

Approximate Automated Campaign Analysis with Density Based Clustering

by

Federico Falconieri

submitted in partial fulfillment of the
requirements to obtain the degree of
Master of Science
in
Computer Science
at the Delft University of Technology,
to be defended publicly on
Thursday October 11, 2018 at 10:00 AM.

This thesis is confidential and cannot be made public until December 31, 2019.

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Thesis committee:

Chair: Dr. ir. J. C. A. van der Lubbe, Faculty EEMCS, Cyber Security Group, TU Delft

Committee Member: Prof. dr. Alan Hanjalic, Faculty, EEMCS Multimedia Computing Group, TU Delft

Supervisor: Dr. Christian Doerr, Faculty EEMCS, Cyber Security Group, TU Delft

Contents

1	Introduction	1
1.1	Contemporary Threat Landscape and Threat Intelligence	1
1.2	State of the art and gaps: Lockheed Martin kill chain model	2
1.3	Research goal and vision	4
2	Related Work	7
2.1	Multi Stage Threat Modelling: the kill chain	7
2.2	Lockheed Martin Intrusion Kill Chain	9
2.2.1	Indicators	9
2.2.2	Course of Action	11
2.2.3	Kill chain analysis & synthesis	11
2.2.4	Campaign analysis	12
2.3	Kill Chain related work	12
2.4	Research gaps	20
3	Methodology	21
3.1	Hypotheses and research questions	22
3.2	Automation: clustering	23
3.2.1	Selecting a suitable clustering method	24
3.2.2	Density based clustering algorithms: DBSCAN and HDBSCAN	24
3.3	Fuzziness: approximate string match	25
3.4	Experimental data: Dutchsec Spamtrap	26
3.4.1	Emails and attachments in the kill chain.	27
3.4.2	Dutchsec Spamtrap	28
3.4.3	Malignity blindness and data suitability	29
3.4.4	Selection of features suitable for approximate match.	29
3.5	Approximate Automated Campaign Analysis with Density-Based Clustering	31
3.5.1	System overview	31
3.5.2	Phase 1: data enrichment	32
3.5.3	Phase 2: preprocessing and feature selection	36
3.5.4	Phase 3-6: the automation engine.	36
3.5.5	Phase 3: indicator level similarity computation	37
3.5.6	Phase 4: indicator level clustering.	40
3.5.7	Phase 5: killchain level similarity computation	44
3.5.8	Phase 6: killchain level clustering (campaign labeling).	45
4	Results	49
4.1	Campaign Analysis, Research Gap and Research Goals of this project	49
4.2	Experimental results	49
4.2.1	Dataset	50
4.2.2	Experimental Setup	50
4.2.3	False Negatives and Recall	51
4.2.4	False Positives and Precision	52
4.3	Analysis	53
4.4	Limitations	53
4.5	Notable Findings	57

5 Conclusion and Future Work	61
A Kill chain Features	65
B Synthetic campaigns	69
C Real Campaigns discovered manually	71
Bibliography	75

Introduction

The modern cybersecurity landscape is characterised by the increasing number of actors capable of performing advanced and highly impactful hacking. The situation has worsened significantly in the last decade because more and more of the critical infrastructure is connected to the Internet, because the capabilities of attackers have improved and because their numbers have increased.

Threat Intelligence emerged as a valuable domain to enhance security defences by studying threats motives, techniques, tools and procedures. Campaign analysis is a process that belongs to this domain and deals with following attackers through time by linking several hack attempts that share a threat actor, a victim and that have a specific goal. Unfortunately, this process is rarely applied in practice because the campaign analysis models available in literature rely on manual investigation by security professionals. This approach can become quickly too expensive, both regarding time and human resources.

In this thesis project, we improve the state of the art by automating a popular campaign analysis framework introduced in 2011 by Lockheed Martin security researchers Hutchins et al. [10]. We do not only automate the process: we also improve its recall performance to provide security analyst with more interesting and complete findings. Hopefully, this will empower all organisations, of any size an security profile, to perform their threat intelligence. Lowering the adoption threshold is a fundamental requirement that is inescapable if we want security to improve horizontally throughout all industry sectors. Widespread adoption of campaign analysis would lead to a broader and quicker understanding of threat campaigns and goals, contributing to a safer society.

1.1. Contemporary Threat Landscape and Threat Intelligence

Advance Persistent Threats, or APTs, are groups with enough time, resources and capabilities to continuously attempt hacking towards a specific target. APTs started to become a severe problem at the beginning of this decade. Not only they grew in numbers as nations scrambled to get their state-sponsored groups, but the impact of their operations started to be significant enough to fall under the definition of warfare (or better cyberwarfare). Indeed, NATO started considering cyberspace the fifth operational domain for warfare after land, sea, air and space ¹. The Pentagon started to do this as early as 2010 [14]. APTs are not all non-profit driven state-sponsored groups: some are large criminal groups using hacking as a source of income. In general, APTs activities range from acts of cyberwarfare (like the Stuxnet hack ²) to both conventional and industrial espionage (like the Gemalto ³ ⁴ and Petrobras ⁵ cases). They are characterised by multi-staged, long-term intrusion attempts, persistence and use of highly sophisticated techniques. State-sponsored APTs techniques tend to be adopted by criminal groups down the line. One excellent and recent example is the WannaCry ransomware campaign of

¹<https://www.militarytimes.com/2016/06/14/air-land-sea-cyber-nato-adds-cyber-to-operation-areas/>

²<https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/>

³<https://www.gemalto.com/press/pages/gemalto-presents-the-findings-of-its-investigations-into-the-alleged-hacking-of-sim-card-encryption-keys.aspx>

⁴<https://theintercept.com/2015/02/19/great-sim-heist/>

⁵<https://www.theguardian.com/world/2013/sep/09/nsa-spying-brazil-oil-petrobras>

May 2017. The United States National Security Agency (NSA) discovered a flaw in Microsoft implementation of an old but ubiquitous version of SMB (Server Message Block). Rather than responsibly disclosing the flaw to Microsoft, the NSA developed an exploit, called EternalBlue, that abused the vulnerability. On April 14th 2017 a group of hackers (the ShadowBrokers) leaked to the public several NSA developed exploits, including EternalBlue. On May 12th 2017 the WannaCry ransomware campaign started and it infected over 230000 computers in more than 150 countries within a day⁶. Wannacry used EternalBlue to infect other computers on the same local network over SMB. After ShadowBrokers publication, the APT developed exploit was accessible to everyone, including criminal groups that then started using it for their profit-driven purposes. The access to these sophisticated methods vastly enlarged the pool of threat actors capable of performing advanced hacking. As seen for Wannacry, tools developed by state-sponsored actors can quickly end up in criminal group hands.

This widespread adoption of APTs techniques when leaked (or sold on the black market to the best offerer) have made the eggshell model of security increasingly insufficient. The eggshell security is the traditional model where there is just one hard layer of defences at the border, around an organisation, and nothing more. Thus, if an attacker successfully breaches the perimeter, he then has instantly gained access to all valuable assets. To cope with the level of complexity and capability of APTs as well as their modus operandi (which as seen can also spread to less skilled threat groups) the defensive side of security has moved from the eggshell model to a layered model, where defensive technologies are deployed both on the border of and within an organisation computer network. Furthermore, because acting only upon discovery of a compromise can be already too late, for example when the goal of the hack is the exfiltration of sensitive information, defensive effort has to focus on multiple aspects like prevention, protection, detection and response.

Threat Intelligence has emerged as a valuable process to improve cybersecurity operations in organisations by shifting the focus from sole defending to additionally understanding the threat landscape. The goal of threat intelligence is to study tactics, techniques and procedures of threats. Findings can be leveraged and shared to select the most appropriate cybersecurity countermeasures, to perform data-driven cyber risk management and to predict & prevent their future moves.

The cyber-security research world has developed a plethora of multi-layered frameworks to perform threat intelligence in the past decade. Among them, one that stands out is Lockheed Martin kill chain model, which was released as a white paper in 2011 [10]. Lockheed Martin is the kind of organisation that state-sponsored APTs target due to the highly sensitive nature of their intellectual property: the corporation operates in weapon manufacturing and aerospace development for the United States government. The kill chain model was seminal in showing how data-driven threat intelligence can be performed in practice and how it can be used to continuously improving defences by studying adversaries over time. In the next section, we briefly present this model and why it stands out. Furthermore, we also show its limitation and how we aim to improve it.

1.2. State of the art and gaps: Lockheed Martin kill chain model

One of the most relevant models developed to perform threat intelligence of APTs intrusion attempts is the Lockheed Martin Intrusion Kill Chain [10], in which Hutchins et al. adapted the military concept of kill chain to the cyber security domain. Intuitively, an attack (traditional or digital) is always composed of a series of steps like for example studying the victim, thinking of a strategy and selecting a weapon. The kill chain model is a framework that envisions a complex attack process as a series of atomical sequential phases. To successfully achieve the attack objective it is required to complete every phase of the chain successfully. Any disruption at any of the phases will lead to failure. The cyber intrusion kill chain of Lockheed Martin has seven phases: reconnaissance (intelligence gathering on the victim), weaponization (developing the computer virus), delivery of the virus to the victim (usually via email), exploitation (convincing the victim to open the virus or exploiting a program flaw like WannaCry), installation of the virus on the victim computer, command and control (the installed virus enables the attacker to control the victim computer remotely) and finally actions on objective (for example exfiltrating information).

Among many other processes performed in Threat Intelligence, one is of particular interest: campaign analysis. The principal goal of campaign analysis is to determine the patterns and behaviours of the intruders, their tactics, techniques, and procedures (TTP), to detect “how” they operate rather than

⁶<http://www.bbc.com/news/world-europe-39907965>

specifically “what” they do [10]. Lockheed Martin defines a clear methodology to perform campaign analysis in practice with email carried spear phishing attacks.

Related incidents can be grouped by looking for commonalities in the metadata of the various attempts after they have been mapped to kill chains. Campaign discovery allows to understand the attacker motives, to track him through time and to understand his capabilities. All these findings can be used to improve defences, leading ideally to a model where the defender iteratively adjusts his defensive posture to the attacker capabilities as the attacker evolve his campaign through time. This process can be particularly valuable because it allows for data-driven security spending. The budget can be allocated efficiently where it is needed based on the actual threats being faced, rather than on a generic idea of the current threat landscape. Lockheed Martin intrusion kill chain and campaign analysis are explained in depth in section 2.2.

Unfortunately, the kill chain model or any other multi-stage threat models are rarely used in practice because frameworks found in literature, including Lockheed Martin’s, rely heavily on manual aggregation performed by experienced security analysts. The cost of hiring dozens of security analysts to spend hundreds of hours to perform this manually is prohibitive for all companies but those that are large enough and where threat intelligence is an absolute priority (like Lockheed Martin). Moreover, due to the absence of automation, it is necessary to hire a large number of experienced analyst, something not only expensive but also rare as the amount of professionals in cybersecurity is far below the (increasing) demand of the market⁷. These complexities are a deal breaker for a smaller organisation but are also not ideal for large organisations, where the amount of data can quickly become overwhelming.

Furthermore, state of the art campaign analysis can be easily circumvented by adding a component of randomness when an attack is carried. Indeed, **only exact match of kill chain features is performed in the original model.** A spear phishing campaign is not going to be discovered by this methodology if the attacker is wise enough to add a random sequence of numbers to his malicious attachment filenames. This kind of randomisation operation is easy to perform, but hard to do in a way that does not retain some form of structure (e.g. partial filename still the same or fixed number of digits). We encountered many examples of this kind of obfuscation techniques in the dataset that we later used to develop and test our automated framework. Figure 1.1 shows an example of “fuzzy” campaign that would effectively escape traditional exact-match-only campaign analysis.

According to a recent SANS survey⁸ 75% of the responders (representing organisations heterogeneously distributed across many sectors) find cyber threat intelligence critical to security. Yet, only 27% indicates that their teams have fully embraced the concept of CTI and integrated response policies across systems and staff. When asked what is holding their organisation back in achieving integrated CTI capabilities respectively 34.9% and 10.8% mentioned lack of budget and staff to perform it and lack of management buy-in/understanding of its usefulness. We can not retrieve an exact figure on how many companies perform Lockheed Martin version of campaign analysis, but the data available seem to indicate that, together to other CTI processes, it is not performed very often.

Automation could undoubtedly help to lower the threshold (cost) in the adoption of this process. A methodology that aids in automated campaign analysis would be very helpful for example in security operation centres to keep track of malicious actors over time and improving detection rules by consequence. The impact of such a methodology would be two-fold. On the one hand, organisations that already do perform campaign analysis could have their analysts use their time more efficiently because the manual grouping could be avoided. Instead, these analysts could focus on evaluation and data exploration. On the other hand, organisations that do not perform any CTI and specifically campaign analysis could finally perform it without the need for a large analyst team or a significant increase in budget spending.

A solution for automatic campaign analysis would significantly lower the adoption hassle for many organisations. More organisations performing threat intelligence means more indicators collection and possibly sharing, and indicator sharing leads to a more secure society.

Bottom Line:

⁷<https://www.forbes.com/sites/jeffkaufman/2017/03/16/the-fast-growing-job-with-a-huge-skills-gap-cyber-security/#20ed9c515163>

⁸<https://www.alienvault.com/docs/SANS-Cyber-Threat-Intelligence-Survey-2015.pdf>

- sophisticated techniques developed by APTs eventually leak/sold to less skilled groups...
- ...threat landscape increasingly larger and more capable of causing serious damage
- to cope with this: new defensive processes, including threat intelligence
- kill chain model and campaign analysis are effective forms of threat intelligence...
- ...but often too expensive because heavily manual and thus not performed in practice
- solution: automation

1.3. Research goal and vision

The problem we are trying to solve, automating campaign analysis, is a form of machine learning task. Specifically, in the absence of training data, we are dealing with unsupervised learning. We can think of this problem as a form of clustering because we are trying to aggregate similar items together. Indeed, we are trying to automatically infer a structure from the raw data, without any a priori knowledge. The goal of this thesis project is to produce and evaluate a methodology to automate campaign analysis employing clustering. We obtained a large set of malicious emails from Dutchsec, an Utrecht-based cybersecurity company; we use this data to design and evaluate our solution. We develop this solution in practice with a pipeline of tools that:

- have as input non-labelled malicious emails
- extract valuable features from the emails
- extract further features by executing the malicious attachments and recording their activity
- cluster together related emails, allowing for different threat profiles to be selected for the clustering (e.g. APT targeted attacks or mass phishing scams campaigns)
- present the results back to the analyst

A diagram of our solution is shown in Figure 1.2, where the scientific contribution is highlighted in the blue area.

It is essential to understand to what degree we want to introduce automation in this domain. The presence of an analyst is fundamental to evaluate the results of the clustering. Our deliverable should be seen as an aid to perform exploratory data analysis, rather than a perfect system that removes the need for the human in the loop. The goal is to automate the process of aggregating kill chains, which is the substantially mechanical portion of the job. This task can take hours or even be infeasible if the amount of data or its multidimensionality are large. We expect our system to produce some noise because no clustering can be perfect. We believe that the tradeoff between recall and precision will be worth it because if the linkage can take very long to be performed manually, whereas getting rid of noise can be done quickly by an experienced analyst because of his domain knowledge. Furthermore, the benefits of introducing automation would not be limited to the task of labelling campaigns. Indeed, after analyst evaluation, verified campaigns will have been found. These could then be used as a training set for further automatic detection and classification.

This thesis report is structured as follows.

Section 2 contains a thorough explanation of the concept of kill chain, from its military origin to Lockheed Martin cybersecurity adaptation and a survey of the state of the art in kill chain models for threat intelligence .

Section 3 presents the research methodology together with the choices and implementing challenges that characterised its development.

Section 4 shows the final results.

Section 5 finally describes the conclusion of this research project and the possible future work.

	m-from	m-subject	m-xmaller	p-hashes	p-filenames
0	Selena Ruiz <RuizSelena445@predatorcheck.com>	FW: Payment #239446	[e393bd1cd878ce3606a55ad93df6e515221cfbd3,5cb...	[urgent_payment_239446.zip,]	
1	Gonzalo Mendez <MendezGonzalo88@ultranet.inf.br>	FW: Payment #035137	[59ddd32d5411de8ca0f0a2b91e5a0ee27b1f164f,5ae...	[, payment_info_035137.zip]	
2	Barbara Perry <PerryBarbara2526@dhcp-089-098-1...>	FW: Payment #394763	[5ae0a84432e34ce23e0a02cd2f36fcc6d911fa97,7b8...	[urgent_payment_394763.zip,]	
3	Shelia Sargent <SargentShelia85@chello.ni>	FW: Payment #352372	[d9f8669cd7812db941733771286f0cab4d419b07,e39...	[, details_payment_352372.zip]	
4	Rachelle Holmes <HolmesRachelle7609@chello.pi>	FW: Payment #336635	[50d911baaa2c1a12fd5a713962b2c1973148a7b1,e39...	[, payment_urgent_336635.zip]	
5	Marilyn Hendrix <HendrixMarilyn512@stephenrain...>	FW: Payment #533935	[e393bd1cd878ce3606a55ad93df6e515221cfbd3,010...	[details_payment_533935.zip,]	
6	Karla Chapman <ChapmanKarla81958@ono.com>	FW: Payment #790221	[cb80804900f8b91cf94aef6aadd3840e610ace41,e39...	[, details_payment_790221.zip]	
7	Merlin Cardenas <CardenasMerlin7552@impsat.net...>	FW: Payment #346417	[b7cc7d47d56e3b1e28de3d3fa8646a33e03fe86c,e39...	[, payment_urgent_346417.zip]	
8	Anibal Acosta <AcostaAnibal67@gtinternet.com>	FW: Payment #368845	[a7ac9fa7b9a82d64da8409198976fcb408522cc2,e39...	[, wire_payment_368845.zip]	
9	Bryon Nguyen <NguyenBryon41@51-151.wind.it>	FW: Payment #211300	[b57c465d17affd45a167b6b473929b188f147a2d,6bb...	[urgent_payment_211300.zip,]	
10	Shelley Crosby <CrosbyShelley52985@plus.pl>	FW: Payment #386144	[a5b61968a3ef34c76f3d04bc11890da87f3c2e44,9e2...	[, urgent_payment_386144.zip]	
11	Ivy Noel <Noellivy25038@propulsestudio.com>	FW: Payment #931560	[db39e9d11f67f3358b13d5a2365ab1978f029dd0,rthn	[, payment_info_931560.zip]	

Figure 1.1: Example of a fuzzy spear phishing campaign encountered in the dataset at our disposal. All indicators seems to almost always be randomized to a degree, but they also seem to maintain an overall logical structure.

