspm-knowledge-base/triage-cyber-security

[158] Éireann Leverett. 2011. Quantitatively assessing and visualising industrial system attack surfaces. *University of Cambridge, Darwin College* 7 (2011), 21. https://www.cl.cam.ac.uk/~fms27/papers/2011-Leverett-industrial.pdf

[159] Keane Lucas, Mahmood Sharif, Lujo Bauer, Michael K. Reiter, and Saurabh Shintre. 2021. Malware Makeover: Breaking ML-based Static Analysis by Modifying Executable Bytes. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS '21)*. Association for Computing Machinery, New York, NY, USA, 744–758. https://doi.org/10.1145/3433210.3453086

[160] John Matherly. 2022. Shodan Search Engine. https://www.shodan.io/dashboard

[161] René Mayrhofer. 2019. Disclosing Proof-of-Concept (PoC) exploits for vulnerabilities: A defender's point of view. https://www.mayrhofer.eu.org/post/vulnerability-disclosure-is-positive/

[162] Scott Menard. 2002. *Applied logistic regression analysis* (2 ed.). Number 106 in Quantitative Applications in the Social Sciences. Thousand Oaks: SAGE Publications, Inc. https://doi.org/10.4135/9781412983433

[163] Fernando Mercês and Joey Costoya. 2020. *Telfhash: An Algorithm That Finds Similar Malicious ELF Files Used in Linux IoT Malware*. Technical Report. Trend Micro. https://github.com/trendmicro/telfhash?tab=readme-ov-file#Resources

[164] Metasploit. 2024. Metasploit | Penetration Testing Software. https://www.metasploit.com/.

[165] Louis Meuleman. 2008. Public Management and the Metagovernance of Hierarchies, Networks and Markets: The feasibility of designing and managing governance style combinations. http://hdl.handle.net/1765/12324

[166] Trend Micro. 2019. Uncovering IoT Threats in the Cybercrime Underground. https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/the-internet-of-things-in-the-cybercrime-underground.

[167] Seumas Miller and Michael Selgelid. 2008. Ethical and Philosophical Consideration of the Dual-use Dilemma in the Biological Sciences. *Science and engineering ethics* 13 (01 2008), 523–80. https://doi.org/10.1007/s11948-007-9043-4

[168] Felipe Moreno-Vera. 2023. Inferring Discussion Topics about Exploitation of Vulnerabilities from Underground Hacking Forums. In *2023 14th International Conference on Information and Communication Technology Convergence (ICTC)*. 816–821. https://doi.org/10.1109/ICTC58733.2023.10393244

[169] Felipe Moreno-Vera, Mateus Nogueira, Cainã Figueiredo, Daniel S. Menasché, Miguel Bicudo, Ashton Woiwood, Enrico Lovat, Anton Kocheturov, and Leandro Pfleger de Aguiar. 2023. Cream Skimming the Underground: Identifying Relevant Information Points from Online Forums. , 66-71 pages. https://doi.org/10.1109/CSR57506.2023.10224941

[170] Philipp Morgner, Christoph Mai, Nicole Koschate-Fischer, Felix Freiling, and Zinaida Benenson. 2019. Security Update Labels: Establishing Economic Incentives for Security Patching of IoT Consumer Products. arXiv:cs.CR/1906.11094 https://arxiv.org/abs/1906.11094

[171] Miftahul Munir. 2025. The Effectiveness of Bug Bounty Program for Technology Company Ecosystem. *Proceedings Series on Social Sciences & Humanities* 21 (Apr. 2025), 85–88. https://doi.org/10.30595/pssh.v21i.1517

[172] Munitio. 2025. A Deep Dive into the Russian Cybercrime Forums Shaping 2023's Landscape. https://munit.io/a-deep-dive-into-the-russian-cybercrime-forums-shaping-2023s-landscape/

[173] Azqa Nadeem, Christian Hammerschmidt, Carlos H. Gañán, and Sicco Verwer. 2020. *Beyond Labeling: Using Clustering to Build Network Behavioral Profiles of Malware Families*. Springer International Publishing, 381–409. https://doi.org/10.1007/978-3-030-62582-5_15

[174] Asuka Nakajima, Takuya Watanabe, Eitaro Shioji, Mitsuaki Akiyama, and Maverick Woo. 2019. A Pilot Study on Consumer IoT Device Vulnerability Disclosure and Patch Release in Japan and the United States. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (Asia*

CCS '19). Association for Computing Machinery, New York, NY, USA, 485–492. https://doi.org/10.1145/3321705.3329849

[175] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. 2015. The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. In *2015 IEEE Symposium on Security and Privacy*. 692–708. https://doi.org/10.1109/SP.2015.48

[176] Trey Herr Nathaniel Kim and Bruce Schneier. 2020. *The Reverse Cascade: Enforcing Security on the Global IoT Supply Chain*. Technical Report. Atlantic Council. https://www.schneier.com/wp-content/uploads/2020/06/Reverse-Cascade-Report-v3.1-3.pdf

[177] National Institute of Standards and Technology (NIST). 2021. National Vulnerability Database. https://nvd.nist.gov/ (Accessed on 2021-06-16).

[178] National Telecommunications and Information Administration (NTIA). 2016. *Vulnerability Disclosure Attitudes and Actions*. Technical Report. https://www.ntia.doc.gov/files/ntia/publications/2016_ntia_a_a_vulnerability_disclosure_insights_report.pdf

[179] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitraş. 2014. Some Vulnerabilities Are Different Than Others - Studying Vulnerabilities and Attack Surfaces in the Wild. In *International Symposium on Recent Advances in Intrusion Detection*, Vol. 8688 LNCS. Springer Verlag, 426–446. https://doi.org/10.1007/978-3-319-11379-1_21

[180] Lily Hay Newman. 2020. 'DDoS-For-Hire' Is Fueling a New Wave of Attacks. https://www.wired.com/story/ddos-for-hire-fueling-new-wave-attacks/

[181] Nilan Johnson Lewis. 2020. Recent IoT Class Actions Highlight Need for Manufacturers & Vendors of Connected Products to Be Aware of Liability Risks. https://nilanjohnson.com/recent-iot-class-actions-highlight-need-for-manufacturers-vendors-of-connected-products-to-be-aware-of-liability-risks/

[182] Weina Niu, Xiaosong Zhang, Xiaojiang Du, Teng Hu, Xin Xie, and Nadra Guizani. 2019. Detecting Malware on X86-Based IoT Devices in Autonomous Driving. *IEEE Wireless Communications* 26, 4 (2019), 80–87. https://doi.org/10.1109/MWC.2019.1800505

[183] Arman Noroozian, Elsa Turcios Rodriguez, Elmer Lastdrager, Takahiro Kasama, Michel Van Eeten, and Carlos H. Gañán. 2021. Can ISPs Help Mitigate IoT Malware? A Longitudinal Study of Broadband ISP Security Efforts. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. 337–352. https://doi.org/10.1109/EuroSP51992.2021.00031

[184] NSA. 2020. Ghidra. https://ghidra-sre.org/

[185] NVD. 2014. CVE-2014-8361. https://nvd.nist.gov/vuln/detail/CVE-2014-8361

[186] NVD. 2017. CVE-2017-17215. https://nvd.nist.gov/vuln/detail/cve-2017-17215.

[187] NVD. 2018. CVE-2018-17173. https://nvd.nist.gov/vuln/detail/CVE-2018-17173

[188] NVD. 2021. CVSS Severity Distribution Over Time. https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cvss-severity-distribution-over-time

[189] NVD. 2024. Common Weakness Enumeration (CWE). https://cwe.mitre.org/.

[190] Markus F.X.J. Oberhumer, László Molnár, and John F. Reiser. 2020. UPX - the Ultimate Packer for eXecutables. https://github.com/upx/upx

[191] Robert O'Brien. 2016. Dropping Highly Collinear Variables from a Model: Why it Typically is Not a Good Idea. *Social Science Quarterly* 98 (04 2016), 360–375. https://doi.org/10.1111/ssqu.12273

[192] Hamed Okhravi. 2008. Evaluation of patch management strategies. *International Journal of Computational Intelligence: Theory and Practice* 3 (12 2008).

[193] Thomas Olsson, Martin Hell, Martin Höst, Ulrik Franke, and Markus Borg. 2019. Sharing of vulnerability information among companies - a survey of Swedish companies. In *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE - Institute of Electrical and Electronics Engineers Inc., United States, 284–291. https://doi.org/10.1109/SEAA.2019.00051

[194] Joost Oortwijn and Carlos Gañán. 2024. Patch Pilgrimage: Exploring the Landscape of TCP Reflective Attacks and User Patching Expedition. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24)*. Association for Computing Machinery, New York, NY, USA, 1395–1404. https://doi.org/10.1145/3605098.3635982

[195] Ofri Ouzan. 2023. Advanced Shodan Use for Tracking Down Vulnerable Components. https://medium.com/@ofriouzan/advanced-shodan-use-for-tracking-down-vulnerable-components-7b6927a87c45.

