Revisiting Mirai

Murtuza Ali

Characterising Botnet Scans
through network telescope data

Revisiting Mirai

by

Murtuza Ali

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 20, 2024 at 10:00 AM.

Student number:	659311534	
Project duration:	November 20, 2024 – June 20, 2024	
Thesis committee:	Prof. dr. G. Smaragdakis,	TU Delft, supervisor
	Dr. ir. H. J. Griffioen,	TU Delft, Daily Supervisor
	Dr. W. Böhmer,	TU Delft
	ir. F. Dominguez,	Hunt and Hackett

TU Delft

Preface

Working on this thesis over the past 7 months has helped me gain a thorough understanding of my field and has also helped me to grow as an individual, to be able to formulate questions and to work towards answering those questions. I look forward to continuing head first into researching the things that interest me.

I would like to thank my supervisors, Dr. Georgios Smaragdakis and Dr. ir. Harm Griffioen for their continued support, insight and encouragements throughout the course of this thesis.

Learning to work in a research capacity has its ups and downs and can often be a very isolating task. For helping me stay afloat and to overcome this seemingly insurmountable task I would also like to thank everyone around me that kept me motivated.

Murtuza Ali
Delft, June 2024

Abstract

The interconnected nature of the internet has led to rapid advancements due to the surge in global communication and collaboration. The proliferation of IoT devices within this interconnected world has created a tightly integrated society, where multiple devices in an ecosystem are accessible from the internet. However, this connectivity also introduces various weak points that malicious actors can exploit. Devices with known vulnerabilities can be targeted and hijacked from anywhere in the world, as long as they are connected to the internet. To find these vulnerable devices, bad actors scan the entire internet address space. By monitoring a range of unused IP addresses also known as a Network Telescope or Darknet, we can gain a better understanding of their activities and tactics, techniques, and procedures (TTPs). We exploit several weaknesses in the design of the prevalent Mirai botnet to gather information on the state of devices, their activities, and lifetimes. Additionally, we use known characteristics to understand the nature of newer strains of these botnets. By using the well studied vanilla Mirai infections as a benchmark, we can classify and categorize the activities of other botnets, refining our data collection to enhance the insights generated from this data.

List of Figures

Contents

1 Introduction

The internet, a quintessential feat of modern engineering, encompasses a global network that interlinks a vast array of devices. The predominant protocol currently in use, IPv4 [1], supports 2^{32} addresses, amounting to almost 4.3 billion possible devices. This capacity has been further expanded through the implementation of Network Address Translation (NAT) [2] and Private Address Ranges, which allow multiple devices to share a single public IP address. Despite these measures, the amount of connected devices has nearly exhausted the available IPv4 addresses. This highlights the rapid global adoption of the internet and its integration into most activities.

The lightning-fast and extensive interconnectivity of companies, institutions, and individuals worldwide through the Internet has led to significant advancements. However, this also introduces substantial vulnerabilities. As every device connected to the internet becomes a node for communication, it also becomes a potential target for exploitation. This vulnerability has led to the emergence of various malicious entities, from highly skilled individuals to online crime syndicates and nation-states who exploit this interconnectedness for malevolent purposes. The widespread access to devices provides these bad actors with unprecedented opportunities for attacks, ranging from data breaches to Distributed Denial of Service (DDoS) and ransomware attacks to name a few, thereby posing serious challenges to global stability.

Earlier, these attacks were aimed at particular entities, each with a specific target in mind, but the usage of shared protocols and widespread manufacture of devices sharing similar software or architecture led to the creation of malware known as worms, which would target a specific vulnerability which exists in large groups of devices and spread rapidly, sometimes using already compromised devices. These were the first forms of networks of targeted devices. In most cases they would wreak havoc by destroying the devices, these attacks have even occurred recently such as the NotPetya attack, which affected organizations, hospitals, and other critical infrastructure across the world. However advances in security practices keep making these attacks much harder to pull off, requiring much more effort required in crafting these attacks.

The proliferation of IoT devices has significantly increased the number of everyday objects connected to the internet. These devices are typically engineered to minimize power consumption and network bandwidth, often at the expense of robust security features. Consequently, many of the advancements in computer security are not incorporated into these devices due to their limited capabilities and the need for cost efficiency. This compromise has resulted in a vast proliferation of devices that are either poorly configured or inherently vulnerable due to architectural flaws. Opportunistic adversaries have exploited these weaknesses, gaining unauthorized control over countless devices.

This aggregation of internet-connected devices compromised via security vulnerabilities has gained the moniker of a ”botnet.” A botnet is characterized by a large number of devices that have been infected by an attacker. These devices are then utilized for various malicious activities, including the distribution of spam emails and the execution of DDoS attacks. DDoS attacks involve overwhelming a target with a flood of internet traffic, resulting in the victim being unable to operate any services. Additionally much like worms, devices within a botnet may engage in scanning the network to identify and compromise further vulnerable devices, thereby expanding the botnet's reach. They may also support the maintenance and updating of the botnet's infrastructure, ensuring its continued operation and efficacy.

These botnets have existed in some form for a long time, with the first well known and acknowledged botnet, being the Earthlink Spammer [3], which was used for phishing attacks, One of the most prevalent botnets in recent times was the Mirai botnet, this botnet used credential stuffing attacks targeted at default login credentials in several internet connected devices, growing to an enormous size. It is also responsible for some of the most devastating DDoS attacks. Although the main Mirai botnet has subsided in apparent activity, it is still a major contributor to the overall scanning activity on the internet. Since its source code got leaked, there have been several offshoots of the original Mirai botnet, often specialised to target newer vulnerabilities. These variants have also been modified to diverge from the original Mirai botnet and do not share all their characteristics.

To set up and create a botnet, an attacker would have to find some commonly available vulnerability and then employ either some scanning tools or software either on their own or through commonly used hosting services to find an initial set of devices, after which they would aim at infecting these devices and getting them under their control. These devices would then continue scanning the internet for more vulnerable devices and then spread to them. One of the most important components of setting up a botnet and then maintaining it is the practice of scanning the internet to identify devices that could be infected and added to the botnet.

These scanning activities can be observed through various means and can be used to understand the state of current vulnerabilities and the extent of their severity. Passive measurement techniques exist that serve this exact purpose, allowing researchers to study and further understand the nature of these botnets. Data collected from a network telescope is used for this study. A network telescope is a set of IP addresses that are not connected to any devices or services, and as such do not send any traffic into the internet. This would mean that any packets received by a network telescope would be unsolicited. This could happen either as a result of a misconfiguration that coincidentally included one or more IP addresses from the network telescope set, or in the case of internet-wide scanning.

Even though passively measuring these scanning activities provides us less contextual information than other methods such as reverse engineering binaries, dynamic analysis, or using C2 milkers among other techniques, it is still an essential field of study due to its largely scalable nature. One can process much larger amounts of data coming through passive measurements with less infrastructure or manual effort required. Moreover, the amount of human effort required to manually analyze samples, or fine-tune honeypots or milkers to account for different variants becomes unviable very fast.

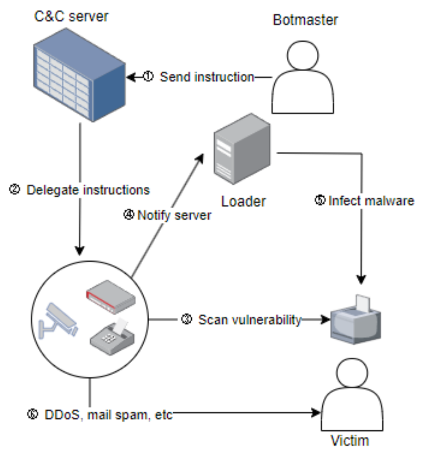


Figure 1.1: Overview of botnet architecture and activities

Despite the importance of understanding botnet behavior and classifying their variants, it is a vastly under-researched topic. Several works highlight certain aspects of botnets which focus on finding fingerprints for these botnets to classify them accordingly, however, it is also essential to understand the nature of the botnet itself. Gaining insight into the lifecycle of a bot, with an understanding of its birth, initial setup, and subsequent activities could provide some invaluable information that would supersede individual strains of a particular botnet. Knowing certain aspects such as how long these bots are usually active, whether their behavior is consistent or instruction-oriented, and whether the botmasters partition portions of the network and delegate separate responsibilities would allow us to gain a much higher top-down view of the inner workings of a botnet and provide us invaluable insights that could help in the perpetual war between botmasters and defenders.

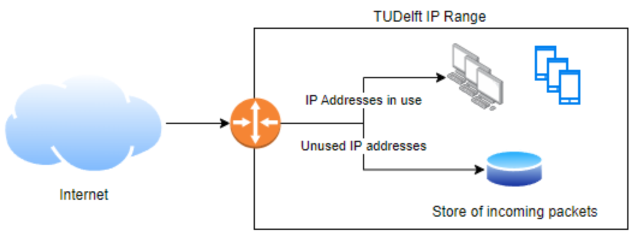


Figure 1.2: Setup of Network telescope used in this work

Taking into account the current state-of-the-art research in this field and the type of data made available throughout this thesis. The main questions that this thesis aims to address are the following:

RQ1. *What is the current state of Mirai?*

RQ2. *Can we find discriminatory features between botnet traffic, and regular scanning traffic?*

RQ3. *Can we find botnet traffic that does not have known signatures such as the Mirai signature?*

RQ4. *What does the lifecycle of a bot look like from a network telescope perspective? What activities does it perform over its lifetime and are they regularly changing?*

2 Background

This section aims to introduce all relevant concepts that are pertinent to the matter of this thesis, such as the Infrastructure of the internet, and its protocols which are (ab)used for the malicious purposes that are investigated later.

2.1. Internet

Since its inception traced back to a United States project in the late 1960s [4] to robustly interconnect several systems that were geographically separated, the internet has grown into a monumental system of interconnected computer systems all over the world. A series of protocols was created, to ensure that all systems can communicate despite their heterogeneity and to make sure that they adhere to the requirements of the underlying low-level infrastructure that forms the physical backbone of the Internet. These protocols are designed following a strict code so that all systems are either capable of inter-communication or signal compatibility or incompatibility at the very least. We will discuss some of these protocols to understand their workings and how they are abused by malicious actors.

2.2. Internet Models

There are several ways to look at the various layers that the internet has been divided into. Two of the most popular frameworks for classifying the internet are the OSI framework and the TCP/IP framework, which will used interchangeably in this document.

2.2.1. OSI model

The OSI model [5] is an abstraction of the various functions employed in the internet into seven layers. These layers all handle a particular crucial function. This helps to provide a series of abstractions such that someone can rely on the lower layers to perform as expected and not have to worry about ensuring guarantees provided by the lower layers. A brief explanation of the 7 layers specified by the OSI model is provided below [6].

The **Physical Layer** is focused on transmitting the raw bits over a physical medium, this involves making sure that the two sides can negotiate specific details such as ensuring the representation of the bits, the rate at which data is sent, etc. to ensure that both endpoints have a common understanding of the specifications of the data transfer. The **Data Link Layer** ensures that the data sent in the physical layer can be checked for correctness and facilitates acknowledgment and methods to have some form of congestion control. The **Network Layer** focuses on routing the packets to ensure that they reach their intended destination and also implements some form of congestion control. The **Transport Layer** focuses on receiving data from higher layers, breaking them into smaller units, and ensuring that they reach their intended destination. The transport layer ensures end-to-end communication. The **Session Layer** handles the creation and upkeep of sessions. The **Presentation Layer** handles the variance in different systems by identifying and converting the syntax of the underlying information when needed. The **Application Layer** focuses on the end functionality required by users in the form of commonly defined protocols such as HTTP, FTP, etc.

The OSI model involves ”packing” the data required by each layer into a lower level layer and exposes higher layers only when required, for example when a switch has to route the packet or the packet is being reconstructed at an endpoint. The relevant information is only exposed when it is needed. This level of abstraction allows for the compartmentalization of responsibilities and allows for applications to be built without having to account for each layer.

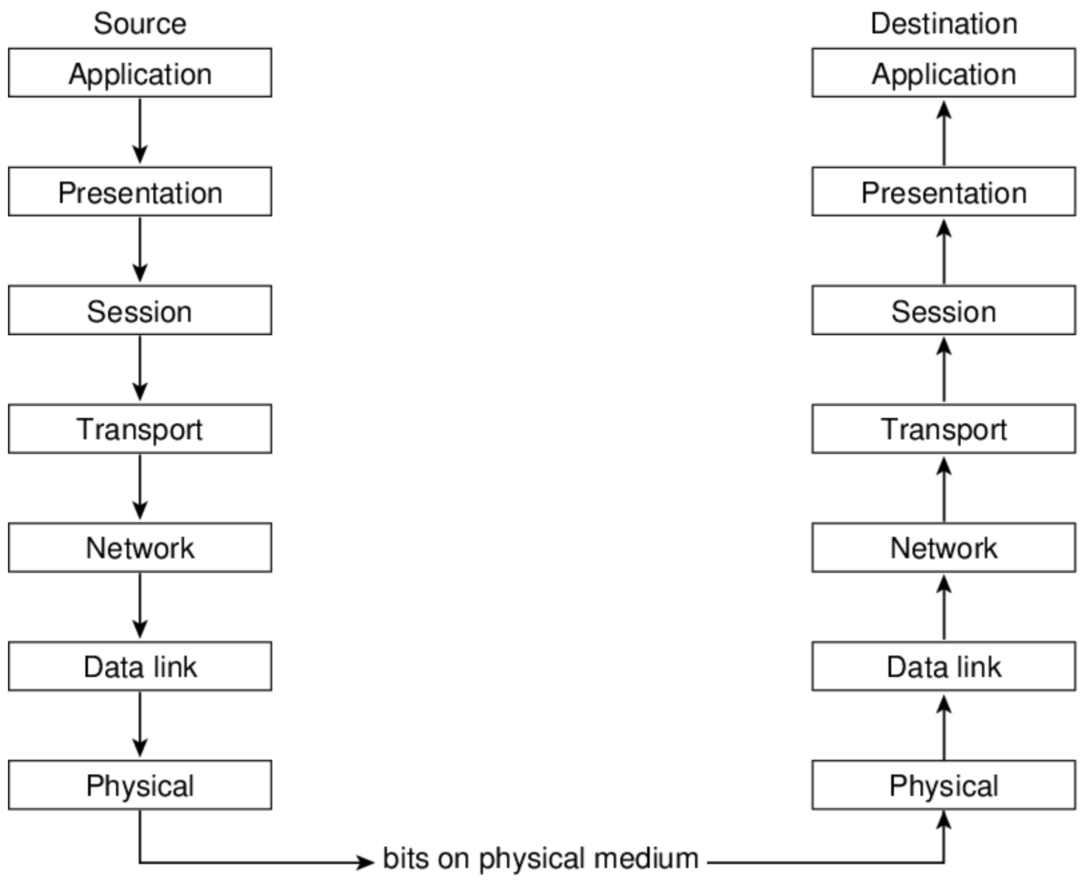


Figure 2.1: OSI model reference model

2.2.2. TCP/IP model

The TCP/IP model was the reference model used in ARPANET to account for the addition of a wide variety of heterogenous networks such as satellites and radio to already existing connections that used phone lines. The layers that make up the TCP/IP model are discussed below.

The **Link Layer** deals with the interfacing between hosts and the lower physical architecture, taking into account the way the hosts would have to interact with the physical medium, addressing over the physical layer, error correction of frames sent over the layer, and so on. The **Internet Layer** handles the sending of packets in an independent fashion over a series of routes to reach the required destination. The internet layer uses the IP or Internet Protocol along with an ICMP or Internet Control Message Protocol to provide additional support or transparency for debugging to the hosts. The **Transport Layer** sits on top of the Internet Layer and provides functionality based on the protocol used, the TCP or Transport Control Protocol provides reliable and in-order end-to-end delivery of packets, while the UDP or User Datagram Protocol is a best-effort connectionless protocol. The **Application Layer** provides unique functionality to the end-user.

The TCP/IP network came to be called so because of the two main protocols used in this architecture. This leads to there being a 5-layered architecture as opposed to the 7-layered architecture in the OSI model.

These models are introduced as they form the backbone of a large amount of the communications on the internet, and are essential to know in order to understand what mechanisms exist for communication between billions of devices on the internet and how they are abused by botnets to propagate and perform attacks. Moreover, an outline of these protocols also highlights the several tradeoffs one has to consider when looking at network security and balancing factors such as performance, availability, and accessibility against more secure implementations.

2.3. Internet Protocol

The version of the internet protocol that we use namely, IPv4 is the fourth version of the internet protocol and was standardized in RFC 791. IPv4 uses 32 bits to represent the address space which provides $2^{32} = 4.294.967.296$ unique addresses. The IPv4 datagram consists of a header and a body, the IPv4 header as seen in 2.2 contains several fields that provide information required to communicate information about the connection and other details such as the address, the size of the datagram, etc. We will highlight some of the headers that are of interest to the rest of the thesis.

The version header mentions the version in use, mostly set to 4 and sometimes set to 6 in case IPv6 is used. The IHL field is used to provide the length of the header in chunks of 32 bits and ranges from 5 to 15. The total length field mentions the total size of the datagram including the header and the body. The identification field is used for reconstructing fragments of a packet. The Time-To-Live field is used to cap the number of hops that a packet can go through before they are dropped, preventing it from traveling indefinitely in case of a misconfiguration. The header checksum is used to ensure that there are no errors in the propagation of the packet. The Source and Destination addresses are used to indicate the addresses of the source and destination endpoints.

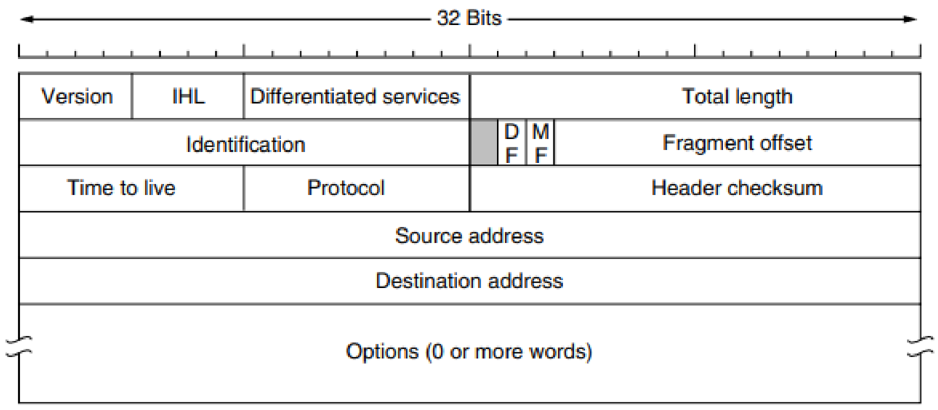


Figure 2.2: The IPv4 protocol header

Although IPv4 is an integral part of the internet, it has recently faced issues with a shortage in the number of available addresses, which has led to the use of technologies such as carrier-grade NAT and DHCP which create interesting research issues and are also explained in later sections.

2.4. TCP: Transmission Control Protocol

The transmission control protocol is important in the Transport Layer. It aims at providing a reliable, in-order, end-to-end transport of data over an unreliable medium. It aims to account for the differences in network characteristics such as the transmission speed, permissible maximum size, etc. It also guarantees that the packets if not received will be retransmitted and rebuilt in order. It also tries to maximize the utilization of the bandwidth, while ensuring that no side is overwhelmed.

Much like the IP datagram, the TCP datagram consists of a header and a body. The TCP headers consist of several fields that provide more information that is used on top of the information provided by the IP headers. The source and destination port fields are used to identify the endpoints used by the endpoints to identify the connection. The Seq and Ack numbers are used to initiate and maintain an understanding of the sequence of packets throughout a connection. The window size is used for congestion control and to signal whether the receiver is capable of handling more bandwidth.

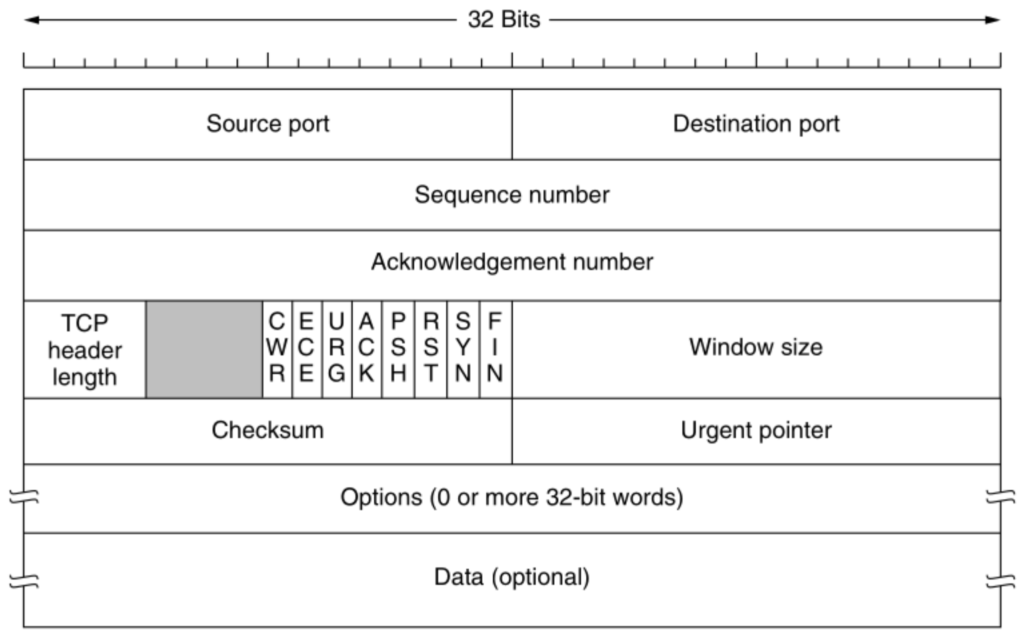


Figure 2.3: The TCP header

2.5. TCP 3-way handshake

In the context of Transmission Control Protocol (TCP) [7], which is inherently stateful, the establishment of a session is imperative for the initiation of communication between entities. The process commences with the initiating party sending a connection request, denoted by a TCP header where the SYN (synchronize) bit is activated, and directed towards the designated port on the target IP address. Upon receiving this request, if the port is engaged by an active process, the responding entity issues a reply with both the SYN and ACK (acknowledge) bits set, initiating the handshake process. The sequence number, selected randomly, and the acknowledgment number, derived by incrementing the sequence number of the incoming packet by one, play pivotal roles in this exchange.

2.6. Internet Scanning

Several botnets abuse this mechanism, using the nature of the 3-way handshake to send only the TCP SYN packets without completing the handshake, increasing the speed with which the host can scan the internet for other devices [8]. This mechanism has come to be known as stateless scanning, However when sending out scanning packets to random hosts on the internet, a challenge arises in maintaining the continuity of this communication process, particularly in remembering the sequence numbers for all outgoing requests or active connections. This complexity is addressed through the implementation of SYN cookies, a mechanism that employs a one-to-one protocol, such as a cryptographic hash, to generate a value based on various parameters including the IP address and port number. This innovative approach enables the host to verify the authenticity of incoming packets by comparing them against the hash function’s output, thus confirming their association with an ongoing session negotiation without necessitating the storage of these values in memory.

This mechanism, notable for its utilization by the Mirai botnet with a unique signature where the destination IP address is equal to the TCP sequence number, is also adopted by other variants employing more sophisticated methods. Such techniques underscore the evolving landscape of network security and the continuous adaptation required to safeguard communications within the TCP framework.

2.7. Modern Address handling methods

Modern ISPs have a variety of methods that deal with allocating and addressing addresses to devices within the network. This includes the usage of methods to preserve the limited amount of IP addresses

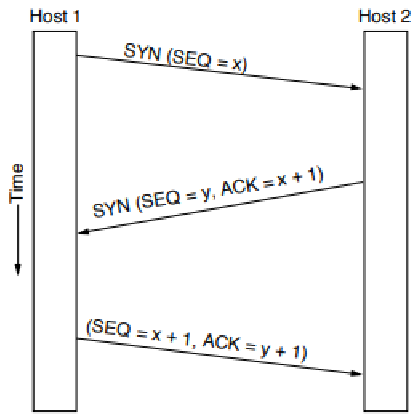


Figure 2.4: The TCP 3 way handshake

such as a Network Address Translation. Other methods involve providing an IP address to a host and then deciding when a host is not active to repurpose it for use for some other devices.