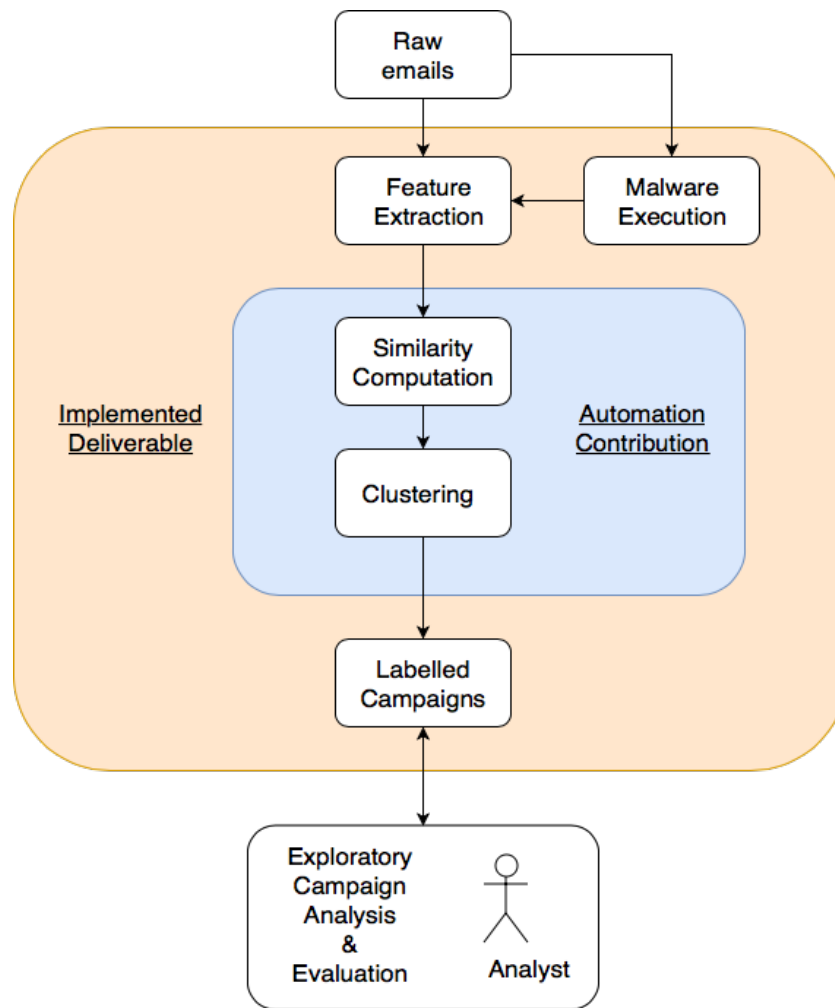


Figure 1.2: High level overview of the system envisioned for this thesis project. The orange area is implemented as a deliverable. The blue area represents the automation engine, which is the scientific contribution of this model.

2

Related Work

This section contains a summary of the state of the art on multi-stage threat models for cybersecurity. The concept of kill chain is explained, as well as all the insight that can be derived from it, with a focus on threat intelligence and cyber risk management. We summarise here the related work and point to the research gap. Section 2.1 explains the general theory and common military applications of kill chain models. Section 2.2 describes in detail the paper that adapted the concept of kill chain to the cybersecurity domain: Lockheed Martin's "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains"[10]. Section 2.3 is a broad summary of the most relevant literature inspired by Lockheed Martin intrusion kill chain.

2.1. Multi Stage Threat Modelling: the kill chain

The kill chain model is a framework that maps attacker behaviour to phases in an adversarial setting, for example conventional symmetric warfare or unconventional asymmetric warfare (like terrorism or guerrilla fighting). In a kill chain model, an attack is broken in atomic phases.

Figure 2.1 exemplarily shows a well known and used in practice model: the F2T2EA kill chain used by the United States Air Force. F2T2EA stands for the 6 phases find, fix, track, target, engage, assess.

A short example of an attack mapped through this model is the following:

1. Find: a phone call by a person of high interest is detected in a heavily populated urban area of a city neighbouring a war zone.
2. Fix: GSM triangulation is used to establish the location of the subject
3. Track: a surveillance drone is deployed in the area to get visual confirmation of the target identity. Visual confirmation is established when the subject leaves the building. The drone keeps following him.
4. Target: the chain of command authorises a strike as soon as the subject leaves the urban area to minimise collateral damage.
5. Engage: an F16 jet takes off from the nearest airbase and reaches the target. Then strikes the target with a missile as he is moving in a car convoy out of town
6. Assess: the surveillance drone provides visual confirmation of the attack result, the subject car was successfully hit and the subject is confirmed killed. The F16 jet can return to base

In a kill chain model, the attacker needs to go through each phase consecutively to reach his objective. A disruption between any of the steps will stop the attack. In this example losing visual contact of the target at phase Track or a mechanical failure of the F16 jet at phase Engage would stop the attack. Aside from increasing chances of success in stopping/executing an attack, applying this model usually comes with further benefits:

- enhanced intelligence on adversary intent, capabilities and methods; this follows from the fact that the modus operandi of the studied actor is dissected in detail.

F2T2EA Killchain
Gen. John Jumper (U.S. Air Force), 1990

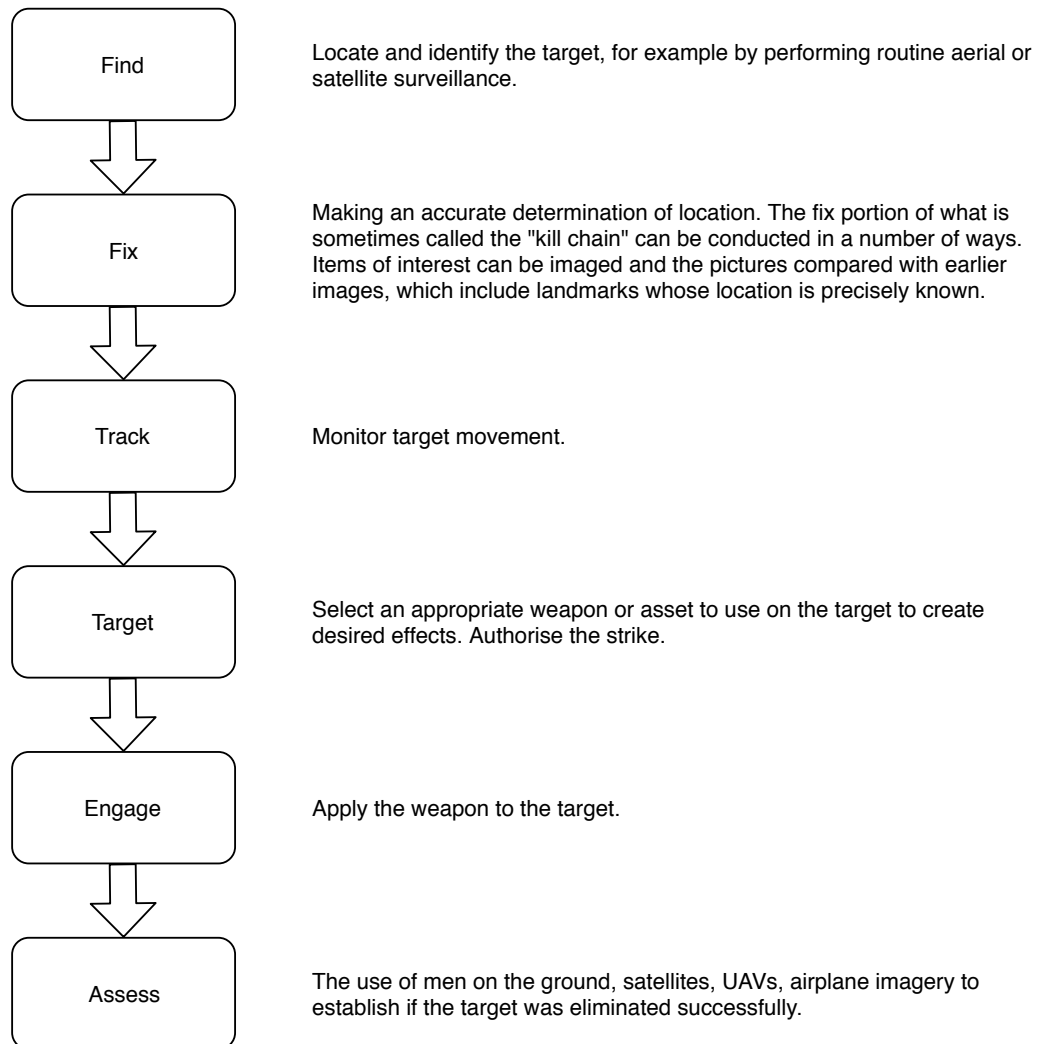


Figure 2.1: F2T2EA killchain model, used since 1990 by the U.S. Air Force.

- enhanced defence/offence by optimising resource allocation with the gathered intelligence.

General John Jumper, The Chief of Staff of the U.S. Air Force in the late 1990s is the father of the modern interpretation of the kill chain. He is the creator of the F2T2EA kill chain mentioned above, visible in Figure 2.1. The United States have been using this model in several conflicts since the first Gulf War, both with offensive and defensive goals.

An example of a defensive use of the model is to use it to disrupt IED ¹ attacks from funding to attack execution.

2.2. Lockheed Martin Intrusion Kill Chain

In 2011 Hutchins, Cloppert and Amin adapted the concept to cybersecurity with the paper “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains”[10] for Lockheed Martin Corporation. The goal of the researchers was to establish a framework to enhance defensive capabilities against intrusions of Advanced Persistent Threats (APTs).

Hutchins et al. define a cyber intrusion as follows:

“The essence of an intrusion is that the aggressor must develop a payload to breach a trusted boundary, establish a presence inside a trusted environment, and from that presence, take actions towards their objectives, be they moving laterally inside the environment of violating the confidentiality, integrity, or availability of a system in the environment.”[10]

The phases of Lockheed Martin Intrusion Kill Chain are:

1. Reconnaissance: research, identify and select targets.
2. Weaponization: coupling of a remote access tool/backdoor with an exploit, often automatically with a tool called weaponizer, into a deliverable payload.
3. Delivery: transmission of the payload to the target.
4. Exploitation: triggering of the payload on the target environment through a software vulnerability or user action.
5. Installation: a remote access tool/backdoor is installed on the target environment to maintain persistence and allow lateral movement.
6. Command and Control: the installed malware must establish a communication channel with the attacker to allow manual interaction.
7. Actions on Objectives: data exfiltration, lateral movement within the network, etc.

Figure 2.2 shows a diagram of the model with extensive phase explanation from the original paper. The authors developed this model to assess the disruptive rise of APTs in the threat landscape in the early 2010s and the limits of traditional incident response, which mostly occurred only after a successful breach. Hutchins et al. publication was very impactful and introduced several concepts and processes that are relevant for this thesis project: indicators, course of action, kill chain analysis & synthesis and campaign analysis.

2.2.1. Indicators

Indicators are any feature belonging to the data used or produced by the attacker. Examples of indicators in a spear phishing attack are the email headers, metadata of the payload, the vulnerability exploited, the hash of the payload, the files dropped by the malware, the IPs and domains that the malware communicates to and many more. Indicators are essential for two reasons: they are traces that the attacker leaves, and as such, they can be used to fingerprint him and track him through campaign analysis (more on this in a few paragraphs). Furthermore, they allow for practical risk management because they show the capabilities of the threat agent. These capabilities can be mapped to counter-measures in what Hutchins et al. call a Course of Action matrix.

¹An improvised explosive device (IED) is a bomb constructed and deployed in ways other than in military action. It may be composed of conventional military explosives, such as an artillery shell, attached to a detonating mechanism. IEDs are commonly used as roadside bombs, and they are generally seen in unconventional asymmetric warfare. In the second Iraq War, IEDs were used extensively against US-led invasion forces

Lockheed Martin Intrusion Kill Chain, Hutchins et al., 2008

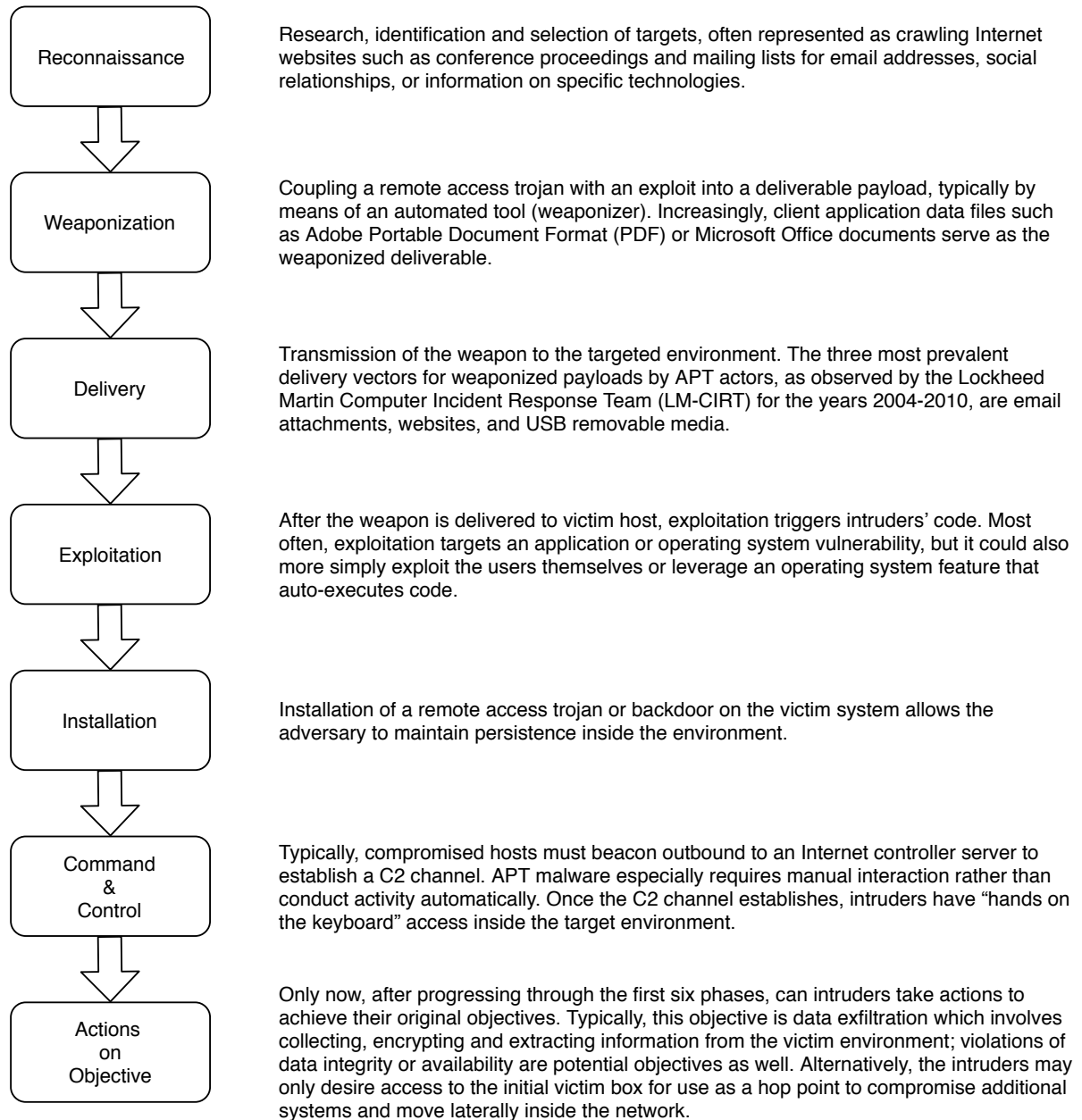


Figure 2.2: Lockheed Martin Intrusion Kill Chain Model [10]

Courses of Action Matrix

Phase	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Web analytics	Firewall ACL				
Weaponization	NIDS	NIPS				
Delivery	Vigilant user	Proxy filter	In-line AV	Queuing		
Exploitation	HIDS	Patch	DEP			
Installation	HIDS	"chroot" jail	AV			
C2	NIDS	Firewall ACL	NIPS	Tarpit	DNS redirect	
Actions on Objectives	Audit log			Quality of Service	Honeypot	

Figure 2.3: Course of Action Matrix from Lockheed Martin "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains"[10]. It maps the intrusion kill chain phases to the actions available to the defender. At the intersections are the countermeasures available

2.2.2. Course of Action

The Course of Action is a matrix that maps every phase of the kill chain to the six actions available to a defender. These actions come from the U.S. Department of Defense Information Operations Doctrine and are: detect, deny, disrupt, degrade, deceive, destroy. Countermeasures are placed at the intersection between phase and action. For example, a firewall is a countermeasure to deny some reconnaissance activity (e.g. Nmap port scan). This matrix can be used to effectively map defensive capabilities to attacker capabilities, leading to wiser investments in security. The Course of Action matrix from the paper can be seen in Figure 2.3

2.2.3. Kill chain analysis & synthesis

Kill chain analysis is a guide for analysts to understand what information is available for defensive courses of action [10]. Upon detection of an intrusion attempt, a kill chain is created. Immediately it can be assumed that mitigations of phases before the one where detection happened failed. For example, if detection occurs at the Command and Control phase, this means that the attacker was able to go through delivery, exploitation and installation successfully. Furthermore, the indicators belonging to these failed (from the defender perspective) phases can now be leveraged to improve the quality of the countermeasures deployed. Studying failed mitigations is called Analysis, whereas the study of post-detection phases is called Synthesis and is equally important. For example, if the email client blocks the email with a malicious attachment upon delivery, indicators of the malicious attachment can be used to discover new techniques and capabilities on later phases (for example a new zero-day vulnerability) that perhaps would have been successful if not for early detection. In other words, analysis and synthesis allow to increase security from both failed and successful intrusions.

If detection happens at any given phase, not only the attack is stopped but all the attacker techniques, intentions and to an extent identity are revealed, and this information can be leveraged against him to mitigate against his future attacks. This active defensive posture will force the attacker to spend a lot more time and resources in recursively developing always new methods, as he can

not recycle anything from his toolchain without being detected and linked to previous attempts.

2.2.4. Campaign analysis

Kill chain analysis and synthesis over multiple intrusion attempts from an APT will likely reveal commonalities and overlapping indicators. Attackers tend to reuse parts of their toolchain when they persistently attack a target. It is merely natural and resource-wise to do so. If an attack is stopped at delivery, why should an attacker waste a perfectly working payload with a zero-day vulnerability that he purchased on the black market for a high price? Overlapping indicators can be used for what Lockheed analysts call Campaign Analysis. Campaign Analysis allows tracking attackers through time by linking intrusion attempts (as kill chains) on overlapping indicators, getting a deep understanding not only of what they do but of their patterns, behaviours, tactics, techniques and procedures as well as their intent. This process can also be used to attempt to establish attribution (mainly from time/geospatial indicators, victim and methods). Figure 2.4 shows an example of repeated spear phishing campaign observed by Lockheed Martin.

Phase	Intrusion 1	Intrusion 2	Intrusion 3
Reconnaissance	[Recipient List] Benign PDF	[Recipient List] Benign PDF	[Recipient List] Benign PPT
Weaponization	Trivial encryption algorithm		
	Key 1		Key 2
Delivery	[Email subject] [Email body]	[Email subject] [Email body]	[Email subject] [Email body]
	dn...etto@yahoo.com		ginette.c...@yahoo.com
	60.abc.xyz.215	216.abc.xyz.76	
Exploitation	CVE-2009-0658 [shellcode]		[PPT 0-day] [shellcode]
Installation	C:\...\fssm32.exe C:\...\IEUpd.exe C:\...\IEXPLORE.hlp		
C2	202.abc.xyz.7 [HTTP request]		
Actions on Objectives	N/A	N/A	N/A

Figure 2.4: Case study of three intrusion attempts experienced by Lockheed Martin Computer Incident Response Team from [10]. The attack vector of the intrusion was targeted malicious email. Some indicators are shown for each phase. One overlapping indicator links together the three attempts, leading the researchers to label this as a campaign.

2.3. Kill Chain related work

At the time of writing this section, Google Scholar counts 373 citations for Lockheed Martin's whitepaper [10]. Of those, over 50% are between 2016 and 2017, but the paper was published in 2011. This citation trend shows that the article was probably quite ahead of its time. Since its publication, the work of Hutchins et al. has been broadly discussed, criticised and extended by the cybersecurity scientific community. Figure 2.5 shows the distribution of citations over time according to Google Scholar.