[196] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the Rise of IoT Compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/woot15/workshop-program/presentation/pa

[197] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the Rise of IoT Compromises. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT) (WOOT'15)*. USENIX Association, USA, 9. https://www.usenix.org/conference/woot15/workshop-program/presentation/pa

[198] Packtsecurity. 2022. Spectre or Meltdown vulnerabilities on IoT. https://security.packt.com/spectre-or-meltdown-vulnerabilities-on-iot/.

[199] Fabio Pagani, Matteo Dell'Amico, and Davide Balzarotti. 2018. Beyond Precision and Recall: Understanding Uses (and Misuses) of Similarity Hashes in Binary Analysis. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy (CODASPY '18)*. Association for Computing Machinery, New York, NY, USA, 354–365. https://doi.org/10.1145/3176258.3176306

[200] Sergio Pastrana, Alice Hutchings, Andrew Caines, and Paula Buttery. 2018. Characterizing Eve: Analysing Cybercrime Actors in a Large Underground Forum. In *RAID 2018: 21st International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer Nature. https://doi.org/10.17863/CAM.31808

[201] Sergio Pastrana, Daniel R. Thomas, Alice Hutchings, and Richard Clayton. 2018. CrimeBB: Enabling Cybercrime Research on Underground Forums at Scale. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1845–1854. https://doi.org/10.1145/3178876.3186178

[202] Roberto Perdisci, Wenke Lee, and Nick Feamster. 2010. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 10)*. USENIX Association, San Jose, CA. https://www.usenix.org/conference/nsdi10-0/behavioral-clustering-http-based-malware-and-signature-generation-using

[203] Jennifer Petersen and Janet Shibley Hyde. 2014. Chapter Two - Gender-Related Academic and Occupational Interests and Goals. In *The Role of Gender in Educational Contexts and Outcomes*, Lynn S. Liben and Rebecca S. Bigler (Eds.). Advances in Child Development and Behavior, Vol. 47. JAI, 43–76. https://doi.org/10.1016/bs.acdb.2014.04.004

[204] Yangheran Piao, Temima Hrle, Daniel Woods, and Ross Anderson. 2025. Study Club, Labor Union or Start-Up? Characterizing Teams and Collaboration in the Bug Bounty Ecosystem . In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 20–20. https://doi.org/10.1109/SP61157.2025.00020

[205] Portnox. [n. d.]. Examining IoT Security Issues. https://www.portnox.com/cybersecurity-101/iot-security-issues Accessed July 4, 2025.

[206] Walter Powell. 1990. Neither Market Nor Hierarchy: Network Forms of Organization. *Research in Organizational Behaviour* 12 (01 1990), 295–336. https://web.stanford.edu/~woodyp/powell_neither.pdf

[207] C.G.J. Putman, Abhishta, and Lambert J. M. Nieuwenhuis. 2018. Business Model of a Botnet. In *2018 26th Euromicro International Conference on Parallel, Distributed*

and Network-based Processing (PDP). IEEE, 441–445. https://doi.org/10.1109/PDP2018.2018.00077

[208] Sandra Rivera Pérez, Michel Van Eeten, and Carlos H. Gañán. 2024. Patchy Performance? Uncovering the Vulnerability Management Practices of IoT-Centric Vendors. In *2024 IEEE Symposium on Security and Privacy (SP)*. 1198–1216. https://doi.org/10.1109/SP54263.2024.00154

[209] Qrator. 2021. Mēris botnet, climbing to the record. https://blog.qrator.net/en/meris-botnet-climbing-to-the-record_142/

[210] Damiano Ravalico, Mauro Farina, Martino Trevisan, and Alberto Bartoli. 2025. Analysing the Temporal Dynamics of the Exploit Prediction Scoring Systems (Epss). https://doi.org/10.2139/ssrn.5147459

[211] Elsa Rodríguez, Max Fukkink, Simon Parkin, Michel van Eeten, and Carlos Gañán. 2022. Difficult for Thee, But Not for Me: Measuring the Difficulty and User Experience of Remediating Persistent IoT Malware. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. 392–409. https://doi.org/10.1109/EuroSP53844.2022.00032

[212] Elsa Rodríguez, Arman Noroozian, Michel van Eeten, and Carlos Gañán. 2021. Super-spreaders: Quantifying the Role of IoT Manufacturers in Device Infections. *Annual Workshop on the Economics on Information Security* (2021), 18. https://weis2021.econinfosec.org/wp-content/uploads/sites/9/2021/06/weis21-rodriguez.pdf

[213] Elsa Rodríguez, Susanne Verstegen, Arman Noroozian, Daisuke Inoue, Takahiro Kasama, Michel van Eeten, and Carlos H Gañán. 2021. User compliance and remediation success after IoT malware notifications. *Journal of Cybersecurity* 7, 1 (07 2021), tyab015. https://doi.org/10.1093/cybsec/tyab015

[214] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2018. Deep Learning is Robust to Massive Label Noise. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1705.10694

[215] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. 2015. Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In *24th USENIX Security Symposium*. USENIX Association, Washington, D.C., 1041–1056. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sabottke

[216] Margarete Sandelowski. 2000. Whatever happened to qualitative description? *Research in nursing & health* 23, 4 (2000), 334–340.

[217] Silvia Sebastián and Juan Caballero. 2020. AVclass2: Massive Malware Tag Extraction from AV Labels. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. 42–53. https://arxiv.org/abs/2006.10615

[218] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. AVclass: A Tool for Massive Malware Labeling. *International Symposium on Research in Attacks, Intrusions, and Defenses* 9854 (09 2016), 230–253. https://doi.org/10.1007/978-3-319-45719-2_11

[219] Offensive Security. 2023. OptiLink ONT1GEW GPON 2.1.11_X101 Build 1127.190306 - Remote Code Execution. https://www.exploit-db.com/exploits/49955.

[220] Paulo Shakarian. 2018. Dark-Web Cyber Threat Intelligence: From Data to Intelligence to Prediction. *Information* 9, 12 (2018). https://doi.org/10.3390/info9120305

[221] Mario Silic. 2013. Dual-use open source security software in organizations - Dilemma: Help or hinder? *Comput. Secur.* 39 (Nov. 2013), 386–395. https://doi.org/10.1016/j.cose.2013.09.003

[222] Yuba R. Siwakoti and Danda B. Rawat. 2023. Detect-IoT: A Comparative Analysis of Machine Learning Algorithms for Detecting Compromised IoT Devices. In *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23)*. Association for Computing Machinery, New York, NY, USA, 370–375. https://doi.org/10.1145/3565287.3616529

[223] Internet Society. 1986. The Internet Engineering Task Force (IETF). https://www.ietf.org/

[224] Software Engineering Institute, Carnegie Mellon University. 1988. CERT Division. https://insights.sei.cmu.edu/divisions/cert/.

[225] Jinke Song, Shangfeng Wan, Min Huang, Jiqiang Liu, Limin Sun, and Qiang Li. 2023. Toward Automatically Connecting IoT Devices with Vulnerabilities in the Wild. *ACM Trans. Sen. Netw.* 20, 1, Article 6 (oct 2023), 26 pages. https://doi.org/10.1145/3608951

[226] Gaurav Sood. 2021. VirusTotal. https://www.virustotal.com/gui/home/search

[227] Kyle Soska and Nicolas Christin. 2014. Automatically Detecting Vulnerable Websites Before They Turn Malicious. In *23rd USENIX Security Symposium*. USENIX Association, San Diego, CA, 625–640. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/soska

[228] Georgios Spanos and Lefteris Angelis. 2018. A multi-target approach to estimate software vulnerability characteristics and severity scores. *Journal of Systems and Software* 146 (2018), 152–166. https://doi.org/10.1016/j.jss.2018.09.039

[229] Jonathan Spring. 2022. Probably Don't Rely on EPSS Yet. Carnegie Mellon University, Software Engineering Institute's Insights (blog). https://insights.sei.cmu.edu/blog/probably-dont-rely-on-epss-yet/

[230] Jonathan Spring, Eric Hatleback, Allen Householder, Art Manion, and Deana Shick. 2021. Time to Change the CVSS? *IEEE Security & Privacy* 19, 2 (2021), 74–78. https://doi.org/10.1109/MSEC.2020.3044475

[231] Jonathan M. Spring, Eric Nelson Hatleback, Art Manion, and Deana Shick. 2020. Prioritizing vulnerability response: A stakeholder-specific vulnerability categorization (version 1.0). https://api.semanticscholar.org/CorpusID:268076578

[232] Neville Stanton. 2006. Hierarchical task analysis: Developments, applications, and extensions. *Applied ergonomics* 37 (02 2006), 55–79. https://doi.org/10.1016/j.apergo.2005.06.003

[233] Strings. 2009. Strings(1) - Linux man page. Retrieved November 1, 2020 from https://linux.die.net/man/1/strings

[234] Walter W Stroup, George A Milliken, Elizabeth A Claassen, and Russell D Wolfinger. 2018. *SAS for mixed models: introduction and basic applications*. SAS Institute. https://support.sas.com/content/dam/SAS/support/en/books/sas-for-mixed-models-an-introduction/68787_excerpt.pdf

[235] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitras. 2022. Expected Exploitability: Predicting the Development of Functional Vulnerability Exploits. In *31st USENIX Security Symposium*. USENIX Association, Boston, MA, 377–394. https://www.usenix.org/conference/usenixsecurity22/presentation/suciu

[236] ST Tajalizadehkhoob, Hadi Asghari, Carlos Gañán, and MJG Van Eeten. 2014. Why them? Extracting intelligence about target selection from Zeus financial malware. In *Proceedings of the 13th Annual Workshop on the Economics of Information Security, WEIS 2014, State College (USA), June 23-24, 2014*. WEIS.