2.7.1. NAT

One of the major limitations of the current IPv4 protocol is the size of the address field which is limited to 32 bits. This leads to a hard ceiling of 2^{32} or approximately 4.2 billion addresses. The last block of IP addresses was issued in November 2019. This problem had been anticipated a long time ago and led to a search for alternatives to IPv4, which led to the creation of the IPv6 protocol. The IPv6 protocol handled the issue faced in IPv4 by having a 128-bit field for addresses which would greatly overtake the range provided by IPv4. However, adopting this protocol would require an overhaul of the current systems that make up the Internet. NAT was created as a solution to the lack of available addresses.

NAT works by assigning a single or a handful of external IP addresses to a block. The customer network has a range of internal IP addresses which are:

$$\begin{gathered} 10.0.0.0 - 10.255.255.255 \\ 172.16.0.0 - 172.31.255.255 \\ 192.168.0.0 - 192.168.255.255 \end{gathered}$$

These addresses are used to identify devices within the network and when an internal address wants to communicate with an external address, the gateway or **NAT box** that it passes through transforms the outgoing packet by changing the source address and source port of the packet to the external IP and a random value between 0 and 65536. It also makes an entry in a table it maintains. When the NAT box receives a reply, it performs a lookup at the destination port of the packet and then replaces the value with the cached internal IP and source port, ensuring that the port reaches the right device in the internal address.

While this is a good solution to the problem of the lack of IP addresses, this creates an issue in our analysis. While NAT initially works on an internal network, greatly reducing the chance of several infections linked to one external address, Several ISPs have adopted this on a larger scale using what are known as carrier-grade NATs. These lead to several home networks and devices behind one IP which could greatly skew any analysis we perform while measuring botnet activity.

2.7.2. IP Churn

As explained in section:2.7.1 IP addresses are a valuable commodity. To ensure that none of the IP addresses issued to them are wasted. ISPs often ”lease” out addresses instead of allocating an IP address to each user. This can be done using a variety of protocols such as DHCP/RADIUS/PPP etc. Broadly, these protocols ensure that addresses are leased to a user when it logs on and refreshes the lease duration if it receives requests from the user to keep it active. If there is no request to keep the lease, it would free the IP and return it to the pool of IP addresses. This allows the ISP to have a pool of IP addresses for active connections and not for total devices/networks.

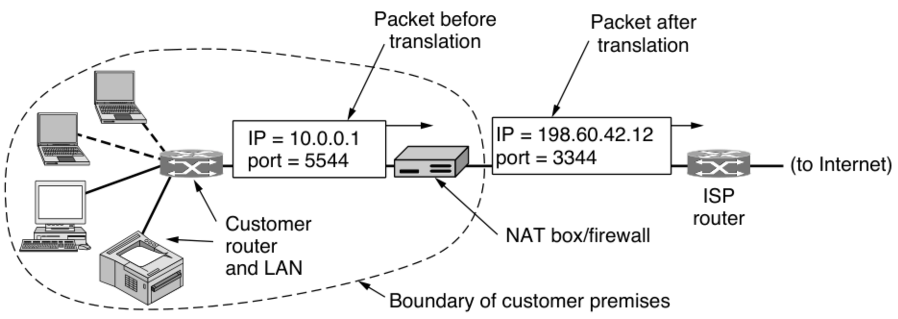


Figure 2.5: Illustration of NAT

In doing so it also means that an IP address is not a reliable identifier for measuring activity of botnets using IP counts. A study by Stone-Gross et al. [9] shows that there is great variability in bots per IP depending on factors such as the geographical location or the ISP. This could lead to a drastic overcounting or undercounting of the IPs by orders of magnitude.

2.8. Botnet monitoring methods

Over the years, the prevalence of botnets has led to an increase in interest in understanding their propagation and measurement. This led to the usage of several techniques that would take advantage of the activities performed by botnets to survive, propagate, and coordinate. These will be discussed in the following sections.

2.8.1. Network telescope/ Darknet

A Network Telescope, also referred to as a Darknet or Internet Sink, is a collection of IP addresses that do not actively run any services or host any infrastructure. Due to the lack of infrastructure, any traffic received is either unsolicited or due to misconfigurations. As most botnets scan the internet randomly and indiscriminately, having a large telescope or several distributed telescopes can allow the observer to easily recognize major events such as scanning campaigns, or attacks, by monitoring scanning packets or backscatter traffic from large DDOS attacks. Analyzing data captured from network telescopes is a challenging task due to the heterogeneity of the scanning traffic received. However, using methods such as clustering or monitoring trends over time often leads to interesting insights that make network telescopes an effective method for understanding the behavior of botnets over time.

2.8.2. Honeypot

A Honeypot builds on the functionality provided by a Network Telescope but adds a reactive element by emulating services that would be usually contacted, These could range from popular services such as telnet, FTP, etc to more obscure services that would be subject to exploitation such as routers, IoT devices with known vulnerabilities. This allows for dynamic interaction with the malicious parties, this is used to identify the nature of the exploit and even to understand the nature of the malware by analyzing the loading mechanism and the malicious executable.

2.8.3. Malware analysis

Several malware samples can be obtained by using honeypots or lifting them off of compromised devices. These samples can then be reverse-engineered to identify the activities of the botnet. By understanding the methods used to communicate with the C2 server and how scans are performed.

2.8.4. Sandboxing

Another useful way of understanding the way botnet malware works is to execute it in a protected environment. This way we can observe the interactions made by the malware, both with the host itself and the various interactions it has over the internet towards other devices. This helps gain more insight than static analysis of code by avoiding measures taken by adversaries to obfuscate the code or other important sections such as default credential lists that may be used to brute force the victims.

2.8.5. C2 milkers

C2 milkers are devices that emulate a bot that communicates with the Command and Control or C&C server, this allows the observer to understand the inner workings of the botnet and monitor botnet activity and directives over time.

2.9. Mirai

Mirai is one of the largest and most popular botnets in recent times and it along with its variants is the focus of this thesis. Mirai gained traction and popularity in late 2016 and performed some of the largest DDoS attacks ever on targets such as Dyn, OVH and Krebs on Security. Mirai utilised a simple vulnerability in the form of default credentials on commonly found IoT devices, specifically targeting telnet ports 23 and 2323. This simple vector of infection along with the rapid stateless scanning as discussed in Section 2.6 allowed it to grow to a steady state value of 200.000 to 300.000 infected devices. Mirai uses a distinct pattern in its scanning packets for its implementation of stateless scanning, this pattern involves setting the Destination IP address value in the TCP Sequence field. This can be expressed in the form:

$$IP.DstAddress = TCP.Seq$$

For the remainder of this thesis the terms Mirai packets or Vanilla Mirai packets will be used interchangeably to refer to these packets. At a later time, the Mirai source code got leaked, leading to a lot of other actors modifying the code to target new vulnerabilities and often using different implementations of stateless scanning that do not have the same signature as Vanilla Mirai, these packets will be referred to as non Vanilla Mirai or non-Mirai variants in this thesis.

2.10. LFSR

A lot of processes require a steady supply of random values for a variety of use cases such as generating keys, session ids, nonce values and so on. An ideal use case would use a TRNG or True Random Number Generator, however it becomes infeasible to use TRNG's for repetitive purposes due to the dependance on hardware for the repetitive generation of values.

In order to efficiently generate a large number of values, a PRNG or Pseudo Random Number Generator is used in lieu of a TRNG, A popular example of a PRNG is an LFSR or a Linear Feedback Shift Register. An LFSR consists of a series of registers, each holding a particular value. In every iteration of the LFSR a feedback function takes a combination of the values of the registers at particular positions known as taps, and performs an operation, usually XOR on these values, This new value is then set to the end of the registers and the other values are shifted by 1 position, this operation then continues every time a new value is generated, providing a reliable stream of values, that can be reproduced given the same starting values for the registers, known as the seed. Several programs use external sources of entropy to generate the seed value for the LFSR, such as epoch time, keystroke frequency, etc.

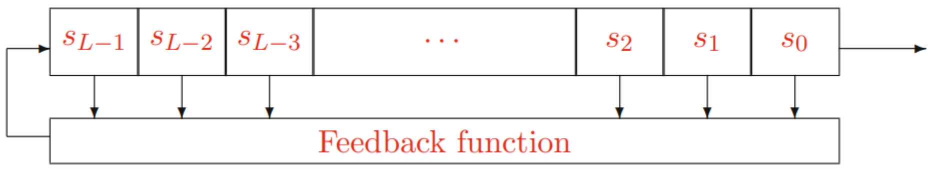


Figure 2.6: Generalised Linear Feedback shift register.

3 Related Work

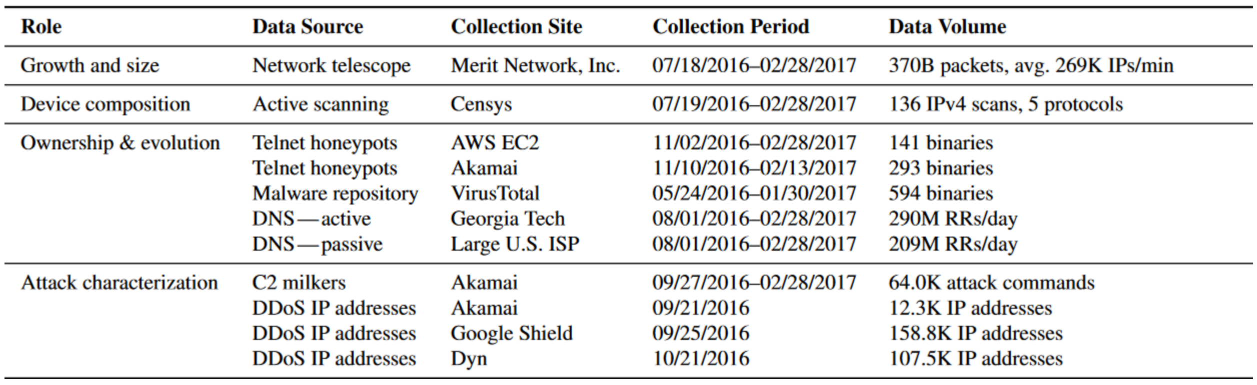
There have been a variety of studies that aim at quantifying and understanding the scope and nature of scanning activity that is undertaken by botnets. As the work in this thesis aims at understanding the evolution in activity of the botnets since previous studies, they will be briefly discussed in this section to provide an understanding for existing standards that have been used in this thesis. This way, the use of similar metrics allows for a more comprehensive understanding of the evolution in the scanning activities of botnets and other actors on the internet in general. Other studies that have been reproduced and then further explored for new insights are also included to provide sufficient background, the research gaps that were identified, and the work done over the course of the thesis that aimed at addressing them.

The studies picked for exploration fall in line with the work done over the course of the thesis and vary in nature with some focusing on statistics of scanning traffic while others focus on visualising scanning patterns to find discerning qualities in malicious and benign scans. There is also work done on understanding the lifetime of a bot within a botnet and also on trying to cluster botnet variants based on data gathered using passive measurement techniques.

The major work that provides an in depth analysis of Mirai on several fronts is the study done by Antonkakis et al. [10]. They use several sources of information such as network telescope for passive measurement data, banner scans, honeypots, C2 milkers, DNS traces and victim logs as shown in Figure 3.1 to view the spread and activity of the Mirai botnet from a high vantage point. They also employ static code analysis to understand the evolution of Mirai and its several offshoots. They utilise data from all these sources to understand the spread of Mirai from its initial appearance and then its steady state size. They also analyse the C2 infrastructure and DNS information to understand the various clusters that were active and to track the evolution and spread of the variants. The victim DDoS information is analysed to understand what kind of attacks the botnets usually perform. They also talk about the factors that lead to the Mirai botnet becoming as big of a threat as it is such as the lack of security controls in the IoT device industry, and discuss measures that could be taken to curtail further botnets. This work provides a detailed insight into every aspect of the Mirai botnet operation and proves to be an invaluable source of information for any further work in this domain. They do however depend on correlating information from several sources to gain these insights which in some cases can tend to be impractical. More work should be done on techniques that can gain similar insights with more constrained data sources.

3.1. Understanding scanning activity from passive measurements

One of the major studies that proves to be a cornerstone for initial analysis in this thesis is the work done by Durumeric et al. [12] where they provide an overview of scanning activities over the internet. This paper serves as an important framework for any analysis of scanning traffic, providing several statistics that provide context about the scanning activities, heuristics that can prove to be useful to decide what data to pick and discard based on factors such as the interarrival time and frequency of



Role	Data Source	Collection Site	Collection Period	Data Volume
Growth and size	Network telescope	Merit Network, Inc.	07/18/2016–02/28/2017	370B packets, avg. 269K IPs/min
Device composition	Active scanning	Censys	07/19/2016–02/28/2017	136 IPv4 scans, 5 protocols
Ownership & evolution	Telnet honeypots	AWS EC2	11/02/2016–02/28/2017	141 binaries
	Telnet honeypots	Akamai	11/10/2016–02/13/2017	293 binaries
	Malware repository	VirusTotal	05/24/2016–01/30/2017	594 binaries
	DNS — active	Georgia Tech	08/01/2016–02/28/2017	290M RRs/day
	DNS — passive	Large U.S. ISP	08/01/2016–02/28/2017	209M RRs/day
Attack characterization	C2 milkers	Akamai	09/27/2016–02/28/2017	64.0K attack commands
	DDoS IP addresses	Akamai	09/21/2016	12.3K IP addresses
	DDoS IP addresses	Google Shield	09/25/2016	158.8K IP addresses
	DDoS IP addresses	Dyn	10/21/2016	107.5K IP addresses

Figure 3.1: Data sources used in work by Antonkakis et al. [11]

packets from a source. Moreover some insight into understanding probabilistic models that could be used to understand whether a scanning packet would be detected by our darknet and the estimated number of packets that a darknet would observe is also provided here. They also show the temporal changes in port scanning activity over a larger period of time that would reflect how the targeted vulnerabilities or services of interest change over time. Other metrics such as countries of origin, packet interarrivals, etc are also discussed. Although this provides a solid foundation on what metrics are important to be discussed to be able to compare the state of botnets over time and variants, there is no work on actually using these discriminatory factors.

Another paper that closely aligns with the analysis we perform on the telescope data is that of Affinito et. al [13], where they focus on the evolution of the Mirai botnets and its various offshoots. This analysis provides an interesting overview of how newer variants propagate, based on new exploits and vulnerabilities and provides insight on shared characteristics between variants that could be used to identify new variants and the vulnerabilities that they target. They provide a comprehensive view on the rise and upkeep of various Mirai variants such as Reaper [14] which was a variant that focuses on HTTP vulnerabilities in IoT devices instead of the traditional telnet credential stuffing attacks. Other variants of interest mentioned include the Satori [15] and WICKED [16] variants which target ports 52869 and 37215, and ports 8080, 8443, 80 and 81 respectively. Other variants discussed in the study were Okiru, Masuta, PureMasuta, Miori, Yowai, Moboot, Sora, Unstable, Mukashi, Corona, Mange, ZHTrap, and VG34 [17–26].

This study provides a comprehensive review of the variants over a long period of time and the evolution in behaviour of attackers ranging from simple credential stuffing attacks, to a wide range of vulnerabilities in IoT devices. However this study is mostly retrospective and does not higlight ways to identify the state of the art attacks and ways to reliably cluster variants on live telescope traffic.

Sadegh et al. [27] also discuss botnet traffic from network telescopes at great length, but complement this information with device information from shodan to understand the types of devices that are infected and whether they are consumer products or belong to CPS realms to provide additional context to the scanning activities. They also track and draw correlations between the passive measurements and Threat Intelligence to understand the spread of various IoT botnet variants.

The works mentioned provide a good basis to begin the investigation into the dataset and to understand key metrics. However gaining insights such as bot lifetimes, time to infection, scanning activity, visibility of telescopes is an important and challenging research topic.

3.2. Visualising traffic

The work by Dainotti et al. [28] looks at a /0 scan event from a botnet and visualises it to understand the methodology undertaken by botnets when it comes to scanning large swathes of the internet address space.They provide interesting visualisations both of the temporal aspect of the scan as well as packet characteristics that are also used in this thesis. They also make a case for the fact that these visualisations may provide more insight for manual detection or correlation for scanning activities. Although this work was performed on the Sality botnet and does not focus on understanding factors such as , the methods and inferences drawn serve to be very useful when it comes to initial analysis of our data.

3.3. Understanding botnet sessions

Griffioen et al. [29] found a weakness in the implementation of the PRNG in the Mirai source code, this allowed them to gain a deep understanding into the lifetime of a bot and how the sessions behave. Furthermore they also investigate how various botnets compete to infect devices and show how it tends to become a zero sum game where no new infections occur and the variants cycle throught the existing pool of devices that are vulnerable. They also provide valuable information on how these infections are often lost due to reboots of the devices or are removed by competing infections. These activities can often be used as a discriminating factor over botnet variants and geographical locations. They also show that some botnet groups affect particular geographical regions as they target the particular models with the vulnerability in question.

Another paper by Griffioen et al.[30] takes advantage of the vulnerability in the RNG of Mirai as discussed before and uses the seed from incoming traffic to provide as a unique identifier of a particular infection. This would persist over extraneous factors such as NAT conversions or DHCP churn which are prevalent in modern carrier networks. They take advantage of the widespread Mirai infection and this characteristic in the scanning packets of the Mirai botnet to gain an extensive understanding of churn and NAT in consumer networks in several regions in the world.

3.4. Research gaps

There is a need to understand the state of the Mirai malware in recent times. As shown in some of the related works, the Mirai botnet is still prevalent and active, with some new variants still showing up which target newer vulnerabilities and exploits instead of just relying on credential stuffing attacks.

Being able to find disciriminatory factors between traffic from botnets and regular scans is also an important topic as several new variants modified the original source code to remove the trademark Mirai signature, finding ways to characterise them is essential to be able to keep monitoring and defending from the botnet threat.

There is little work done to understand the efficacy of telescopes in capturing the true state of scanning traffic and the few works that do address this issue often rely on comparisons between telescopes which only provide us a relative understanding instead of an empirical measure.

Finally, being able to succesfully understand the scanning rate of devices in the botnet and also to be able to understand the time between the device infection and scanning hits provides an important understanding of when packets hit the telescope and may actually provide some more information about the scanning rates and behaviour of devices present in the botnet.

The aim of this thesis is to provide an updated view on the state of the Mirai botnet, to be able to find discriminatory factors in scanning traffic that may allow us to characterise botnet traffic that does not have the Mirai signature and to gain additional understanding on the visibility of our telescope from additional insights that we gain from cracking the RNG seed from packets received

4 Dataset Characteristics

The dataset used in this thesis is traffic collected from a network telescope operated by TU Delft. The network telescope operates on 3 /16 ranges leading to a total of $3 * 2^{16} = 196.608$ addresses, however all of the addresses in the ranges are not consistently used to log incoming packets. Some of these addresses are in use by hosts connected to the TU Delft and are not used for logging any incoming traffic. The sources of traffic logged are mainly TCP traffic which is also the traffic that is used in this study.

In this chapter we will take a deeper dive into the various characteristics of the packets that are captured by the network telescope. Furthermore we also try to understand whether there are any discerning factors or trends between known Mirai traffic and other scanning traffic when looking at it from a much broader perspective. In order to do so using telescope traffic, we try to look at patterns or visualisations for both mirai and non-mirai traffic in this section.

4.1. Packet Statistics

We start by taking a look at aggregate statistics on all the TCP packets to understand the difference in nature for sources that send known botnet packets that contain the Mirai signature, in order to identify any discriminatory factors as per our first research question.

4.1.1. Flags

As only TCP packets are considered for this study, the distribution of other packets is not considered. However, there are some interesting observations made when looking at the flags that are set for the incoming TCP packets. All possible combinations of flags are looked at to find any anomalies or unexplainable behaviour. When looking at only one field set as shown in Table 4.1, most packets sent have only the SYN flag set, followed by ACK and RST. The other single flag packets are negligible.

Flag	Number of occurrences
SYN	10.026.882.738
ACK	19.261.207
RST	3.855.029
Other	≈2500

Table 4.1: Frequency of packets with singular flag

Combinations of two set flags also provide insight as to other internet activities.. Table 4.2 shows the most common two flag combinations of incoming packets. For instance, the SYN+ACK or RST+ACK packets may indicate some form of backscatter from a host that is suffering a DDOS attack, where the attacker uses a spoofed IP address that falls within our range. Looking at these packets may indicate the intensity, frequency and techniques used in DDOS attacks. PSH+ACK packets are usually used to request the remaining data to be sent to the receiving party.

There are also other packets that have 3 or more fields set, as most of these are negligible in number, they have been omitted apart from a few that are still significant. Although these are used in

Flag	Number of occurrences
SYN+ACK	98.829.534
PSH+ACK	4.809.132
RST+ACK	2.420.713
SYN+CWR	2.177.786
Other	≈150.000

Table 4.2: Frequency of packets with two flags set

normal TCP exchanges, it is interesting that they appear in a passive monitoring system such as the network telescope. This could be due to a recently active IP being added through the telescope or due to a misconfiguration on the senders side.

Flag	Number of occurrences
SYN+ECE+CWR	12.903.745
SYN+PSH+ACK	369.015
FIN+SYN+PSH+ACK	35.774
FIN+SYN+RST+PSH+ACK+URG	225.215

Table 4.3: Miscellaneous packet fields with high occurrences

4.1.2. Cumulative scanning activity

An interesting observation made when looking at the total number of Source IPS that sent Mirai traffic to those that didnt was that IPs that sent atleast one packet with the Mirai signature was 46% of the total number of IPs that sent any traffic to the network telescope. However the number of packets sent to the telescope with the Mirai signature contributes only 1.5% of the total packets received at the telescope over the duration of the entire month of June.

In order to ensure that the numbers are not inflated due to factors such as IP churn, the same ratios are calculated over a much shorter period of one day. The ratio of IPs that send Mirai scanning traffic to those that dont is 45% and the ratio of packets is 1.9% for the 20th of June, these values fall within a reasonable margin of error to assume that the values obtained over a month period are also reasonable.

This would align with the consensus over other studies [10] which estimate the scanning speed of Mirai to be around 250 bytes/second while other famous bots such as SQL Slammer scanned faster by a magnitude of difference, which points to the fact that Mirai targets IoT devices which are low powered devices, some of which are geographically located in countries with less robust internet speeds and availability, which would mean that these factors lead to a much slower scanning speed compared to other botnets or worms in the past.

Table 4.4: Ratio of Source IPs and packets for Mirai and other activities

Type	Month	Day
Unique Sources	46%	45%
# Packets	1.5%	1.9%

4.1.3. Source IPs

Source IPs signify what address the packets are sent from. As the senders usually require a response, it is safe to assume that these values are actual indicators of the infected devices themselves and are not spoofed. We look at trends in the Source IPs and distinguish on the presence of the Mirai signature. Figures 4.1 and 4.2 show that the number of packets received that have the Mirai signature are way more uniform, with most src ips close to the average amount of packets sent, with a few that send much larger number of packets, while this difference is a lot more pronounced in Source IPs that send packets without the Mirai signature. The tall thin lines and negligible height of the average number of packets show us the vast difference between the normal scanners and the ”heavy hitters” in normal scanning traffic.

This behaviour can also be understood by looking at the CDF of the number of packets sent from indiviudal source IPs shown in Figure 4.3. It is clearly observed that there is a much higher concentration of Mirai source IPs at the lower values compared to packets without the signature.

Another noticeable difference between Source IP addresses as seen in Figure 4.4 in packets sent with or without the Mirai signature ($IP.dstIP == TCP.seq$) is the difference in average interarrival times, On average Mirai packets tend to have a relatively lower time between packets sent, while normal scanning packets have a lower peak early and a much longer tail meaning that packets sent from other scanning sources also keep sending traffic over a much longer time. The fact that a large number of IP addresses send packets with low interarrival time suggests that these scans occur in short bursts of concentrated activity followed by longer pauses of inactivity.