Dan Guido's "A case study of Intelligence-Driven Defense"[9] is the first work that used and modified the concept of the kill chain for threat intelligence. It was published in 2011. Guido's work is an engaging study because it uses the kill chain idea to gain knowledge on the economics of cybercrime and it unveils the myth of complexity around mass malware campaigns. The author adapted Lockheed

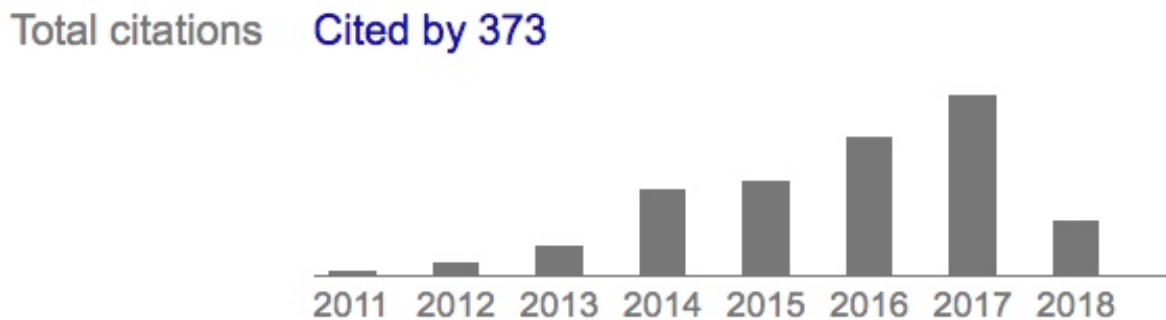


Figure 2.5: Google Scholar citation count for “Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains”[10] on June 20th 2018

Martin intrusion kill chain to the realm of mass malware; his modified chain has the steps: gain exposure, acquire capabilities, establish a delivery network, exploit the target, install malware, establish a command and control, perform actions on objectives.

This kill chain is more abstract than Lockheed Martin’s, and thus less valuable if the goal is mapping threat behaviour to countermeasures planning and deployment. Guido’s goal with this work is mainly to do data-driven threat intelligence on mass malware cybercrime; improved defence is secondary.

Guido’s work is more interesting for his in-depth study of exploit kits than for the adapted kill chain that he developed. He focused on the top 15 exploit kits of the first quarter of 2011 by popularity. His findings are that the distribution of malicious URLs hosting these exploit kits is exponential: the top four kits were responsible for 85% of the volume of malicious URLs. Furthermore, he collected the source code of the exploits and mapped each kit to the vulnerabilities exploited. Guido found that the kits developed over the past two years implemented only 27 exploits and targeted only five applications: Microsoft Internet Explorer, Adobe Reader, Adobe Flash, Oracle Java, Apple QuickTime [9]. This fact is striking, as more than 8000 vulnerabilities were added at MITRE Common Vulnerability and Exposure (CVE) project in 2010 alone. In other words, Guido found out that the most exploit kits target a minimal set of vulnerabilities and focus on most commonly used applications. This finding makes perfect economic sense, as mass malware authors want to hit as many targets as possible minimising costs of development. The insight here from the defensive point of view is that by patching against those few vulnerabilities has a considerable impact on increasing resilience. The myth of complexity behind mass malware is further unveiled when Guido shows the entirety of the 27 exploits code comes from two freely available sources:

- publicly disclosed targeted attacks by APTs, including vulnerabilities and possibly even implemented exploits;
- 0-days vulnerabilities responsibly disclosed by security researchers, possibly with implemented exploits;

The exploits that corrupted memory were developed predominantly by advanced persistent threats and used in targeted attacks. After this code became public, crime pack authors copied it verbatim and used it for mass attacks [9]. The exploits that abused logic flaws originated entirely from security researcher disclosures. A single researcher had disclosed three of the eight flaws [9].

In most cases, the attackers copied the exploit code verbatim. In other words, the authors of mass malware exploit kits rely almost entirely on others’ work. The most interesting conclusion of this work is that the kill chain model can be successfully extended from APT’s activity to untargeted mass malware, as authors of exploit kits recycle partially/wholly APTs modus operandi and tools. A very recent example of this is the WannaCry ransomware campaign of May 2017. The United States National Security Agency discovered a vulnerability of the SMBv1 protocol, CVE-2017-0144 ², that allowed for remote code execution. The NSA did not disclose the vulnerability and developed an exploit known as

²<https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0144>

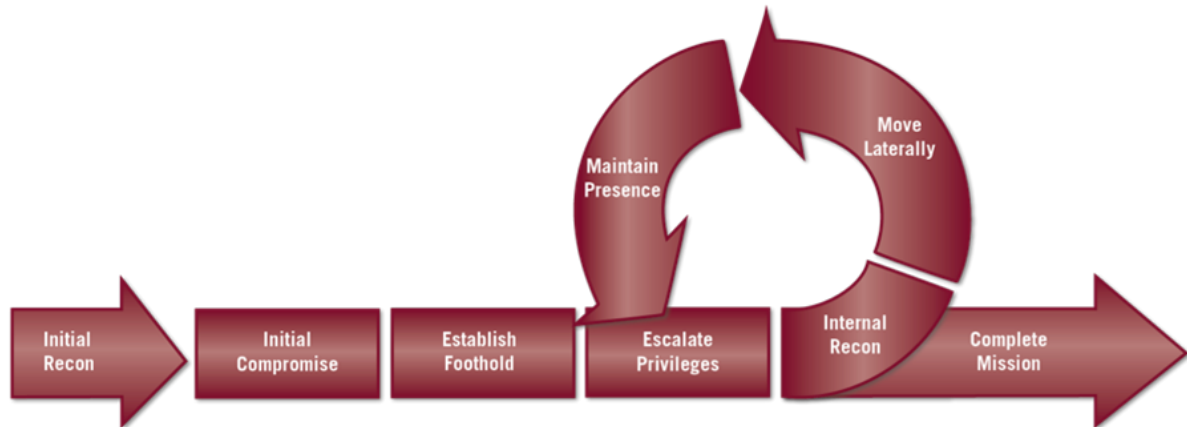


Figure 2.6: Mandiant Attack Lifecycle model

EternalBlue³. The exploit was leaked with others by the Shadow Brokers hacker group on April 14, 2017⁴. On May 12, 2017, the WannaCry ransomware campaign started, and it infected over 230000 computers in more than 150 countries within a day⁵. Wannacry used the EternalBlue exploit to infect other computers on the same local network over SMB. EternalBlue was also used for the NotPetya ransomware campaign that took place on June 27, 2017.

In 2013 the American cybersecurity firm Mandiant released a report on the activity of the advanced persistent threat known as APT1⁶. Mandiant did an impressive study of techniques, targets and motives of the unit by observing the operations of APT1 since 2010. Researchers at the company likely identified the unit as the 2nd Bureau of the Chinese People's Liberation Army General Staff Department's 3rd Department also known as Unit 61398. Little information is officially available on Unit 61398 and Mandiant makes the case that it is a cyber warfare division of the Chinese Army. In this work, Mandiant presents the multi-stage attack model that is used internally to study APTs like APT1: the Mandiant Attack Lifecycle. Mandiant Attack Lifecycle has eight phases: initial recon, initial compromise, establish a foothold, escalate privileges, internal recon, move laterally, maintain presence, complete mission. A diagram of the Mandiant Attack Lifecycle is showed at Figure 2.6. Escalate privileges, internal recon, move laterally and maintain presence are iterative and can be repeated until the mission is completed. The cyclic nature of Mandiant kill chain is a good representation of what happens in reality, where attackers generally have to do quite some lateral movement from the first point of entry to find valuable information on other machines. The attack lifecycle was developed with the mindset of the digital forensics professional; unsurprisingly it is used by Mandiant only after successful exfiltration have been discovered.

Mandiant model may describe APTs behaviour better than Lockheed Martin original kill chain by adding a cyclic/iterative component, but it does not seem to be leveraged in practice for data science at least in this report. Indeed, at least in this report, Mandiant does use their attack lifecycle as a reliable tool for mapping intrusions, comparing indicators, improving risk management and countermeasures with analysis and synthesis but instead merely as a framework to describe APT activity more understandable.

Bhatt et al. designed and implemented a "security event collection and analysis system" following the original kill chain model in "Towards a framework to detect multi-stage advanced persistent threats attacks"[2]. The system is developed using Apache Hadoop technology and uses system logs as data. The process is described as semi-automatic because the detection and centralised logging is automatic, but the correlation process still requires the command of an experienced analyst. Furthermore,

³<https://en.wikipedia.org/wiki/EternalBlue>

⁴<https://arstechnica.com/information-technology/2017/04/nsa-leaking-shadow-brokers-just-dumped-its-most-damaging-release-yet/>

⁵<http://www.bbc.com/news/world-europe-39907965>

⁶<https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf>

Bhatt et al. define the automatic event correlation to detect APTs as a research issue. The tool developed has various modules, among which one of high interest for this research: the intelligence module for the log analysis process and the malware analysis module. The former uses a predefined table called the Defense Plan. The table links intrusion kill chain phases to specific attack mechanism. The list of attack mechanism was extracted from CAPEC⁷ list from the non-profit MITRE corporation⁸. The researchers tested an implementation without the malware analysis module and therefore were not able to test realistically kill chain phases after delivery. Furthermore, the intrusion synthesis module (for automatic correlation of kill chains into campaigns) is also missing and was replaced by manual work by a security analyst. This short paper is a practical application of Lockheed Martin kill chain model and although the implementation lacks several critical elements the overall design is quite interesting. The main research issue, automatic correlation of kill chains in campaigns, remains open four years after publication.

In 2015 Mitre developed its framework for threat assessment: ATT&CK⁹. The Cyber Attack Lifecycle, yet another revised kill chain, is one of the elements of ATT&CK and has the phases recon, weaponise, deliver, exploit, control, execute, maintain. ATT&CK is described as a threat data-informed adversary model, and it focuses on the post-exploit phases of the cyber attack lifecycle. Control, execute and maintain are decomposed in nine sub-phases: persistence, privilege escalation, defense evasion, credential access, host enumeration, lateral movement, execution, c2, exfiltration. The deconstruction is similar to what done in the Bryant kill chain [3] and has the same goal: to allow a practical mapping of alerts and indicators to a multi-staged threat model. For each sub-phase, the techniques available to adversaries are listed in a Tactics and Techniques table. Techniques may fall into one or more sub-phases. Finally, each technique is labelled as detect, partially detect or no detect to show gaps in current cybersecurity countermeasures. Figure 2.7 shows an example of what is called the Notional Defense Gap table. This framework does not use the kill chain idea for enhanced threat intelligence, but rather it is used for cyber risk management. Using a variation of the kill chain model for risk management is not groundbreaking, as Hutchins et al. already did mention that the model could be used for that, but it is one of the few frameworks actually developed to do so.

Rutherford et al. [15] came up with an improved cybersecurity kill chain in “Using an Improved Cybersecurity Kill Chain to develop an Improved Honey Community” with the following phases: reconnaissance, weapon selection, delivery, exploitation, intelligence gathering, objective execution and exfiltration of information. Interestingly, this kill chain features bidirectional attacker movement between the last three steps. Whereas Lockheed work seems to give equal importance to each step, Rutherford’s work explicitly states the importance (for the defender) of the intelligence gathering phase (it replaces the installation phase). This phase is here broader, and the rationale behind is similar to that of Bryant kill chain [3].

The most significant change to the original kill chain introduced in this paper is the introduction of bilateral arrows among the last phases. Non-linear kill chain navigation is not a new idea though, as seen in Mandiant Attack Lifecycle from 2013. Like in Mandiant case, Rutherford et al. do not show the practical use of this kill chain, so the complications of introducing movement in the kill chain are ignored.

“A novel kill chain framework for remote security log analysis with SIEM software” by Bryant and Saiedian [3] is one of the latest (2017) and most inspirational interpretation at the kill chain concept. In this work, the authors develop a new kill chain framework centred on the use case of the internal security analyst of an organisation. The new model leads to improved forensics and incident response but does not leverage the kill chain concept for improved threat intelligence and defences.

Security analysts often have to deal with incomplete or missing data when dealing with incident investigation. Alerts are generated by a heterogeneous variety of sensors and countermeasures (IDS, firewalls, antiviruses, etc...), and even one single intrusion can trigger an overwhelming and disorganised amount of alarms if proper automatic clustering is not in place.

⁷Common Attack Pattern Enumeration and Classification (CAPEC) is a list of software weaknesses curated by the non-profit MITRE Corporation

⁸<https://capec.mitre.org/index.html>

⁹<https://www.mitre.org/sites/default/files/publications/pr-15-1288-adversarial-tactics-techniques-common-knowledge-presentation.pdf>

Notional Defense Gaps

Persistence	Privilege Escalation	Defense Evasion	Credential Access	Host Enumeration	Lateral Movement	Execution	C2	Exfiltration
Legitimate Credentials			Credential Dumping	Account enumeration	Application deployment software	Command Line	Commonly used port	Automated or scripted exfiltration
Accessibility Features		Binary Padding	Credentials in Files	File system enumeration	Exploitation of	File Access	Comm through removable media	Data compressed
AddMonitor		DLL Side-Loading	Network Sniffing	Group permission enumeration	Vulnerability Logon scripts	PowerShell	Custom application layer	encrypted Data size limits
DLL Search Order Hijack		Disabling Security Tools	User Interaction	Local network connection enumeration	Pass the hash	Registry	protocol	Data staged
Edit Default File Handlers		File System Logical Offsets	Credential manipulation	Local networking enumeration	Pass the ticket	Rundll32	Custom encryption cipher	Exfil over alternate channel to C2 network
New Service		Process Hollowing		Operating system enumeration	Peer connections	Scheduled Task	Service Manipulation	Exfil over physical medium
Path Interception		Rootkit		Owner/User enumeration	Remote Desktop Protocol	Third Party Software	Data obfuscation	Exfil over network medium
Scheduled Task				Process enumeration	Windows management instrumentation		Fallback channels	Exfil over physical medium
Service File Permission Weakness				Security software enumeration	Windows remote management		comm Multilayer encryption Peer connections	From local system
Shortcut Modification				Service enumeration	Remote Services Replication through removable media		Standard app layer protocol	From network resource
Web shell				Window enumeration	Shared webroot		Standard non-app layer protocol	From removable media
BIOS	Bypass UAC				Taint shared content		Standard encryption cipher	Scheduled transfer
Hypervisor Rootkit	DLL Injection	Indicator blocking on host			Windows admin shares		Uncommonly used port	
Logon Scripts	Exploitation of Vulnerability	Indicator removal from tools						
Master Boot Record		Indicator removal from host						
Mod. Exist'g Service		Masquerading						
Registry Run Keys		NTFS						
Serv. Reg. Perm. Weakness		Extended Attributes						
Windows Mgmt Instr. Event Subsc.		Obfuscated Payload						
Winlogon Helper DLL		Rundll32						
		Scripting Software Packing						
		Timestomp						

Detect
Partially Detect
No Detect

Approved for Public Release; Distribution Unlimited. Case Number 15-1288




Figure 2.7: Example of Notional Defense Gap table produced by MITRE's ATT&CK framework

Bryant and Saiedian explain that the state of the art in multi-staged attack modelling is not addressing the issues of security analysts adequately: although a step in the right direction, actually implementing and using established models is described as cumbersome and hardly successful if not in toy scenarios. The main issue with Lockheed Martin kill chain and Mandiant Attack Lifecycle is their high level of abstraction. Phases are described vaguely, leading to subjective choices by the analyst in labelling indicators. This vagueness, even if not critical in small investigations, points to potential inconsistencies when more analysts work together on large cases or when organisations want to collaborate in cyber threat intelligence sharing.

Furthermore, the phases do not always align precisely with the countermeasures that generate alarms with additional confusion in labelling indicators. The first two phases of Lockheed Martin kill chain (Reconnaissance and Weaponisation) extend beyond the scope of a single organisation network and may require data unavailable to internal security teams. Vulnerabilities at these early phases may be technical as well as non-technical (e.g. job posting on LinkedIn looking for professionals trained on specific systems gives away information on the systems used internally). In general, vulnerabilities that don't produce data in either way (out of scope or non-technical) are especially hard to use in a practical, easy to implement, way during an investigation. Concluding, the researchers agree that multi-stage attack modelling is the right idea but show that the state of the art models fail at implementation because they are too abstract and high level.

The authors present an improved multi-stage attack model called the Bryant kill chain. It is not just another abstract tool to map APT's intrusion activities but an entire methodology to systematically handle complex incidents in an organisation with countermeasures deployed at multiple levels. The Bryant kill chain has four phases and seven sub-phases:

1. Network phase: Reconnaissance (Pre hack)
2. Network phase: Delivery (Pre hack)
3. Endpoint phase: Installation (Hack)
4. Endpoint phase: Privilege Escalation (Hack)
5. Domain phase: Lateral Movement (Compromise)
6. Domain phase: Actions on Objective (Compromise)
7. Egress phase: Exfiltration (Theft)

The seven sub-phases are further deconstructed in thirteen tasks that need to be executed progressively. The tasks are: (Reconnaissance) probing and enumeration, (Delivery) host access and network delivery, (Installation) host delivery and software modification, (Privilege Escalation) privilege escalation and privilege use, (Lateral Movement) internal reconnaissance and lateral movement, (Action on Objective) data manipulation and obfuscation, (Exfiltration) external data transfer. The tasks are mapped to the following sensors: perimeter firewalls, network IDSs, domain controller logs, endpoint monitoring on the operating systems, host-based anti-malware programs, edge routers, network load balancers, remote access points. These sensors generate either logs (endpoint operating systems, firewall, etc...) or alerts (IDS, host-based anti-malware, etc...). Finally, tasks are mapped to indicators, features that can be extracted from the generated logs or alert. For example, in the case of probing the available indicators are port scans, default credential attempts, blacklisted traffic and the two specific Windows Event IDs 4625 and 5157 (respectively "Failed Logon" and "Firewall Deny"). A figure of the deconstructed endpoint phase is provided in Figure 2.9 to have a better idea of how phases, tasks and indicators overlap.

The Bryant kill chain is designed to facilitate logical data aggregation. This is essential to link alerts from several sensors or phases into a single kill chain. Specifically, the model has a specific indicator as the aggregate field for each sub-phase and then uses this aggregate field to link more phases together. For example, in Reconnaissance the aggregate field is the Source IP address. This source IP address is present also in the Delivery indicators, and therefore the two phases can be linked together. Continuing through the kill chain, the Destination IP indicator of the Delivery phase can be mapped to the Computer Name at Installation phase, and so on. This methodology is the same of Lockheed Martin's kill chain. What is innovative is that this is the first time a practically deployable framework is described in detail

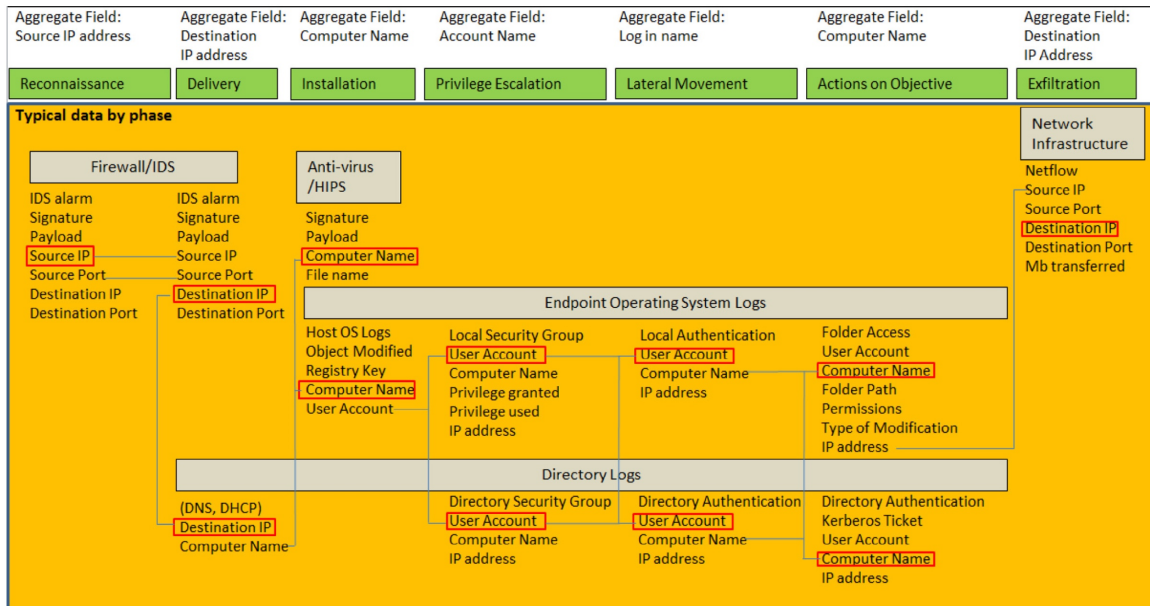


Figure 2.8: Indicator relationship between phases in Bryant kill chain model [3]. Field used for data aggregation among phases is in a red box.

for network intrusion. Whereas the original paper was an abstract description of a methodology, this paper provides a working implementation. Figure 2.8 is provided for added clarity.