[237] Rui Tanabe, Tsuyufumi Watanabe, Akira Fujita, Ryoichi Isawa, Carlos Gañán, Michel Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. 2022. Disposable Botnets: Long-term Analysis of IoT Botnet Infrastructure. *Journal of Information Processing* 30 (09 2022), 577–590. https://doi.org/10.2197/ipsjjip.30.577

[238] Nazgol Tavabi, Palash Goyal, Mohammed Almukaynizi, Paulo Shakarian, and Kristina Lerman. 2018. DarkEmbed: Exploit Prediction With Neural Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (04 2018). https://doi.org/10.1609/aaai.v32i1.11428

[239] The MITRE Corporation. 1999. CVE - Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/index.html

[240] The Shadowserver Foundation. 2004. https://www.shadowserver.org/.

[241] Bill Toulas. 2024. Hackers use PoC exploits in attacks 22 minutes after release. https://www.bleepingcomputer.com/news/security/hackers-use-poc-exploits-in-attacks-22-minutes-after-release/

[242] Trend Micro. 2020. Caught in the Crossfire: Defending Devices From Battling Botnets. https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/caught-in-the-crossfire-defending-devices-from-battling-botnets

[243] H. Tschofenig and S. Farrell. 2017. RFC 8240: Report from the Internet of Things Software Update (IoTSU) Workshop 2016. https://doi.org/10.17487/RFC8240

[244] UK Parliament. 2022. Product Security and Telecommunications Infrastructure Act 2022. https://www.legislation.gov.uk/ukpga/2022/46/contents

[245] Unixfreaxjp. 2018. Behind the scene of: "Unpacking the Non-Unpackable". https://github.com/radareorg/r2con2018/blob/master/talks/unpacking/README.md

[246] URLhaus. 2018. URLhaus | Malware URL exchange. https://urlhaus.abuse.ch/

[247] U.S. Congress. 2020. IoT Cybersecurity Improvement Act of 2020. https://www.congress.gov/bill/116th-congress/house-bill/1668

[248] Kami E. Vaniea, Emilee Rader, and Rick Wash. 2014. Betrayed by updates: how negative experiences affect future security. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2671–2674. https://doi.org/10.1145/2556288.2557275

[249] VARIoT. 2024. VARIoT – Vulnerability and Attack Repository for IoT. https://www.variot.eu/.

[250] Marie Vasek and Tyler Moore. 2014. Identifying Risk Factors for Webserver Compromise. In *Proceedings of the International Conference on Financial Cryptography and Data Security (FC)*, Nicolas Christin and Reihaneh Safavi-Naini (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 326–345. https://doi.org/10.1007/978-3-662-45472-5_22

[251] Jay M Ver Hoef and Peter L Boveng. 2007. Quasi-Poisson vs. negative binomial regression: how should we model overdispersed count data? *Ecology* 88, 11 (2007), 2766–2772. https://doi.org/10.1890/07-0043.1

[252] VirusTotal. 2020. VirusTotal Reference API | Files. https://developers.virustotal.com/v3.0/reference#files

[253] VirusTotal. 2024. VirusTotal – Learning resources - Vulnerability management. https://www.virustotal.com/getstarted/vulnerability-management.

[254] Daniel Votipka, Rock Stevens, Elissa Redmiles, Jeremy Hu, and Michelle Mazurek. 2018. Hackers vs. Testers: A Comparison of Software Vulnerability Discovery Processes. In *2018 IEEE Symposium on Security and Privacy (SP)*. 374–391. https://doi.org/10.1109/SP.2018.00003

[255] VulnCheck. 2024. A Peek into the Last Decade of Vulnerability Exploitation. https://www.vulncheck.com/blog/state-of-exploitation-a-decade. White paper.

[256] VulnCheck. 2024. VulnCheck KEV. https://vulncheck.com/kev.

[257] VulnDB. 2020. Comprehensive Vulnerability Intelligence. https://vulndb.cyberriskanalytics.com/#statistic (Accessed on 2021-05-01).

[258] Vulners. 2024. All-in-one vulnerability intelligence. https://vulners.com/.

[259] T. Walshe and A.C. Simpson. 2022. Coordinated Vulnerability Disclosure programme effectiveness: Issues and recommendations. *Comput. Secur.* 123, C (Dec. 2022), 14. https://doi.org/10.1016/j.cose.2022.102936

[260] Huanran Wang, Weizhe Zhang, Hui He, Peng Liu, Daniel Xiapu Luo, Yang Liu, Jiawei Jiang, Yan Li, Xing Zhang, Wenmao Liu, Runzi Zhang, and Xing Lan. 2021. An Evolutionary Study of IoT Malware. *IEEE Internet of Things Journal* 8, 20 (2021), 15422–15440. https://doi.org/10.1109/JIOT.2021.3063840

[261] Yanjun Wang and Qun Liu. 2006. Comparison of Akaike information criterion (AIC) and Bayesian information criterion (BIC) in selection of stock–recruitment relationships. *Fisheries Research* 77, 2 (2006), 220–225. https://doi.org/10.1016/j.fishres.2005.08.011

[262] Chris Wanstrath, P. J. Hyett, Tom Preston-Werner, and Scott Chacon. 2022. Github Database. https://github.com/.

[263] White House & Federal Communications Commission. 2025. U.S. Cyber Trust Mark Program. https://www.fcc.gov/CyberTrustMark

[264] Suzanne Widup, Marc Spitler, David Hylender, and Gabriel Bassett. 2018. 2018 Verizon Data Breach Investigations Report. https://www.verizon.com/business/resources/reports/DBIR_2018_Report.pdf

[265] Allan Wigfield and Jacquelynne S. Eccles. 2000. Expectancy–Value Theory of Achievement Motivation. *Contemporary Educational Psychology* 25, 1 (2000), 68–81. https://doi.org/10.1006/ceps.1999.1015

[266] Dumidu Wijayasekara, Milos Manic, Jason L. Wright, and Miles McQueen. 2012. Mining Bug Databases for Unidentified Software Vulnerabilities. In *2012 5th International Conference on Human System Interactions*. 89–96. https://doi.org/10.1109/HSI.2012.22

[267] J. Wunder, A. Kurtz, C. Eichenmüller, F. Gassmann, and Z. Benenson. 2024. Shedding Light on CVSS Scoring Inconsistencies: A User-Centric Study on Evaluating Widespread Security Vulnerabilities. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 57–57. https://doi.org/10.1109/SP54263.2024.00058

[268] Chaowei Xiao, Armin Sarabi, Yang Liu, Bo Li, Mingyan Liu, and Tudor Dumitras. 2018. From Patching Delays to Infection Symptoms: Using Risk Profiles for an Early Discovery of Vulnerabilities Exploited in the Wild. In *27th USENIX Security Symposium*. USENIX Association, Baltimore, MD, 903–918. https://www.usenix.org/conference/usenixsecurity18/presentation/xiao

[269] Wei Xie, Chao Zhang, Pengfei Wang, Zhenhua Wang, and Qiang Yang. 2021. ARGUS: Assessing Unpatched Vulnerable Devices on the Internet via Efficient Firmware Recognition. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS '21)*. Association for Computing Machinery, New York, NY, USA, 421–431. https://doi.org/10.1145/3433210.3453685

[270] Soufian El Yadmani, Robin The, and Olga Gadyatskaya. 2022. Beyond the Surface: Investigating Malicious CVE Proof of Concept Exploits on GitHub. *arXiv* (2022). https://arxiv.org/abs/2210.08374