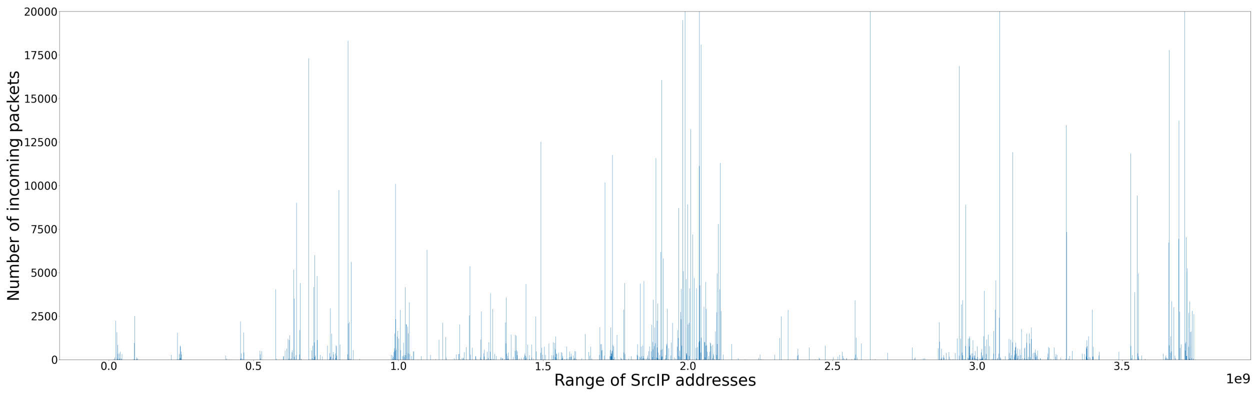


Figure 4.1: Distribution of source IPs that send Mirai packets as observed over 3 days

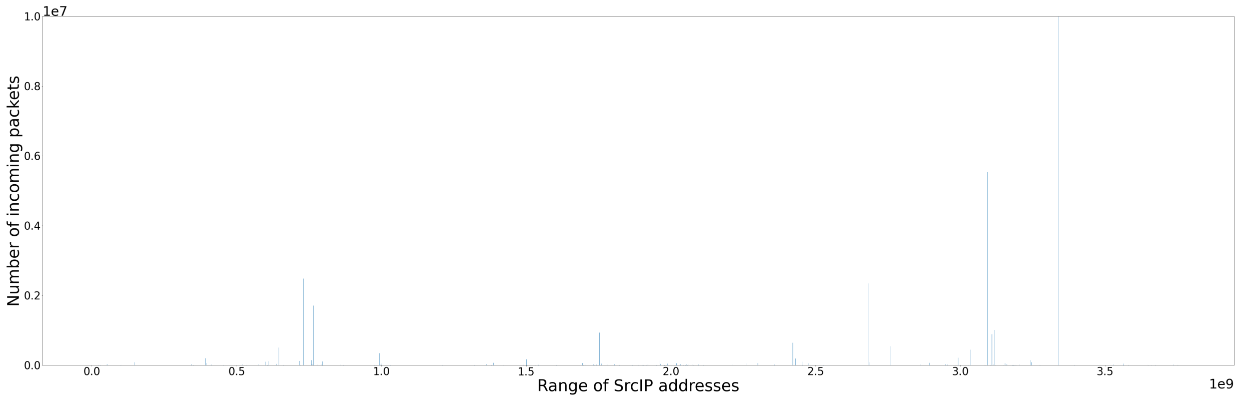


Figure 4.2: Distribution of source IPs that send packets without Mirai signature as observed over 3 days

4.1.4. Destination IPs

Looking at traffic received in Figure 4.1 shows a uniform distribution of packets received over one selected /16 range out of 3 available ranges, although traffic from all 3 is used in the analysis, similar behaviour is observed on the other two /16 ranges as well. The empty bands in between are due to the nature of allocation of IP addresses to the telescope, as traffic directed to some ranges which are in use is not captured by the telescope. The uniform distribution of packets received is expected as most scanners either ensure equal coverage or employ the usage of a Random Number Generator or RNG to decide the scan target, both these cases would provide a similar observation and it is hard to understand whether the behaviour is due to an advanced adversary performing a coordinated scan that covers the exact space or whether it is due to randomly picked IP addresses with a decently well designed Random Number Generator.

4.1.5. Destination Ports

The Destination Port of a TCP packet is used to determine the service that the packet is sent to. These are among the primary indicators that help us understand what kinds of services are being targeted by

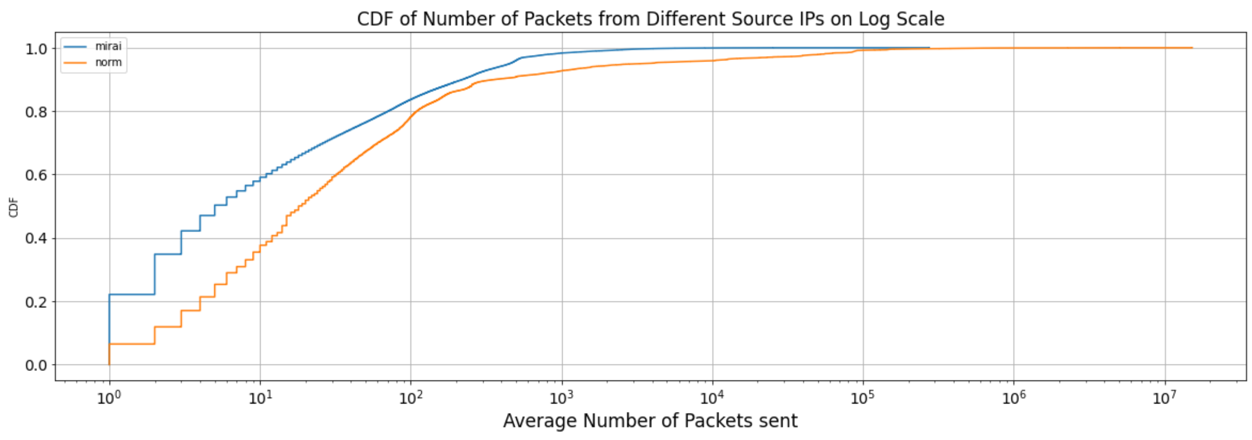


Figure 4.3: CDF of packets sent from individual source IPs as observed over a 3 day period.

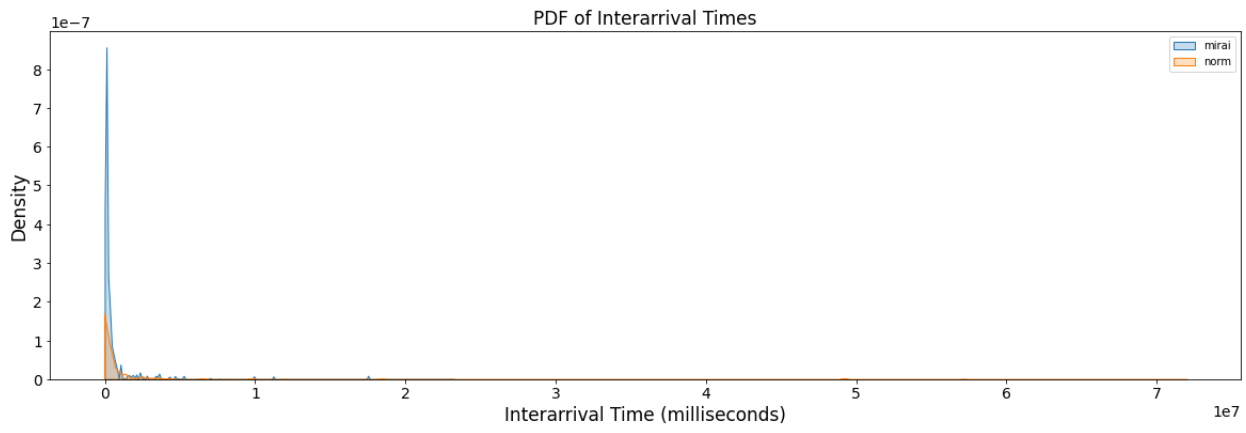


Figure 4.4: PDF of interarrival times between packets

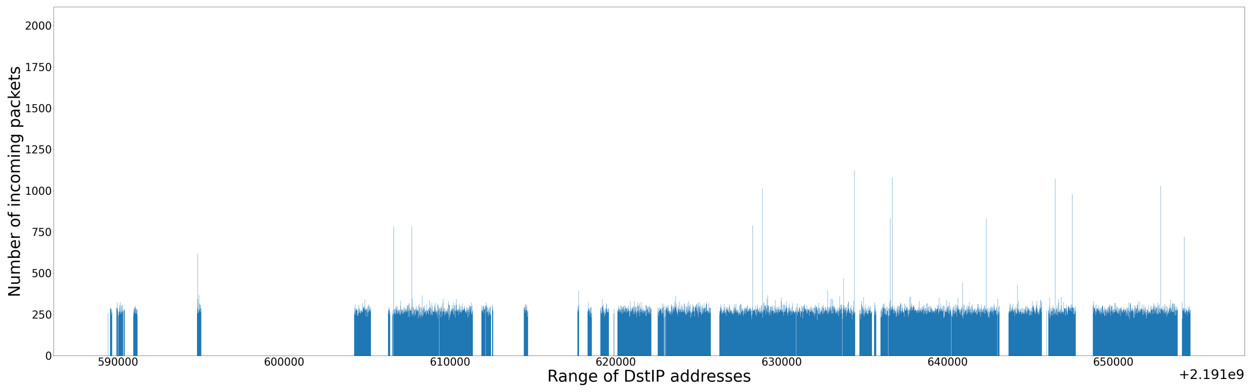


Figure 4.5: Number of packets sent to Telescope IP addresses

botnets. There is a significant difference in the ports that are targeted by Mirai and non-Mirai scanning packets. Table 4.5 shows the top 10 ports that both scanning packets target.

Some of the ports more commonly targeted by mirai are 52869 and 37215 which are mainly considered to be introduced in a Mirai variant named Satori[15], which target the specific vulnerabilities CVE-2014-8361 [31], which is a Remote Code Execution vulnerability in the miniigd SOAP service in the realtek SDK and CVE-2017-17215 [32] which targets a Remote Code Execution vulnerability in Huawei routers. Port 5555 is also a commonly targeted port, which targets open ADB or Android Debug Bridge which allows the attacker to remotely execute code on these vulnerable devices[33, 34]. These

Mirai	Percentage	Non-Mirai	Percentage
52869	22.24%	80	1.87%
2323	19.44%	22	1.72%
60023	13.53%	8080	1.6%
22	10.53%	3389	1.56%
8080	6.6%	443	1.24%
5555	5.51%	5555	0.93%
443	4.37%	81	0.64%
37215	2.87%	3128	0.61%
56575	2.78%	8081	0.59%
80	2.42%	8443	0.59%

Table 4.5: Most frequently targeted destination ports by packets with and without Mirai signature

ports mainly target RCE or Remote Code Execution Vulnerabilities and are relatively newer additions to the Mirai botnet. Other ports such as 37777, 7547, and 6789 are also vulnerabilities that are targeted by known Mirai variants [35].

The ports 2323. 22, 8080, 3389, 8081, and 8443, are all ports that are used by well known services, these ports are also targeted by researchers, and measurement companies such as shodan. These are also targeted by botnets through credential stuffing campaigns, services such as telnet and RDP are often configured with default or simple credentials which are then subject to a dictionary attack. Unlike the ports that target known vulnerabilities, it is fairly difficult to discern whether the scan is malicious or not.

4.1.6. Source Port

Looking at the distribution of the source port values used by the scanning packets yields some interesting results. The packets sent with the Mirai signature seem to be somewhat biased towards some values as seen by the largely varying number of packets received, which could be used to make some observations on the distribution of values generated by the PRNG used by the Mirai botnet.

The packets sent without the Mirai signature are more uniform on average, but three distinct groups can be noticed as a step function. This could be due to how operating systems allocate ports to applications. For instance windows operating systems usually allocate these ephemeral ports to applications in the range $49152-65535$ [36] and the linux kernel allocates ephemeral ports in the range $32768-65535$. However there is also a bias towards the lower values of these ranges as seen in 4.6. We can use the insights from this distribution to gain a ballpark understanding of the systems that are used by these scanners. These are under the assumption that there is no major difference in the scanning rate and behaviour in these separate systems, but gives us a rough estimate of the distribution of systems.

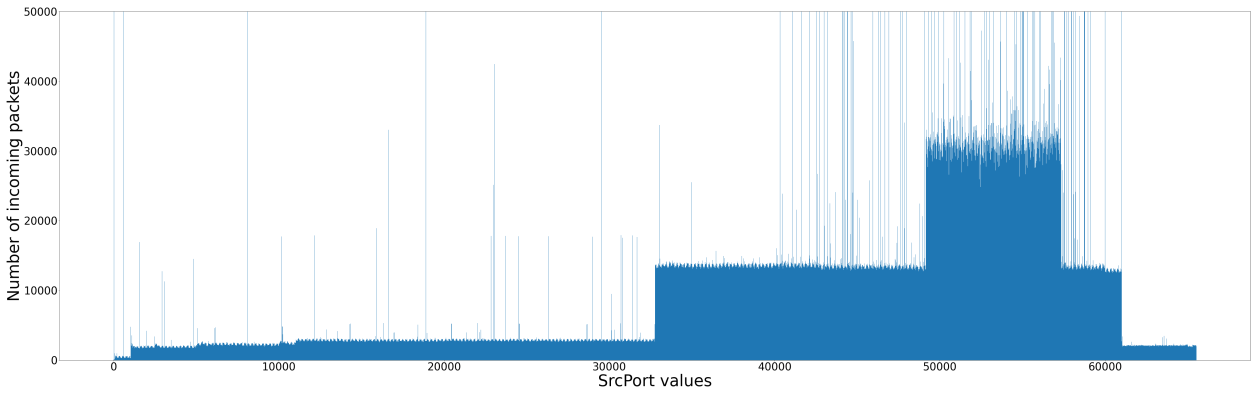


Figure 4.6: Number of packets without the Mirai signature sent by a Source Port

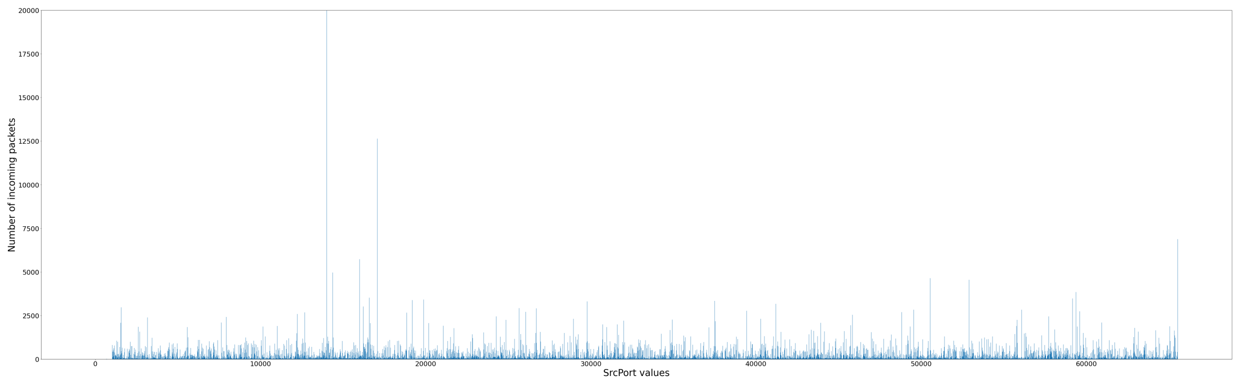


Figure 4.7: Number of packets with the Mirai signature sent by a Source Port

4.1.7. IPId

Mirai packets have uniformly distributed IPIds, while packets sent without the Mirai packet have mostly uniform distribution as well while having a few values that are much larger than the average spread. These could be due to the presence of certain scanning tools that use hard coded values such as 54321 as is used by ZMap. Zmap contributes a large amount of scanning traffic contirbuting to almost 40% of the total scanning packets seen in the telescope.

IPID	Percentage
54321	39.99%
0	2.30%
46280	0.4%
256	0.31%
58544	0.27%

Table 4.6: Most frequent IPId values by packets without Mirai signature

4.2. Packet Field combinations

Apart from trends noticed in single packet fields, there are also combinations of fields in scanning packets that may help determine interesting features in either botnet or normal scanning. Often, visual inspection provides interesting views of the data that may otherwise be hard to understand. This subsection will highlight some interesting combinations.

4.2.1. SrcPort and DstPort

Looking at a scatter plot provides little information apart from what is already seen when inspecting single values of packet fields due to a lack of visualisation of several packets being sent with the same SrcPort and DstPort values as these would just overlap each other and be represented with a singular point. Similar inferences can be made regarding commonly targeted destination ports, as can be seen as the horizontal lines in Figure 4.8.

More interesting insights can be found when looking at a bubble plot of the same data. As seen in Figure 4.9 there are much larger bubbles seen along $x = 0$, which could signify some botnets set a default value for the SrcPort at 0 as opposed to randomly generating values, or a flaw in the LFSR used that biases it to certain values, Other large bubbles could exist due to much larger session times/scanning speeds for bots that could be due to some devices staying online much longer or existing in areas with higher bandwidth capabilities, this hypothesis can be made due to the fact that vanilla Mirai variants randomly set a SrcPort and Window value at the start of a session before sending out scanning packets, These large values indicate that the session with the particular SrcPort value lasted longer. This could also be confirmed by looking at the bubble plot of SrcPort and Window combinations.

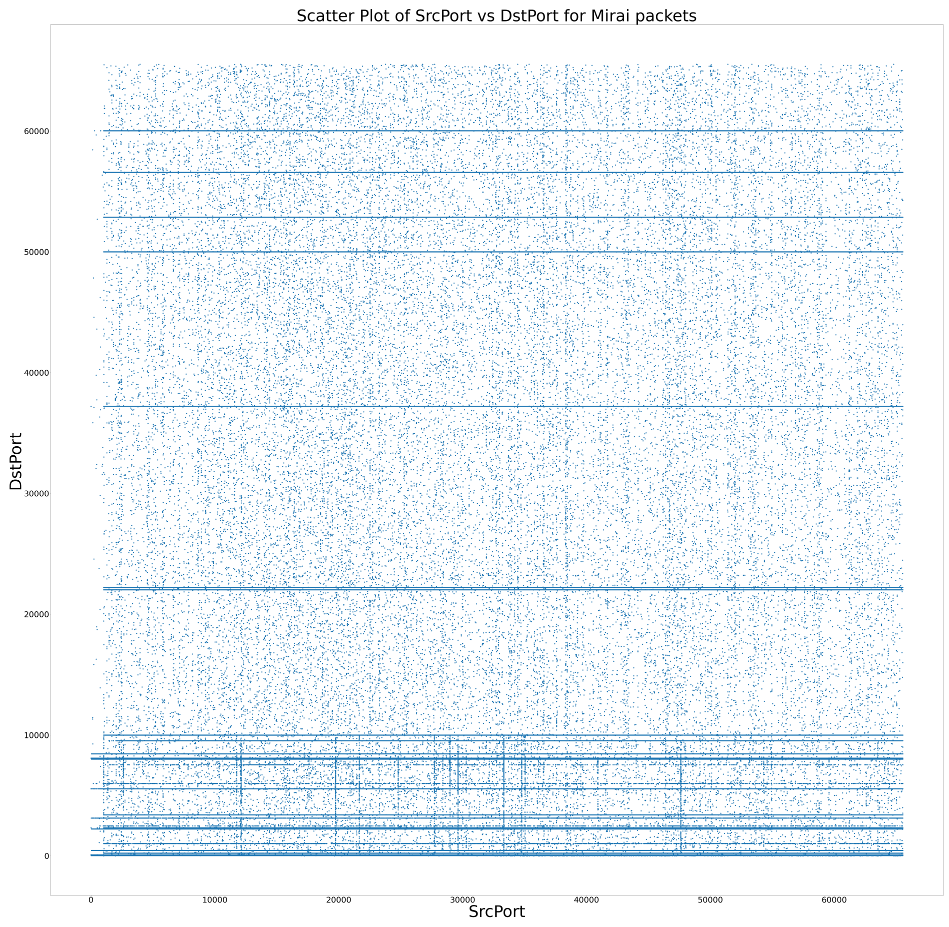


Figure 4.8: Scatter plot of Source and Destination ports of packets sent with the Mirai signature.

4.2.2. SrcPort Window

Looking at Source Port and Window size should provide an overview of how many packets were received during a particular session of a vanilla Mirai bot. As seen in Figure 4.10, there are several bubbles with uniform size which could reflect the average expected activity of a bot during its lifetime. A lot of these bubbles are also concentrated along the $x = y$ line. As these values are picked as consecutive values of the LFSR, there might be a relation due to the generated consecutive values being in close proximity of each other, however this was not verified.

The source port and window combinations as shown in Figure 4.11 show distinct horizontal lines which would indicate commonly used window size values which are in line with what values are usually negotiated over handshakes or even over the course of a TCP connection.

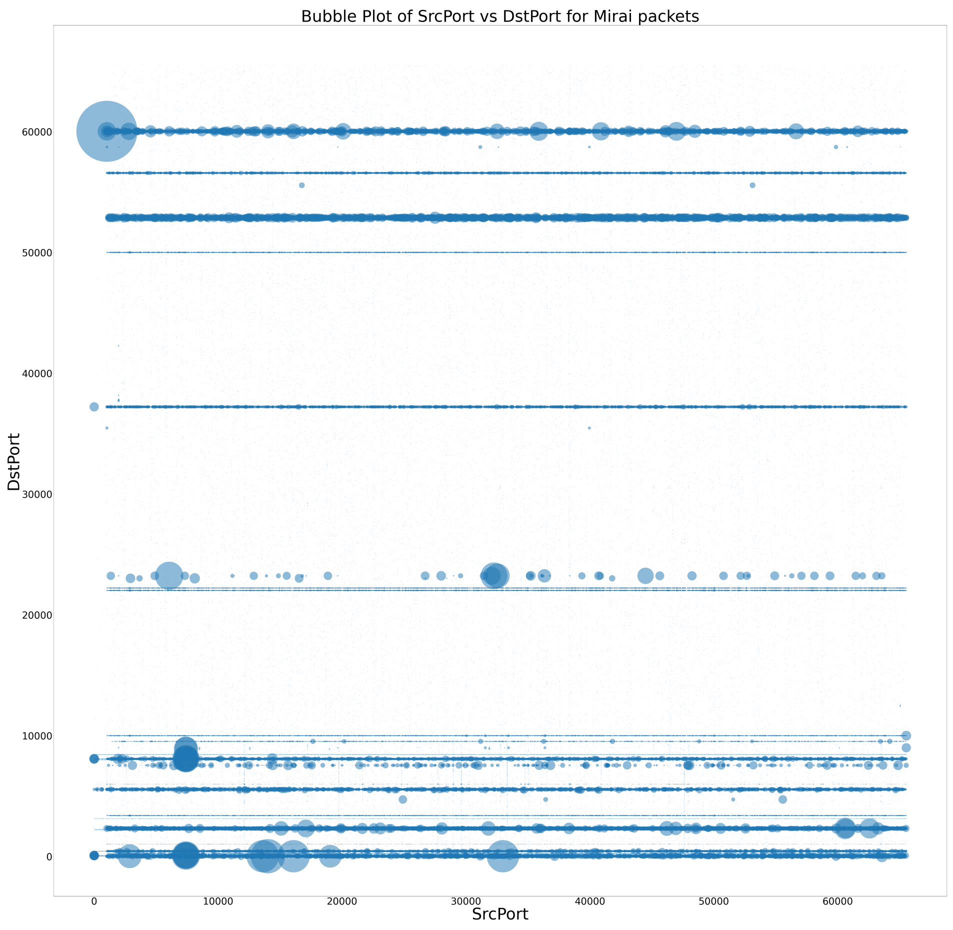


Figure 4.9: Bubble plot of Source and Destination ports of packets sent with the Mirai signature.

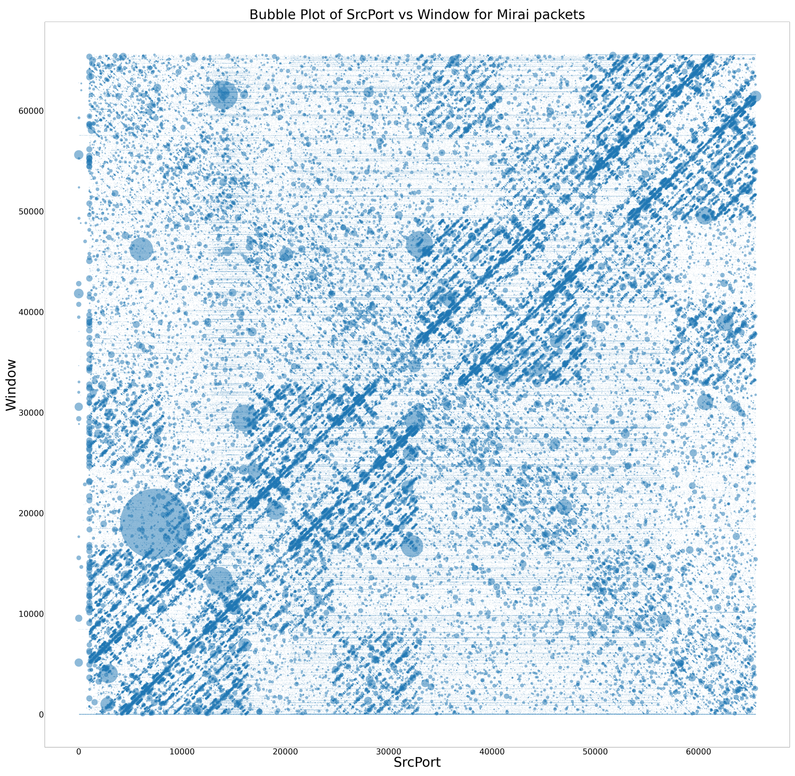


Figure 4.10: Bubble plot of Source Port and Window Size of packets sent with the Mirai signature.