The authors deployed and tested the framework in conjunction with a SIEM¹⁰. The model proved effective in reducing the number of false positives and negatives alerts, in reducing the overall number of alerts by aggregating them and in providing a standard methodology for incident investigation. The work of Bryant and Saiedian provides a remarkable evolution of the multi-stage attack model idea. The proposed model is scalable, efficient and easy to use in practice. Less abstraction leads to less subjectivity when creating kill chains and therefore to improved consistency. The Bryant kill chain works well regardless of the analyst and allows organisations to share data in the same format. Furthermore, the linkage between phases by using aggregate indicator is a simple and very effective implementation. This approach is not exempt from shortcomings, because a less abstract framework comes at the cost of reduced adaptability to unusual attack patterns. Furthermore, it is true that logs and sensor alarms can easily provide attack data, but at the same time it is not addressed how indicators are extracted in vastly common scenarios like attacks that use emails as the attack vector. As we have previously stated, the majority of attacks are The fundamental concepts of kill chain analysis and synthesis are ignored, indicators of phases that are not triggering an alert are not leveraged during incident investigation or afterwards to improve threat intelligence and defensive posture. Likewise, campaign analysis is also not mentioned. The reason for this lies in the primary goal of the authors: improving the workflow of security analysts and forensics professionals with multi-stage attack models. Bryant and Saiedian achieved this goal by describing in detail a practical, non-abstract, methodology for these stakeholders, but sadly throughout the work there is no mention of threat intelligence or leveraging indicators for improved defence and risk management. These were the main goals of Hutchins et al. for Lockheed Martin.

¹⁰SIEM: security information and event management. Software to visualise and handle security incidents in an organisation in a centralised manner

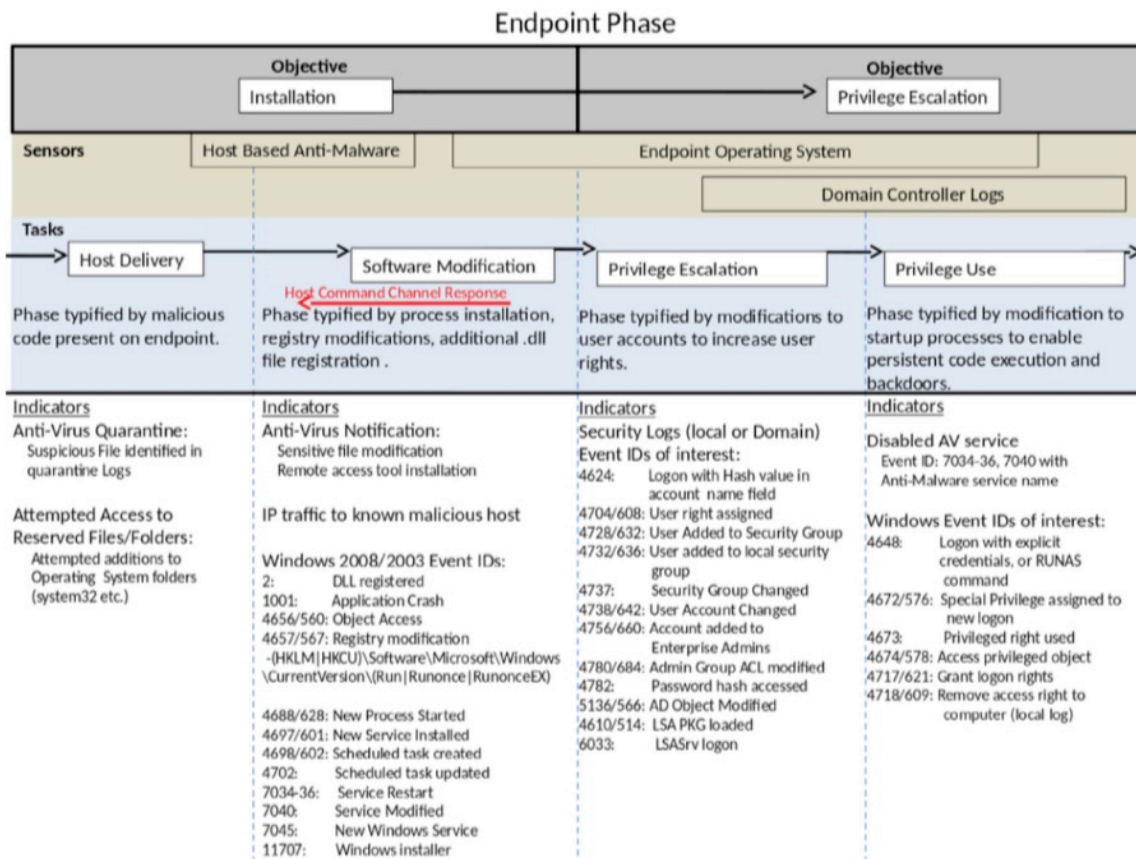


Figure 2.9: Deconstructed endpoint phase from Bryant kill chain [3]. This table shows clearly how sub phases, sensors, tasks and indicators overlap logically.

2.4. Research gaps

In the previous sections, the original kill chain paper from Lockheed Martin[10] was presented together with the most relevant related work that followed its publication in 2011. We want to highlight two research gaps in the literature: the lack of automation and the increasingly anachronistic nature of exclusively exact match of indicators in campaign analysis.

Campaign analysis is a process rarely applied in practice, possibly due to the lack of a readily applicable and adaptable framework as well as for the lack of automation. Campaign analysis can be decomposed into three sub-processes:

1. Detection of intrusion: automated with HIDS and NIDS technologies.
2. Killchain feature extraction from intrusion data: somewhat automated in the form of malware analysis tools and threat intelligence sources (e.g. Remnux, Virustotal, Cuckoo).
3. Correlation of kill chains in campaigns by looking for overlapping indicators: not automated.

Without automation, campaign analysis will remain an activity performed only by large organisations capable of spending many resources in cybersecurity staff to manually look for overlapping indicators.

In all the kill chain models presented in section 2, campaign analysis was performed exclusively by searching for overlapping indicators that matched exactly. This approach is very simple, and it works well enough in practice, especially with highly specific indicators like hashes, vulnerabilities CVE identifiers or IP addresses. For other more “human oriented” indicators like email subjects or filenames, the exact match may not be the most optimal strategy. Changing one single character in an indicator will make exact match fail, and an attacker can easily automate the generation of always new subjects and filenames, easily circumventing campaign analysis. Furthermore, this makes countermeasures like static filtering based on databases of blacklisted indicators ineffective. What we see in practice, on the other hand, is that often the automatic generation of new indicators tends to follow some form of a pattern, possibly capturable by approximate matching. We believe approximate matching of some indicators could lead to not only more effective cyber defences but also to more exciting discoveries (at the inevitable price of a component of false positives).

3

Methodology

In section 2.2 we thoroughly explained the concept of cyber kill chain and campaign analysis developed by Lockheed Martin in 2011. The kill chain is a model that maps an attack to phases in an adversarial setting. In this model the attacker needs to go through each phase consecutively to reach his objective: a disruption of the chain at any phase will cause the attack to fail. Aside from phases, the other characteristic atomic item of the kill chain model is the concept of **indicator** and its use. Indicators are any feature belonging to the data used or produced by the attacker. The usage of a specific exploit is an indicator, an IP address is an indicator, an email address is an indicator and so on.

A **campaign** is a repeated series of intrusion attempts, or in other words a collection of related kill chains. Lockheed Martin researchers observed that in practice malicious actors, even sophisticated ones, tend to recycle to some extent at least a parts of their toolchain. In other words, separate intrusion attempts can be linked together (in a campaign) if specific-enough indicators are found to occur across them. When intrusions are targeted, frequent and persistent the volume of collectable indicators augments, making it easier to spot patterns. The factors that lead attackers to recycle parts of their kill chains are economical, threat-oriented and characteristic of the human nature.

Cybercriminals and APTs, no matter how large or advanced, have limited resources like any other organisation. These resources are money, time, workers at disposal. When a criminal group invests in an expensive vulnerability exploit acquired on the black market, they are incentivised to **maximize the return over the investment** by using it more often rather than less often. Similarly, when a state-sponsored APT spends months developing a highly advanced malware that exploits a zero-day vulnerability, he will likely use it in more than one single attack.

Like many other highly specific professions, APTs hackers tend to develop a particular style, or modus operandi, in the way they perform their attacks. For example, the way they decide to move laterally in a network or to pick their targets and the techniques they use to socially engineer them are all influenced by their experiences, education, culture, upbringing. The modus operandi includes not only the techniques and decision making but also the tools used. Some of these have to be developed internally and are then shared by actors in the same collective but are not seen in other groups (similarly to conventional weapons for governments, state-sponsored APTs do not like to share their tools with the enemy).

Moreover, these individuals and groups may actually not care about concealing their identity at all (for example if they are protected by their country as long as they do hack organizations exclusively abroad, like Russia allegedly does ^{1 2}) or value this less than reaching the objective of their campaign.

Finally, hackers are humans, and humans make mistakes. Sloppiness, laziness and poor choice making can lead actors to recycle old indicators (e.g. usernames, passwords, emails).

Unfortunately, the models found in Literature to perform campaign analysis rely heavily on manual work by an experienced security analyst, leading to prohibitive costs and complexity. Furthermore, we also highlight the issue of the increasing inadequacy of exact-only match of indicators.

The goal of this thesis project is to explore and implement a solution to significantly lower the cost of performing this process.

¹<https://slate.com/technology/2018/02/why-the-russian-government-turns-a-blind-eye-to-cybercriminals.html>

²<https://www.justice.gov/opa/pr/us-charges-russian-fsb-officers-and-their-criminal-conspirators-hacking-yahoo-and-millions>

In this chapter, we provide a detailed description of the methodology adopted. In section 3.1 we first explore two hypotheses on how to bridge the research gap mentioned and thus we formulate our research questions and working items. Then, in sections 3.2 and 3.3 we study the hypotheses domains in depth. In section 3.4 we describe the data used to build and test our solution. Finally, in section 3.5 we present in detail our implemented solution.

3.1. Hypotheses and research questions

The ultimate research goal of this thesis project is to improve the state of the art in campaign analysis. By studying the state of the art, we have highlighted two shortcomings of the models in the literature. The first research gap is the following:

Research Gap 1 *Campaign labelling is a heavily manual, not automated, task and it is thus often prohibitively expensive to perform in practice.*

Campaign analysis is a process where separate incidents are grouped when relevant indicators overlap exactly. We can, therefore, say that campaign analysis is a task where unlabelled data (kill chains) are assigned a label (campaign). Performing this job manually becomes infeasible when the dataset or its dimensionality are large. We have therefore stated that we want to improve the state of the art by introducing automation. Automating the process of labelling unlabelled categorical data is a very well known machine learning problem called classification. In this specific scenario, we do not know a priori the number of categories (the number of campaigns). Furthermore, we want to develop a solution that does not require a training set of existing campaigns. We need to automate the discovery of *new* campaigns together with the labelling of old campaigns. In machine learning performing classification without a training set is called clustering or cluster analysis. Now that we understood that the problem at hand can be reduced to a form of well-known machine learning task, we formalise it as our first hypothesis and thus come up with a related research question:

Hypothesis 1 *Automating campaign analysis is a form of unsupervised classification (cluster analysis) task.*

Research Question 1 *Can campaign analysis be performed with an existent clustering technique?*

To perform unsupervised clustering analysis, it is necessary to use a measure of similarity to assess if two data points belong to the same cluster. In principle, more similar data points should be closer and thus more likely to end up in the same cluster whereas more different data points should be farther and thus less likely to cluster together. We will therefore also need to find a suitable measure of distance. In section 3.2 we will dive deeper into the domain of clustering in search of a suitable technique to perform it in our scenario.

The second research gap identified in the related work is the following:

Research Gap 2 *Kill chains are grouped in campaigns only if indicators match exactly*

In Lockheed Martin's campaign analysis two intrusion attempts are grouped when one or more indicators belonging to their reciprocal list of indicators match *exactly*. Indicators are any feature belonging to an intrusion attempt, for example, the hash of the malicious payload, the CVE code of the vulnerability exploited, the email address of the sender of the malicious email, the IP that the malware contacts after successful installation. Although exact match of indicators surely performs well regarding precision, it can hardly keep up with the increasingly popular trend of **obfuscation**. To cope with increasingly effective forms of filtering, malware creators often add a component of randomness to indicators like filenames, subjects and email addresses. This technique makes the exact match of indicators obsolete, even though it was not developed for the precise reason of bypassing campaign analysis. We, therefore, formulate a second hypothesis and two related research question:

Hypothesis 2 *Exact only indicator match likely does not perform optimally (in terms of recall^a). A form of approximate match could improve the performance of the model for some indicators.*

^ain this context, the recall is maximum when the number of false negatives is zero (items that actually belong to a campaign but are not labelled as such by the system)

Research Question 2 *Does approximate match of suitable indicators improve recall performance of campaign analysis?*

Research Question 3 *Can a methodology be developed to assess what indicators are most suitable (and what are not) for approximate match?*

In section 3.3 we explore the domain of similarity of information theory, looking for a suitable technique to perform approximate string matching and in section 3.4.4 we develop a methodology to understand what indicators are most suitable for this technique.

3.2. Automation: clustering

The research problem we are dealing with is to automate the process of linking related malicious emails, by following the idea of Campaign Analysis introduced in Lockheed Martin kill chain model[10]. As mentioned in the previous section, this is a form of machine learning task called classification. Specifically, because we want to perform it without a training set, we are dealing with a clustering problem. Indeed, we want to build a system capable of performing unsupervised learning: we will feed it unlabelled kill chains, and we want it to output their labels (the campaigns to which they belong).

In general, the goal of clustering is to separate an unlabeled data set into a finite and discrete set of “natural” hidden data structures [21]. Although many forms of clustering exist, the most prominent and related to the problem at hand are the following:

- Connectivity models: Hierarchical Clustering
- Centroid based clustering
- Density based clustering

Hierarchical clustering (HC) algorithms organise data into a hierarchical structure according to a measure of proximity (also known as connectivity). The results of HC are usually depicted by a binary tree or dendrogram. The root node of the dendrogram represents the whole data set, and each leaf node is regarded as a data object. The intermediate nodes, thus, describe the extent that the objects are proximal to each other; and the height of the dendrogram usually expresses the distance between each pair of objects or clusters, or an object and a cluster. The ultimate clustering results can be obtained by cutting the dendrogram at different levels [21]. HC algorithms are mainly classified as agglomerative methods and divisive methods. Agglomerative clustering starts with clusters, and each of them includes precisely one object. A series of merge operations are then followed out that finally lead all objects to the same group. Divisive clustering proceeds in an opposite way [21]. A clear advantage of HC is that no a priori knowledge is required on the number of clusters, or categories. The common criticism for classical HC algorithms is that they lack robustness and are, hence, sensitive to noise and outliers [21]. A frequent phenomenon is chaining, or the production of a few large clusters, which are formed by virtue of nearness of points to be merged at different steps to opposite edges of a cluster[20].

In contrast to hierarchical clustering, which yields a successive level of clusters by iterative fusions or divisions, **centroid based or partitional clustering** assigns a set of objects into clusters with no hierarchical structure [21]. Here, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k , k -means clustering gives a formal definition as an optimisation problem: find the k cluster centres and assign the objects to the nearest cluster centre, such that the squared distances from the cluster are minimised³. Although it is theoretically possible to guess the number of k partitions in the data set, this can become increasingly hard as the size of the dataset increases. In some cases not knowing a priori the exact number of partitions or even the order of magnitude makes using these techniques unfeasible.

Density based clustering (DBC) is an evolution of hierarchical clustering. The idea is to similarly agglomerate or to divide the data points to find the final clusters, but instead of using a cut off value based on the distance of points, a more complex and granular measure is used: density.

In DBC, a cluster is a set of data objects spread in the data space over a contiguous region of high

³https://en.wikipedia.org/wiki/Cluster_analysis

	Parameters	Scalability on large datasets
Hierarchical	cut off distance	technically yes, but cut off value hard to assess on very large dataset
Centroid Based	number of clusters	yes
Density Based	density parameter	yes

Table 3.1: Comparison of three well known categories of clustering techniques

density of objects. Clusters, “dense” regions, are separated from each other by contiguous regions of low density of objects. Data objects located in low density regions are typically considered noise or outliers [12]. DBC requires that the density in a neighbourhood for an object should be high enough if it belongs to a cluster. DBC creates a new cluster from a data object by absorbing all objects in its neighbourhood. The neighbourhood needs to satisfy a user-specified density threshold [21]. The algorithm that decides if two points should be merged in a cluster or not is called linkage method. DBC is often used because of its efficiency on large datasets compared to other approaches and because no a priori knowledge of the number of clusters is required. Similarly to HC, DBC can suffer from chaining at a varying degree, depending on the linkage method used.

3.2.1. Selecting a suitable clustering method

In the previous section, we have discussed the constraints of our scenario. The most important is that there should be no requirement to know a priori the number of clusters. We need the clustering technique to automatically infer the correct number of campaigns present in the data, even if this number varies significantly. Furthermore, although our deliverable will be only a proof of concept, it is desirable that we use a technique efficient on large datasets and resistant to noise. Finally, we ideally would like to use a technique with a publicly available implementation. Table 3.1 shows that based on these constraints density-based clustering is the most suitable options among the categories we assessed. We now look in depth at two popular density-based clustering algorithms: DBSCAN and HDBSCAN.

3.2.2. Density based clustering algorithms: DBSCAN and HDBSCAN

DBSCAN⁴, Density-Based Spatial Clustering of Application with Noise, is the original algorithm that implements density-based clustering. It was created by Ester et al. in 1996 [7].

The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. Due to this rather generic view, clusters found by DBSCAN can be of any shape, and not exclusively convex. DBSCAN has two parameters: `min_samples` and `eps`. These define the concept of density. `min_samples` sets the minimum amount of neighbours necessary to form a cluster. `eps` is the threshold distance that is used to establish the neighbourhood relationship. Higher `min_samples` or lower `eps` indicate higher density necessary to form a cluster. The central component to the DBSCAN is the concept of core samples, which are samples that are in areas of high density. A cluster is, therefore, a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples). More formally, we define a core sample as being a sample in the dataset such that there exist `min_samples` other samples within a distance of `eps`, which are defined as neighbours of the core sample. This tells us that the core sample is in a dense area of the vector space. A cluster is a set of core samples that can be built by recursively taking a core sample, finding all of its neighbours that are core samples, finding all of their neighbours that are core samples, and so on. A cluster also has a set of non-core samples, which are samples that are neighbours of a core sample in the cluster but are not themselves core samples. Intuitively, these samples are on the fringes of a cluster. Any core sample is part of a cluster, by definition. Any sample that is not a core sample, and is at least `eps` in distance from any core sample, is considered an outlier by the algorithm. The one drawback of DBSCAN is that it works well only if clusters have a similar, constant, density. Because the `eps` parameter is fixed, the algorithm will not handle well scenarios where cluster density varies significantly.