[271] Wei Yang, Jiakun Jiang, Erin Schnellinger, Stephen Kimmel, and Wensheng Guo. 2022. Modified Brier score for evaluating prediction accuracy for binary outcomes. *Statistical Methods in Medical Research* 31 (08 2022), 096228022211223. https://doi.org/10.1177/09622802221122391

[272] Zynamics. 2021. BinDiff. https://www.zynamics.com/bindiff.html

[273] Orçun Çetin, Carlos Gañán, Lisette Altena, Samaneh Tajalizadehkhoob, and Michel van Eeten. 2019. Tell Me You Fixed It: Evaluating Vulnerability Notifications via Quarantine Networks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. 326–339. https://doi.org/10.1109/EuroSP.2019.00032

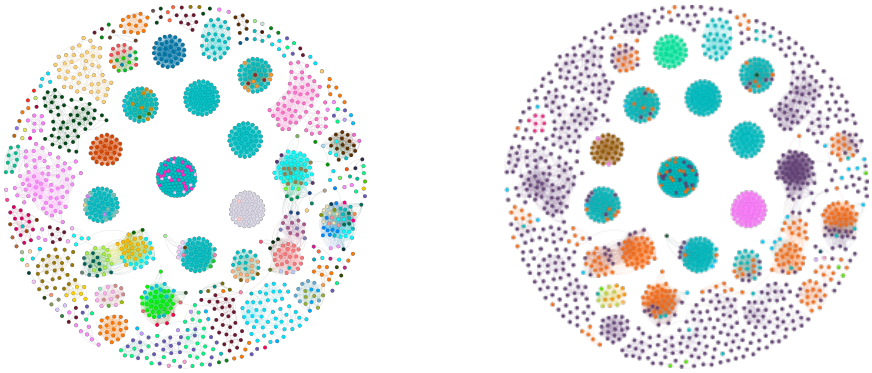
[274] The Connectivity Standards Alliance ("Alliance"). 2024. The Connectivity Standards Alliance Product Security Working Group Launches the IoT Device Security Specification 1.0. https://csa-iot.org/newsroom/the-connectivity-standards-alliance-product-security-working-group-launches-the-iot-device-security-specification-1-0/

A

Clustering and Classification

A.1. AUXILIARY ANALYSIS: AVCLASS2 AND VHASH CLUSTERING

Figure A.1 provides auxiliary information for our classification and clustering based on AVClass2 and Vhash. As discussed in Section 2.7, we used BinDiff [272] to compare binary similarity on the packed and unpacked URLhaus dataset.



(a) Coloured by Vhash

(b) Coloured by AVClass2

Figure A.1: Results of pairwise similarity calculation of the URLhaus samples using Ghidra 1.9.2 and BinDiff 6. 1,725 out of 2,298 samples could be analyzed, shown are only samples with a similarity score and confidence above 0.8. Colours indicate distinct clusters of Vhash and AVClass2 respectively. For Vhash, dark teal indicates a missing Vhash while for AVClass2 dark teal indicates a singleton class.

B

CrimeBB Dataset and GLM Model

B.1. SUMMARY OF CRIMEBB DATASET

Table B.1: Summary of CrimeBB forums ordered by the number of posts.

Forums	# Posts	Oldest post
hack-forums	42,474,326	2007-01-27
zismo	11,127,167	2010-05-26
blackhatworld	10,320,120	2005-10-31
multiplayer-game-hacking	10,217,579	2005-12-26
nulled	6,675,498	2013-04-02
lolzteam	6,196,005	2013-03-10
ogusers	3,608,306	1900-01-01
cracked	2,977,801	2018-04-03
mmo4me	2,899,930	2010-04-01
antichat	2,630,906	2002-05-29
v3rmillion	2,459,519	2016-02-02
unknowncheats	2,403,995	2002-11-02
raidforums	1,231,126	2015-03-20
elhacker	980,523	2002-08-21
probiv	822,671	2014-11-05
breached	737,922	2022-03-16
forum-team	431,695	2017-10-31
indetectables	328,024	2006-02-20
xss-forum	310,796	2004-11-13
dread	294,596	2018-02-15
runion	240,632	2012-01-11
offensive-community	161,492	2012-06-30
underc0de	92,247	2010-02-10
the-hub	88,753	2014-01-09
ifud	72,851	2012-05-10
piratebay-forum	60,678	2013-10-23
torum	28,485	2017-05-25
safe-sky-hacks	27,018	2013-03-28
kernelmode	26,815	2010-03-11
freehacks	26,471	2013-07-27
deutschland-im-deep-web	20,185	2018-11-22
greysec	11,925	2015-06-10
garage-for-hackers	8,710	2010-07-06
stresser-forums	7,069	2017-04-09
envoy-forum	2,163	2019-07-06
Hackers Armies	-	-
Total	11,0003,999	

B.2. MODEL

B.2.1. FEATURES WITH FACTORS INFLUENCING THE TARGETING OF CERTAIN IOT VULNERABILITY USING GLM.

Table B.2: Regression results of the model including the Engagement feature. Only the 38 features with p-value < 0.05 are shown.

Feature	Coefficient	Standard Error
PoC	1.272^{***}	(0.165)
PR:L	-0.595^{*}	(0.240)
Vendor:INTEL	1.159^{***}	(0.261)
Vendor:FEDORAPROJECT	0.891^{**}	(0.338)
Vendor:CITRIX	2.013^{***}	(0.480)
Vendor:MITEL	2.051^{**}	(0.733)
Vendor:DLINK	0.785^{***}	(0.223)
Vendor:DRAYTEK	3.399^{***}	(0.629)
Vendor:WESTERNDIGITAL	2.111^{*}	(0.920)
Vendor:WAVLINK	1.849^{*}	(0.787)
Vendor:MINIO	2.591^{**}	(0.853)
Vendor:APACHE	2.304^{***}	(0.356)
Vendor:HUAWEI	-3.882^{**}	(1.217)
Vendor:ZYXEL	1.440^{**}	(0.463)
Vendor:MICROCHIP	3.544^{*}	(1.495)
Vendor:FORTINET	1.183^{***}	(0.273)
Vendor:FREEBSD	-3.447^{***}	(0.909)
Vendor:EMBEDTHIS	1.943^{*}	(0.803)
Vendor:OPENSSL	2.992^{***}	(0.555)
Vendor:GRANDSTREAM	1.566^{*}	(0.798)
Vendor:LINKSYS	1.546^{*}	(0.736)
Vendor:SONICWALL	1.446^{**}	(0.492)
Vendor:DAHUASECURITY	2.230^{**}	(0.722)
Vendor:NUUO	1.540^{*}	(0.653)
CWE-22	0.590^{*}	(0.266)
CWE-352	-1.488^{*}	(0.739)
CWE-843	1.658^{**}	(0.535)
CWE-287	0.562^{*}	(0.250)
CWE-78	1.152^{***}	(0.182)
CWE-94	1.107^{*}	(0.493)
CWE-116	2.509^{*}	(1.086)
CWE-294	1.733^{*}	(0.747)
Ref:Third Party Advisory	-0.389^{*}	(0.175)
Ref:Exploit	0.358^{*}	(0.171)
Ref:Release Notes	0.562^{*}	(0.261)
Ref:VDB Entry	0.416^{*}	(0.196)
Ref:Issue Tracking	0.746^{**}	(0.266)
Engagement	0.237^{***}	(0.017)
Observations	15,600	
Akaike Inf. Crit.	3,810.212	
Note:	*p<0.05; **p<0.01; ***p<0.001	