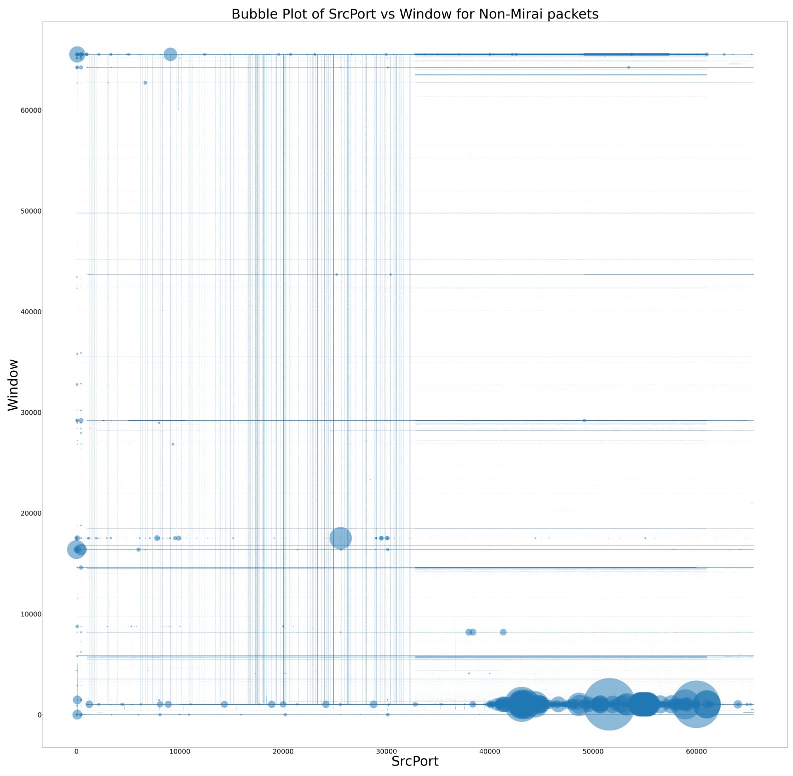


Figure 4.11: Bubble plot of Source Port and Window Size of packets sent without the Mirai signature.

5 Methodology

This section details how further investigation was performed to gain a better understanding of the insights gained when the telescope data was explored. Technical explanations of implementations done to extract more information about the data are also included in this section.

5.1. Reinfections

Due to the large amount of scanning traffic received, it becomes fairly hard to characterise which packets are sent from botnets, and more specifically botnets of interest to us which are variants of or have similar characteristics to the Vanilla Mirai botnet that has been extensively studied. In this section we detail some insights gained when analysing the telescope data that could help us use our existing knowledge to filter out scanning packets which dont have the Mirai signature, and how we can create rules that can be used to filter out various types of variants and behaviours.

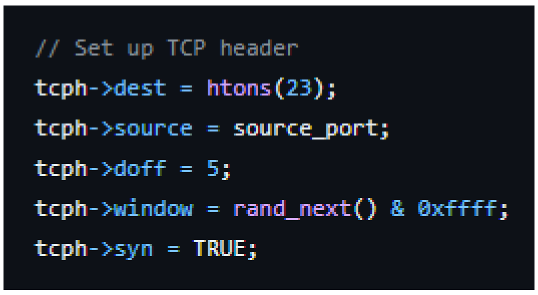
When looking through the data we can observe certain IP sources where packets with and without the Mirai signature are observed subsequently or interleaved between each other. This phenomenon could be explained by referring to the nature of infections usually seen when dealing with botnets and vulnerable devices, which is seen in work done by Griffioen et al. [29] which in turn refers to the work of Kermack et al. [37]. In their work Griffioen et al. posit that the infections of vulnerable devices on the internet by botnets follow the SIS (Susceptible-Infected-Susceptible) process where a device could be infected by a botnet variant and then the device would either restart or the adverse effect is noticed by the device owner prompting them to reset or format the device without addressing the root cause, which is the vulnerability or the SIR (Susceptible-Infected-Recovered) process where the device might be updated to address the vulnerability that the botnet was propagating through. The infected devices undergo the SIS process as the vanilla Mirai variant and other variants often do not employ any means of persistence on the device. This would mean that the malware does not persist on the device between reboots. This section focuses on the SIS process which would present in a variety of scenarios:

- Devices that have not been secured or are still vulnerable to these attacks are more prevalent in developing countries, which may have unstable electric grids or network connectivity. This may lead to the devices often restarting entirely when the power returns or having to establish a new connection once the network connectivity is regained.
- Competing botnets might take over the same device, this was noticed earlier among the vanilla variants and was even addressed by some newer variants which would entirely shut off the services once they infect a device to make sure another strain does not infect the device. However only a few services are restricted which might mean that there are other variants that may still take over the already infected device.
- The code written for the botnets could be unstable and subject to crashes due to poor error handling or coverage of cases in case the existing code was adapted by a less experienced attacker.

This fact paired with the knowledge that there are several variants that actively scan the internet, with a large amount of overlap of vulnerabilities targeted, means that the devices that undergo the SIS process would eventually be reinfected, either by the same botnet variant or a new variant, depending on which random scans discover it first.

Vanilla Mirai scans have another discerning feature that assists in inferring whether a period of inactivity is due to a pause in scanning or rather due to the device reverting to the Susceptible stage of the SIS process. Looking at the Mirai source code as in Figure 5.1 shows that the SrcPort and Window values are randomly generated at the start of the infection and then are used for every scanning packet that is sent by the infected device. Tracking this SrcPort Window combination can help in tracking the state of the device with a reasonably high confidence, as the probability of collision in values of 2 16-bit fields is $\frac{1}{2^{32}}$.

When looking at incoming packets with the Mirai signature and providing unique values in increasing order to SrcPort and Window combinations, some IPs present in the fashion as seen in Figure 5.2. Different combinations are given incremental numerical values much like Label encoding to make a comprehensive visualisation of the reinfection of these devices. The blue dots at $y = 0.5$ signify incoming packets without the Mirai signature. The blue dots at $y = 1$ signify incoming packets with the Mirai signature, and the orange dots at values $y = 0, 1, 2, 3$ and so on signify the session of the Mirai packet depending on the SrcPort and Window combination. This allows us to understand the progession in sessions from a singular host and also the occurences of non-Mirai packets in between.



```
// Set up TCP header
tcph->dest = htons(23);
tcph->source = source_port;
tcph->doff = 5;
tcph->window = rand_next() & 0xffff;
tcph->syn = TRUE;
```

Figure 5.1: Portion of Mirai Code that sets the Source Port and Window combination at the start of the session [38]

In some cases as seen in Figure 5.3, there are multiple unique sessions sending packets with the Mirai signature simultaneously, which has a possible explanation by Griffioen et al [29] to be due to multiple device infections behind a NAT.

In certain cases however, there are some packets not having the Mirai signature observed before, in between or after packets sent with the Mirai signature. These could be hypothesised to be scanning packets sent from a Mirai variant that does not implement the Mirai signature, opting for a different method to verify the response to the scanning packet from the target device. An example of this can be seen in Figure 5.4 where certain non-Mirai scanning packets close to other Mirai scanning packets are observed in the horizontal line $y = 0.5$.

As part of the work in this thesis, the scanning packets without the Mirai signature that are observed in close proximity of Mirai reinfections are filtered and stored in a dataset for examination. There are a number of measures taken to ensure that there can be a reasonably high confidence that the packets that have been filtered are what they are hypothesised to be:

- Packets must only be considered if there are Mirai reinfections before and after, to ensure that the device is not in the SIR or Susceptible-Infected-Recovered process or was sending out scanning packets as part of a legitimate process prior to its infection.

- The previous and subsequent Mirai scanning sessions should be within a particular window of time to provide a certain degree of confidence that the IP address is associated with that of the infected device. Keeping a short time frame along with the chances of a churned IP also sending scanning packets with the Mirai signature provides a relatively high degree of confidence that these are not separate infections that underwent churn.

- The number of unique infections that the non mirai packets are observed between should be limited to discount cases where minimal packets are sent

- The time window for analysis should be within a particular threshold to avoid any errors in device attribution due to network churn.

Keeping in mind these precautions certain parameters were set, including setting a 2 hour window before and after and ensuring that the non-mirai packets having greater periods between mirai scanning packets were not considered. Looking at works by Griffioen et al. [30] and Moura et al. [39] leads us to believe that the sessions observed are not long enough to have to factor in churn. The observed sessions range from a few hours to less than a day while these works show that IPs would usually churn from 15-60 hours. However to have even more reliable results, the same observations could be made using a sliding window of a shorter period.

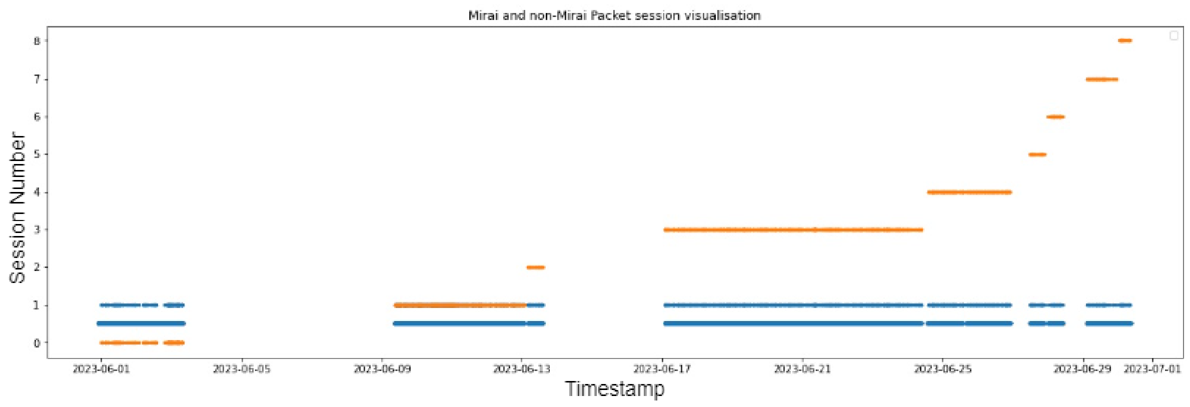


Figure 5.2: Visualisation of reinfections of device with variance in SrcPort/Window combination

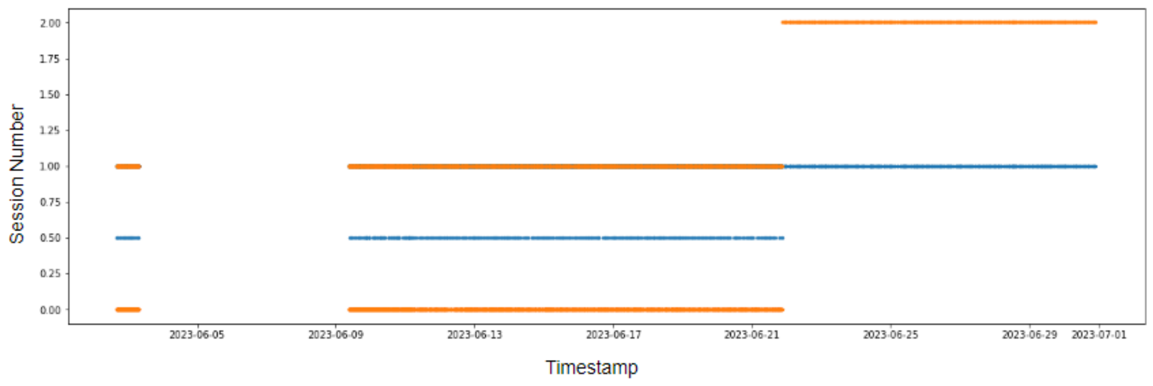


Figure 5.3: Visualisation of multiple device infections assumed to be behind a NAT.

5.2. PRNG

As seen in previous works such as by Griffioen et al. [30], there exist certain oversights in the design of the random number generator used by vanilla mirai that can help provide some interesting insights on the nature of the Mirai infection. In this section, we aim to build on the work and to use the information we gain to supplement the existing data we get from the network telescope.

Looking at the source code of the Mirai Botnet reveals the mechanism that is used to generate the random values that are often used in various activities of the bot itself. The bot uses generated random values to set values for various packet fields such as the SrcPort, the Window Size, the IPId, etc. It also employs a random scan of the internet by randomly picking the Destination IP to target for scanning next. Moreover, the same Random Number Generator is also used for other activities such as packet

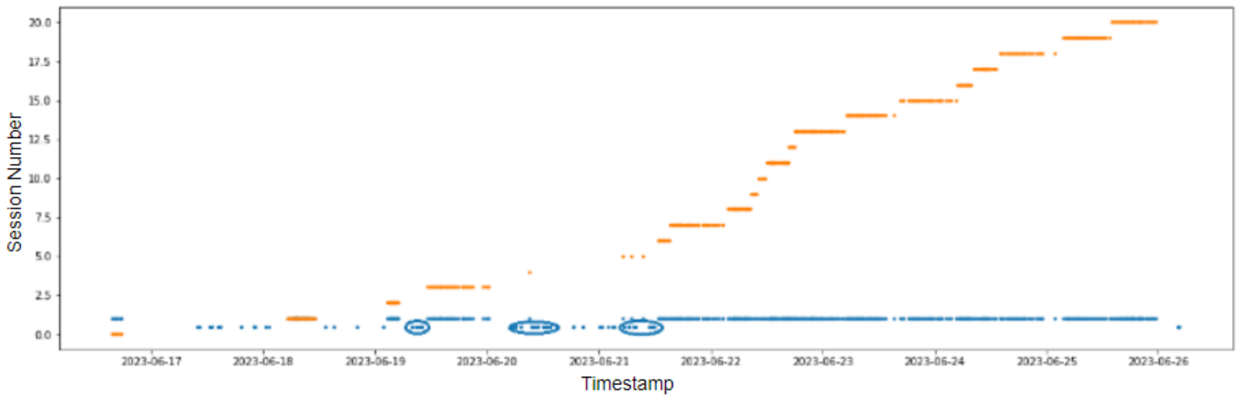


Figure 5.4: Visualisation of non-Mirai scanning packets between Mirai scanning packets.

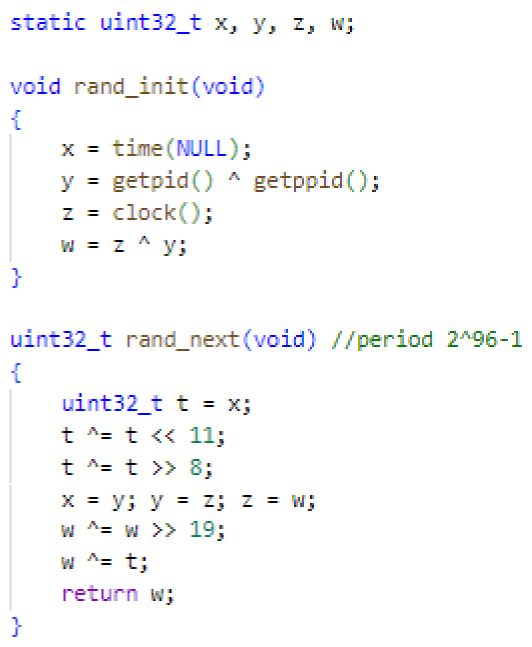
value generation for DDoS attacks as seen in the calls to the same function , where the packet fields alongside the spoofed source IP are also set. As such the PRNG used is an integral part of the bot design.

The PRNG used by the vanilla Mirai variant is a Linear Feedback Shift Register or LFSR. It also has sufficiently even bit distributions and has passed the Dieharder test suite[40] that is commonly used to check the randomness of PRNGs as per the work of Riegel [41]. It suffices to say that the PRNG produces output that is sufficiently indistinguishable from a truly Random Number Generator. However LFSRs are only effective when they can ensure that the state of the n registers that is uses is hidden at all times. Moreover leaking n consecutive values provides us enough information to identify any values generated by the LFSR. Due to the nature of an LFSR even a subset of registers along with some educated guesses can greatly reduce the complexity of identifying the initial starting state or "seed" of the PRNG. However it is fairly difficult to consistently generate past values from when the state is known due to the usage of the XOR function in conjunction with the bit shifts, so it is beneficial to be able to identify the seed values to the PRNG to ensure that all possible values are identified.

The implementation of the PRNG in Mirai has 4 registers that store the current state of the PRNG. The PRNG has a large cycle length of $2^{96} - 1$, so it is fairly improbable for repetitions to occur. The initial values of the PRNG are seeded by taking some environmental variables as seen in 5.5. The first value seen as x, is taken as the epoch time at the time the PRNG is initiated. The second value y, is the XOR of the PID of the current process and the PID of the parent process that spawned the current process. The third value, z is the time in ms since the start of the process. The fourth value, w is the XOR of the last two generated state variables.

As seen in the Mirai code snippet in Figure 5.6 [38] we can see how Mirai bots scan the internet. We can see the main thread create the scanning thread at ①. The random number generator is then initialized with the seed values at ②. The source port is set at ③, making sure that the generated value is truncated to include the last 16 bits and also that it does not exist within the protected range of source port values. The IPID is then set at ④. Then it begins the infinite scanning where it sets the IPId again at ⑤ and sets the destination IP at ⑥, which is the address that it picks to scan randomly. There is a discrepancy caused by this method. As seen in the code for get_random_ip in Figure 5.9 it generates a random IP address and checks whether it is in a restricted range [11], if so it regenerates an address. This would mean that the values are not consecutively generated by the PRNG, even though we can account for this, the likelihood of it occurring is rare, so we do not check for it.

In the case of Mirai, there is no output transformation of the values generated by the PRNG when they are used to set the values of the TCP fields in the scanning packets. This means that if we were to observe four PRNG values in the TCP fields we would have the state of the PRNG at that instant. However this is not possible for a variety of reasons. Firstly, when looking at values that remain persistent over transmissions, there are two reliably set values by the PRNG that are observed, which are



```
static uint32_t x, y, z, w;

void rand_init(void)
{
    x = time(NULL);
    y = getpid() ^ getppid();
    z = clock();
    w = z ^ y;
}

uint32_t rand_next(void) //period 2^96-1
{
    uint32_t t = x;
    t ^= t << 11;
    t ^= t >> 8;
    x = y; y = z; z = w;
    w ^= w >> 19;
    w ^= t;
    return w;
}
```

Figure 5.5: Mirai PRNG implementation [38]

the IPID and the Destination IP. In order to get four consecutive values, two consecutively generated packets would have to hit the telescope. Considering that the size of the network telescope is 2^{16} addresses, the odds of two consecutively generated packets hitting the telescope is $\frac{1}{2^{16}}$, which means that 1 in 65536 packets from the same host would have a consecutive packet also hit the telescope. Considering the scanning rate of hosts observed, the odds of finding this occurence are astronomical, and there is still missing information as the IPID field would have a truncated 16 bit value instead of the entire 32 bits.

Secondly, in the case that telescope receives the first generated packet after the initial SrcPort and Window values are set, 4 consecutive values of the PRNG will be recorded, however the values of the SrcPort and Window size are actually truncated 16 bits of the generated 32 bit values. This would mean that there would still be insufficient information to deduce the state of PRNG.

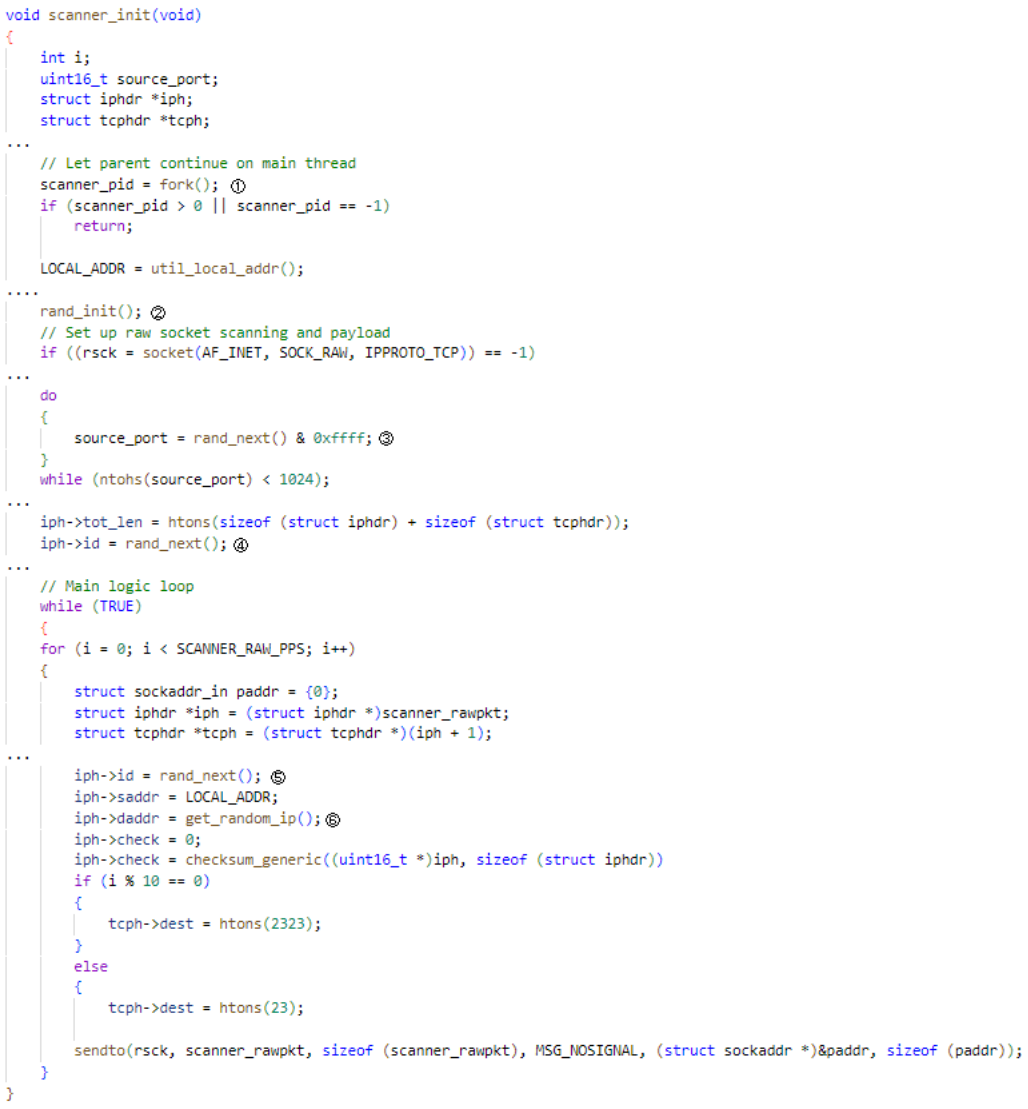
However, as characterised in the work by Griffioen et al. [29], there exist some oversights in the generation of the seed of the PRNG that make it susceptible to brute force. This leads to a much lower entropy of the seed space than initially assumed. The implementation errors that make this possible are:

- The value x, which is determined by the epoch time at the time of seed generation has a theoretical maximum of $2^{31} - 1$. However, we can take advantage of the knowledge of average bot lifetimes to gain a much tighter bound on the possible values of x. This could range from 15 min as set in the original work and has been set as high as 2 weeks for experimental purposes.

- The value y, is determined by getting the XOR of the PID and the PPID, giving a theoretical bound of 2^{16} values. However this value is even lower in practise due to the nature of the PID allocation by several systems and is discussed later.

- The value z is determined by the internal clock at the time of seeding. This value records the amount of ticks since the parent process forks to create the scanning process to the current clock tick. As very few instructions execute since the fork, this value is also very low. Further, in older versions of glibc, the granularity of clock ticks is modulus 10,000, which makes the solution space for this value even lower.