HDBSCAN: Hierarchical Density-Based Spatial Clustering of Applications with Noise [4, 5] is based

⁴<http://scikit-learn.org/stable/modules/clustering.html#dbscan>

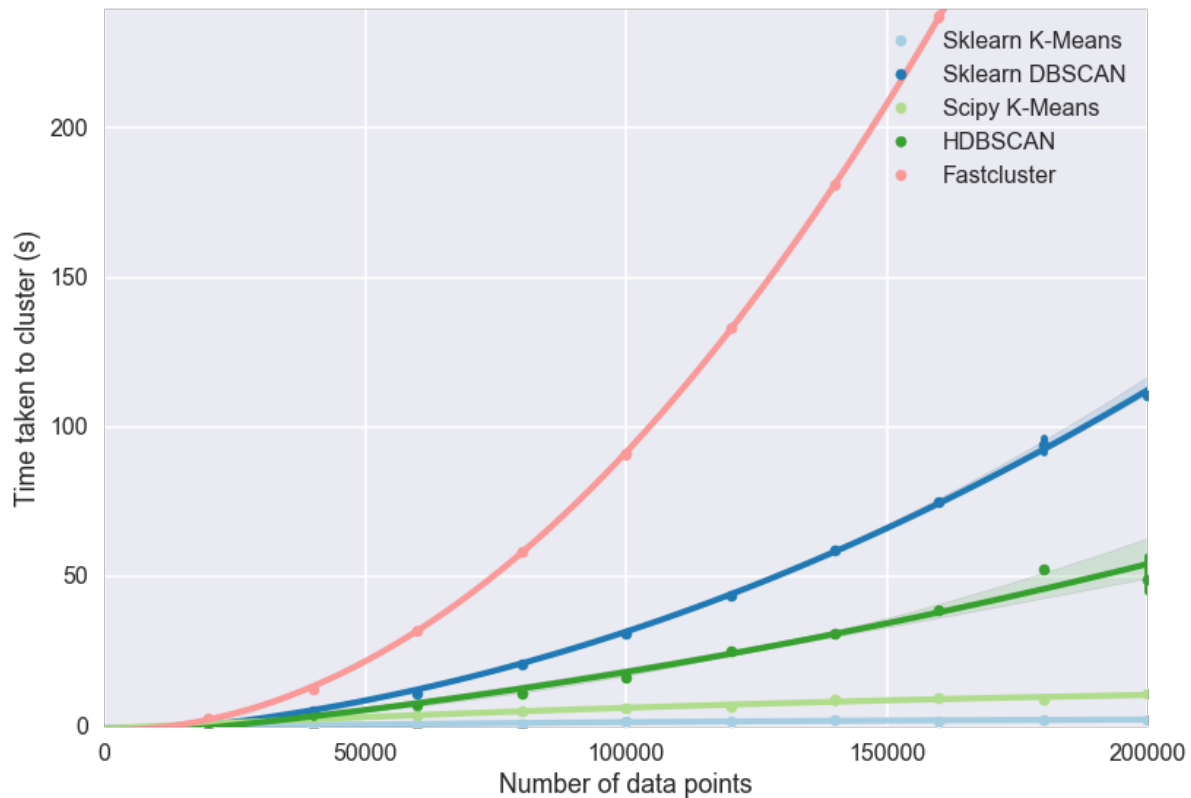


Figure 3.1: Performance comparison of several clustering techniques. HDBSCAN performs better than DBSCAN. Taken from HDBSCAN documentation.

on the aforementioned DBSCAN algorithm. HDBSCAN has just one very intuitive parameter: `min_cluster_size`. A cluster will be formed exclusively if this value is reached. This parameter is easy to set, whereas `eps` and `min_samples` are not very intuitive. The most significant advantage of HDBSCAN over DBSCAN is that it performs well on a dataset with variable density clusters. HDBSCAN recursively performs DBSCAN with varying `eps` parameters, and then outputs the most optimal configuration automatically based on a metric of stability of the clusters found. This is a significant improvement over DBSCAN, because not only it is easier to configure (DBSCAN requires to try different values of `eps` extensively manually to find the best one, whereas HDBSCAN single parameter is the minimum cluster size), but also because it allows for different `eps` in different regions of the data space. For instance, HDBSCAN is capable of identifying a cluster within another cluster, whereas DBSCAN is not capable of doing so and it will label everything as the same one. Furthermore, HDBSCAN implementation is also faster than DBSCAN, see Figure 3.1.

3.3. Fuzziness: approximate string match

In section 2.4 we highlighted that one aspect missing from related work in campaign analysis literature is fuzziness in indicator matching. To assess approximate matching, it is necessary to address first the concepts of similarity.

The concept of similarity in Information Theory has been studied since the 1960s, mainly in the domain of Probabilistic Record Linkage. It was based mainly on the seminal paper by Fellegi and Sunter “A Theory for Record Linkage”[8] In the 1990s the topic was explored practically and not just theoretically by a large number of researchers; most of today’s similarity algorithms and functions were developed in this period [6]. Lin [13] gives a formal definition of Similarity in Information Theory:

The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to describe fully what A and B are.

Similarity is an especially important core element of processes that use features comparison for higher level purposes, for example for Machine Learning.

As seen in the previous sections, features (or indicators) are the atomic element of existing kill chain models. No implementation seems to exist that uses systematic feature similarity methods with indicators in the related work on multi-stage threat modelling. This aspect is one of the novelties introduced in this thesis project. It was thus relevant to study the related literature on similarity methods to learn the best techniques for our features.

String metrics are used mainly in search engines and in systems that check spelling or auto correct queries. "A comparison of String Distance Metrics for Name-Matching Tasks"[6] by Cohen, Ravikumar and Fienberg provides an excellent reference point on the existing string metrics methods and how they perform at different tasks. Four families of methods are presented:

- Edit-distance like functions: (Levenstein distance, Jaro distance and others) in which distance between two strings a and b is the cost of the best sequence of edit operations that convert a into b (edit operations: character insertion, deletion, substitution etc...).
- Token-based distance function: two strings can be considered as multisets of substrings (or tokens or n -grams). Jaccard similarity, TF-IDF or cosine similarity and other methods fall in this category. The distance of two strings is computed as the difference between the two multisets. The tokens are also called n -gram based on the size n of the tokens in characters (bi-gram and tri-gram if $n = 2$ or $n = 3$); this size is the most important parameters of token-based distance functions.
- Hybrid distance functions: they combine edit and token based distance functions in two or more steps.
- "Blocking" or pruning methods: when matching large lists of strings it is often not computationally practical or useful to measure the distance between all pairs. To solve this problem strings can be grouped on some a priori variable and compare only groups that are interesting.

The Information Retrieval branch of Computer Science has produced countless methods for assessing document similarity (in this domain called relevance). The most pervasive application is in text search engines for the world wide web. These techniques usually involve creating a reverse index of a text document [11], usually giving different weight to different words based on how often they occur (the idea is to make vastly common words like preposition less relevant). A reverse index is a table that tracks the number of occurrences and location of every word in the document.

One of the most common technique is called term frequency over inverse document frequency or TF-IDF. It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modelling. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. It can be applied to words as well as to token, for example in query completion applications and keyboard auto-correction.

3.4. Experimental data: Dutchsec Spamtrap

To develop and test our improved campaign analysis system we need a dataset of kill chains. Looking for such a dataset, we started a collaboration with Dutchsec, a young cybersecurity company based in Utrecht that developed a spamtrap which collected almost half a million malicious emails. We found the dataset to be rich in features and mostly malicious, precisely what we needed. In this section we first address the role of emails and attachment in Lockheed Martin kill chain model. Indeed, our dataset is composed of emails, not kill chains, so some form of data enrichment will be necessary for our system to expand the former ones in the latter ones. Then we describe the dataset more in detail, and we also discuss the problem of malignity blindness that will inevitably affect our system. Finally, we showcase a methodology to decide which features are suitable for approximate indicator match.

3.4.1. Emails and attachments in the kill chain

Emails are one of the most widespread methods of communication in the world. Private and public organisations use emails for communication, either internally or with the outside Internet or both. According to the Email Statistics Report 2011-2015 released in 2011 by the Radicati Group “In 2016, the number of business and consumer emails sent and received per day totals over 215.3 billion, and is expected to grow at an average annual rate of 4.6% over the next four years, reaching over 257.7 billion by the end of 2020”⁵. The report estimates 2.76 billion unique email users by the end of 2017, or 37% of the Earth population (although the 7.442 billion figure used to compute the percentage includes sets of the population not in the working age like children and elderly). Emails are also a prominent way to share documents through attachments within companies as well as towards clients, partners and more.

Email popularity makes it a very appealing attack vector in cybersecurity. Verizon’s 2015 Data Breach Investigation Report ⁶ indicates that 77.3% of cyber espionage attacks studied used emails as the attack vector in malware installation. Respectively, 39.9% used email attachments and 37.4% used email links. 2016 version ⁷ of the same report states that “phishing remains a dominant cyber espionage attack vector” and that “the majority of phishing cases feature phishing as a means to install persistent malware”. Unsurprisingly, examples and case studies presented in the kill chain models literature often involve emails in the Delivery (or equivalent) phase [2, 3, 10]. Another reason to use emails is that indicators from almost the entirety of the kill chain phases can be extracted from just one email: recipient lists and email style for reconnaissance, email metadata for delivery, post exploitation phases from attached malware and many more. This is quite convenient as it is not necessary to deal with data coming from a vast number of sensors, a very common burden in previous kill chain research [3].

Although no related work focuses exclusively and explicitly on emails within the kill chain model, a lot of related work exists on email as attack vector and countermeasures for it. Most of the techniques proposed involve some form of feature extraction which is quite relevant for this research, regardless of the original motive (usually to train a classifier for anti-spam filters).

Broadly speaking, when dealing with malicious emails a somewhat clear separation can be done between low-quality mass spam and phishing campaigns and high-quality emails with a precise target. Although the line is blurring as the quality of phishing campaigns is improving to avoid current filtering methods, the diversity in motive is still evident: the former usually has an economic motive (scams, phishing⁸, advertisement) whereas the latter is more prominently used for espionage and data exfiltration. Well crafted malicious emails are known in the literature as Targeted Malicious Emails or TMEs. TME is often explicitly tailored on a particular victim, uses social engineering to convince the victim of its legitimacy and is used by APTs to carry spear phishing or reconnaissance activity.

Whereas mass spam can be considered an almost solved problem thanks to Machine Learning methods developed in the first decade of the 2000s [18, 19], TMEs are still an open research issue, as it is tough to separate them from non-malicious email automatically and with an acceptable level of false positives and negatives [1]. TMEs do not contain the usual giveaways of mass spam: spelling mistakes, bizarre content, recycled schemes that use specific keywords (sex improving drugs, Ponzi schemes, Nigerian prince etc...). TMEs is usually the kind of email observed in APT’s activity.

Amin [1] has published several works on identifying TME. In “Detecting Targeted Malicious email through classification of persistent threat and recipient-oriented features”[1] Amin developed a system to effectively identify TMEs out of a corpus composed of regular emails, mass spam and TMEs. Amin trained a supervised classifier using persistent threat features (threat actor locale, weaponisation tools...) and recipient-oriented features (victim email and other email header features). Specifically, a

⁵http://www.radicati.com/wp/wp-content/uploads/2016/01/Email_Statistics_Report_2016-2020_Executive_Summary.pdf

⁶http://www.isaca.org/chapters2/Luxembourg/Documents/201510%20Digital%20Forensics%20Master%20Class/5_DBIR%202015%20-%20CLUSIL-ISACA.pdf

⁷www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf

⁸The terms phishing, spear phishing and whaling are used quite often confusingly in the cybersecurity community. For clarity sake, throughout this paper the following definitions are used. Phishing: a non-tailored attack that involves an attempt to obtain sensitive information such as usernames, passwords, and credit card details (and money), often for malicious reasons, by disguising as a trustworthy entity in an electronic communication. It is typically carried out by email spoofing and it often directs users to enter personal information at a fake website, the look and feel of which are identical to the legitimate one and the only difference is the URL of the website in concern [17]. Spear phishing: tailored attack where the attacker sends the victim an email with malware attached or a link that leads to malware download. Whaling: spear phishing directed towards a very high profile target.

random forest classifier was used to separate TMEs from non-targeted malicious email (spam). The performance was measured regarding false negatives and compared to traditional email filtering techniques. Three are the main findings of this work:

- TME shows persistent threat features whereas mass spam does not (because the threat actor will repeatedly try to perform the attack);
- TME shows persistent recipient-oriented features whereas mass spam does not (this is straightforward as the former is targeted whereas the latter is non-targeted and on a mass scale);
- detection of TME through the combined use of persistent threat and recipient-oriented features results in significantly lower false negatives than detection of TME using conventional email filtering techniques (with an acceptable level of false positives).

Amin's work is critical as it provides a scientific confirmation to the hypothesis of Hutchins et al. [10] that features (or indicators) of APT email activity can be found persistently in what can be called a malicious campaign. He foresaw future work in using more features from other elements of emails like attachments and links, which is done in this thesis project.

In "Malicious PDF Detection using Metadata and Structural Features"[16] by Smutz and Stavrou built a random forest tree classifier to detect malicious PDF documents. This work further confirms that features extracted from metadata of malicious files can be used against their creators to perform enhanced threat intelligence and map campaigns. Interestingly, the authors mainly performed kill chain and campaign analysis similarly to what is showed in Hutchins et al. [10] (although limited to the weaponisation phase) without any mention or reference to the original kill chain or any other multi-stage threat model.

3.4.2. Dutchsec Spamtrap

At the beginning of 2016, Dutchsec built a spam trap to collect and study spam emails directed to well-known spam infested mail domains. A spam trap is a honeypot dedicated to malicious emails. Dutchsec registered a large number of domains that have been active in the past but not anymore, and then they changed the mail exchanger record (MX record) to point to their spamtrap. These domains mostly belonged to organisations based in the Netherlands, and there was no specific methodology on how they were selected. Dutchsec requested not to disclose the domains involved.

The dataset collected consists of 490649 emails (490649 MIME messages composed by 835075 MIME parts) collected in the period from February 2016 to May 2017. The entirety of the emails seems to be malicious, being either spam or a form of phishing. Of course, given the amount of data, it was impossible to check 100% emails for maliciousness. Aside from some test emails sent during the set up of the spam trap, no other was found to be non-malicious.

The dataset has several hundreds of features. The most substantial majority of these features is composed by information describing the emails themselves: all possibly imaginable email header fields (e.g. body and subject), information on the email routing through the internet, geolocation data, email metadata (e.g. size and date). Furthermore, other features describe the content and metadata of attachments (e.g. size, filename, hash and links). Aside from the emails of the spamtrap, Dutchsec also provided us with the raw attachments that we can execute in a malware lab to extract further indicators to fill our kill chains. The attachments collected are 79576 files, of which 38028 unique:

- 29834 javascript scripts (.js)
- 27860 zip archives (.zip)
- 14367 Microsoft Word documents (.doc, .docx)
- 4060 pdf documents (.pdf)
- 2840 Microsoft Excel documents (.xls, .xlsx)
- 79 Microsoft Windows executables (.exe)
- 17 Microsoft PowerPoint documents (.ppt, .pptx)
- 1520 others

Eventually, we selected 68 features to be included in our system. Some of these belonged to the spamtrap whereas some others were extracted/created when testing the attachments in the malware lab we later set up. These features can be seen in the Appendix at table A.1.

3.4.3. Malignity blindness and data suitability

DutchSec spamtrap contains spam and phishing emails. Is this a suitable data set to perform campaign analysis, a process developed by Lockheed Martin against APT carried attacks? According to literature, mass malware and phishing campaigns are becoming increasingly similar to APTs in the techniques employed [9]. Therefore the answer seems to be yes, it makes sense to apply campaign analysis to Dutchsec dataset of spam and phishing emails.

A more interesting problem is the following: does the campaign analysis methodology require malicious data to work? To answer this question, we need to understand precisely what it is performed in Lockheed Martin's kill chain whitepaper. In the examples provided in the document, emails are first flagged as malicious from the CERT team of Lockheed Martin, and only afterwards they are escalated to the Threat Intelligence team where Hutchins et al. perform campaign analysis on them by looking for overlapping indicators. Thus, campaign analysis itself does not flag emails as malicious or benign, and it is performed only on malicious emails. What would happen if the initial filtering step was to be skipped? The methodology of campaign analysis is quite data independent on its own. Emails are grouped because some indicators overlap. This could be performed flawlessly on a set of benign emails too. The difference would be that the campaigns found would be benign, something distinctly odd since campaigns are commonly intended to be malicious by definition.

The critical point here is that the methodology itself, the grouping by looking for overlapping meta-data, is independent of the data being malign, benign or mixed. We call this property **malignity blindness**, and our system will also be affected by it because we are not going to modify the manual process but rather only automate it. Because the system will automatically infer campaigns from the input data, it will be up to the analyst to either make sure that only malicious emails are used or to ignore benign campaigns after they are clustered. Once more we stress that the system we are developing must keep the human in the loop and it is designed as an auxiliary tool to the analyst: it is not intended to replace the human but rather empower it.

3.4.4. Selection of features suitable for approximate match

Hutchins et al.[10] approach when performing campaign analysis is to look for indicators overlapping exactly. Although exact match is most definitely a good choice, we believe that this can be improved by additionally performing approximate match on a selection of indicators. In section 3.3 we presented the state of the art in string comparison and in section 3.1 we formulated research question 3: Can a methodology be developed to assess what indicators are most suitable (and what are not) for approximate match? Because all the indicators we selected are strings, we will use techniques from the domain of approximate string matching to assess the similarity of suitable indicators.

To select which indicators are suitable for approximate string matching we adopted an approach based on statistical analysis and domain knowledge.

For some indicators, this was not necessary, as it was entirely evident that they are not suitable for approximate string matching. This is the case for example for highly specific indicators like hashes, payload sizes, IP addresses, location coordinates, YARA signature names, vulnerabilities exploited and many more. Looking for overlap in these indicators makes sense exclusively when the match is exact: changing just one bit of a file leads to a hash function at least 50% different from the original due to the Avalanche Effect. Similarly, changing one digit of a number can alter its meaning significantly. Likewise, a few indicators are suited for approximate string matching, for example, email subject, email body, email sender address, filename.

To further justify our decision we conducted a statistical analysis on the entropy of the indicators values. In section 3.5.5 we describe how we compute a distance matrix for every indicator. This distance is the cosine distance ($1 - \text{cosinesimilarity}$) between all vector representations of the indicator values.

We then computed for each of these distance matrices the average and the standard deviation. The standard deviation is a measure that is used to quantify the amount of variation or dispersion of a set of data values. A low standard deviation indicates that the data points tend to be close to the mean (also called the expected value) of the set, while a high standard deviation indicates that the data points are

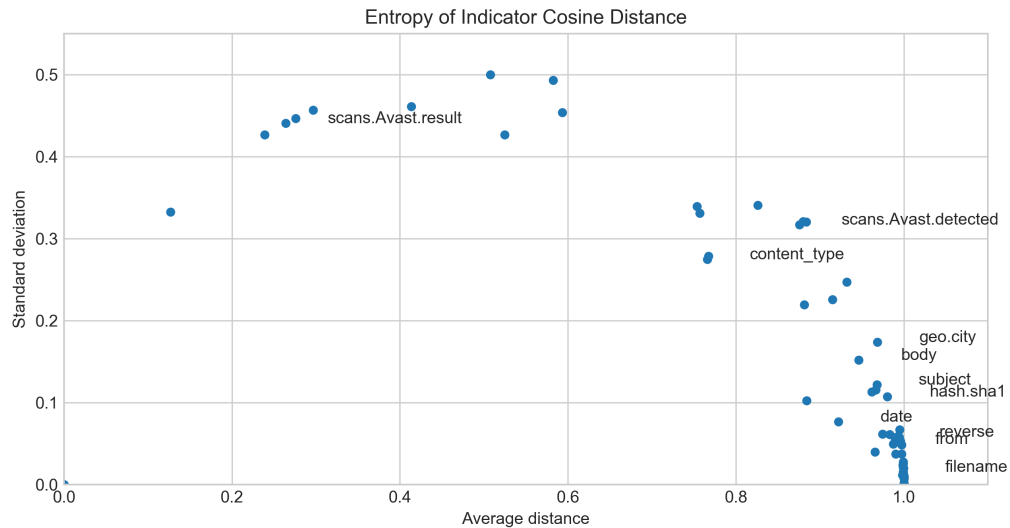


Figure 3.2: Distribution of entropy of cosine distance among indicators. Horizontal axis represents average distance, vertical axis represents standard deviation

spread out over a broader range of values ⁹.