B.2.2. MODEL FEATURE COEFFICIENTS

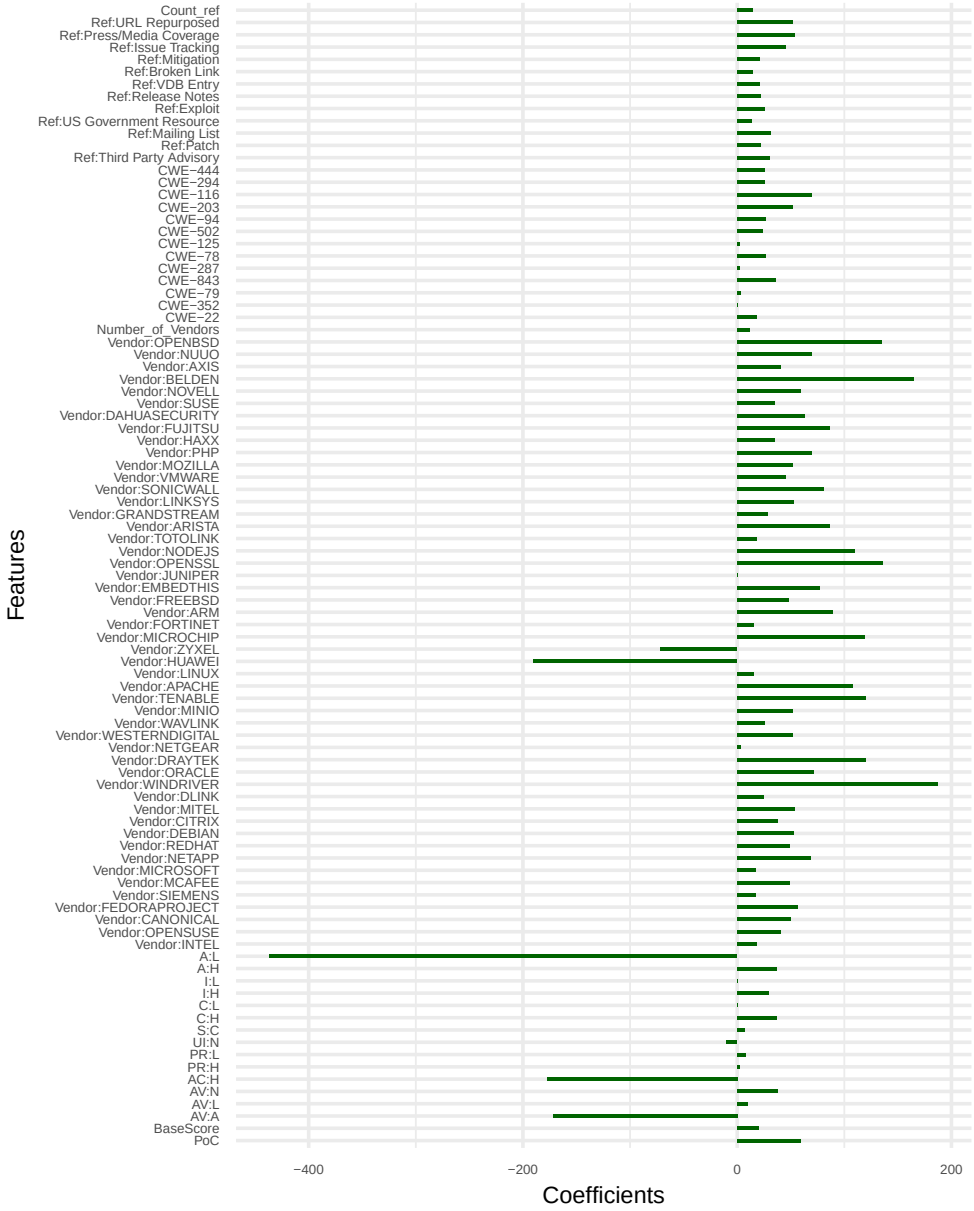


Figure B.1: The 93 features used selected by elastic net regularization.

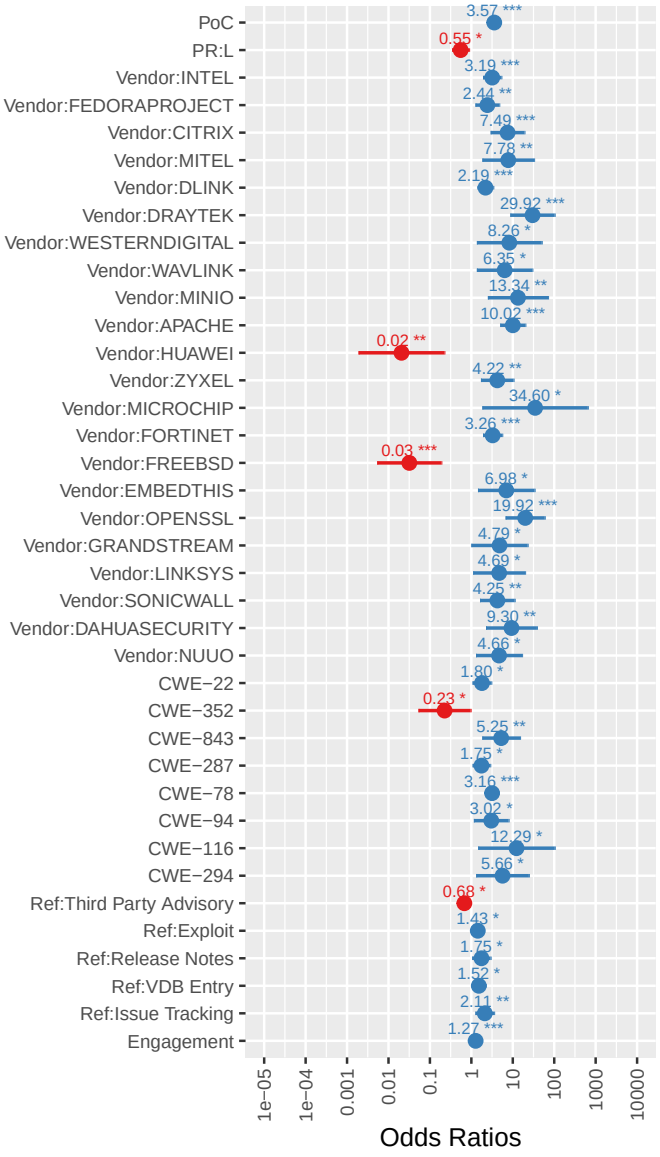


Figure B.2: Odds ratio of the regression model with hacking community PC in regularization. Only the features with p-value < 0.05 are shown.

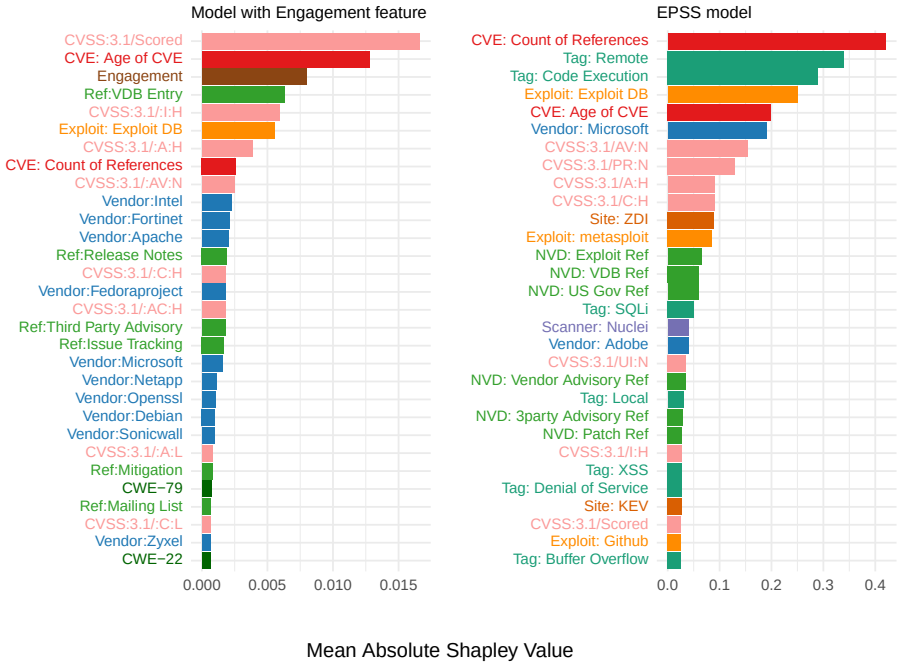


Figure B.3: Mean absolute Shapley value for the top 30 most influential feature of our model with Engagement vs. EPSS.

C

Interview Protocol and Codebook

C.1. INTERVIEW PROTOCOL

This section outlines the semi-structured interview protocol used in our study. The protocol is organized into four thematic areas: (1) Background and Experience, (2) Motivation, (3) PoC Exploit Development Process, and (4) Disclosure, Publication, and Potential Misuse.

BACKGROUND AND EXPERIENCE

1. What is your main job?
    - 1.1. How many years have you been in this position?
2. How does your job relate to PoC exploit development?
    - 2.1. If it is not directly related, how would you describe your involvement?
3. How many years have you been involved in developing PoC exploits?
4. How frequently do you work on developing PoC exploits?
5. On a scale of 1–5 (1 = beginner, 5 = expert), how would you rate your PoC exploit development skills?
6. How many PoC exploits have you developed (publicly disclosed or not)?
    - 6.1. How many of them are public?
    - 6.2. Where can they be found?
    - 6.3. What do you count or consider as a PoC exploit?
7. For which types of software do you develop PoC?
    - 7.1. Why do you focus on those types?
    - 7.2. Can you briefly describe the last three PoC you developed (or any three you recall)?

MOTIVATION

1. What motivates you to develop PoC exploits for vulnerabilities?

POC EXPLOIT DEVELOPMENT PROCESS

1. Can you walk me through your typical process for developing a PoC exploit?
    - 1.1. How do you identify or search for vulnerabilities?
    - 1.2. What factors do you prioritize when deciding to develop a PoC?
    - 1.3. Do you collaborate with others? Why or why not?
        - 1.3.1. If yes, what role do you usually play in the collaboration?
    - 1.4. Why do you follow this particular process?
    - 1.5. Have you ever used alternative methods? If so, what made you switch?