These values then undergo a series of bit shifts and XOR operations, and are then shifted by one value to accomodate the newly generated value, which is also the output of that round as shown in



```
void scanner_init(void)
{
    int i;
    uint16_t source_port;
    struct iphdr *iph;
    struct tcphdr *tcph;
...
    // Let parent continue on main thread
    scanner_pid = fork(); ①
    if (scanner_pid > 0 || scanner_pid == -1)
        return;

    LOCAL_ADDR = util_local_addr();
....
    rand_init(); ②
    // Set up raw socket scanning and payload
    if ((rsck = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) == -1)
...
    do
    {
        source_port = rand_next() & 0xffff; ③
    }
    while (ntohs(source_port) < 1024);
...
    iph->tot_len = htons(sizeof (struct iphdr) + sizeof (struct tcphdr));
    iph->id = rand_next(); ④
...
    // Main logic loop
    while (TRUE)
    {
    for (i = 0; i < SCANNER_RAW_PPS; i++)
    {
        struct sockaddr_in paddr = {0};
        struct iphdr *iph = (struct iphdr *)scanner_rawpkt;
        struct tcphdr *tcph = (struct tcphdr *)(iph + 1);
...
        iph->id = rand_next(); ⑤
        iph->saddr = LOCAL_ADDR;
        iph->daddr = get_random_ip(); ⑥
        iph->check = 0;
        iph->check = checksum_generic((uint16_t *)iph, sizeof (struct iphdr))
        if (i % 10 == 0)
        {
            tcph->dest = htons(2323);
        }
        else
        {
            tcph->dest = htons(23);

        sendto(rsck, scanner_rawpkt, sizeof (scanner_rawpkt), MSG_NOSIGNAL, (struct sockaddr *)&paddr, sizeof (paddr));
    }
}
```

Figure 5.6: Mirai scanning packet PRNG set values [38]

Figure 5.7.

Due to these oversights in implementation the LFSR has a much lower entropy than expected, which makes it much easier to brute force the seed. Furthermore the operations in an LFSR are simple bit shift and XOR operations which are extremely inexpensive to perform, leading to a very fast implementation of the brute-forcing software.

As a large number of values is generated, it becomes instrumental to ensure that collisions are minimised, and in case of collisions, we should be able to verify whether the obtained solution is infact the correct solution or a spurious result.

5.3. Implementation

As established in the previous section, the method used to generate seeds for the PRNG that is used by Mirai has much lower entropy than its theoretical value. This section details the workflow utilised to brute force seeds from packets that are observed in the network telescope and methods used to further enrich the information.

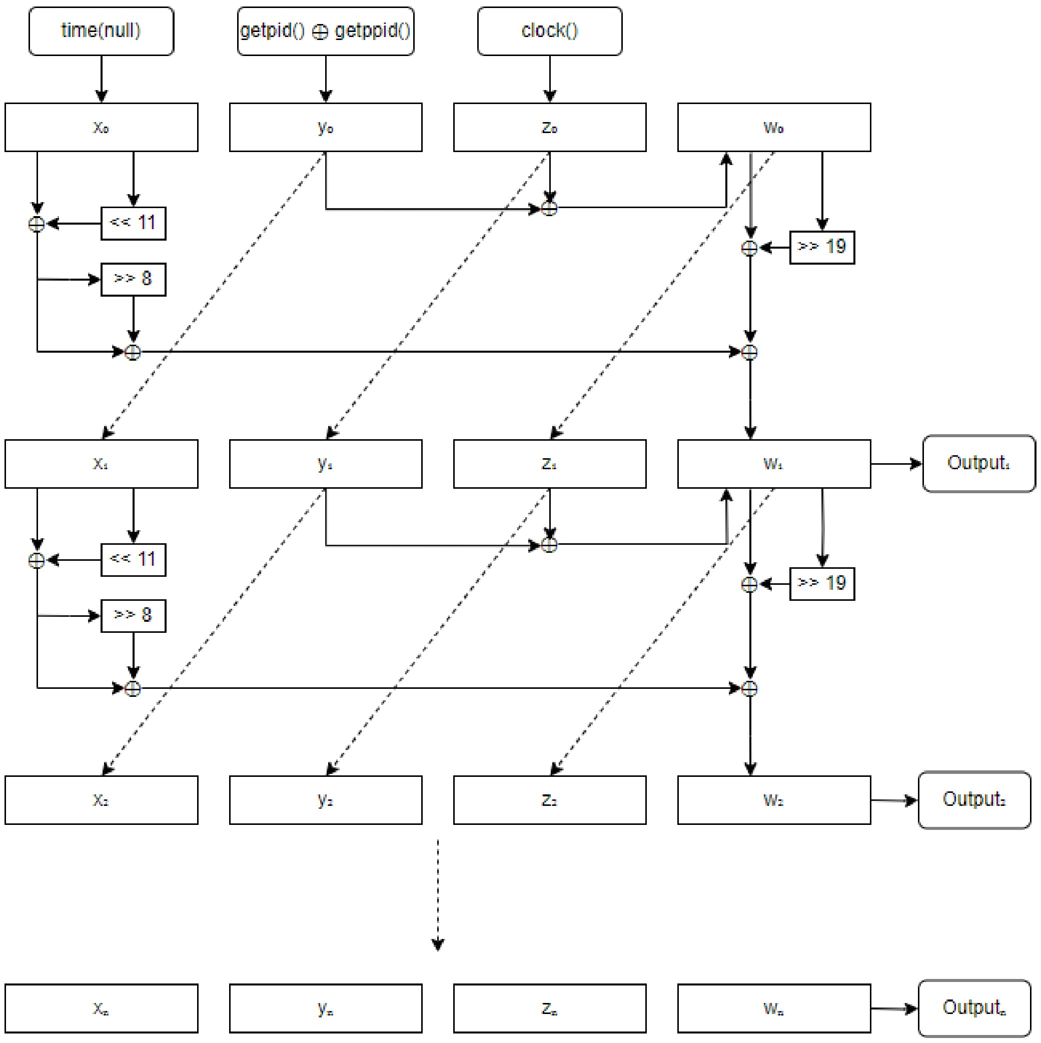
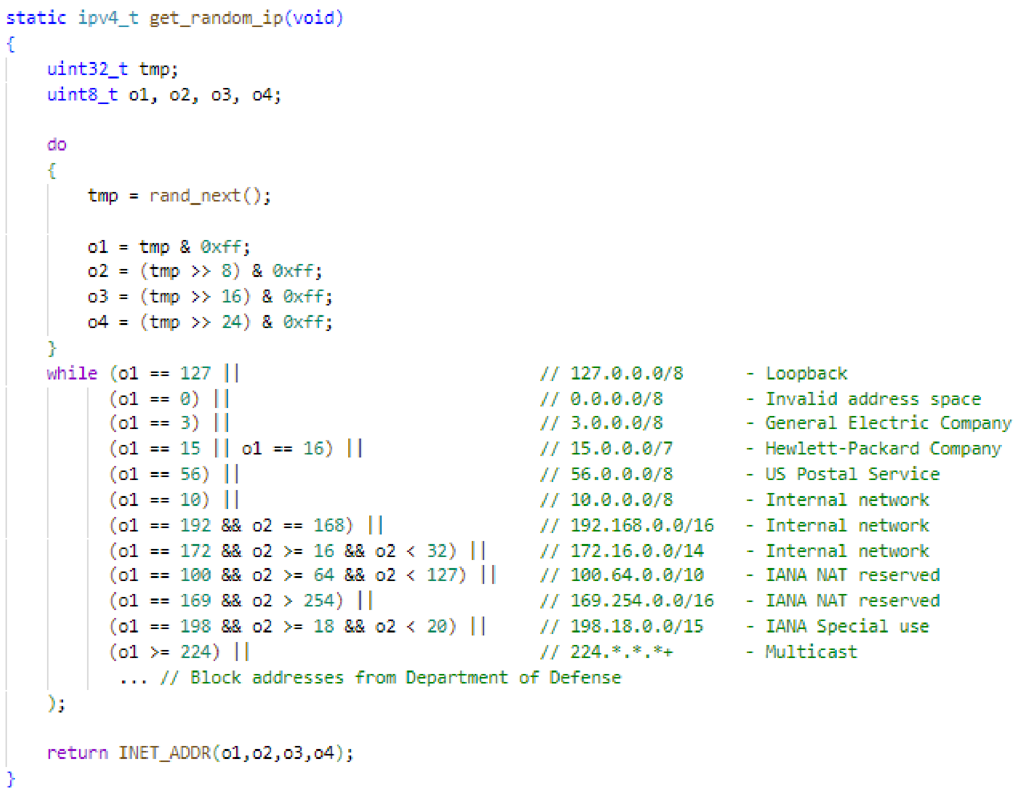


Figure 5.7: Mirai LFSR design

The implementation used to brute force the seed for incoming packets and to get further context on the state of the bot at the time of sending the packet is as detailed below:

1. We filter out packets that have the Mirai signature from a particular period of time, 6 hours in this instance, and store them for further use.
2. The brute force algorithm then runs on each individual packet. Based on the Timestamp of when the packet is received, a range of time is picked before then. These values range from 15 minutes before the packet arrives, all the way upto two weeks before the packet is seen at the telescope.
3. For each packet 2^{16} possible values of the PID ⊕ PPID are iterated over.
4. Due to the granular nature of the clock() function, there are only two possible values, when brute-forcing the seed, only 0 and 10000 are checked.
5. The combination of all these values is then considered to be the starting seed of the LFSR. Once the intial seed is set, a large series of the outputs of the LFSR is generated. To ensure that possible solutions are not missed a sufficiently large value has to be set. However, when considering the scale of operations required and the amount of traffic that is taken, a very large value would result in exponentially increasing computation times. Thus a tradeoff between maximising solutions and runtime has to be reached. For most of the bruteforce operations, 2^{25} sequential values are generated by the LFSR.



```c
static ipv4_t get_random_ip(void)
{
    uint32_t tmp;
    uint8_t o1, o2, o3, o4;

    do
    {
        tmp = rand_next();

        o1 = tmp & 0xff;
        o2 = (tmp >> 8) & 0xff;
        o3 = (tmp >> 16) & 0xff;
        o4 = (tmp >> 24) & 0xff;
    }
    while (o1 == 127 ||                             // 127.0.0.0/8      - Loopback
          (o1 == 0) ||                              // 0.0.0.0/8        - Invalid address space
          (o1 == 3) ||                              // 3.0.0.0/8        - General Electric Company
          (o1 == 15 || o1 == 16) ||                 // 15.0.0.0/7       - Hewlett-Packard Company
          (o1 == 56) ||                             // 56.0.0.0/8       - US Postal Service
          (o1 == 10) ||                             // 10.0.0.0/8       - Internal network
          (o1 == 192 && o2 == 168) ||               // 192.168.0.0/16   - Internal network
          (o1 == 172 && o2 >= 16 && o2 < 32) ||     // 172.16.0.0/14    - Internal network
          (o1 == 100 && o2 >= 64 && o2 < 127) ||    // 100.64.0.0/10    - IANA NAT reserved
          (o1 == 169 && o2 > 254) ||                // 169.254.0.0/16   - IANA NAT reserved
          (o1 == 198 && o2 >= 18 && o2 < 20) ||     // 198.18.0.0/15    - IANA Special use
          (o1 >= 224) ||                            // 224.*.*.*+       - Multicast
           ... // Block addresses from Department of Defense
    );

    return INET_ADDR(o1,o2,o3,o4);
}
```

Figure 5.8: Mirai get_rand_ip function [38]

6. These values are then compared with the IPId and Destination IP values extracted from the logged packet. The LFSR in Mirai does not perform any transformations on the byte order of the generated values when setting them as the values for the packet fields before sending them. So we need to ensure that there is coherence between the Byte order used when matching. So we convert the values extracted from the packets from Host byte order to Network Byte order.

7. These two values are then compared with the generated string of values for each possible seed.

8. On getting a match the value is stored in a file for verification later, as there is still a high chance for collisions considering that there is 24 bits of entropy and we generate 2^{25} values for each seed.

9. In order to verify whether a seed is legitimate and not a collision, we generate a whole string of values from the seed and compare the fields to all incoming packets over a period of time since the startup time value found in the seed.

10. Finding several packets verifies that the seed was authentic and not a collision as the chances of matching several packets to a collision is extremely low. After doing so, we have a final set of known authentic seeds and packets that where generated by the seed that hit our telescope.

5.3.1. Improving Performance

Despite having success with the previous method, the time taken to brute force seeds for all incoming packets over a long period of time is largely infeasible due to the vast amount of computation required. To gain an idea of the scale of the computations needed, we can look at how the time needed grows exponentially with the various parameters.

Filtering packets that have the Mirai signature over a period of 6 hours, yields around 1.3 million packets. Each packet is then checked for a period of 15 minutes, for 2^{16} PID values and for 2 possible

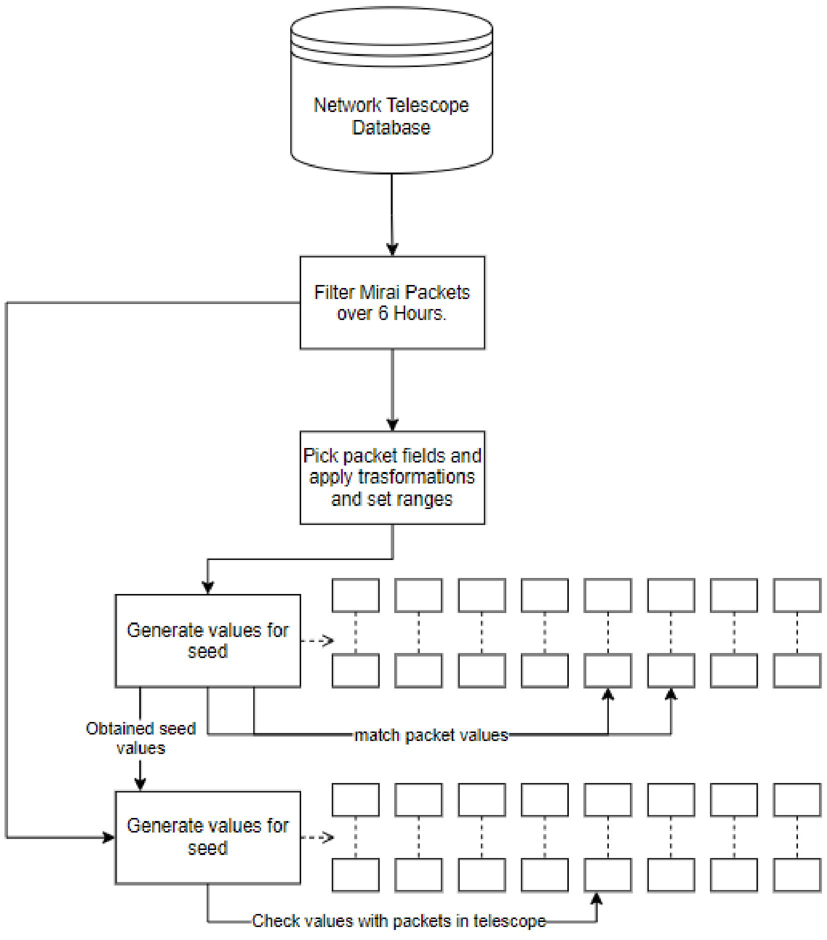

Network Telescope Database
Filter Mirai Packets over 6 Hours.
Pick packet fields and apply trasformations and set ranges
Generate values for seed
Obtained seed values
match packet values
Generate values for seed
Check values with packets in telescope

Figure 5.9: Workflow for bruteforcing seed and verification.

values of clock(). This would lead to a theoretical maximum number of calculations

$$\approx 10^6 * 900 * 2^{16} * 2 \approx 1.5 * 10^{14} \tag{5.1}$$

Although these calculations were performed on a powerful server with parallelisation, this still takes a large amount of time, even considering that we often use larger values for time in the past such as 2 weeks.
In order to make brute forcing the seed sustainable, certain changes were made to ensure that scaling with more packets, larger ranges of time or even for a larger amount of values generated by the LFSR for comparison. Some of these changes were considered after understanding certain occurences in the brute force process and others were performed after further inspection of the source code and understanding of the workings of the Mirai LFSR. The changes made are:

- It was observed that the PID values for most seed values are generally very low. This could be attributed to the fact that the parent and child process are spawned in very little time between each other. This would mean that the PID values allocated to both the processes would usually be very close to each other. The XOR of these close values would then usually be a small value as well. Taking this into consideration, we can reduce the range of possible values to check from 2^{16} to 2^7.

- As seen in Figure 5.1, the Source Port and Window are the first fields to be set. Having a check for these values has two benefits. Firstly, It would mean that a large portion of the entire solution space can be discarded upon checking the first few values instead of having to go through the whole range of generated values. Second, the probability of having a collision also reduces significantly as now the total number of bits that are compared are 80 compared to the previous 48 bits.

Taking these considerations into account greatly reduces the amount of computations required in general and makes it possible to analyse larger amounts of data in a more efficient manner.

5.4. Bruteforcing seeds from DDOS packets

Further looking at the Mirai code shows us the built in functionality of bots to perform Distributed Denial of Service or DDoS attacks. The code, contained in attack.c contains functionality to parse and setup necessary requirements in order to carry out an attack. There is support for several attack methods on the TCP protocol which consist of TCP syn , TCP ack and TCP stomp attacks. There are also attacks for other protocols such as UDP and GRE. We only focus on some TCP attacks as those are only ones that we can detect through the current setup on the network telescope.

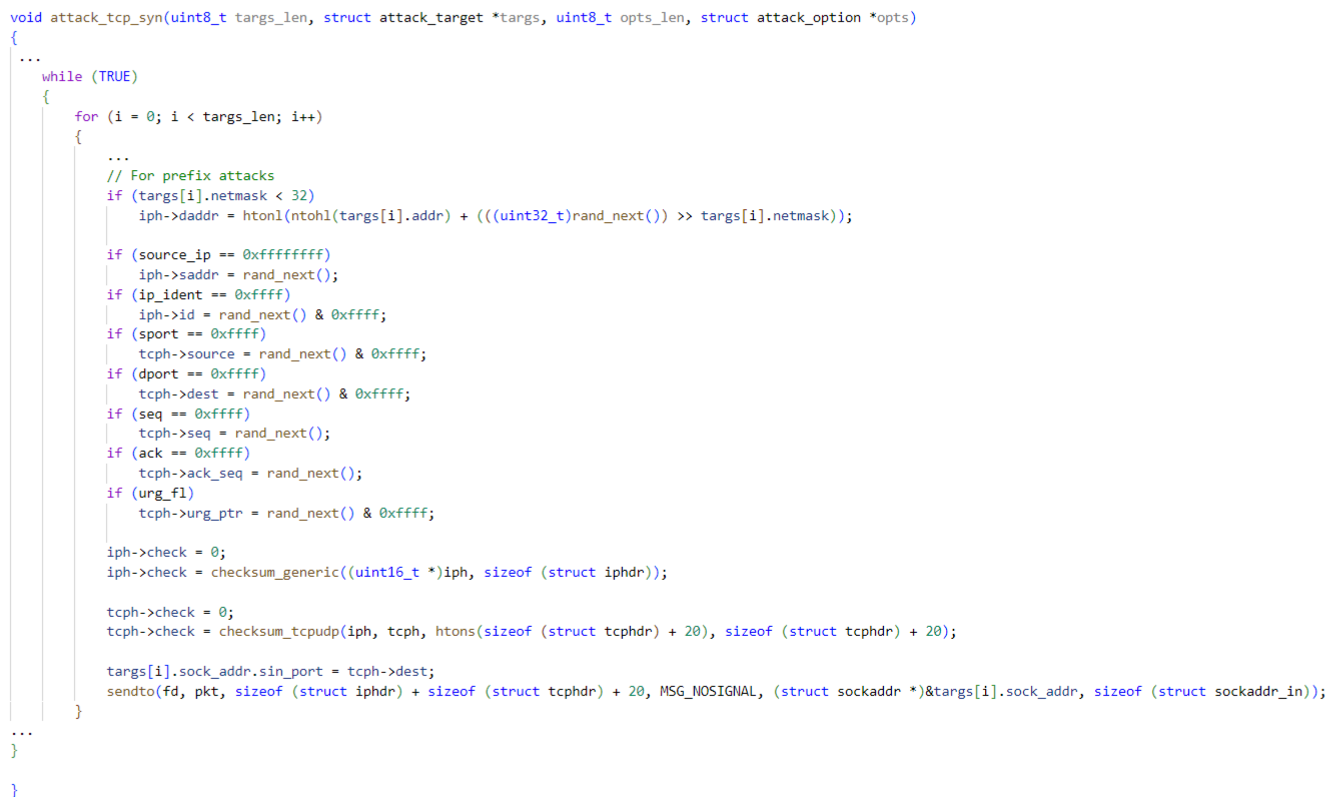
The TCP attack that we focus on is the TCP syn attack. The attack aims at abusing the mechanism of the 3-way handshake by sending SYN packets to the host, causing them to respond to the request and wait for the response on an allocated port, upon sending enough traffic the hosts available ports are continuously overwhelmed, preventing any legitimate traffic. Looking at the code in the vanilla Mirai botnet in figure 5.10, we see that there are a variety of fields set. Some of these are set randomly while others are set based on some input parameters.

The values that are set randomly use the output from the same rand_init() function, which leads us to believe that these values could also be used to bruteforce the seed used to set these values. However, we have to ensure that these values either do not undergo any transformation, or are transformed in a way that we account for when bruteforcing the value of the seed.

The transformations that we account for when comparing the values of the LFSR output and the packets we observe in the telescope are:

- **Destination IP:** The destination IP is the value that is generated as a spoofed Source IP by the attacker in order to hide the IP address of the bots participating in the attack. We receive the packet at our telescope when this generated value coincidentally falls within its range.
- **Source IP:** As we are looking at perceived backscatter traffic from a DDoS attack, the value of Source IP set in the packet we receive is actually the generated Destination IP value which is set by the PRNG. Once the packet is sent to the victim, the victim would respond by setting its IP address as the source IP. This field gives us information as to who the intended target of the attack was. This value is not matched to any field as it is not entirely randomly generated, apart from the last few bits to attack all addresses in a subnet.
- **Source Port:** Similar to the IP addresses the value of the destination port is now swapped to the value of the source port in the response by the victim, This also gives us information as to what port the attack was targeted on, and the nature of the response tells us whether an active service was on the port.
- **Destination Port:** In order to account for backscatter, we will have to match this value to the Source Port generated in the PRNG.
- **Ack:** As per the TCP handshake, we will receive a value in the ACK field which is the value of the SYN field in the first packet sent by the bot to the victim incremented by a value of 1. This would mean that we would have to match the value of the ACK field to that of the SYN value in the position generated by the PRNG after reducing it by 1, all while keeping in mind that we take into account the conversion of byte order to ensure coherence.

Seeing as these values added together provide around 48 bits of entropy, getting a match would not require us to take any further steps to validate the seed, as the chances of collision are fairly low. A similar setup is used as in the case of brute forcing the seeds of the scanning packets.



```c
void attack_tcp_syn(uint8_t targs_len, struct attack_target *targs, uint8_t opts_len, struct attack_option *opts)
{
 ...
    while (TRUE)
    {
        for (i = 0; i < targs_len; i++)
        {
            ...
            // For prefix attacks
            if (targs[i].netmask < 32)
                iph->daddr = htonl(ntohl(targs[i].addr) + (((uint32_t)rand_next()) >> targs[i].netmask));

            if (source_ip == 0xffffffff)
                iph->saddr = rand_next();
            if (ip_ident == 0xffff)
                iph->id = rand_next() & 0xffff;
            if (sport == 0xffff)
                tcph->source = rand_next() & 0xffff;
            if (dport == 0xffff)
                tcph->dest = rand_next() & 0xffff;
            if (seq == 0xffff)
                tcph->seq = rand_next();
            if (ack == 0xffff)
                tcph->ack_seq = rand_next();
            if (urg_fl)
                tcph->urg_ptr = rand_next() & 0xffff;

            iph->check = 0;
            iph->check = checksum_generic((uint16_t *)iph, sizeof (struct iphdr));

            tcph->check = 0;
            tcph->check = checksum_tcpudp(iph, tcph, htons(sizeof (struct tcphdr) + 20), sizeof (struct tcphdr) + 20);

            targs[i].sock_addr.sin_port = tcph->dest;
            sendto(fd, pkt, sizeof (struct iphdr) + sizeof (struct tcphdr) + 20, MSG_NOSIGNAL, (struct sockaddr *)&targs[i].sock_addr, sizeof (struct sockaddr_in));
        }
...
}

}
```

Figure 5.10: Mirai code for DDoS tcp syn attack. [38]

6 Results

6.1. Finding non-Mirai variants

As discussed in Section 5.1, We can leverage the nature of botnet infections to filter out packets sent from non-mirai variants with a relatively high degree of confidence. Doing so would allow us to understand the nature of infections by other variants beyond what we know currently about the Mirai botnet. There are several botnet variants that target more than one vulnerability. We can track these by looking at the different destination ports that are targeted during a ”session”. This also helps us to verify our results by making observing known exploit ports among the list of ports in the packets that we have filtered.