The full table of average and standard deviation for the seventh week in the dataset, the 12th week of 2016, is in the Appendix at A.2. These values were found to be similar among all weeks.

Figure 3.2 shows the distribution of entropy in our indicators. The horizontal axis measures the average value of the cosine distance; the vertical axis measures the standard deviation from the average value.

Very high values of average cosine distance are to be expected in indicators with high entropy like hashes or human-oriented strings (e.g. subject or filename). On the other hand, intuitively, lower values of the average cosine distance imply less entropy. A lower amount of possible values showing up a high number of times will lower the average distance between all items. In other words, the average of the pairwise cosine distance among all values of an indicator gives an idea of the entropy or variability of these values. Indicators with a low number of highly repeated values (low entropy) have low average cosine distance, whereas indicators with a high number of rarely repeated values (high entropy) have a higher average cosine distance. The standard deviation gives us another measure of entropy by telling us how much of the distribution is close to the average.

Three areas can be observed in the figure. On the bottom right we can see a high number of features characterised by very high average cosine distance and very low standard deviation. This is the case for email body, subject, receiver address, sender address, filename, sha1, reverse DNS, dates, timestamps and other similar ones. This means that most values for these indicators are utterly different from most other values. These indicators carry highly specific information (dates, hashes, other numbers) or human-oriented information (subject, email addresses). On the top left, we can see another cluster of features characterised by low average cosine distance and medium standard deviation (around 0.5). This is the case for all antiviruses' result signatures and cuckoo signatures (e.g. if the malware is a dropper, trojan, ransomware or other). These can be highly specific strings containing detailed information on the payload. These indicators are used for general information and repeat a lot in the dataset. Finally, we observe a group of indicators between these two, with a relatively high average cosine distance (between 0.7 and 0.9) and a low standard deviation (around 0.25). Indicators in this area are most notably the email content-type header and the maliciousness score of the antiviruses (a boolean True/False). A medium entropy characterises these indicators compared to the other two groups. They tend to have a medium fixed set of possible values. It is sufficiently large to raise the average distance in this range, but not quite large enough to push it to the first group of highly specific and rare indicators. The content type of the email header is typically a string indicating the file format of the following MIME part plus if present the filename attached. The presence of the filename helps

⁹https://en.wikipedia.org/wiki/Standard_deviation

#	Indicator	Source	Info
15	exiftool.ZipFileName	VirusTotal	Name of zip archive, if present
20	body	SpamTrap	MIME body of email
25	filename	Cuckoo	Name of extracted file if archive, otherwise name of attachment
27	file.name	SpamTrap	Name of attachment
29	from	SpamTrap	MIME from header of email
36	headers.Return-Path	SpamTrap	MIME Return-Path, may differ from "from" header
52	reverse dns	SpamTrap	Retrieved by spamtrap
64	subject	SpamTrap	MIME subject of email

Table 3.2: Features selected for approximate string matching

explaining why this indicator lays between the two predominant extreme groups in the figure.

Our findings corroborate our domain knowledge and intuition.

Features characterised by low entropy are abstract specific labels (like antivirus signatures) or boolean values that are not suitable for approximate string matching. For example, we need the signature `LockyRansomware8963.11` to match exactly because `PetyaRansomware896.1` is a similar string but an utterly different ransomware family.

On the other hand, features characterised by high entropy are suitable for approximate string matching but only if they are human-oriented strings of text like the subject or the body of an email. Hashes and dates are also highly diverse, but we need them to match exactly because allowing the opposite would lead to severe false positives. The same reasoning holds for indicators in the middle group.

With this methodology, eight features were deemed suitable for approximate string matching. They can be seen in Table 3.2.

3.5. Approximate Automated Campaign Analysis with Density-Based Clustering

In section 3.1 we have broken our research problem, automating campaign analysis, into working items, hypotheses and specific research questions. We, therefore, identified the two subdomains of computer science that we identified as relevant to build a working solution: clustering (section 3.2) and approximate string matching (section 3.3). In this analysis, we found some promising techniques and methodologies: density based clustering to automate campaign labelling and TF/IDF for approximate string match. Furthermore, we have obtained a dataset of malicious emails suitable to our needs, although it will be necessary to enrich it and preprocess it. Now that we have found all the necessary building blocks for our system, in this section we can finally propose a complete framework to perform automated campaign analysis and bridge the research gap we have identified.

Section 3.5.1 first contains a high-level overview of the system. Then, sections 3.5.2 to 3.5.8 explain the low level implementation details.

3.5.1. System overview

The goal of our system is to aid a security analyst in performing campaign analysis. We have identified clustering and TF/IDF tokenisation + cosine distance as promising methods to perform this. We have also found the necessity to expand our dataset, as it covers only a portion of the kill chain model. We propose to realise our system as a pipeline of scripts and modules. Most of the scripts are written in Python 3, with a minority being in Logstash to interface with the DutchSec SpamTrap (stored in an Elasticsearch index).

This pipeline starts from unlabelled emails, enriches them by increasing the number of indicators (and kill chain phases covered) through a malware analysis module and then performs two steps of clustering, ending with a campaign label for every email. We define six logically divided phases:

1. Data enrichment following the killchain model: malware analysis module: Cuckoo sandbox + VirusTotal.

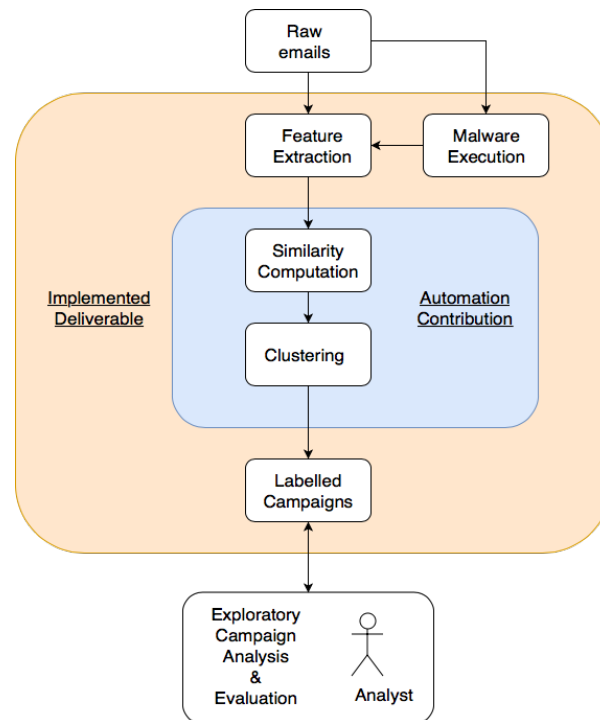


Figure 3.3: High level overview of the system envisioned for this thesis project. The orange area is implemented as a deliverable. The blue area represents the automation engine, which is the scientific contribution of this model.

2. Preprocessing and feature selection: from ElasticSearch to HDF5.
3. Indicator/feature level similarity: TF/IDF + cosine similarity
4. Indicator/feature level clustering: DBSCAN/HDBSCAN
5. killchain level similarity: ad hoc similarity metric dependent on threat module
6. killchain level clustering / campaign labelling: DBSCAN/HDBSCAN

A simplified diagram of our solution is shown at Figure 3.3, whereas Figure 3.4 additionally shows the details of each phase input and output. The system is fully modular so that we can replace specific techniques used at every specific phase. Modularity allows us to test different setups (with or without approximate string match of indicators) and different clustering algorithms (DBSCAN and HDBSCAN). Different setups allow to formally evaluate performance and understand which techniques are to be preferable. A formal evaluation of results is done in chapter 4. In the next six sections, we illustrate the inner workings of each phase.

3.5.2. Phase 1: data enrichment

Indicators from phases 1, 2 and mostly 3 (reconnaissance, weaponisation and delivery) can be extracted directly from emails. Features of reconnaissance are benign attachments and recipient lists. The metadata of a malicious attachment can be classified under weaponisation. Most email header fields are clear examples of delivery indicators (from, subject, x-mailer...). As mentioned in section 3.4.3, raw emails lack indicators from phases 4-7. These are the phases after delivery, where the malicious payload first uses an exploit or social engineering(4) to trigger installation(5) and only afterwards it is executed. The malicious executable may try to establish a communication with the remote command and control server (6). Regardless, the next step is performing whatever action the payload was designed to perform (7: actions on objective; like ransomware encryption, data exfiltration, installing more malware...).

Phases post delivery are precious in the kill chain model because they reveal interesting technical indicators on the attacker modus operandi. Furthermore, as obfuscation techniques have become

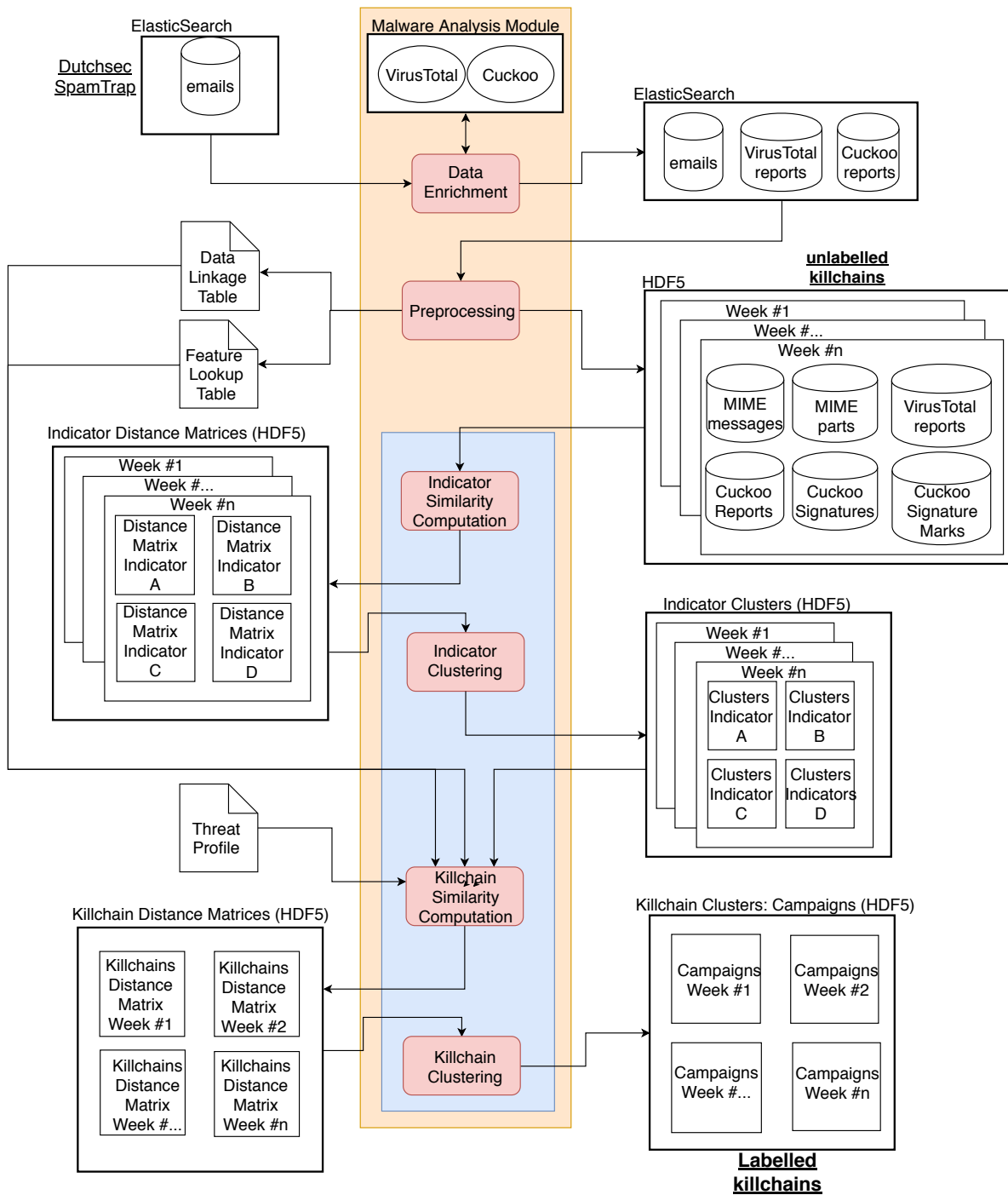


Figure 3.4: Detailed diagram of our system. The system can be seen as a pipeline of 6 steps (in the red boxes) executed in series. The orange area is implemented as a deliverable and the blue area represents the automation engine, which is the scientific contribution of this model.

widespread, techniques that do not analyse in depth attachments properties and execution are not useful anymore. The inclusion of a malware analysis module was then deemed critical for the quality of this project. Malware analysis techniques can be broadly divided into two categories: static and behavioural. Both can provide a vast deal of post-delivery indicators. The malware analysis module of our presented system is composed of the combined usage of VirusTotal APIs and a custom-built Cuckoo Sandbox malware lab. A diagram of this phase can be seen in Figure 3.5.

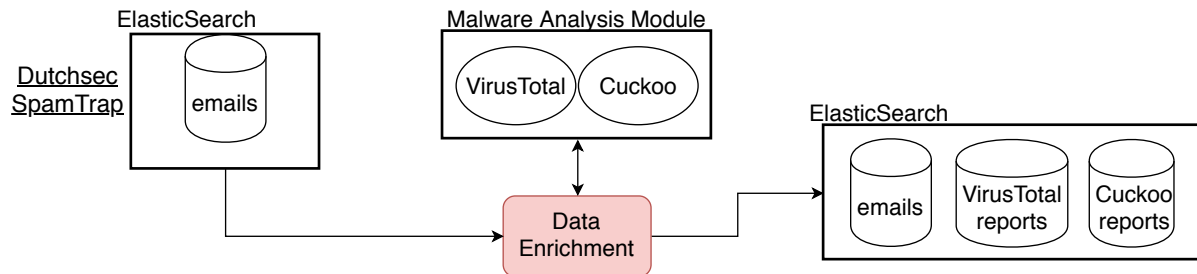


Figure 3.5: Phase 1 of our system. The attachments are extracted from the spamtrap, and then both are submitted to a custom-made cuckoo malware lab and VirusTotal for in-depth analysis. This greatly augments the number of indicators, filling for missing phases in the kill chain model (everything after delivery)

VirusTotal.com is an aggregator of antivirus products and online scan engines. The user uploads a file, a hash of the file, a link or an IP and the system returns a report. This report contains the result of 59 antivirus programs, an overall score of malignity and many interesting additional pieces of information like the date of first submission, the ssdeep of the file and its metadata. VirusTotal comes with APIs, but the free tier does not allow to retrieve large volumes of reports in a short amount of time. Luckily VirusTotal was interested in collaborating to this project and gave us free limited access to some of the paid APIs for three months.

As mentioned in section 3.4.2, the dataset is composed of roughly 500000 emails spreading over 16 months. The spamtrap itself labelled the attachments as malicious or not at collection time using ClamAV. According to this analysis, the dataset contains roughly 11000 malicious attachments, and of those only 6000 were unique. We found this figure suspiciously low, so we decided to perform an analysis on our own. It is possible that because the analysis was done at collection time a large number of false negatives would exist. Antiviruses perform in general worse on brand new malware compared to old malware, as knowledge of malicious indicators becomes widely available and vendors update their filters.

As a first thing, the malware analysis module hashes the attachments and then submits the hashes to VirusTotal through its APIs. VirusTotal returns a JSON report which we store in ElasticSearch. Only reports for files that had been analysed in the past can be retrieved, due to the limited access to the paid APIs. No new report can be generated. Luckily, the attachments belonged to emails sent between February 2016 and August 2017, so rather old concerning malware lifecycle. Indeed, we can retrieve a report for 35078 out of the 38028 hashes.

Although the VirusTotal reports contained quite interesting information, they mostly strictly included only static analysis. The behavioural analysis had been performed on just 0.6% of our samples. This low percentage is quite logic, as the behavioural analysis is a feature available only to paid users. VirusTotal on its own is therefore insufficient to fill the kill chain model because we are missing all indicators of phases after exploitation. To obtain these, we decided to set up our own malware analysis laboratory to test all the attachments.

At this time, Cuckoo Sandbox is one of the most popular open source dynamic malware analysis systems. It is built in Python and requires minimum effort to set up (at least in its purest form). Cuckoo has a host-guest architecture. The main cuckoo process runs on the host, which then spawns a guest running the target environment in a virtual machine. The host then passes the file that needs to be tested to the guest, generates a report and then finally kills the guest. Everything that the malware does is recorded (creation of files, internet connections, system calls etc...). This report contains both static and behavioural analysis features. Of great interest and value are the YARA signatures that cuckoo can match over malware. YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples automatically. YARA signatures are, simply put, abstractions of behaviours. To be more clear, they are a unique and understandable label for a behaviour that

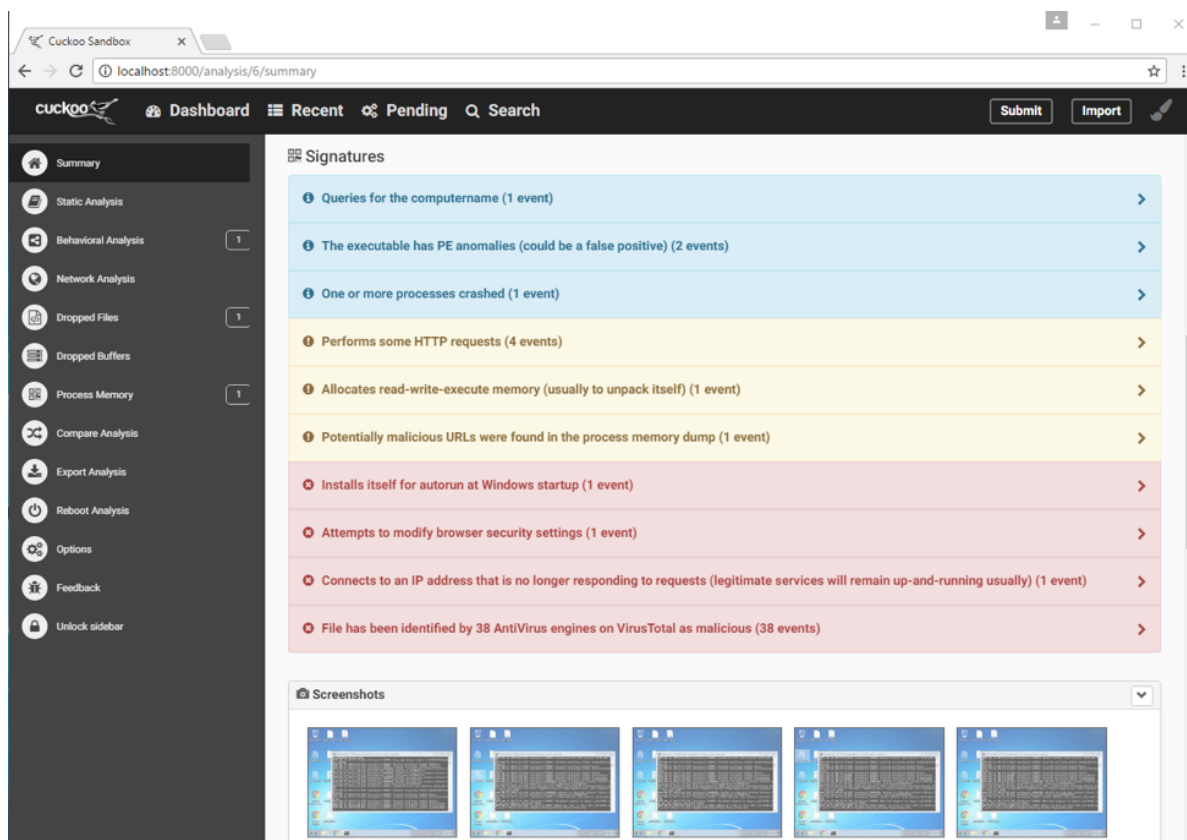


Figure 3.6: Screenshot of Cuckoo Malware Analysis UI. Specifically this screenshot shows the YARA signatures found during execution of a malware sample.

can be composed of a multitude of simple atomic actions. Examples of YARA signatures are “HTTP connection to blacklisted domain”, “connection to the TOR network”, etc. More can be seen in Figure 3.6.