2. Have you developed PoC for known vulnerabilities discovered by others that lacked a PoC? Why?
3. Have you developed PoC for vulnerabilities that already had existing ones? Why?
4. What programming languages or tools do you typically use, and why?
5. How do you validate or determine that your PoC exploit is successful?

PoC Disclosure, Publication, and Potential Misuse

1. After developing a PoC, what steps do you take for reporting it?
   - 1.1. If disclosing, to whom do you disclose?
     - 1.1.1. What guidelines or principles do you follow to ensure responsible disclosure?
   - 1.2. Do you usually report full or partial disclosure? Why?

     Full disclosure: Sharing both vulnerability details and the complete PoC.

     Partial disclosure: Sharing only vulnerability details.
   - 1.3. What are your reasons for making PoC public?
     - 1.3.1. Where do you typically publish them?
2. Are there cases where you chose not to disclose or publish a PoC you developed? Why?
3. What specific impacts or responses have you observed from the community or vendors regarding your PoC?

C.2. CODEBOOK

The qualitative analysis of our semi-structured interviews with PoC exploit developers was guided by systematic coding procedures consistent with established best practices in qualitative cybersecurity research [109, 216]. Interview recordings were transcribed verbatim and anonymized to remove all identifying information in compliance with ethical protocols.

Our analysis employed a multi-stage coding process following the methodological framework established by [48] for thematic analysis and adapted principles from [39] for multi-phase qualitative coding. This approach with two coders enabled both depth and reliability in capturing the complex motivational and contextual factors influencing PoC exploit development. Initially, two analysts independently conducted open coding on a purposive sample of transcripts, engaging in iterative memoing and discussion to identify emergent codes and competing interpretations [59]. This iterative refinement process resulted in an initial codebook capturing distinctive categories relevant to the dual-use dilemmas and participant experiences.

Following established protocols for qualitative reliability assessment [39, 154], we conducted multiple rounds of coding with systematic comparison between coders. Subsequent rounds of coding applied this evolving codebook to the full dataset, with coders regularly convening to reconcile discrepancies and refine definitions. To assess inter-coder reliability, we calculated Krippendorff's alpha using established computational methods [55], achieving a high consistency score ($\alpha = 0.857$), surpassing thresholds commonly accepted in qualitative security studies [153]. This ensured that the codebook was both comprehensive and applicable across our diverse sample.

The development of our codebook followed systematic thematic analysis principles [121], ensuring theoretical saturation was achieved within our sample of 16 participants. Our hybrid inductive-deductive approach [109] allowed for both data-driven code emergence and theory-informed analysis through our application of EVT constructs.

The codebook presented in Table C.1 summarizes the final set of coding categories, including definitions and representative examples. It underpinned both thematic analysis and framework analysis phases, providing a transparent and replicable foundation for interpreting the nuanced motivational calculus underpinning PoC development decisions. This structured approach facilitated the inductive emergence of key themes that illuminate the interplay of expectancy beliefs, subjective task values, moral reasoning, and contextual factors in dual-use cybersecurity research.

Table C.1: Interview Codebook.

Code	Description
Background and Demographics	
role-primary	Main job role of the participant
role-secondary	Additional roles beyond primary job
job-involve-poc	Job involve PoC exploit development
job-no-poc	Job does not involve PoC exploit development

Continued on next page

Code	Description
years-job	Number of years in current position
years-security	Total years of experience in security field
years-poc	Number of years involved in PoC exploit development
skill-level	Self-assessed PoC development skill level on scale from 1 to 5
no-vuln-discovered	Number of vulnerabilities discovered by participant
no-poc-developed	Total number of PoC developed (public and private)
no-poc-public	Number of PoC made publicly available
freq-poc	How often participant works on developing PoC
poc-lasts	Last three or any PoC developed
poc-platforms	Platforms used for sharing PoC
software-targets	Types of software targeted for PoC development
PoC Perceptions and Definitions	
poc-def-code-centric	Only as code-based
poc-def-demo	Simple crash demonstrations
poc-def-weaponized	Full automation and weaponization
poc-def-spectrum	Continuum from crash to full exploit
poc-def-any	Any form from text description, to video tutorial, to code base
poc-def-0day	Only for newly-discovered or zero-day vulnerabilities
Motivational Factors	
mot-ego	Prove technical understanding and demonstrate expertise
mot-curiosity	Intellectual satisfaction and technical challenges
mot-sec-improv	Improve cybersecurity for society
mot-demonstrate	Showcase technical abilities
mot-personal-protection	Protect personal accounts, data, and digital assets
mot-protection	Safeguard organizational systems and colleagues
mot-financial	Seek monetary gain through sales or career advancement
mot-recog	Community acknowledgment and professional reputation
mot-fun-challenge	Fun and game-like activity
mot-civic-duty	Feel responsibility as good digital citizen
mot-vendor-pressure	Pressure vendors for faster patching
mot-education	Improving security community skills
Development Tools and Methods	
tool-python	Use Python for PoC development due to simplicity and/or AI compatibility
tool-ai-assisted	Leverage AI tools for code generation and development assistance

Continued on next page

Code	Description
tool-other-languages	Use other programming languages than Python
tool-shodan	Utilizing Shodan for target identification and validation
validation-local	Test PoC in isolated local environments
validation-physical	Test on physical devices
validation-systematic	Follow structured validation processes across multiple environments
approach-known-vulns-poc	Develop PoC for disclosed vulnerabilities with existing exploits
approach-known-vulns-no-poc	Develop PoC for disclosed vulnerabilities without existing exploits
approach-unknown-vulns	Develop PoC for newly discovered or zero-day vulnerabilities
approach-patch-diffing	Analyze version differences to identify vulnerabilities in silent patches
approach-fuzzing	Use automated testing techniques to discover vulnerabilities
approach-poc-adaptation	Leverage and validate existing public PoC
approach-poc-scratch	Develop PoC completely from scratch
approach-poc-scratch-reason	Avoid potential hidden malicious intent
Workflow and Process	
method-systematic	Follow structured, repeatable development processes
method-opportunistic	Adapt approach based on available resources and circumstances
work-solo	Work independently throughout PoC development lifecycle
work-collaborative	Work with others throughout PoC development lifecycle
work-both	Work either with peers or solo throughout PoC development lifecycle
work-style-reasons	Reasoning for opting for solo or collaborative work styles
work-collaborative-role	Role(s) when collaborating with others
Disclosure Strategies & Reporting Channels	
disclosure-full	Share both vulnerability description and complete PoC code
disclosure-partial	Share only vulnerability details while withholding PoC code
disclosure-none	Not disclosing vulnerabilities or PoC publicly
disclosure-delayed	Wait for vendor patches and user update before public disclosure
disclosure-immediate	Shift to publication to pressure unresponsive vendors for remediation

Continued on next page

Code	Description
disclosure-conditional	Wait for vendor patches before any public disclosure
disclosure-reasons	Reasons for chosen disclosure strategy
disclosure-reasons-threat	Evaluate potential threat of PoC publication
disclosure-reasons-impact	Consider scope and severity of potential misuse
disclosure-reasons-legal-risk	Concerns about legal consequences of PoC development or sharing
disclosure-reasons-legal-supportive	Operate within permissive legal environments
disclosure-reasons-reput	Worries about damage to professional reputation
disclosure-reasons-opport	Considering alternative uses of time and resources
disclosure-reasons-moral	Psychological burden of potential harmful consequences
disclosure-reasons-effort	Physical and mental effort required for PoC development or disclosure
channel-cvd-national	Use national coordinated vulnerability disclosure programs
channel-vendor-direct	Report directly to affected vendors or organizations
channel-internal-client	Disclose within corporate or client environments only
channel-public-platforms	Publication platforms
channel-restricted	Share through private groups or underground channels
Publication Practices	
full-poc-public	Complete PoC available to general public
selective-poc-community	Share PoC only within trusted security communities
pub-kowdg-share	Contribution to community education and skills
pub-pressure-tactic	Use PoC publication to pressure vendors for remediation
pub-recognition	Pursue acknowledgment and status within community
pub-monetization	Sell PoC through private or underground markets
community-trust-high	High trust in security community members and platforms
community-trust-low	Skepticism about community members' intentions

Continued on next page

Code	Description
community-anonymous	Prefer anonymous participation in security communities
exp-positive-disclosure	Beneficial outcomes from previous disclosure activities
exp-negative-disclosure	Harmful consequences from past PoC sharing
exp-vendor-cooperation	Previous positive interactions with responsive vendors
exp-vendor-hostility	Past negative experiences with uncooperative vendors
exp-misuse-witnessed	Direct knowledge of PoC being used maliciously
learning-community-driven	Acquire skills through community interaction and collaboration
learning-self	Expertise through independent study and practice
identity-ethical	Strong identification as responsible security researcher
identity-expert	Primary identity centered on technical capabilities
identity-community	Self-perception as contributor to collective security
identity-conflict	Experience tension between different professional identities
values-open-source	Strong belief in open sharing of security knowledge
values-prevention	Prioritize prevention of potential misuse over transparency
values-neutrality	Believe PoC are neutral tools independent of use context

ACKNOWLEDGEMENTS

Reaching the end of this journey has been both a privilege and a challenge. I owe a great deal to the people who have supported me along the way, through their kindness, guidance, and unwavering belief. This is my small way of thanking the people, places, and moments that shaped this path.