Although it is fairly easy to assume that all packets we receive have to be botnet scanning packets, as legitimate devices would never send packets to a telescope or a darknet, it is fairly important to ensure that we account for cases where this could actually happen. The packets we receive could have reached us for a variety of reasons:

- These packets could be the result of a misconfiguration which would mean a device is repeatedly trying to establish a connection, considering that Mirai and several of its variants target some of the most commonly used services on the internet such as Telnet, SSH, HTTP, RDP, etc, this overlap could easily provide us with a large number of false positives.
- These packets could be from a malicious adversary that is running a high speed scanner from a hosting service. In this case a specialised tool such as ZMAP is used instead of relying on the slower scanning speed of several bots. Although this is also a precursor to infection of devices, this would throw off the statistics when studying the activities and inner workings of a botnet.
- Some scans of popular services are routinely performed by services such as Shodan. This would mean that there is an overlap of scanning packets towards services of interest which would have to be addressed before performing any kind of study.
- There could also be scans on commonly targeted ports based on new CVE’s carried out by security companies or researchers which could also provide us with false positives.

In order to filter out packets that may be interesting to us based on the phenomenon we observed earlier, we have to set some constraints on the nature of the packet and its temporal proximity to known botnet scanning packets such as packets which have the Mirai signature $(IP.dstIP == TCP.seq)$. In this thesis, we take 4 such cases, each more restrictive than the last, in order to understand the quality of the results that we get. Moreover these cases also correlate to certain phenomenon that we notice in the telescope data through manual inspection. This also helps us to be able to isolate unique ”sessions”, a challenge that is much more difficult for variants, as they lack the distinct Source Port and Window combination as seen in Vanilla Mirai. If we take all the non-Mirai scanning traffic and aim to classify variants, being able to filter out scanning activity with a high degree of confidence, and also finding common patterns in botnet activity to isolate sessions, allows us to create much more meaningful datasets that can then be used to train much more accurate models for detecting botnet variants.

These cases are not exhaustive and there may be several more occurrences, however in the case of scanning traffic, there are more exceptions than rules, and it becomes fairly hard to account for all possible cases. However, the cases built here provide very helpful context when it comes to understanding botnet scanning, and could prove to be useful for later works.

The four cases that have been used to filter out prospective botnet scanning packets which do not contain a known Mirai signature are:

- **Filtering all packets from a source that has sent atleast one Mirai scanning packet as prospective scanning packets:** In this case we only check whether a known source IP has sent any Mirai scanning packets over the duration of a whole month. If so, then any other packets that we receive are considered to be scanning packets sent from a Mirai variant. This is an extremely loose constraint that could lead to a large amount of false positives, as there could be multiple devices that are associated with an IP over a range that long due to IP address churn, Moreover, if the device was not churned, it could also have restarted, thereby losing the infection.
- **Filtering packets from a source that are within 2 hours of a Mirai packet being sent:** In this case we further restrain the previous case, limiting the packets that we consider to have been sent in a two hour window before and after the last packet received with the Mirai signature. This would mean that we pick out Non-Mirai packets that exist in windows of length 4 hours where atleast one mirai packet was sent on each end. This would help prevent false positives from devices that might have churned to a different address, or recovered from an infection. When we consider the average interarrival times as seen in Figure 4.4. this is a very loose constraint still, but will allow for a good point of comparison, perhaps allowing us to capture cases in which a device reboots but then gets infected with a newer variant. In this case having a much tighter bound would miss out on these packets.
- **Filtering packets from a source that are within 2 hours of more than 5 Mirai packets being sent and the scanning packets are part of at most 5 sessions in an 8 hour window:** During manual inspection of traffic from several sources, there are cases where the amount of packets sent per session are much lower and more sporadic. As these sessions are extremely short lived, the confidence in the non-Mirai packets originating from a variant is much lower. Furthermore, we have cases such as packets originating from large ranges with one identifier. This can be seen in cases such as Figure 6.1, where a large amount of devices exist behind a single identifier. In this case it becomes trivial to accept incoming non-Mirai packets due to the large amount of incoming Mirai packets. Setting an upper limit on the amount of active sessions within the considered time window would filter out such occurrences.
- **Filtering packets from a source that are within 2 hours of more than 5 Mirai packets being sent and the those scanning packets are part of one session:** This case is set up to identify packets that might be a part of a concurrent infection or those that exist exclusively between two Mirai infection. This is the most constrained case and would, in theory yield packets with the highest confidence in their being a part of a malicious mirai variant. However, there are other known cases which this would exclude leading to loss of some amount of information.

6.1.1. Case 1

When considering all non-Mirai packets from Sources that have sent atleast one Mirai packet over network telescope data over the period of the month of June, we accumulate 17,000,000 packets. The most commonly targeted ports are well known services as seen in Table 6.1. This would mean that there are a lot of other scanning packets that are also included alongside the packets we need. Although there are mirai variants that do target several of these services, it becomes fairly hard to understand whether the packets originate from a heavy hitter or a botnet scan.

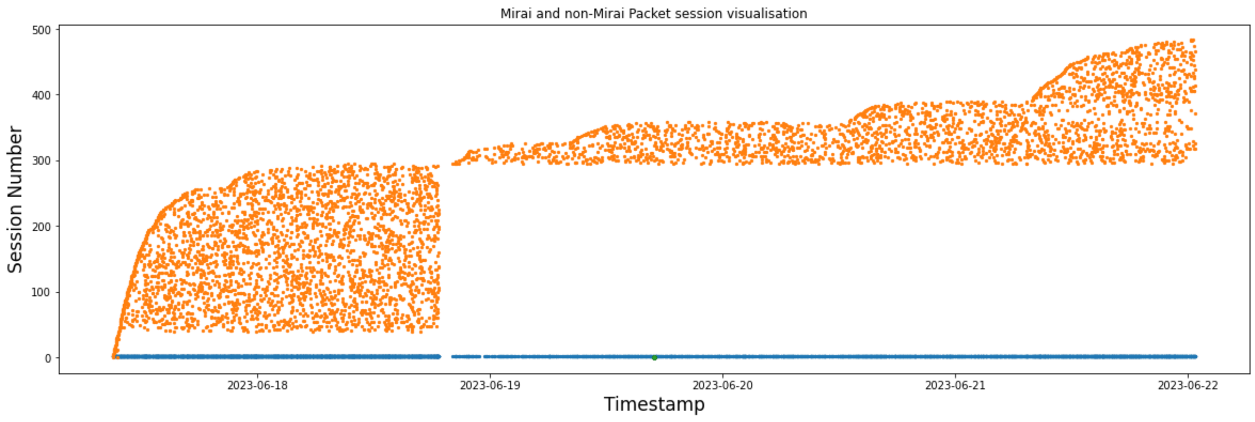


Figure 6.1: Mirai session visualisation of incoming packets from a large IP range

Also as seen in Figure 6.3, the regions marked in the black boxes are not close to any Mirai infections, this could be traffic from a botnet, but cannot be classified with as high a degree of confidence as the other marked packets using our hypothesis of reinfections.

Port	Percentage
80	19.18%
5555	16.85%
81	15.43%
8080	10.04%
22	9.86%
25565	7.38%
52869	6.1%
60001	2.09%

Table 6.1: Most frequently targeted destination ports in filtered non-Mirai packets

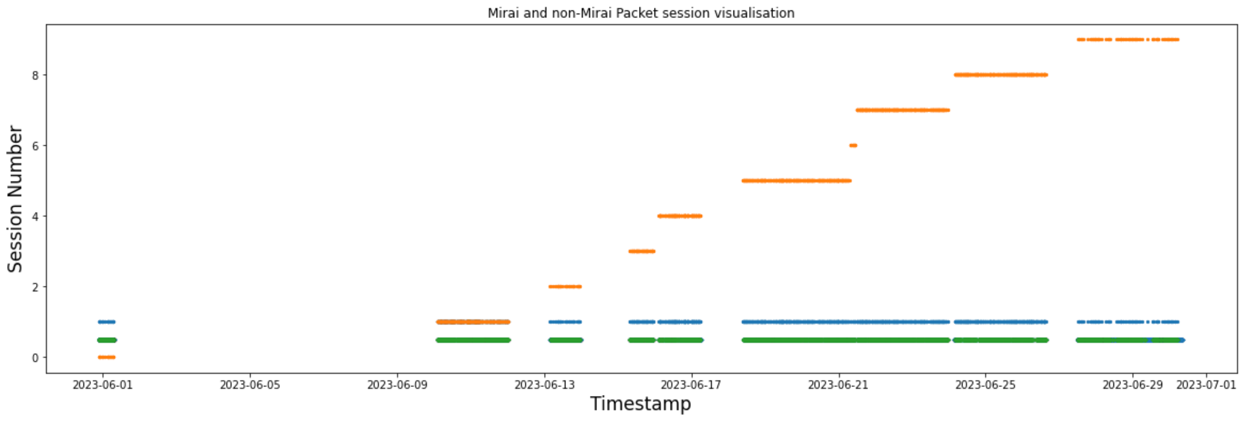


Figure 6.2: Mirai session visualisation of incoming packets with no constraints

6.1.2. Case 2

In order to prevent packets from being incorrectly picked as seen in subsection 6.1.1, we further narrow the constraints, to include packets that would only be considered if there were Mirai packets present in a 2 hour window before and after the observed packet. Looking at Figure 6.4 we see that the earlier packets that were included in 6.3 are now not included, giving us the packets that we want based on the hypothesis.

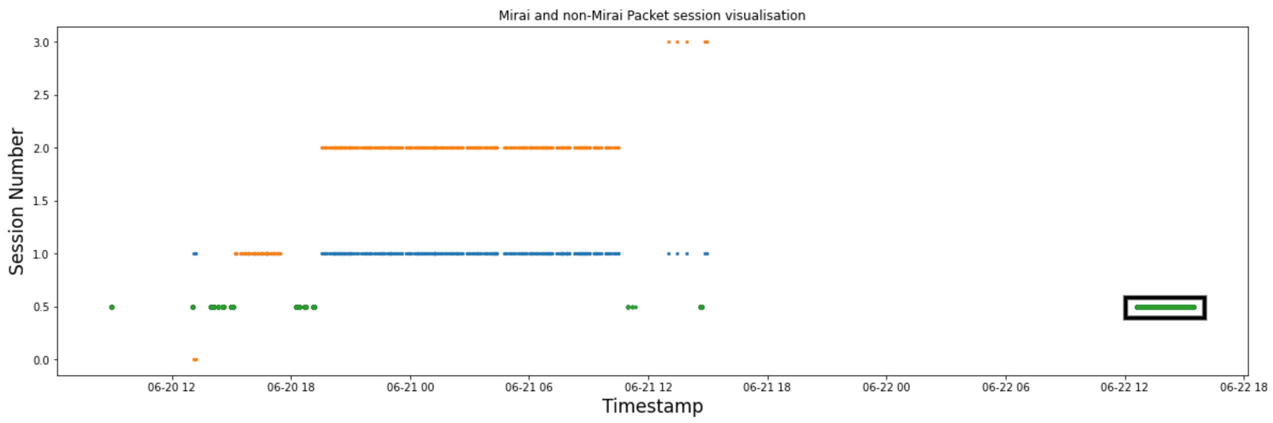


Figure 6.3: Mirai session visualisation of incoming packets with no constraints with unintended behaviour

In some cases this constraint might not be enough, especially when there seem to be several infections behind one IP address as in the case seen in Figure 6.1, upon investigating, this address is seen to be a national backbone with a /12 range. In this case there are several mirai infections and legitimate devices that exist behind one identifier. Due to the large amount of traffic that comes from this address, any non-Mirai packet would be marked as per our criteria. In order to avoid these cases, we have to further constrain our conditions.

Looking at the most common ports, we see that the ports with known vulnerabilities now have a much larger share compared to the popular services, which would imply that some of the scanning traffic from hosting services has been suppressed. An interesting insight is the appearance of port 10911, which had not been observed this prevalently in the earlier case. Further investigation shows that an RCE vulnerability in the RocketMQ service [42] was recently discovered in 2023, and several botnets such as the ”Dreambus” botnet have started to actively scan and infect devices with this vulnerability [43]. As we were not aware of this port being used for attacks, it proves a characterisation of the intended use case of highlighting ports of interest that could then be investigated to understand the newer vulnerabilities that botnets are targeting.

Port	Percentage
5555	23.25%
52869	20.05%
80	18.12%
81	15.49%
8080	9.26%
22222	2.79%
60001	2.74%
10911	1.83%

Table 6.2: Most frequently targeted destination ports in filtered non-Mirai packets sending atleast 5 packets in a 2 hour window

6.1.3. Case 3

We notice a lot of cases where there are IP addresses that have several devices behind one identifier. In this case all incoming packets are accepted, giving us results similar to Case 1 rather than that of Case 2. In order to address this, we put a cap on the amount of sessions in a sliding window. This would mean that most packets that are accepted are truly originating from a unique device. As seen in Table 6.3 there is a much higher prevalence of packets with destination port 37215 appearing among the filtered packets.

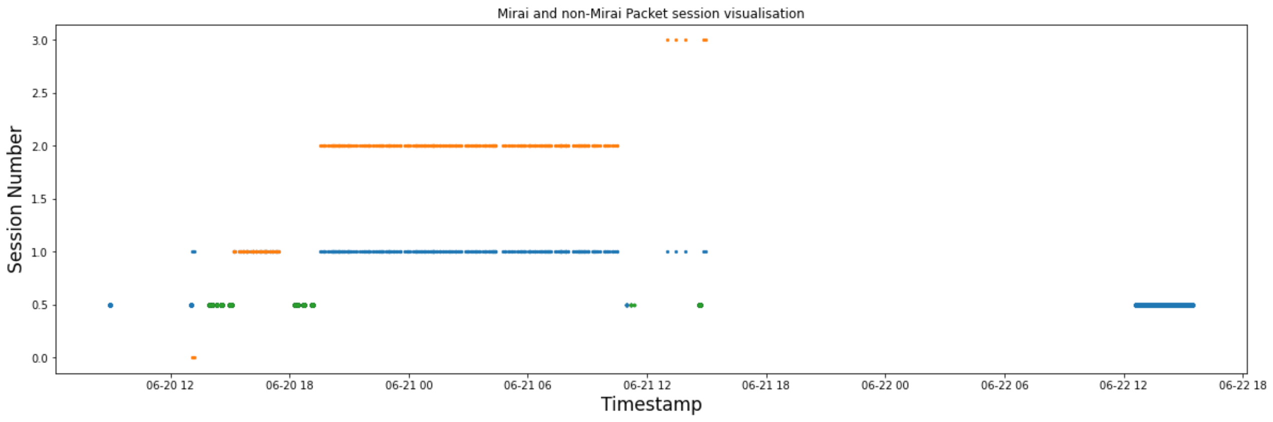


Figure 6.4: Mirai session visualisation of incoming packets marked within 2 hour window

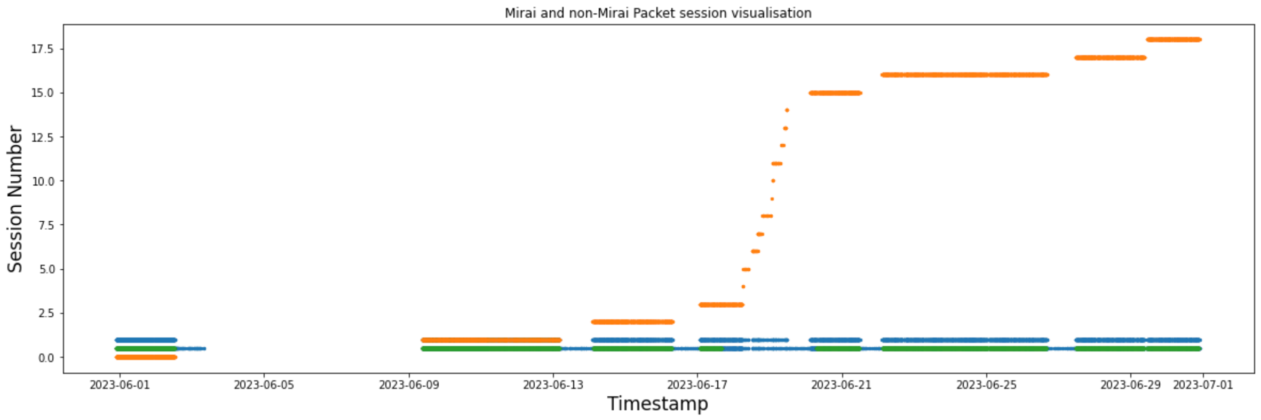


Figure 6.5: Mirai session visualisation of incoming packets marked within 2 hour window with a maximum of 5 sessions

Port	Percentage
5555	36.53%
52869	22.08%
81	13.68%
8080	12.31%
80	9.95%
22	1.54%
37215	0.84%
2222	0.48%
8443	0.45%

Table 6.3: Most frequently targeted destination ports in filtered non-Mirai packets sending atleast 5 packets in a 2 hour window with a maximum of 5 sessions

6.1.4. Case 4

We aim at looking if we can draw focus to particular events, in this case we only consider traffic from long lasting sessions and those where one infection ends and gives rise to a new session. Looking at Figure 6.6 we see that the portions where there is a quick switch in infections is ignored. When using this filter, this could mean that certain cases isolate certain types of botnets based on their behaviour. Looking at Figure 6.6 shows that the marked packets stop much earlier around the timestamp of the 18th of June than in Figure 6.5 due to the tighter bound on allowed number of sessions.

As seen from these cases, we can filter out traffic from mirai variants, giving us a more concise dataset that can be used for further analysis, and as a data source for training models to detect these

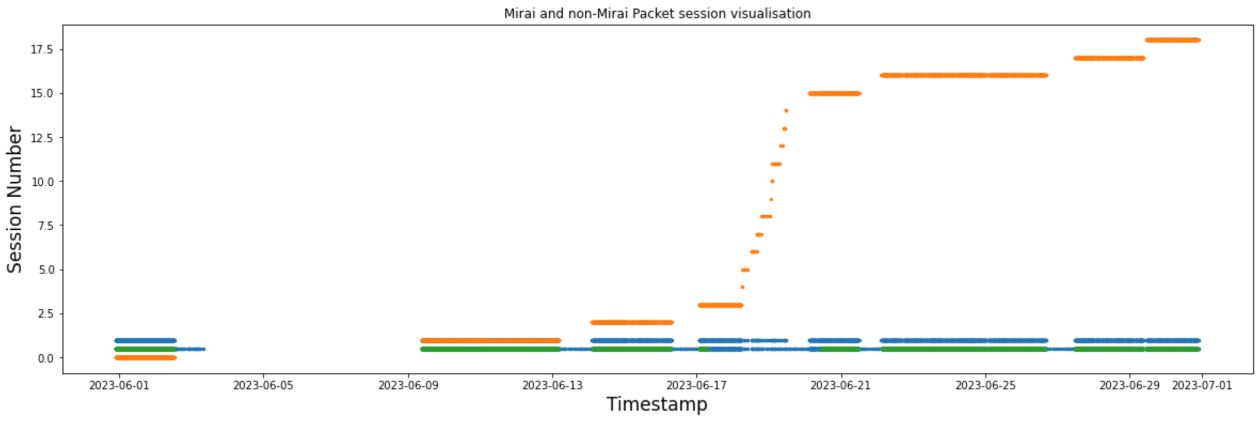


Figure 6.6: Mirai session visualisation of incoming packets marked within 2 hour window with a maximum of 2 sessions

variants. More cases could be created to find different variants and behaviours. Case 4 represents a very conservative case with the highest restrictions which should, in theory yield the packets with the highest certainty, however this is difficult to verify without other data sources and will be discussed in Future Works.

6.2. Bruteforce PRNG seeds

In order to understand our capabilities of brute forcing seeds from incoming Mirai packets, we need to be able to understand the rates at which we can extract seeds from the incoming scanning packets. Doing so provides us an understanding of how many packets with the Mirai signature have completely unmodified code, seeing as the open-source code can make it fairly easy to modularly tweak portions of the botnet, including the parameters of the LFSR that is used in the generation of random values.

While brute-forcing the seeds for these incoming packets, we also enrich the data by including the internal counter of the PRNG when the values matching the packet fields of the incoming packet were generated. This extra information allows us to understand the progression of the internal ”state” of the infection, an internal ”clock” of the activities of the malware as such. Noting this value relaxes our reliance on the time that we receive the packet, as that can be subject to a number of extraneous factors beyond our control.

Looking at iterations alongside the other incoming details in a Network Telescope reduces the dependance on the size of the telescope. There is no longer a requirement for a large amount of incoming packets to reliably judge sessions, rather just a couple of packets could be used to ideally characterise the activity of a scan, including details such as the start of the infection time and also the scanning rate.

6.2.1. Seed Yield

We start by taking the first packet that has been sent by a Source IP address to our network telescope over the course of a 6 hour period. We do so as the period we pick is relatively short and reduces the chance of an infection switching to a different IP address. When we pick the first packets from the IP’s, we pick a total of 37,817 packets out of a total of approximately 1.4 million packets. We pick the first packet seen so that we have to perform fewer iterations to brute force the seed, as subsequent packets would have value that were generated at a later time, furthermore, brute-forcing the first packet also provides us a more efficient way to match the subsequent packets to the seed as discussed in a later section.

We look to brute force the seeds, setting the range of the seed value of time() to a period of the last 30 days. Upon doing so, we manage to get 4200 seeds from the total of ≈38,000 packets. This provides us with a yield of 11.05%. We expect to miss a few seed values when bruteforced due to the presence of certain conditions in the Mirai source code as seen in Figure 5.9 where if the random IP generated is regenerated in case it falls into one of the restricted ranges, in this case the values of the field of ipid and Destination IP are not sequential.

We would also lose some values when the randomly generated value of the source port would land in the restricted range i.e between $0 - 1024$ out of a possible range of 65535 values, as we assume that the Source Port and Window values are the 3rd and 4th generated value respectively. However looking at the number of values in the restricted range compared to the total amount of values, the amount is negligible and definitely would not contribute to such a large discrepancy between the number of packets and the generated seeds.

A possible reason for this could be due to traffic from seeds that persist a longer period, that was beyond our current range, despite the long search window we have. We then refine the dataset to include only fresh sessions, by only counting those incoming packets from sources that havent sent us any Mirai traffic in the last 6 hours from the window that we take the first packets from. This would eliminate most of the sources that were active before we check for seeds as these sources usually send scanning packets in fairly quick succession, giving us mostly fresh infections. This serves the purpose of providing us a view of the iterations from the start of the infection. When filtering for new infections, the yield of seeds rose to approximately 2,200 out of 12,000 fresh packets giving us a yield of around 18.2%.

This is still a much lesser value than expected. The values generated were validated by checking subsequent packets in the Network Telescope with the corresponding seed value, considering the entropy value of the fields to be matched, the chances of a collision are sufficiently low enough and these are reduced even further with the amount of packets matched, this makes the chances of this being an implementation error extremely low. As such it is fair to conclude that the low yield is not due to a fault in the brute forcing algorithm. Another possible hypothesis could be that as the parameters of the PRNG are fairly easy to change, there could be actors that modify the amount of bit shifts in the steps of the LFSR or some other changes that are fairly easy to implement. This would mean that these packets are not bruteforceable by our current implementation.