We set up a distributed cuckoo malware analysis system. The system itself is composed of two Asus home servers and one Alienware gaming computer running Ubuntu 16.04 LTS. As suggested by cuckoo in their setup guide, we used an official Windows 7 64 bit image. We also purchased a licence for Microsoft Office 2013 and installed Adobe Acrobat Reader for opening PDFs. Finally, all default security mechanism of Windows are disabled and macro execution is enabled on every program where it is an option (Microsoft Office, Adobe Acrobat Reader). The two home servers were capable of running ten virtual machines at the same time, whereas the Alienware pc could run up to four, for a total of twenty-four hosts running on the three machines. The front end user interface of the malware lab can be seen in Figure 3.6.

As usual, the malware landscape evolves in parallel with the countermeasures that the security research community develops to stop it. Cuckoo, like any other behavioural analysis system that uses virtual machines as a testing environment, also suffers from this dynamic. Malware has evolved to detect if it is running within a virtual machine and if so to stop execution altogether. This mechanism is also called sandbox evasion. Malware can perform it in a multitude of ways, for example by looking at the MAC address of the network interface (by default VirtualBox and others have a unique MAC address) or by looking at the size of the Hard Drive (e.g. nowadays a 10GB hard drive is not typical). Finally, we tested the “detectability” of our target system with the open source project Pafish¹⁰. Somewhat unintentionally following the idea of the kill chain, Pafish was developed by collecting indicators on malware sandbox evasion techniques.

With the cuckoo distributed system successfully set up, we submitted the 38028 unique attachments to it for analysis. After eight days of continuous computation, we had a behavioural report for all of them.

¹⁰<https://github.com/a0rtega/pafish>

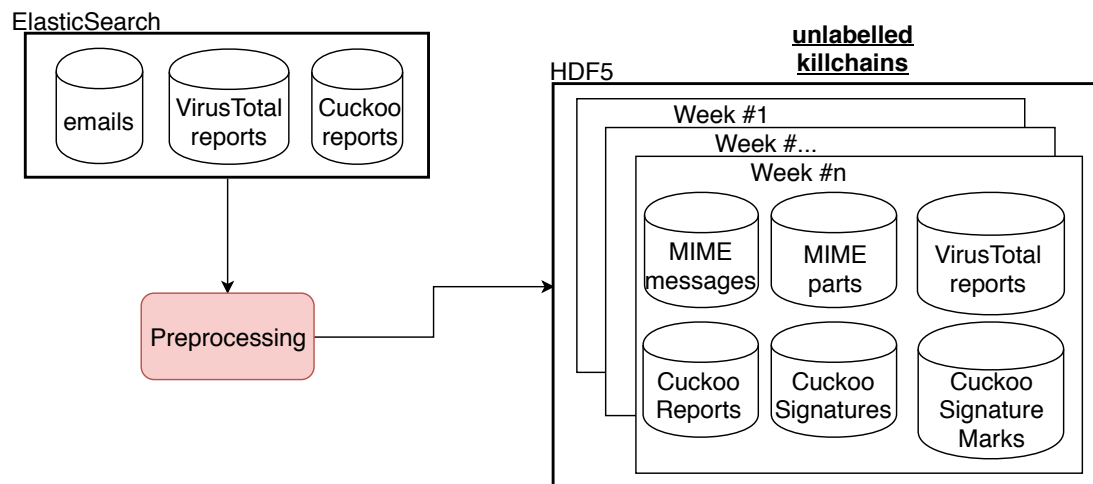


Figure 3.7: Phase 2 of our proposed system. Data are unified and migrated from ElasticSearch to the more Python friendly HDF5 format. The dataset is also divided in weeks.

The overwhelming majority of them was found to be malicious, with only roughly 3000 attachments not triggering any signature. We can now say it was a good call to not trust the analysis performed by ClamAV at collection time (it labelled only roughly 6000 as malicious). Indeed, the quality and amount of indicators obtained through our malware lab makes it an undeniably fundamental component of any proper campaign analysis framework based on emails as the source data.

3.5.3. Phase 2: preprocessing and feature selection

In phase 2, we tidy up our data, unifying the three databases (Dutchsec Spamtrap, VirusTotal analyses and Cuckoo reports) containing all the indicators discovered. Figure 3.7 shows a diagram of this phase. We migrate from three elastic search indices to one HDF5 data structure, where each item is a kill chain with its plethora of indicators. HDF5 is a data model, library, and file format for storing and managing data ¹¹. It is used often in conjunction with Python to do data science.

Whereas ElasticSearch uses JSON as a data storage format, HDF5 uses numpy tables. JSON allows for a tree structure, but numpy tables are comparable to those of a database. Several fields in all of the indices, unfortunately, had a variable number of items. For example, one message is composed of multiple parts, one cuckoo report can have a variable amount of YARA signatures triggering, and these signatures can trigger an arbitrary amount of time. In other words, to convert the data structure from JSON to HDF5 it was necessary to index a tree structure for every kill chain. This indexing proved to be quite challenging at first but was later solved using pandas DataFrames and their property of storing variable-size lists as items.

The final working item of this phase is to select a subset of the indicators. The reason why we include this aspect in the system is that it is dependent on the dataset. A widely different dataset will probably come with a different indicator set.

In our case, not all indicators present in the spamtrap or found later by the malware analysis module are of interest. Furthermore, some of these are often empty or unreliable, so we also have to exclude them. We select the remaining 68 indicators with the criterium of relevance (in cybersecurity maliciousness sense) and by making sure we have some indicators for each Lockheed Martin's kill chain phase. The full list of features is in the Appendix, at A.1.

3.5.4. Phase 3-6: the automation engine

Now that our data is ready to perform campaign analysis we approach the first real scientific contribution of this thesis project: the automation engine. Since the problem seems suitable for it, we have decided

¹¹<https://www.hdfgroup.org/HDF5/>

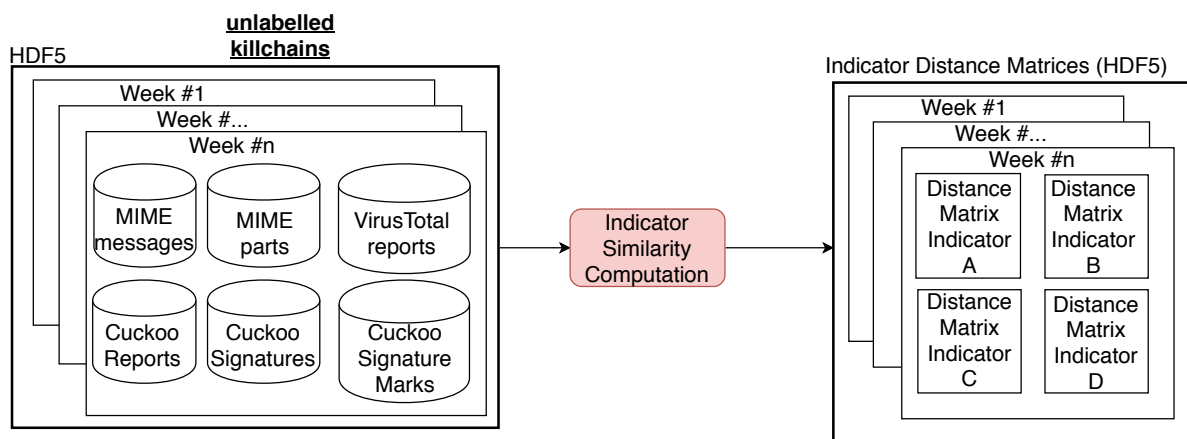


Figure 3.8: Phase 3 of our proposed system. The input of this phase is the enriched dataset of killchains, the output is a similarity matrix for each feature.

to try to implement automation through clustering. Our dataset is highly multidimensional (68 features, some of which with more than just one value) and non-categorical. Some features are numerical, but mostly are strings of arbitrary length and meaning. In order to use a clustering technique, we need to transform the values of our feature from unstructured strings to categorical/numerical. Furthermore, to perform clustering it is necessary to have some metric of either similarity or distance depending on the context, and not all distance metrics are good for all clustering techniques.

We envision the following methodology to automate the process of campaign analysis:

- for each feature: using a measure of similarity, compute the pairwise similarity among all items. (Phase 3, this section). The output of this phase is a feature level $F \times F$ distance matrix
- for each feature: apply density-based clustering on the feature level distance matrix obtained at previous phase. Phase 4, section 3.5.6.
- for each pair of kill chains: compare the result of the clustering at the previous phase. The more labels two kill chains share, the more they are similar and vice versa: this can be used to build a similarity metric. Using this score to build a kill chain-level distance matrix $N \times N$. Phase 5, section 3.5.7.
- use the kill chain-level distance matrix for a second round of density-based clustering. Each clustered label is a campaign. Phase 6, section 3.5.8.

3.5.5. Phase 3: indicator level similarity computation

As pointed out, the goal of this third phase is to, using a measure of similarity, compute the pairwise similarity among all items. Figure 3.8 shows a high level diagram of this phase.

The indicators considered are mostly strings, so we need to look for a similarity metric among strings. First, we try the most obvious option: edit based string metrics (e.g. Levenshtein, Hamming). This method is far too slow: computation of one row of the pairwise distance matrix (1 item compared to all the others) took roughly 1 second on a 6th generation (Skylake) 2 GHz Intel Core i5. One average week of the dataset has roughly 10000 emails. The number of possible combination per week is thus on average $(10000^2)/2 = 50,000,000$. 50 million seconds is roughly 580 days, for each feature. We decided to discard this method and look for better alternatives.

In section 3.3 we pointed at TF/IDF tokenisation of n-grams as an interesting approach to assessing the similarity of strings. This approach is often used in practice, combined with cosine distance, to compute the pairwise similarity of a large set of strings. Looking for implemented examples, we encounter an open source project from ING¹² that uses n-grams and TF/IDF to perform approximate search of strings in a database. This ING library then computes the cosine distance (the measure of the angle

¹²<https://medium.com/wbaa/https-medium-com-ingwbaa-boosting-selection-of-the-most-similar-entities-in-large-scale-datasets-450b3242e618>

between two vectors) to return all the relevant items when querying a database afflicted by the inconsistent use of nomenclature/spelling mistakes. ING themselves resorted to this approach because they found edit distance techniques to be not scalable. The reason for the speedup is mostly due to the usage of efficient linear algebra techniques for the pairwise computation. In our testing, this proved to be faster by a factor of x100 to x1000 when compared to the linear approach of going through all the possible combinations of edit distance implementations. The computation time was reduced from 580 days to hours. Due to bugs in the library, we ended up implementing the approach from scratch.

We now will present the pipeline of steps performed at this phase, but first, in Figure 3.9 we introduce a simple example dataset that will recurrently be used through this chapter. In this simplified kill chain set, there are four kill chains with three indicators: sender, subject and the hash of an attachment. There are two spear phishing campaigns: (A, B) and (C, D).

Killchain	f 1.1 (sender)	f 1.2 (subject)	f 1.3 (hash attachment)
A	foo@kpn.nl	invoice xx1234#	hash 34567896
B	bar@kpn.nl	invoice xx1235#	hash 886548907
C	silvio.berlusconi@forza-italia.it	elezioni!	hash u513672498
D	salvini@leganord.it	elezioni 2018	hash u513672498

Figure 3.9: Example dataset to showcase the similarity computation with TF/IDF and cosine distance computed at phase 3. Killchains A and B belong to one campaign, C and D to another.

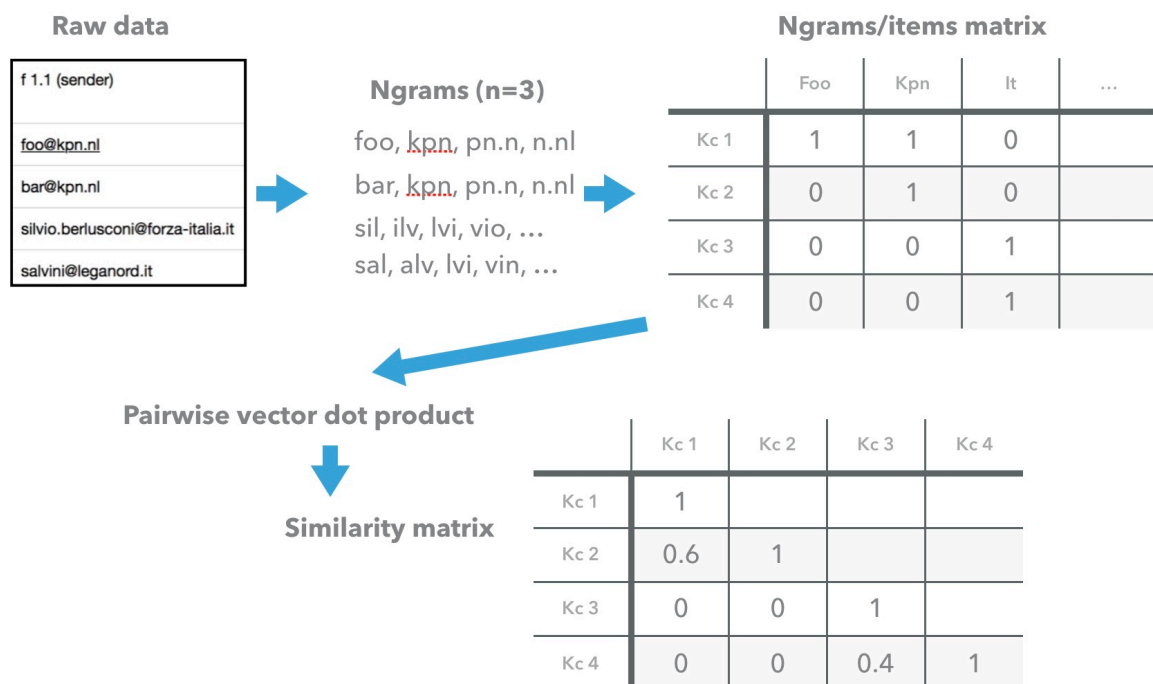


Figure 3.10: Phase 3 TF/IDF + cosine distance for similarity computation of indicator "email sender" on example dataset

Phase 3 TF/IDF + cosine distance for feature level similarity computation:

1. for each feature corpus, the full token list is generated by an n-gram tokeniser from the python library scikit-learn
2. a TF/IDF vectorizer from the python library scikit-learn generates one normalised token vector for each document, computing each component (the value for each n-gram) using TF/IDF. In other words, each vector component measures the TF/IDF relevance score of a specific n-gram.

The token vectors are normalised during computation to have all magnitude 1. The normalisation of the token vectors is necessary for two reasons. The first one is that this allows comparing all term vectors by exclusively looking at the cosine of the angle between them, which is easily computed by performing the dot product of the two. The second one is that TF/IDF is already taking care of giving an appropriate weight for each token for each document. This value already takes into account both the frequency within the document (TF) and the frequency within the document corpus (IDF). Indeed, normalisation of the magnitude of the token vector does not mean ignoring the different size of the documents, nor it means ignoring the amount of time a token is present in a document. Not normalising the token vectors, in this case, would alter the TF/IDF computation by redundantly accounting for the term frequency within each document leading to an inconsistent value of tokens for each document.

In a strictly positive vector space like ours, the cosine of the angle between all pair of possible vectors is always between 0 and 1 because n-grams count can only be non-negative. Two vectors with the same orientation have a cosine similarity of 1 and indicate identical documents, two vectors at 90° have a similarity of 0 and indicate completely different documents.

3. compute the cosine similarity between all pairs of vectors

The cosine of two non zero vectors can be derived from the Euclidean dot product, which takes into account the angle as well as the magnitude:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \cdot \|\vec{b}\| \cdot \cos(\theta)$$

which is the same as:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

Now, in the special case when magnitude is 1 for both vectors, it follows that:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{1 \cdot 1} = \vec{a} \cdot \vec{b}$$

$$\cos(\theta) = \sum_{i=0}^n a_i \cdot b_i$$

To compute this measure for all possible combinations, it is sufficient to perform the dot product between the token vector matrix T generated by the TF/IDF vectorizer by its transpose T^{-1} .

The resulting matrix contains at row x and column y the aforementioned normalised dot product which is simply the sum of the product of the components:

$$\cos(\theta) = \vec{x} \cdot \vec{y} = \sum_{i=0}^n x_i \cdot y_i$$

This computes all pairwise cosine similarities and can be performed in python with the simple one-liner.

$$SIM = np.dot(T, T^{-1})$$

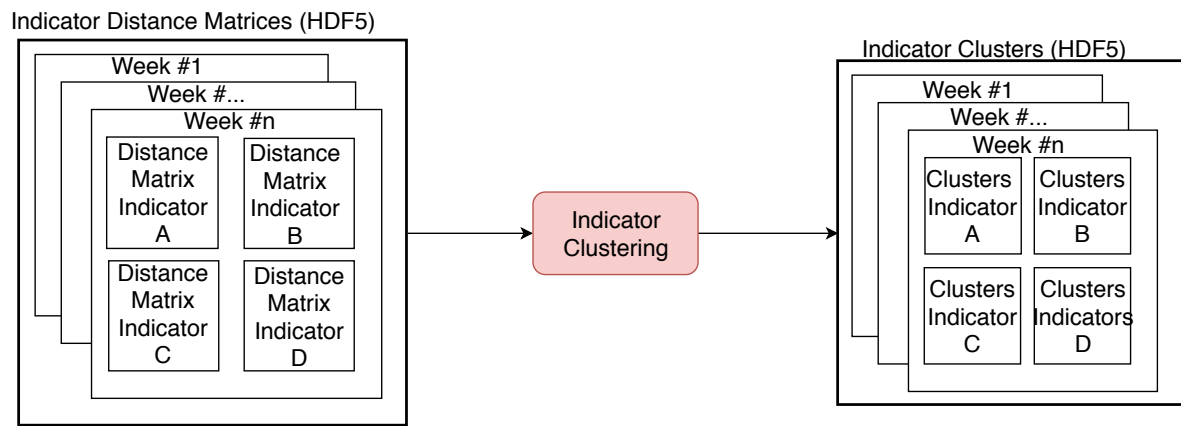


Figure 3.11: Phase 4: indicator level clustering. The input of this phase is the indicator similarity matrices obtained at phase 3, the output are the cluster labels.

4. Finally, as we care about distance and not similarity, and since our token vector space is non negative (TF/IDF scores are never negative, at most 0), we can compute the distance matrix from the similarity matrix by simply performing:

$$DIST = np.subtract(1, SIM)$$

This methodology allows computing a similarity matrix for each of the 68 features in an amount of time on the order of hours (single core), although with burdensome memory requirements (up to 60GBs for the weeks with more emails). Having solved the similarity assessment of features, in the next section we perform a first round of clustering to group related items together under the same label (thus transforming raw uncategorical data in categorical data).