First of all, I would like to thank my amazing promotors, Michel and Carlos. Thank you for taking a chance on me at a moment when I was ready to walk away from my PhD, not knowing what to expect. You created space for me in your group that shaped the entire course of my journey. Without your belief and support, I wouldn't have been able to start and successfully finish this PhD.

Michel, your sharp thinking, hands-on supervision, and the safe, welcoming space you created, along with your willingness to fix text minutes before a deadline and the care you put into our group's dynamics, shaped my PhD experience in ways I'll always be grateful for. Thank you for hosting the Sinterklaas and Christmas dinners, for encouraging me to embrace Dutch directness, and for helping me get a proper bike (and pushing me into that first surprise ride from Delft to The Hague). I may not have lived up to your hopes of cycling to the office regularly (I blame the Dutch weather!), but I promise to take that bike to new places in Saudi and continue your cycling legacy.

Carlos, you became the constant I could always reach, day or night. Your spot-on comments, statistical help, and perfectly timed reminders grounded me when everything felt overwhelming. Your ability to get multiple things done at record speed (aka Carlos time) still baffles me; I'm convinced your cats and chickens have something to do with it. And if *suficiente* is the only Spanish word I truly remember, it's because I screamed it internally every time you started catastrophizing that we'd never make the deadline. Thank you for the endless chats filled with dark humor and laughter that somehow made even the worst chaos feel manageable.

To my wonderful TBM group, I'm so grateful to have had such supportive and amazing colleagues, many of whom became close friends. Elsa, having you as my first officemate and then a close friend made all the difference. Your unwavering warmth, gentleness, and perfectly timed Reels kept me sane. If I'd followed your "don't over think it, just start writing" advice earlier, I might've finished sooner. You are my safe person. Maria, you have always been there to pick me up on moments even when I couldn't ask. Our deep conversations and your amazing cook made my days. The past year hasn't been easy, and I couldn't be there as much as I wanted, but I'm always rooting for you. Mathew, I'm proud of pulling off some reverse cultural integration by getting you to embrace a bit of Middle Eastern spirit. And who knew that simple 1-to-10 scale we started in Japan could mean two completely different things to each of us for years? Thank you for all the fun, crazy, and joyful moments. Lorenz, thank you for introducing me to Keti. You have both been both such amazing friends and ex-neighbors. I miss having you around, especially our shared mission for good food. Watching you become wonderful parents to little Mathilde

has been incredibly heartwarming. I'm truly grateful I got to know you, and I'll always have a soft spot for your little family. Boy, even though we didn't meet as often as I wanted, every time we did, your existential debates and jiu-jitsu takedowns (yes, I let you win, someone had to protect your ego) made up for it. Sandra, you've always been so ready to help, especially when you babysat Yoda so I could take a few days off. Seeing you both become amazing parents to Auri was the ultimate validation of my psychic abilities :p Swaathi, it's probably time to admit that you are my favorite mix of wise, fun, grounding, and endlessly teaseable human. Apparently, we can't leave each other behind even at the finish line. Szu, I'm grateful for our trip to Prague, meeting your cousin in Taiwan, your magic with upgrading slides, and the beautiful drawing you made of me and Yoda. You have a way of making everything feel lighter. To my wonderful officemates Yana, Abrahem, Annebel, and Max, special thanks to you for your generous help connecting me to people with your unbeatable network. To Aksel, Veerle, Cécile, Ronak, Hugo, Fieke, Evi, Yury, Savvas, Simon, Natalia, Xander, Radu, Nicolas Dintzner, Kelvin, Qasim, Arman, Keti, and Urgun. Thank you all for making the journey lighter.

I would also like to thank, Joyce van Velzen for being both a great friend and a true lifesaver that I honestly don't know how I would've navigated the system without you. Abdullah Aljufairi, for your endless help throughout and even before applying for my PhD. Panas, for your tremendous help with dogsitting Yoda so I can focus on work and for being a wonderful friend, not only to him but to me. To my friends back home and around the world, Namaa, Nora, and to all the Saudi White House group for the great continued visit.

To my close friends. Cynthia, I'm beyond grateful for that first PhD course that led me to you, and for you then leading me to your BFF, Chris. Together we formed a little trio that carried me through everything, the highs, the lows, and everything in between. Our silly foodie dances whenever we were blown away by good food, Cynthia. And Chris, that memorable Mac Miller night. Thank you both for being constants in my life! Xandra, I'm so grateful that Yoda and Bruce brought us together; you became family in no time. Thank you and Ahn for being Yoda's second home and favorite humans. Our spontaneous trips, retail therapy, crazy energy, ridiculous laughs, filterless conversations, and especially for the endless bubble teas on blue days. You made "I'm just a text or call away" feel real.

To my cousins Yara, Rasha, aunts Sahar, Ghada, Warda, and uncle Solaiman, thank you all for the support you have given me in various ways.

I owe special thanks to my uncles. Khalid, you helped me begin this PhD journey and but never let me forget how it all started with the absurd number of my suitcases you carried up those impossibly narrow Dutch stairs to the place I had booked, only for us to discover it was right in the heart of the Red Light District. Amo Solaiman, my strongest backbone and softest spot. I'll always remember that year when you came early before work to take me to school. Thank you for never doubting my decisions, unwavering support and love, and your warm voice notes that lifted me every single time. You are the BEST uncle, and human, I could ask for.

أدين بشكر كبير لشخصين. خالي خالد، أنت اللي ساعدتني أبدأ رحلة الدكتوراة، ولو إنك تذكّريني كل شوي ببدايتها خصوصًا بعدد شنطي اللي تكسرنا واحنا نحاول نشيلها بدرجهم الهولندي الضيق والناس تضحك علينا في الشقة اللي اكتشفنا إنها في نص الرد لايت ديستركت! دعمك مستحيل انساه!

عمو سليمان، حبيبي وسندي ونقطة ضعفي وأحنّ قلب فالدنيا. ما أنسى السنة اللي كنت تجي بدري قبل دوامك بس عشان توصلني للجامعة. تعبك معنا ومع بابا. شكرًا لأنك ما شكّيت في قراراتي بيوم، على العكس، كنت دائمًا داعمي الاكبر. رسايلك الصوتية الدافئة ماعمرها خيبت انها ترمم يومي و ترفعني كل مره. إنت أحسن عم فالدنيا. الله لايخلينا منك.

To my wonderful big family. my nieces-Maysa, Najala, Deema, and Lolo-and my nephew Solaiman: your constant “When will you finish and come back?” was the most persistent deadline I had. That question alone pushed me forward. Now we can all rest, and I hope you’ll start to remember my name and that I am, in fact, part of the family.

My sisters, thank you for your support, love, and unwavering belief in me. Your visits, calls, and shared trips gave me the energy I needed more times than I can count. Lamia, your warm check ups and thoughtful deliveries, especially those Ramadan dates, always arrived at just the right time. Afnan, your gentle companionship and generosity; and I’m still sorry for being a reckless host and nearly poisoning you with my French toast. But just so you know, I’m absolutely still claiming my promised graduation trip to Italy! Raghad, for being my true lifesaver when I got stuck in London, and more than anything, for “abandoning” Yoda, ending up with me saved me more than you could ever imagine. Having him by my side made all the difference. Mada, although your medical advice is helpless :p, you’ve been an amazing travel buddy and a SECOND mom to Yoda, knowing he was in such good hands gave me peace of mind to focus on finishing. Sara, thank you for your quiet support and for helping with drawings when I needed it most. If I’m grateful for anything during this time, it’s that it brought us closer. And finally, my little Haifa, your sticky notes always hit me right in the heart, and I only stole your paintings just to have a piece of you in my place. Thank you all for putting up with my stress, sudden bursts, and absences, and for lifting me up every time I felt like quitting. I promise I’ll try to do better now that I’m done.

Mama, no words will ever do you justice. I know how hard it was to let me go as I discovered life on my own, but you supported me with everything you had. Your love, presence, strength, and positive energy kept me going. Even if you still think I’m either doing accounting or something with just resting routers, you always listened to my endless rants without tirelessly. You’re a supermom. I love you.