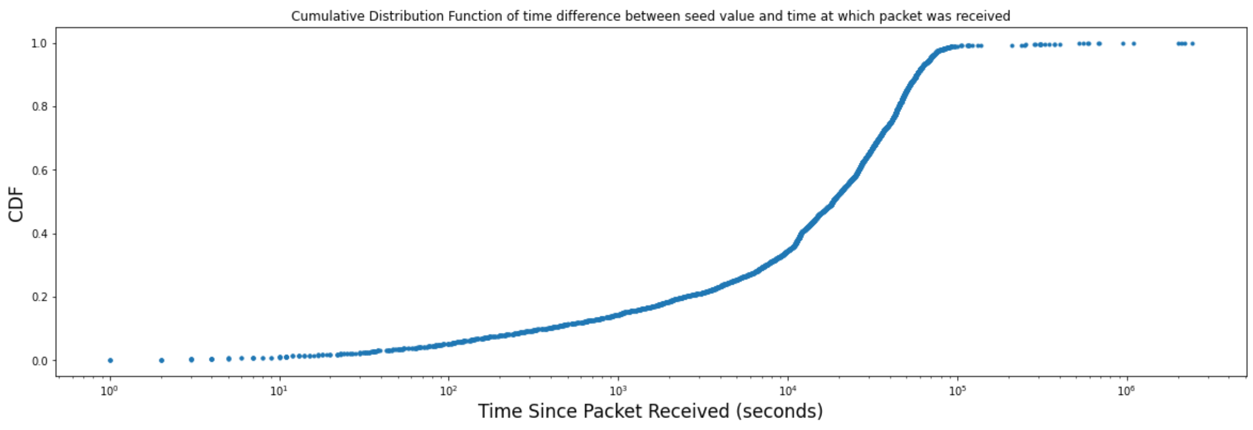


Figure 6.7: CDF of time difference between infection start and packet receipt

When we look at the seeds that we do manage to brute force, we can see that most of the packets usually show up in the telescope very rapidly, with around 40% of packets arriving in the first 2 hours, and almost all the packets arrive within the first day. It is important to realise that these packets might not be the first packet received from the infected host, however, the low frequency of values beyond a day gives us an important insight into the average lifetime of these bots. It is important to understand whether these values saturate due to mechanisms in the botnet that update the existing malware to newer versions, whether a competing infection took over the device and killed the existing infection or if there are two concurrent infections at the same time, or if there was a power/connection outage or if the device was manually reset by the owner. These hypotheses can only be verified by other sources of information such as honeypots, sandboxes or C2 milkers.

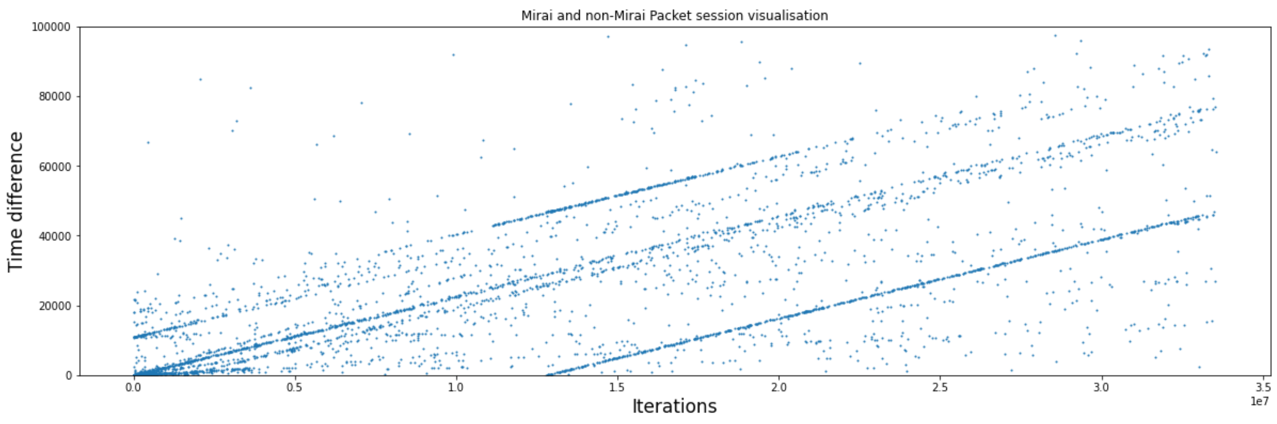


Figure 6.8: Scatter plot of iterations of the PRNG vs the time difference between infection start and packet receipt.

6.2.2. Linear relations in botnet clusters.

We have another interesting observation, when plotting a scatter plot of the number of iterations of the PRNG when the packet fields match the incoming packet. We can see a large number of diagonals that show a linear relationship between the lifetime of the bot and the number of iterations. The linear relationship matches our expectations from analysing the code as the bot is programmed to randomly spew out scanning packets, so it would generally do so in a uniform manner. However we also see that several of these lines are transposed and also have different slope values, suggesting a change in either the internal state of the packets at the start of the infection or the rate at which the bots themselves scan. Figure 6.8 shows us the linear relationship between the first packets found over all the source IPs. These lines are not indicative of the nature of the individual sources themselves, rather they show a trend over a large amount of infections. These upon further analysis could hint at different device type and thus the targeted vulnerabilities.

In order to identify whether the nature of the bots can be determined by the scanning rate shown by the slope, we visualise the distribution of slopes, signifying the rate at which the device scans the internet. Looking at the slopes shows us that the distribution is not random, instead it has peaks, one of which contributes to the majority of observed values while there are other peaks as seen in Figure 6.9. This could be indicative of the nature of the devices themselves, or the region that they are scanning from which could affect the rate of scanning either through the device speed or that of the network connection.

Once again, these insights do not allow us to make conclusive statements from these observations alone, rather they point us in the direction that more targeted methods of information gathering such as banner grabs from services such as Shodan would allow us to correlate with. In this case, getting live device information for an IP address for which a seed has been bruteforced and a scanning rate has been identified could lead to a fingerprint in acitivity that could then be established from scanning activity alone. This could mean that future Telescope data could provide a hint as to the nature of devices that are performing the scans once some correlation has been identified.

Having information on the device type along with the ports scanned could provide an extremely informative combination which could help determine the exact service and vulnerability being targeted.

6.2.3. Scanning nature of individual bots.

When we look at the iterations plotted over time for an individual IP address, we also see a similar linear relation between the iterations and the time passed. In fact when brute forcing seeds over a large period of 3 days, we notice that this behaviour is persistent over infections. In Figure 6.10, a device is

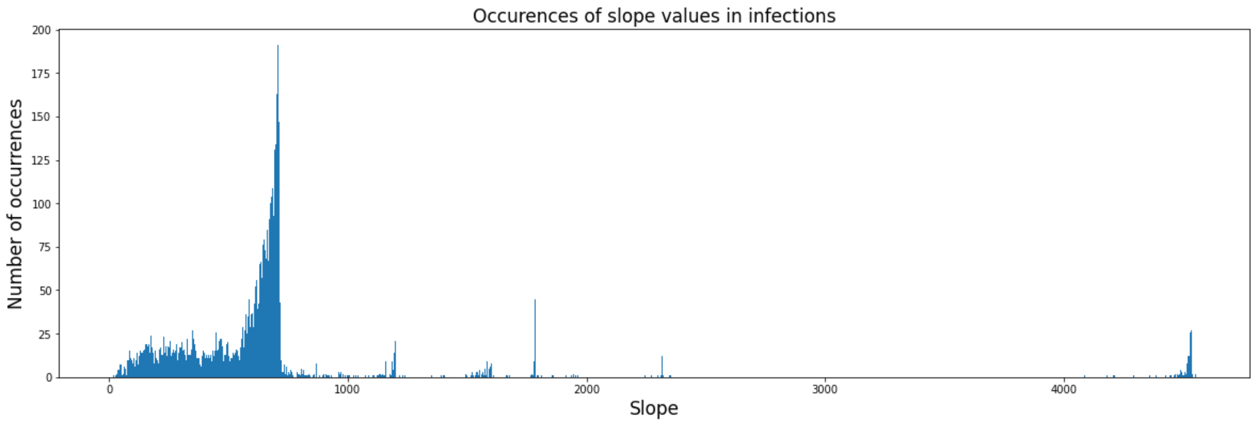


Figure 6.9: Visualisation of slope of iterations vs. time for each unique infection

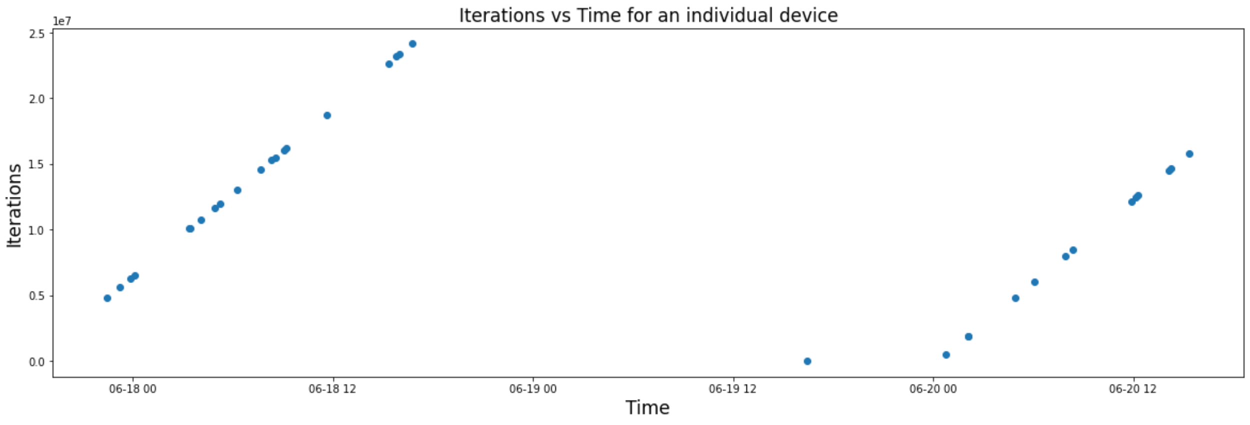


Figure 6.10: Activity of an individual device (IP address)

shown to have several infections over a period of time. We can observe when an infection stops and a new infection starts by following the iteration count. In the example shown there were 3 distinct seed values which correspond to our assumption through visual inspection.

When looking at each IP over a period of 3 days, we see that there are on average 14.7 infections per IP address, but the majority of infections are much lesser as seen in Figure 6.14. We also look at the percentage of time that an active infected device sends packets, we consider the time range for this to be from the first infected packet received to the last infected packet received. This measure would help to understand the amount of time that a vulnerable device is actively sending botnet traffic, especially providing insight on downtime between infections. In order to do so, seed values with a low enough iteration count are picked, signifying fresh infections, then all packets from then on which were succesfully bruteforced are included, this could include subsequent infections as well. If a packet is sent in a 5 minute sliding window which moves from the first packet to the last, then the 5 minute period is marked as active scanning, otherwise it is marked as inactive. The value for the size of the sliding window was picked heuristically by looking at botnet interarrival times and assuming a period by when a source would usually send another packet if it was still active, and can be changed, however this still provides an important understanding of botnet sessions. As seen in Figure 6.11, there are more hosts that are relatively less active, sending scanning packets for around 10-30% of the total time, but a large amount of IP addresses send packets for the entire duration, this could be due to the presence of more than one infection either on the device itself or due to several devices behind one IP address, thereby skewing our metrics. An example of such a case can be seen in Figure 6.12, there were 14 active infections, not necessarily concurrent over the course of the observations.

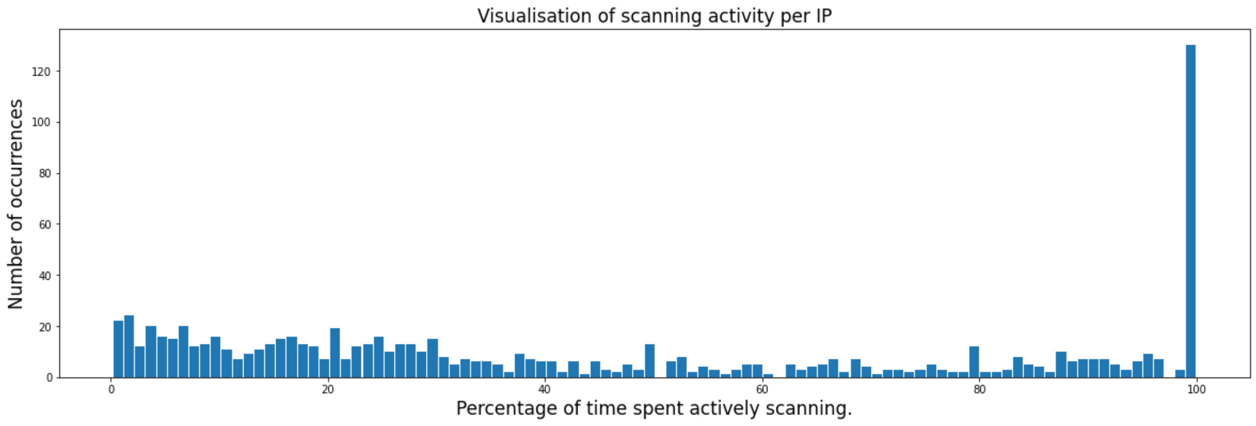


Figure 6.11: Visualisation of percentage a SrcIP is actively sending scanning packets.

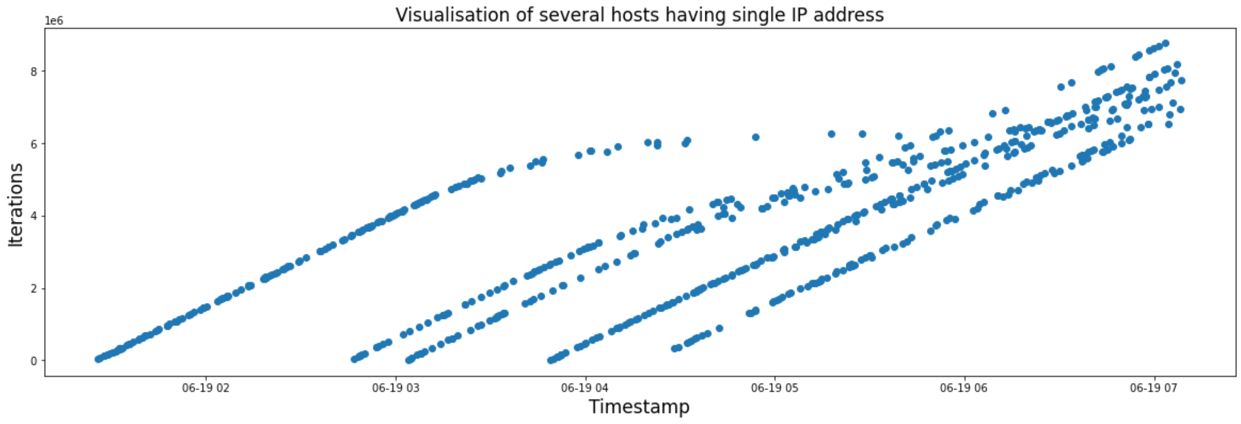


Figure 6.12: Example of several infections from a single IP-address.

In their work Richter et al. [44] discuss the frequency of hosts that perform partial internet as opposed to total internet scans, meaning that the scanners consciously choose to scan targeted regions as opposed to the entire internet address, as is the case with vanilla mirai infections. In order to do so they correlate data gathered from a total of 89,000 hosts in partnership with a CDN. They then perform an analysis of the distribution of packets sent and received on Source and Destination IP addresses, sorted into bins of /8 addresses. A high correlation would mean that this is a random internet wide scan, while a lower correlation value would mean that the scans are targeted in some way. In order to perform such an analysis, a vast amount of IP addresses spread across a variety of ranges is required. In this thesis, we posit that it is possible to try to understand this behaviour, atleast in the case of mirai variants that might have changed the restricted ranges to target particular address spaces, or locations. In order to check whether this is the case we try to visualise the difference in number of iterations between subsequent packets in Figure 6.13, as opposed to checking the rate which involved calculating the number of iterations with regard to time. This measure could be less concrete, as packets can be dropped or received out of order which could skew the statistics, but we considered the trends over a large amount of incoming packets to show clear differences which could atleast hint at such a phenomenon in case clear groups were observed. As seen in the Figure there is one large peak as opposed to the posited unique peaks which would mean that most Vanilla Mirai scans do indeed perform internet wide sweeps instead of more localised scans.

6.2.4. Understanding botnet scanning activity from start of infection.

Filtering on fresh IPs and looking at the iteration values of incoming packets shows a peak at the start and then yields a long tail as seen in Figure 6.15. The majority of the values lie within the $0 - 10^6$

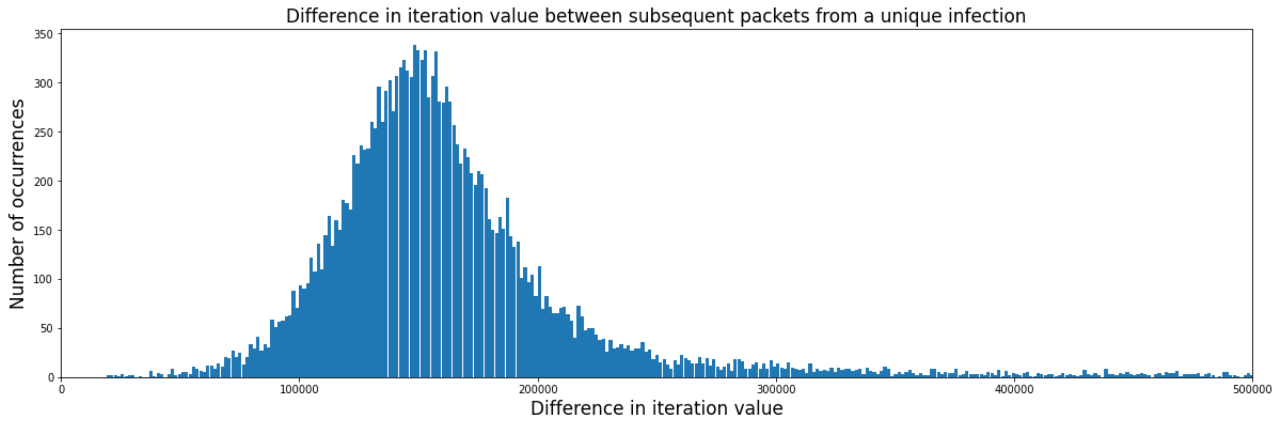


Figure 6.13: Difference in iteration values of subsequent packets from a unique infection.

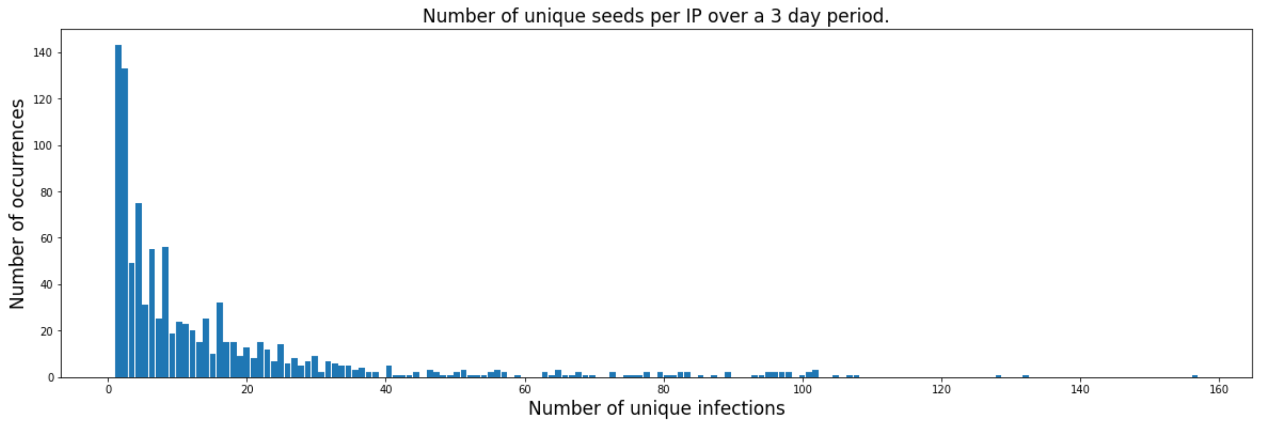


Figure 6.14: Number of unique infections from an IP address not accounting for churn.

range, this gives us an idea of the number of iterations before a packet reaches the network telescope. The long tail could be due to the fact that values are filtered on SrcIP and could be freshly churned addresses, however, due to the short Window that the devices are discounted, the effects due to churn are largely mitigated. This is also confirmed by looking at the difference in time between the first packet received on the telescope and the time set during the start of the infection. Most of the values are concentrated towards the head of the distribution, Looking at the zoomed in graph in Figure 6.16 shows that most packets are received in the first hour with a fair amount arriving in the 2-5 hour range as well.

This information could prove to be useful when conducting a study on the visibility of various telescopes. With the advent of new concepts for Network Telescopes such as Meta-Telescopes as mentioned in the work by Wagner et. al [45], Using the background Mirai scanning activity as a constant to compare the efficacy of telescopes used in the various studies could provide an additional layer of understanding about the nature of the telescopes. Moreover comparisons of the data logged over a larger spread of Address Spaces or ASes would also provide more data about localised variants of botnets or botnets that target vulnerabilities that are more prevalent in certain geographical locations by modifying the list of allowed and disallowed ranges as seen in the source code in Figure 5.9.

6.3. Brute forcing non mirai signature

When attempting to brute force non-Mirai signatures, there are a few issues, firstly it is fairly hard to match the exact same parameters as was done in the case of vanilla Mirai, due to much larger heterogeneity in Mirai variants, finding methods to capture most of them without static analysis, is very hard. If we were to relax the constraints, we see that there is a much larger amount of false positives. Owing

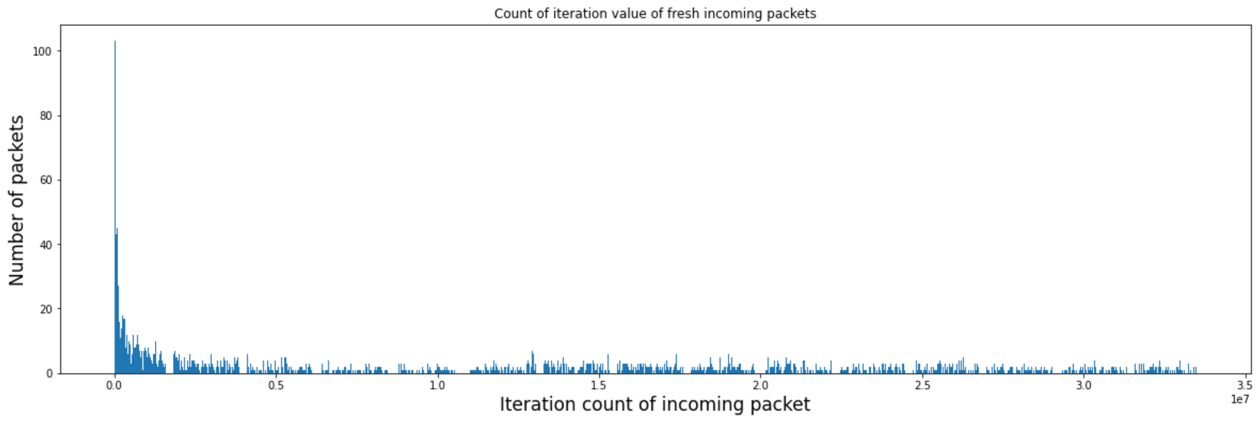


Figure 6.15: Histogram of iteration value of fresh incoming packets.

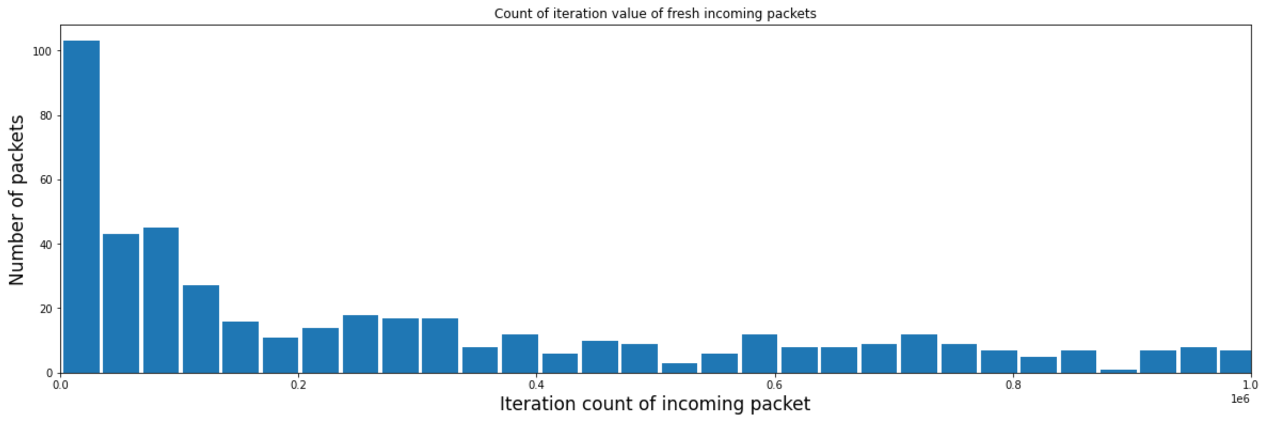


Figure 6.16: Histogram of iteration value of fresh incoming packets focusing on the lower values.

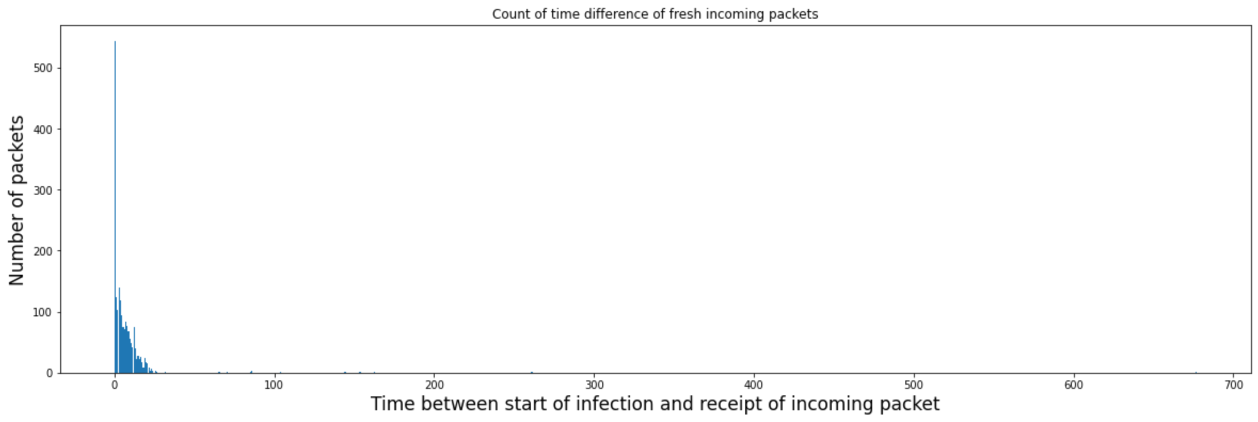


Figure 6.17: Histogram of time difference between receipt and seed time of fresh incoming packets.

to the much larger amount of scanning packets, verification of the false positives is significantly more resource intensive than in the case of vanilla Mirai.

As such, the way to progress with this issue is to rely on static analysis to build profiles of common botnets, as there is evidence of reuse of a lot of code components across variants, using these similarities to filter out other variants is an interesting topic to be explored.

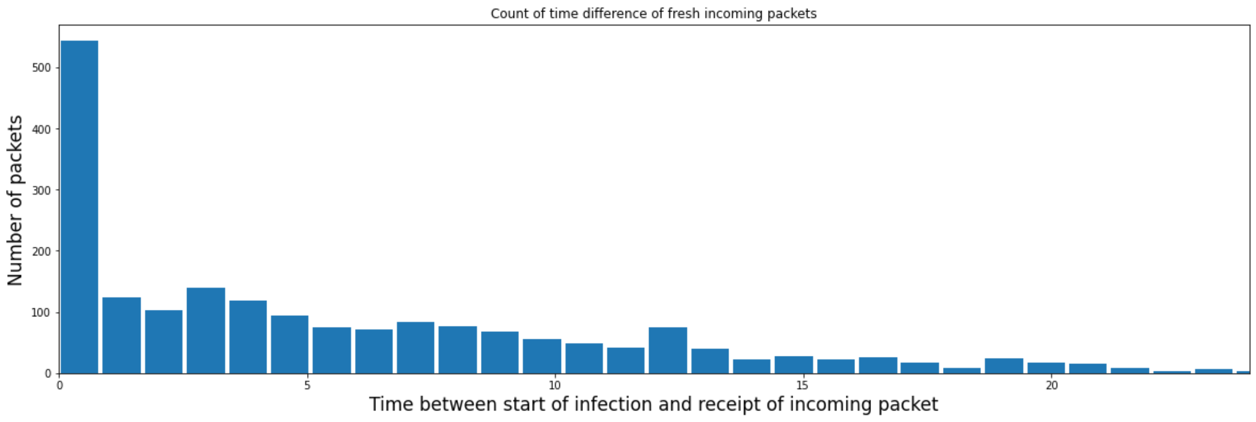


Figure 6.18: Histogram of iteration value of fresh incoming packets focusing on the lower values.

6.4. Bruteforcing DDoS packets

When bruteforcing the seed for DDoS attacks, we didnt manage to find any traces of activity for a DDoS attack, in particular by looking at the SYN-ACK and RST packets that would originate from the TCP DDoS attacks as seen in the Mirai source code. Despite verification of the workings of the code, no observations were made for internet backscatter traffic over a period of 2 days. Some reasons as to why we didnt observe any backscatter traffic are elaborated in Section 7.2

7 Reflections

7.1. Size of dataset

One of the biggest challenges over the course of this thesis was dealing with an extremely large dataset. This created two major issues, The primary issue was the increase in resources required, Performing operations either involved breaking up the data into smaller pieces and performing batchwise analysis to avoid overwhelming the available resources. Even then, certain operations require a large amount of processing power which posed a great challenge especially when considering the limited time frame of the thesis. Upgrading to a more powerful server alleviated the issues but not entirely. Considering the amount of computations required to brute force incoming packets, even a relatively powerful server would take a long time with optimisations in place. This would stack up during the exploratory phase of the thesis as it usually requires to experiment with varying parameters and methods.

Furthermore some checks require requests to be made to a remote server which holds all the data collected by the honeypot. The data that these requests are based on is too large to be stored locally, and requires all requests based on complex logic which depends on a lot of variables to be made in batches to ensure that the storage server is not overwhelmed. Furthermore some calculations require repeated requests to the server based on some logical conditions which can take a large amount of time to execute. These factors played a pivotal part over the course of this thesis, sometimes deciding the path that would be taken in terms of feasibility of answering research questions.

Iteratively pruning the data using clues from certain analyses would help reduce the amount of data to be analysed. An example that was done by us was finding the start of a botnet session from brute-forcing the seed and average bot lifetimes as inferred from the data that helped determine a window for searching for DDoS packets in case the bot also participate in a DDoS attack, using this methodology in more of our analysis would have helped mitigate the effort of working with such a large dataset.

7.2. Working without ground-truth data

When answering particular research questions such as bruteforcing seeds in DDoS backscatter traffic or trying to observe the difference between internet-wide and partial scans using difference in iterations between subsequent packets, it becomes fairly difficult to be able to draw concrete conclusions with the available information.

For instance, when checking for DDoS seeds as mentioned in Section 6.4, a check was made after manually inspecting the code when bruteforcing backscatter packets. When performed on backscatter traffic over the course of the day, no packets were found. This could mean that there is a flaw in the implementation or in the understanding of the working of the Botnet, which could be possible even after intense scrutiny. There are also assumptions that could be made on observing the behaviour such as:

- If no results were found when checking seeds obtained from bruteforcing scanning packets, then it could mean that there is a separation in the duties delegated to the botnet, meaning that we would not be able to match DDoS activity to seeds obtained from bruteforcing scanning packets.

- When bruteforcing seeds from scratch on DDoS traffic, not finding seeds could mean that there were no attacks in the period of time that we checked from vanilla Mirai botnets, and that a separate botnet or even a misconfiguration could have contributed to this traffic.
- There is a certain amount of transformation in the data that occurs in execution that can sometimes be overlooked when just statically analysing the code, some examples of this could be transformations that occur in separate byte order, or language specific quirks that exist which are not accounted for when emulating this behaviour in a separate, C and Golang in this case.

A lot of these doubts can be exhaustively concluded by utilising other sources of data or methods of analysis, however taking into account the time frame and scope of this thesis, these have remained as hypotheses and a foundation for future work.

7.3. Limitations of passive monitoring

One of the biggest challenges faced during this thesis was dealing with the lack of context provided by a Network Telescope. Passive monitoring is much easier to set up and much less resource intensive, However this also has a tradeoff in terms of information gathered. Network telescopes provide abundant information such as:

- Services targeted
- Location of origin of scans.
- Nature of targets
- Scanning activity over time
- Prevalent vulnerabilities.

to name a few. However being able to draw more concrete conclusions from these characteristics becomes exceedingly hard. In this thesis we utilise a flaw in the PRNG of the still prevalent vanilla Mirai variant, to brute force seeds as done in the work by Griffioen et al. [30], and build up on that by using the iterations required to brute force the current incoming packet to gain an understanding of the internal state of the infected device. Using some of these characteristics gives us invaluable insights into both the infected device itself and the nature of the botnet as well. However getting more information than what has already been done from this source becomes much harder, unless there are more breakthroughs made in exposing some misconfigurations in the botnet that leak additional information.

7.4. Importance of Passive Monitoring

Despite the limitations in passive monitoring as mentioned in the previous Section, it also has several redeeming qualities. Passive monitoring requires minimal effort in terms of setting up, and even less effort in terms of upkeep compared to other methods such as setting up honeypots, reactive telescopes or performing dynamic analysis on samples. These methods of investigation require significant infrastructural commitments alongside additional time invested by humans.

For example, setting up honeypots for services would require creating an environment that either allows a service or an accurate enough emulation of the service, Seeing as there are new vulnerabilities that are discovered frequently, keeping up with the ever changing landscape is a very difficult challenge to surmount and an important field of research. This is also applicable to analysing samples where the amount of samples that are collected grow much faster than they can be completed. Moroever, adversaries can identify measures taken to study these botnets and take counter measures to make the current solution obsolete or to make the analysis harder, turning the field of study into a cat and mouse game between the security industry and malicious actors.

In this case, passive monitoring provides a way to channel resources in the right directions. Understanding trends on a macro-level using passive monitoring can help researchers decides which services are the most frequently targeted and critical for research, or which variants are more prevalent and require more study. Understanding trends in the behaviour of botnets can help direct more intensive resources to the best possible research areas.

7.5. Unification of techniques

Working on this thesis has provided a lot of insight into the field of botnets and their prevalence to this day shows how important a field this remains. One of the best ways to deal with the ever changing landscape of botnets, would be to create an iterative process that would rely on a feedback mechanism between various sources of information with varying levels of complexity, For example, a network telescope could be used to identify unique fingerprints in packets received, using methods such as honeypots or device information through banner grabs could be used to confirm conjectures made from the telescope data. Going forward it would be easier to directly correlate telescope occurences to additional information once a line is already drawn between a pattern and a conclusion. This interdependance would ensure that a maximum amount of information is collected in a sustainable and scalable manner.

8 Conclusion and Future Work

Since their origin, botnets have been a monumental security risk by performing several devastating DDoS attacks and other activities such as scam mails and phishing. Despite measures being taken to curb such botnets, the distributed nature of botnets makes it extremely difficult to contain. As such, this is still an important field of study, to ensure that we have an understanding of the current state of botnets over the internet, what trends are currently being followed and upcoming threats. This allows for a more proactive approach to combating these botnets, denying the spread in the initial stages itself.

The proliferation of IoT devices and other internet connected devices provides a wide range of infection vectors for malicious actors on the internet. These devices often lack the protective measures and long term support present in fully fledged systems that make them especially susceptible to exploits through specially crafted/malformed requests. In order to identify these devices, the malicious actors scan the internet for services that are of interest to them and then proceed to exploit them. Although vanilla Mirai was one of the most infamous botnets, there now exist newer variants that are equally if not more potent. This thesis aims at looking at the current state of vanilla Mirai and using insights gained to better understand its newer strains. As a recap the research questions we had aimed at answering over the course of this thesis are:

RQ1. *What is the current state of Mirai?*

RQ2. *Can we find discriminatory features between botnet traffic, and regular scanning traffic?*

RQ3. *Can we find botnet traffic that does not have known signatures such as the Mirai signature?*

RQ4. *What does the lifecycle of a bot look like from a network telescope perspective? What activities does it perform over its lifetime and are they regularly changing?*

We find extensive data that shows that Mirai is still prevalent with a large ratio of total scanning sources attributable to Mirai. We also look at the trends of Mirai scanning traffic using characteristics such as the nature of the TCP/IP fields, the temporal aspects such as interarrival or active scanning time, and the number of packets sent. We also look at several visualisations that reveal unique relations in the Mirai data. We also compare these with the regular scanning traffic to find any discriminatory factors. Although we see a marked difference on a large scale, it becomes fairly difficult to classify traffic from particular sources as botnet traffic with a high confidence, which provides us with a solution to the questions posed in **RQ1** and discriminatory features as posited in **RQ2**

We notice a peculiarity in the behaviour of several sources that send Mirai traffic in that they also send non-Mirai scanning traffic. Using this observation with other works equating botnet infections to classical models as seen in epidemiology allows us to reliably connect these concepts and to find a better filter for certain botnet variants. Setting certain constraints allowed us to isolate what we believe with a high confidence to be scanning traffic from Mirai variants that do not exhibit the classical mirai signature. Setting further constraints also allows to partition these packets into what we believe to be individual sessions which could also help provide a better understanding of Mirai variant lifetimes and activities. This allows us to address **RQ3** by providing a framework to extract packets believed to be from other botnets, and even varying parameters to find botnets that exhibit particular behaviours.

We also build upon existing work by bruteforcing incoming Mirai scanning packets to find the seed that was used to generate random values used in various parts of the bots activities, and improve on it by speeding up the implementation and reducing the need for verification of results by reducing the chances of collisions in values.

These seeds are then supplemented by also providing the internal state of the random number generator at the time of generating the packet, which provides a more accurate internal state of the bot while it scans. Doing so allowed us to make certain conclusions such as the fact that there must be a separation in bots which perform scanning activities and those that take part in a DDoS attack due to the nature of concurrent packets, or that vanilla Mirai performs little to no DDoS attacks. We also find certain factors such as the rate of scanning determined by looking at the delta of iterations vs. the time to identify the scanning rate of bots, to see if it was a discriminatory factor and if certain assumptions regarding the location or device could be reliably drawn. Also looking at the average difference in subsequent packets would allow us to identify if there are changes made to the souce code in terms of targeted geographic locations. Although it is difficult to conclusively state these findings using only telescope data, this work provides a foundation for further sources of information to be used to test these hypotheses to answer **RQ4** .

8.1. Future Work

Over the course of this thesis, several new avenues of research have been explored, which in turn reveal a lot of new questions, which would need to be explored to build on top of this largely exploratory work. Some of the possible directions that could be taken based on the insights are as follows.

8.1.1. Expanding constraints to filter traffic

In this work, 4 cases were taken to highlight the promising nature of the idea of using prevalent infections to filter out newer more well hidden variants, however these cases are not exhaustive, and can be further explored or developed, perhaps even automatically by using certain heuristics to gauge the nature of the results. This would greatly help in characterising non-Mirai variants by taking advantage of the already existing prevalent Mirai infections. Expanding the existing set of constraints could help highlight different strains of botnets based on their targets, behaviour, etc.

8.1.2. Improving discriminatory factors

This study focuses on looking at aggregate statistics to identify discriminatory features between botnet and scanning traffic. Using some methods to cluster and identify groups based on passive measurements could reduce the vast data collected into patterns that could be easier to visualise and manually inspect.

8.1.3. Improving vantage points

In this thesis, the data collected was from 3 /16 blocks of IP addresses, this could lead to a bias in the data as only botnets randomly targeting the internet and the region the telescope is present in would present in the current data. Using the additional information from bruteforcing the seed and adding the iterations can provide much needed context when working with smaller and more distributed telescopes.

8.1.4. Understanding visibility

Understanding how long it takes for a botnet, if at all to reach a telescope is an important question when performing a study using data gathered from the telescope. Using the background vanilla Mirai infection to characterise the nature of traffic arriving at different telescopes could provide a unique insight on the difference in activity from botnets in different local ranges.

8.1.5. Sample analysis

While bruteforcing seeds, the low yield was hypothesised to be due to changes made in the original source code, this could be verified by performing static analysis on various samples. These changes could then be verified in the data found in the telescope, allowing for an attribution of scanning packets to particular strains. Futhermore, certain aspects such as the reasoning for the observed bot lifetimes can be verified by running the bot in a sandbox environment.

8.1.6. Correlating with banner grabs

Certain characteristics were observed when looking at the change in iterations of the random number generator with respect to time. Certain discrete values of the slope were found which could mean that the scanning rate is dependant on the device or some other factor such as the ISP or the geographical location. Correlating these observations with banner grabs or IP information could provide interesting insights, or maybe provide device fingerprints from just passive measurements.

8.1.7. Understanding bot sessions

Looking at the iterations of a bot over time lead to the conclusion that either most bots do not perform any activity apart from scanning or that there are groups of bots that perform scanning and other activities which are separated. Emulating a bot to receive instructions from the C&C server could help confirm the hypotheses.

Bibliography

[1] *INTERNET PROTOCOL*. https://datatracker.ietf.org/doc/html/rfc791.

[2] *Traditional IP Network Address Translator (Traditional NAT)*. https://datatracker.ietf.org/doc/html/rfc3022.

[3] *EarthLink Victorious In Spam Lawsuit*. https://www.cbsnews.com/news/earthlink-victorious-in-spam-lawsuit/.

[4] *ARPANET*. https://www.darpa.mil/about-us/timeline/arpanet.

[5] *What is the OSI Model?* https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/.

[6] Andrew S. Tanenbaum and David Wetherall. *Computer Networks*. 5th ed. Prentice Hall, 2011.

[7] *RFC 793*. https://datatracker.ietf.org/doc/html/rfc793.

[8] Yimu Ji et al. “The Study on the Botnet and its Prevention Policies in the Internet of Things”. In: *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*. 2018, pp. 837–842. DOI: 10.1109/CSCWD.2018.8465280.

[9] Brett Stone-Gross et al. “Your botnet is my botnet: Analysis of a botnet takeover”. In: Nov. 2009, pp. 635–647. DOI: 10.1145/1653662.1653738.

[10] Manos Antonakakis et al. “Understanding the Mirai Botnet”. en. In: ().

[11] Joel Margolis et al. “An In-Depth Analysis of the Mirai Botnet”. In: *2017 International Conference on Software Security and Assurance (ICSSA)*. 2017, pp. 6–12. DOI: 10.1109/ICSSA.2017.12.

[12] Zakir Durumeric, Michael Bailey, and J. Alex Halderman. “An Internet-Wide View of Internet-Wide Scanning”. In: *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 65–78. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/durumeric.

[13] Antonia Affinito et al. “The evolution of Mirai botnet scans over a six-year period”. In: *Journal of Information Security and Applications* 79 (2023), p. 103629. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2023.103629. URL: https://www.sciencedirect.com/science/article/pii/S2214212623002132.

[14] *Reaper Botnet*. https://www.radware.com/security/ddos-threats-attacks/threat-advisories-attack-reports/reaper-botnet/.

[15] *Satori*. https://blog.netlab.360.com/warning-satori-a-new-mirai-variant-is-spreading-in-worm-style-on-port-37215-and-52869-en/.

[16] *A Wicked Family of Bots*. https://www.fortinet.com/blog/threat-research/a-wicked-family-of-bots.

[17] *IoT Botnet: More Targets in Okiru’s Cross-hairs*. https://www.fortinet.com/blog/threat-research/iot-botnet-more-targets-in-okirus-cross-hairs.

[18] *Mirai-Based Masuta Botnet Weaponizes Old Router Vulnerability*. https://www.securityweek.com/mirai-based-masuta-botnet-weaponizes-old-router-vulnerability/.

[19] *Satori Author Linked to New Mirai Variant Masuta*. https://blog.newskysecurity.com/masuta-satori-creators-second-botnet-weaponizes-a-new-router-exploit-2ddc51cc52a7.

[20] *NEW MIORI BOTNET HAS A UNIQUE PROTOCOL FOR C2 COMMUNICATION*. https://securityaffairs.com/88303/malware/miori-botnet-new-c2-protocol.html.

[21] *Hackers Exploiting ThinkPHP Vulnerability To Expand Hakai and Yowai Botnets*. https://gbhackers.com/hackers-exploiting-thinkphp-vulnerability-to-expand-hakai-and-yowai-botnets/.

[22] *Mirai Variant MooBot Botnet Exploiting D-Link Router Vulnerabilities*. https://www.blazeguard.com.au/mirai-variant-moobot-botnet-exploiting-d-link-router-vulnerabilities/.

[23] *SORA and UNSTABLE: 2 Mirai Variants Target Video Surveillance Storage Systems*. https://www.trendmicro.com/vinfo/hk/security/news/internet-of-things/sora-and-unstable-2-mirai-variants-target-video-surveillance-storage-systems.

[24] *Mirai's new variant "Mukashi" attacks network-attached devices*. https://success.trendmicro.com/dcx/s/solution/000283375?language=en_US.

[25] *New Threat: ZHtrap botnet implements honeypot to facilitate finding more victims*. https://blog.netlab.360.com/new_threat_zhtrap_botnet_en/.

[26] *New Mirai botnet variant found operating across 3 different campaigns in 2022*. https://www.cyberdaily.au/tech/8711-new-mirai-botnet-variant-found-operating-across-three-different-campaigns-in-2022.

[27] Sadegh Torabi et al. "Inferring, Characterizing, and Investigating Internet-Scale Malicious IoT Device Activities: A Network Telescope Perspective". In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2018, pp. 562–573. DOI: 10.1109/DSN.2018.00064.

[28] Alberto Dainotti et al. "Analysis of a "/0" Stealth Scan From a Botnet". en. In: *IEEE/ACM Transactions on Networking* 23.2 (Apr. 2015), pp. 341–354. ISSN: 1063-6692, 1558-2566. DOI: 10.1109/TNET.2013.2297678.

[29] Harm Griffioen and Christian Doerr. "Examining Mirai's Battle over the Internet of Things". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 743–756. ISBN: 9781450370899. DOI: 10.1145/3372297.3417277. URL: https://doi.org/10.1145/3372297.3417277.

[30] Harm Griffioen and Christian Doerr. "Quantifying autonomous system IP churn using attack traffic of botnets". In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ARES '20. Virtual Event, Ireland: Association for Computing Machinery, 2020. ISBN: 9781450388337. DOI: 10.1145/3407023.3407051. URL: https://doi.org/10.1145/3407023.3407051.

[31] *CVE-2014-8361*. https://nvd.nist.gov/vuln/detail/CVE-2014-8361.

[32] *CVE-2017-17215*. https://nvd.nist.gov/vuln/detail/cve-2017-17215.

[33] *Satori*. https://www.trendmicro.com/en_nl/research/18/g/open-adb-ports-being-exploited-to-spread-possible-satori-variant-in-android-devices.html.

[34] *ADB.MIRAI: NEW MIRAI BOTNET VARIANT SPREADING VIA THE ADB DEBUG PORT*. https://nsfocusglobal.com/adb-mirai-new-mirai-botnet-variant-spreading-via-the-adb-debug-port/.

[35] Georgios Kambourakis, Constantinos Kolias, and Angelos Stavrou. "The Mirai botnet and the IoT Zombie Armies". In: *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*. 2017, pp. 267–272. DOI: 10.1109/MILCOM.2017.8170867.

[36] *Windows port allocation*. https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/default-dynamic-port-range-tcpip-chang.

[37] William Ogilvy Kermack, A. G. McKendrick, and Gilbert Thomas Walker. "A contribution to the mathematical theory of epidemics". In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 115.772 (1927), pp. 700–721. DOI: 10.1098/rspa.1927.0118. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1927.0118. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1927.0118.

[38] *Mirai Source Code*. https://github.com/jgamblin/Mirai-Source-Code/tree/master.

[39] Giovane C. M. Moura et al. “How dynamic is the ISPs address space? Towards internet-wide DHCP churn estimation”. In: *2015 IFIP Networking Conference (IFIP Networking)*. 2015, pp. 1–9. DOI: 10.1109/IFIPNetworking.2015.7145335.

[40] *Dieharder Test Suite*. https://webhome.phy.duke.edu/~rgb/General/dieharder.php.

[41] Meghan Riegel. “Tracking mirai: An in-depth analysis of an iot botnet”. PhD thesis. Pennsylvania State University, 2017.

[42] *CVE-2023-33246*. https://nvd.nist.gov/vuln/detail/CVE-2023-33246.

[43] *RabbitMQ vulnerability*. https://blogs.juniper.net/en-us/threat-research/dreambus-botnet-resurfaces-targets-rocketmq-vulnerability.

[44] Philipp Richter and Arthur Berger. “Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope”. In: *Proceedings of the Internet Measurement Conference*. IMC ’19. Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 144–157. ISBN: 9781450369480. DOI: 10.1145/3355369.3355595. URL: https://doi.org/10.1145/3355369.3355595.

[45] Daniel Wagner et al. “How to Operate a Meta-Telescope in your Spare Time”. In: *Proceedings of the 2023 ACM on Internet Measurement Conference*. IMC ’23. <conf-loc>, <city>Montreal QC</city>, <country>Canada</country>, </conf-loc>: Association for Computing Machinery, 2023, pp. 328–343. ISBN: 9798400703829. DOI: 10.1145/3618257.3624831. URL: https://doi.org/10.1145/3618257.3624831.