ماما، مهما قلت من كلام ما راح يوفيك حقك. أدري قدّيش كان صعب عليك تخليني أطلع وأكتشف الحياة بنفسي، ومع ذلك ساندتيني بكل شي عندك. محبتك، وجودك، قوّتك، وطاقة الإيجابية هي اللي كانت تدفعني أكمل. وحتى ولو انك ما زلتي مقتنعة إني أدرس محاسبة. كنتِ تسمعين فضفضتي اللي ما تخلص بدون ما تزهقين. خوخة، إنتِ أم خُورافية. أحبك!

Baba, my first believer, the one who inspired it all and will never read this. Carrying my school bag even when I was already in college, standing against the madness back then, and taking me yourself to the other side of the world to pursue my master’s, were my greatest lessons in standing up for what feels right, in being resilient, strong yet kind and humble. Your sharpness, analytical mind, and our long conversations, always starting as intellectual debates and ending in stubborn fights, became my bittersweet reminder whenever I felt lost or doubted myself. This doesn’t feel complete without you. I miss you.

Finally, to those whose paths crossed with mine but whose journeys led elsewhere, your care and support are not forgotten. Thank you for being part of this journey. I wish you well.

Arwa Abdulkarim
Den Haag, May 2025

AUTHORSHIP CONTRIBUTIONS

This dissertation is based on four peer-reviewed papers resulting from collaboration with multiple co-authors. In each of these studies, I was fortunate to receive valuable feedback and diverse contributions from my co-authors. In the following paragraphs, I describe their individual contributions to each study.

For the first study (see Chapter 2), Michel van Eeten, Carlos Gañán, and Martina Lindorfer advised on methodology, refined the argumentation, and polished the manuscript. Katsunari Yoshioka provided access to the IoTPOT honeypot dataset and drafted its setup. Kaichi Sameshima contributed to the dynamic analysis of IoT malware binaries and extracted the observed exploits. Jakob Bleier conducted the binary similarity analysis and produced Figure A. I performed the remaining data collection, static analysis, main data analysis, and wrote the majority of the manuscript.

For the second study (Chapter 3), Katsunari Yoshioka provided access to the IoTPOT dataset and drafted its setup. Kaichi Sameshima extended his dynamic analysis work to extract the exploits. Carlos Gañán supported model selection and co-authored the corresponding section. Michel van Eeten and Carlos Gañán advised on methodology and helped refine the manuscript. I conducted the remaining data collection, developed the model, carried out the main analysis, and wrote the majority of the text.

For the third study (Chapter 4), Katsunari Yoshioka and Takayuki Sasaki provided access to exploits captured through the IoTPOT and X-Pot honeypots. Mathew Vermeer helped implement the model and drafted the related section. Michel van Eeten and Carlos Gañán contributed to shaping the methodology, structuring the paper, selecting the model, and refining the arguments through in-depth discussions and polishing the manuscript. I collected and processed the remaining datasets, performed the data analysis, and wrote the majority of the paper.

In the last study (Chapter 5), I designed the interview protocol, which was improved through feedback from Lorenz Kustosch, Carlos Gañán, and Michel van Eeten. I conducted the interviews, anonymization, analysis, and wrote the manuscript. Lorenz Kustosch joined pilot interviews, helped with feedback, and advised on grounding the findings. Together with Lamya Alowain, we coded all interviews and discussed and summarized results. Carlos Gañán conducted one interview in Spanish and advised on structuring and polishing the paper. Michel van Eeten advised on structuring and revised the introduction.

I reserve special thanks for my promotor Michel van Eeten and co-promotor Carlos Gañán, who have graciously guided me through this entire journey—from the uncertainties of the first study to the completion of the last. Your mentorship has been transformative, and I am profoundly grateful that my research journey has been shaped and supported by both of you. To all of my co-authors, I am deeply grateful for your unwavering support, invaluable contributions, and constructive feedback.

List of Publications

4. **Arwa Abdulkarim Al Alsadi**, Kaichi Sameshima, Jakob Bleier, Katsunari Yoshioka, Martina Lindorfer, Michel van Eeten, and Carlos H. Gañán. 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22). Association for Computing Machinery, New York, NY, USA, 309–321. https://doi.org/10.1145/3488932.3517408*

3. **Arwa Abdulkarim Al Alsadi**, Mathew Vermeer, Takayuki Sasaki, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. Bits and Pieces: Piecing Together Factors of IoT Vulnerability Exploitation. In *Proceedings of the 2025 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Hanoi, Vietnam, https://doi.org/10.1145/3708821.373387*

2. **Arwa Abdulkarim Al Alsadi**, Kaichi Sameshima, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2023. Bin There, Target That: Analyzing the Target Selection of IoT Vulnerabilities in Malware Binaries. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). Association for Computing Machinery, New York, NY, USA, 513–526. https://doi.org/10.1145/3607199.3607241*

1. **Arwa Abdulkarim Al Alsadi**, Lorenz Kustosch, Lamya Alowain, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. "Oh, what people would do with my knife?" Navigating the Dual-Use Dilemma in PoC Exploit Development, Disclosure, and Community Dynamics. *Under submission to USENIX Security '26*

DATASETS

Table 2: Dataset availability

Publication	Dataset(s)
Arwa Abdulkarim Al Alsadi, Kaichi Sameshima, Jakob Bleier, Katsunari Yoshioka, Martina Lindorfer, Michel van Eeten, and Carlos H. Gañán. 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22). Association for Computing Machinery, New York, NY, USA, 309–321. https://doi.org/10.1145/3488932.3517408*	IoT malware datasets, PCAPs, exploit signatures, vulnerability information, and scripts are publicly available at `https://doi.org/10.4121/19248725`.
Arwa Abdulkarim Al Alsadi, Kaichi Sameshima, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2023. Bin There, Target That: Analyzing the Target Selection of IoT Vulnerabilities in Malware Binaries. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). Association for Computing Machinery, New York, NY, USA, 513–526. https://doi.org/10.1145/3607199.3607241*	IoTPOT honeypot data cannot be shared due to our agreements with Yokohama National University but may be requested at `https://sec.ynu.codes/iot/available_datasets`. All remaining data used in this study is available from their publicly accessible sources.
Arwa Abdulkarim Al Alsadi, Mathew Vermeer, Takayuki Sasaki, Katsunari Yoshioka, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. Bits and Pieces: Piecing Together Factors of IoT Vulnerability Exploitation. In *Proceedings of the 2025 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Hanoi, Vietnam, https://doi.org/10.1145/3708821.373387*	Model source code and the URL sources to collect the data used in the exploit prediction model (e.g., VARIoT, NVD, ExploitDB, etc.) are available at `https://doi.org/10.4121/69d111e7-30e6-4206-b27d-22a5f4e33722`. CrimeBB access requires an application and agreement with the Cambridge Cybercrime Centre.
Arwa Abdulkarim Al Alsadi, Lorenz Kustosch, Lamya Alowain, Michel Van Eeten, and Carlos Hernandez Gañán. 2025. "Oh, what people would do with my knife?" Navigating the Dual-Use Dilemma in PoC Exploit Development, Disclosure, and Community Dynamics. *Under submission to USENIX Security '26*	Interview transcripts, even anonymized, cannot be shared.

About the author

Arwa Abdulkarim Al Alsadi (1989) was born in Riyadh, Saudi Arabia. She received her Bachelor's degree in Information Systems from Al-Imam University in 2011 with Second Class Honors.

In 2013, she moved to the United States to pursue her Master of Science degree in Cybersecurity at The George Washington University's School of Engineering & Applied Science, where she graduated with First Class Honors in 2014. During her time there, she also earned graduate certificates in both Computer Security and Information Assurance and Management of Information and Systems Security.

After gaining experience in the cybersecurity field across consulting, research, and technical roles, she joined the King Abdulaziz City for Science and Technology (KACST) in Riyadh, Saudi Arabia, as a Cybersecurity researcher in 2015. During this period, she enhanced her technical expertise by obtaining the GIAC Reverse Engineering Malware (GREM) certification in 2018.

In 2020, she joined Delft University of Technology at the Department of Multi-Actor Systems in the Netherlands as a PhD researcher. Her doctoral research focuses on understanding target selection in IoT malware attacks, addressing critical questions in the rapidly evolving landscape of Internet of Things security threats. Throughout her PhD studies, her collaboration with Yokohama National University resulted in three publications and co-authorship in another research study, and she also co-authored an additional study with her PhD group, both in related areas. Since June 2025, she has also been contributing to serve on the programming committee at the FIRST Vuln4Cast FIRST conference. Her interdisciplinary work bridges empirical security research with a strong understanding of real-world attacker behaviors, drawing from both academic and operational perspectives.