3.5.6. Phase 4: indicator level clustering

In the previous section, we started working on the “automation engine” component of our system. We designed a methodology to compute a similarity score between strings, and we then applied it to our dataset, obtaining a similarity matrix for each indicator. We can now feed this similarity matrix (after converting it into a distance matrix) to a clustering algorithm. At the end of this step, we will have converted our raw, uncategorical dataset (strings) in a strictly categorical dataset (cluster labels, numbers). Figure 3.11 shows a high-level diagram of the input and output of this phase and Figure 3.12 illustrates the effect of this phase in our example dataset.

This step allows us to do two things at once: to easily automate the process of exactly matching indicators (precisely like in Lockheed Martin’s methodology) and to eventually to implement approximate indicator match for the indicators we found to be suitable in section 3.4.4. We can do this as follows:

exact match we force the clustering algorithm to cluster together only elements with a perfect similarity score (1)

approximate match we allow the clustering algorithm to cluster together elements with a similarity score above a certain threshold.

As pointed out previously, clustering algorithms usually allow a distance matrix as input, not a similarity matrix. Distance makes more sense for clustering because when we plot the data points in an n-dimensional vector space, we want more similar elements to be represented closer and more different elements to be represented far from each other. Converting a similarity matrix into a distance matrix is trivial for similarity metrics that have an upper bound. This is our case because our similarity metric is bound to the interval $[0, 1]$.

$$DistMatrix = 1 - SimMatrix$$

In section 3.2 we have given a short survey on clustering techniques in the search for one suitable for our needs and tailoring our constraints. We identified density-based clustering as a good candidate,

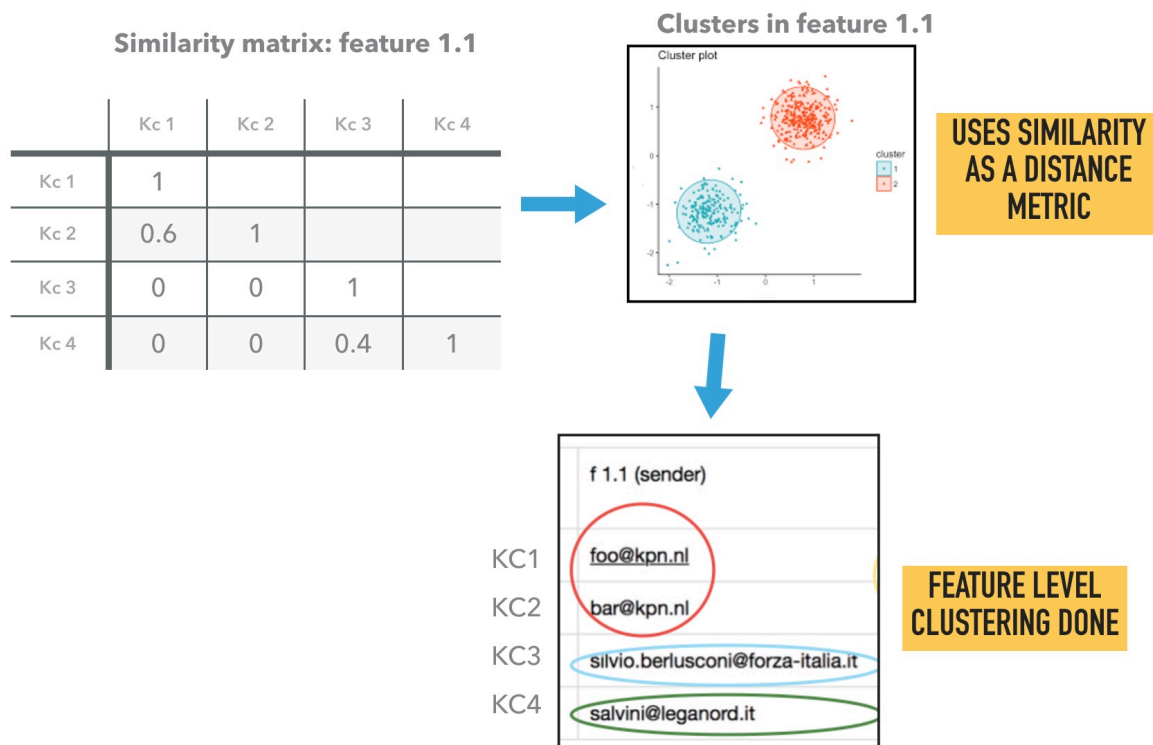


Figure 3.12: Abstract view of phase 4 on our example dataset

and we selected DBSCAN and HDBSCAN as the algorithm to use in our system. We described how the two algorithms work in section 3.2.2. Again, our system is highly modular, and it thus allows to alter the clustering technique or its parameters easily. We, therefore, test both DBSCAN and HDBSCAN to compare the two. In this chapter we focus only on the implementation details, the results can be found in chapter 4.

Fine tuning DBSCAN parameters

The DBSCAN algorithm can be found implemented in the python library scikit-learn. There are two parameters to set: `eps` and `min_samples`. They both affect the measure of density that we desire to be captured. Lower values of `eps` and higher values of `min_samples` lead to denser and less noisy clusters, whereas high `eps` and low `min_samples` lead to less dense, more noisy clusters. The suggested approach to finding the best set up is to start with a `min_samples` values of 1, then play with `eps`. If the clustering is too noisy increase `min_samples` and start over. This is repeated until the clustering quality is acceptable. However, how is the clustering evaluated? Unfortunately, in the absence of a training set, there is no alternative than to manually evaluate the content of the found clusters by counting false positives (how much noise in one cluster). In large datasets, like our case, this is far too time-consuming to perform for all the clusters found (in the order of thousands for a week with 10000 kill chains). Luckily, when the parameters are not ideal, the result is so weak that it is instantly recognisable as such, leading to a faster than hoped cycle of evaluation¶meter-tuning. A further annoyance of DBSCAN is that different dataset will likely have different ideal parameters tuning; therefore it is necessary to perform the parameter discovery for each indicator. Luckily though:

1. parameters found for one period are good for other time periods too. Not only this saves us much work, but it also is an interesting find on its own as it tells us that the distribution of similarity scores (and therefore the patterns in the data) is somewhat coherent across different time series. It is important to note that if this is true for our dataset, it may not be for other datasets. This may be a problem.
2. parameters have to be found only for the indicators selected for approximate indicator match. For indicators where exact match is necessary it is sufficient to set the `eps` parameter to 0, forcing

the clustering to exclusively label together elements with similarity 1 (and therefore identical).

After extensive testing we eventually settled for `min_samples = 1` and found the ideal value of `eps` for all the indicators selected for approximate indicator match. They can be viewed at table 3.3.

Table 3.3: Features selected for approximate string matching with chosen `eps` parameter for DBSCAN

Indicator #	Name	DBSCAN eps
15	<code>additional_info.exiftool.ZipFileName</code>	0.55
20	<code>body</code>	0.35
25	<code>file.name</code>	0.55
27	<code>filename</code>	0.55
29	<code>from</code>	0.45
36	<code>headers.Return-Path</code>	0.45
52	<code>reverse</code>	0.5
64	<code>subject</code>	0.475

HDBSCAN parameter tuning

In section 3.2 we have discussed how HDBSCAN allows for variable density clusters by automatically finding an optimal `eps` parameter in different areas of data space. The only parameter that has to be set in HDBSCAN is the very obvious `min_cluster_size`. What should be the minimum cluster size that our clustering allows? If a cluster is below this threshold, it will be ignored, and its items will be labelled as noise at the end of the algorithm. To answer this question, we need to understand if we want to allow for **singleton campaigns**. A singleton campaign is a campaign composed by just one kill chain, and we believe they should be allowed. The reason for this is two-fold. First of all, there is no advantage in explicitly marking noise in our use case. Secondly, our methodology will eventually allow tracking campaigns through time by cross-checking the final cluster stage among contiguous weeks. Therefore, we should not discard the potential starting occurrence of a campaign only because there is just one kill chain in a specific time interval. For example, imagine an APT that sends several emails per week, starting Sunday the 26th of August. If campaign analysis is performed on a weekly basis with our system, the first kill chain will not end up in a campaign if singleton campaigns are not allowed, and it will be simply discarded as noise. Furthermore, there is no trade-off whatsoever in allowing singleton campaign.

Following this logic, we decide not only to allow singleton campaigns but also that they are a desirable property of our clustering. We thus set `min_cluster_size` to 1. No further tuning is required in HDBSCAN. In section 4 we give a formal comparison of DBSCAN and HDBSCAN concerning performance, but so far we must state that HDBSCAN is far superior in ease of use. It took minutes to set and run HDBSCAN, whereas it took dozens of hours to set and evaluate DBSCAN parameters iteratively.

Campaign Analysis through looser clustering

During the process of fine-tuning DBSCAN parameters, we made an exciting discovery. We ideally desire a value of `eps` that leads to zero false positives in our clusters. We found out that when this parameter was set to be slightly underfitting so that a little amount of false positive is allowed, the results of the clustering are significantly more interesting. For example we show at Figure 3.13 the content of a cluster of email subjects when `eps` is set to 0.45 compared to when it is set to 0.425.

When the variable is set at 0.425 the clustering is more strict, the label contains 215 elements and they all more or less related to the string "Laten we leren kennen?". Furthermore, there are no blatant false positives in the cluster.

When the variable is set at 0.45 the clustering is looser, and the label contains 911 elements. Not all the elements contain the string "Laten we leren kennen?", however, they all seem to follow some similar logical structure and meaning.

What do we want from the clustering? Are these additional elements to be considered false positives? We have to make a decision here in what we want our system to do. The rationale behind a false positive assessment is that we have to have a definition of true positive. In our scenario, we want to cluster together things that are related to a campaign, following Lockheed Martin process of campaign analysis. Although reasonably different, we can see that these subjects are related and most likely


```
<HDF5 dataset "DBSCAN_425": shape (215, 2), type "|0">
[[ '333' 'Laten we leren kennen?']
[ '767' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '899' 'Hello! Laten we leren kennen?']
[ '1172' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '1247' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '1289' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '1301' 'Laten we leren kennen?']
[ '1509' 'Hello! Laten we leren kennen?']
[ '1983' 'Laten we leren kennen?']
[ '2028' 'Laten we leren kennen?']
[ '2236' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '2330' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '2663' 'Hello! Laten we leren kennen?']
[ '2895' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '2897' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '2907' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '2914' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '2945' 'Hello! Laten we leren kennen?']
[ '3009' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '3135' 'Laten we leren kennen?']
[ '3174' '\U000feb0d_ ... Hello! Laten we leren kennen?']
[ '3175' '\U000feb0d_ ... Laten we leren kennen?']
[ '3196' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '3267' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '3343' '\U000feb0d_ ... Laten we leren kennen?']
[ '3388' 'Hello! Laten we leren kennen?']
[ '3492' '\U000feb0d_ ... Hello! Laten we leren kennen?']
[ '3584' '\U000feb0d_ ... Hello! Laten we leren kennen?']
[ '3586' '\U000feb0d_ ... Laten we leren kennen?']
[ '3590' '\U000feb0d_ ... Hello! Laten we leren kennen?']
[ '3598' '\U000feb0d_ ... Hello! Laten we leren kennen?']
[ '3617' 'Hello! Laten we leren kennen?']
[ '3659' 'Laten we leren kennen?']
[ '3802' '\U000feb0d_ ... Laten we leren kennen?']
[ '4204' 'Hello! Laten we leren kennen?']
[ '4372' '\U000feb0d_ ... Laten we leren kennen?']
[ '4461' '\U000feb0d_ ... Laten we leren kennen?']
[ '4488' '\U000feb0d_ ... Laten we leren kennen?']
[ '4585' '\U000feb0d_ ... Laten we leren kennen?']
[ '4650' 'Hello! Laten we leren kennen?']
[ '4655' 'Hello! Laten we leren kennen?']
[ '4905' '\U000feb0d_ ... Laten we leren kennen?']
[ '5021' '\U000feb0d_ ... Hello! Laten we leren kennen?']
[ '5024' '\U000feb0d_ ... Laten we leren kennen?']
[ '5026' '\U000feb0d_ ... Hello! Laten we leren kennen?']
[ '5124' 'Laten we leren kennen?']
[ '5126' 'Hello! Laten we leren kennen?']
[ '5391' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '5430' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '5640' '\U000feb0d_ ... Laten we leren kennen?']
[ '5653' '\U000feb0d_ ... Laten we leren kennen?']
[ '5684' 'Hello! Laten we leren kennen?']
[ '5863' 'Laten we leren kennen?']
[ '5865' 'Laten we leren kennen?']
[ '5958' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '5977' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '5996' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '6282' '\U000feb0d_ ... Laten we leren kennen?']

<HDF5 dataset "DBSCAN_45": shape (911, 2), type "|0">
[[ '101' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik woon in de buurt, ik kan voldoen?']
[ '105' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik woon in de buurt, ik kan voldoen?']
[ '328' 'Laten we ontmoeten! Ik woon in de buurt ..']
[ '333' 'Laten we leren kennen?']
[ '338' 'Ik wil u snel te zien ..']
[ '339' 'Laten we ontmoeten! Ik woon in de buurt ..']
[ '340' 'Ik zou willen ontmoeten met u']
[ '406' 'Laten we ontmoeten! Ik woon in de buurt ..']
[ '416' 'Ik wil u te ontmoeten']
[ '417' 'Ik woon in de buurt, ik kan voldoen?']
[ '445' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ontmoeten']
[ '452' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik zou willen ontmoeten met u']
[ '479' '\U000feb0d_ ... Ik wil u te ontmoeten']
[ '509' '\U000feb0d_ ... Laten we ontmoeten! Ik woon in de buurt ..']
[ '510' '\U000feb0d_ ... Ik wil u snel te zien ..']
[ '693' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ontmoeten']
[ '767' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '828' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we ontmoeten! Ik woon in de buurt ..']
[ '829' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik zou willen ontmoeten met u']
[ '835' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u snel te zien ..']
[ '883' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik woon in de buurt, ik kan voldoen?']
[ '893' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u snel te zien ..']
[ '895' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we ontmoeten! Ik woon in de buurt ..']
[ '898' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik woon in de buurt, ik kan voldoen?']
[ '899' 'Hello! Laten we leren kennen?']
[ '900' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u snel te zien ..']
[ '907' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ontmoeten']
[ '910' '\U000feb0d_ ... Ik wil u snel te zien ..']
[ '1113' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik woon in de buurt, ik kan voldoen?']
[ '1125' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u snel te zien ..']
[ '1126' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ontmoeten']
[ '1131' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we ontmoeten! Ik woon in de buurt ..']
[ '1163' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik zou willen ontmoeten met u']
[ '1172' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '1180' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ontmoeten']
[ '1183' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik woon in de buurt, ik kan voldoen?']
[ '1247' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?']
[ '1248' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik zou willen ontmoeten met u']
[ '1251' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ontmoeten']
[ '1268' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ontmoeten']
[ '1289' '=?UTF-8?Q?=F3=BE=AC=8D_?= ... Laten we leren kennen?']
[ '1297' 'Ik zou willen ontmoeten met u']
[ '1301' 'Laten we leren kennen?']
[ '1307' '\U000feb0d_ ... Ik wil u snel te zien ..']
[ '1311' '\U000feb0d_ ... Ik zou willen ontmoeten met u']
[ '1633' 'Ik wil u snel te zien ..']
[ '1644' 'Ik wil u te ontmoeten']
[ '1683' 'Laten we ontmoeten! Ik woon in de buurt ..']
[ '1689' 'Hello! Laten we leren kennen?']
```

Figure 3.13: Cluster content comparison of two runs of DBSCAN with different eps parameter on SpamTrap email subjects. Left: eps = 0.425, Right: eps= 0.45. The latter is looser but seems to actually capture well a real campaign. This is the effect of the phenomenon called cluster chaining.

were variation within a campaign. Not only the human-readable messages seem to relate, but there is also a clear structure in the strings: some “machine oriented” followed by three dots followed by a message that hints at the opportunity of a date. Without really trying too much we have already found the combination of TF/IDF + cosine distance and density-based clustering to be able to spot campaigns in the data.

If we further set the eps to 0.475 the cluster contains even more items: 1124. This time though, 13 actual false positives, completely unrelated strings, are introduced. 13 out of 213 new items are thus wrongly labelled.

We now have to make a decision. Allowing for these little false positives or getting rid of them at the expense of some false negatives. The goal of our system is to automate the process of campaign analysis to help a security analyst performing threat intelligence. The hard part of this process is the manual cross-reference of possibly thousands of multidimensional kill chains. We believe that a fuller but less precise result is far more interesting for an analyst compared to a less complete but more precise result. Furthermore, an analyst can quickly, almost at a glance discard false positives whereas it would take hours to manually check all the other kill chains to manually find what the clustering missed out for the sake of precision.

Clustering Chaining: a desirable property

The “Laten we leren kennen” example also show us an interesting phenomenon of density based clustering: **chaining**. Wilks defines chaining as

the production of a few large clusters, which are formed by virtue of nearness of points to be merged at different steps to opposite edges of a cluster[20].

Chaining is often mentioned as an undesirable property that makes density-based clustering less robust

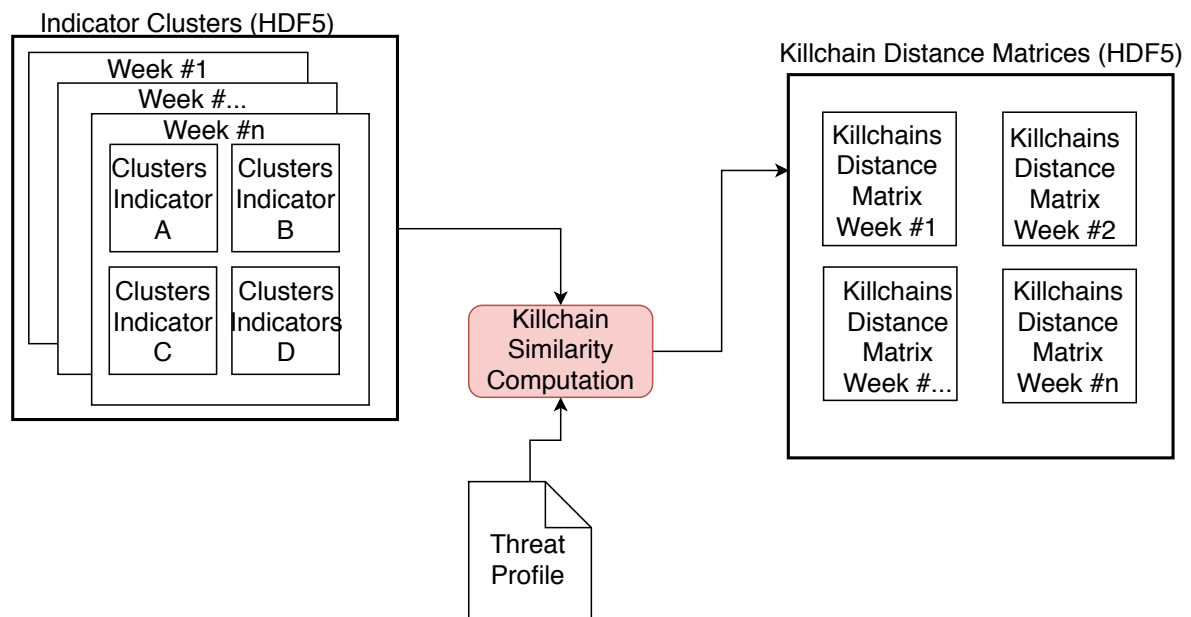


Figure 3.14: Phase 5 of the proposed automated campaign analysis system. In this phase, the similarity between killchains is computed with an ad hoc similarity metric that accounts for variable weight indicator and different threat profile (APT, spam etc)

compared to other clustering techniques[21]. As we will now illustrate, chaining is actually producing a quite desirable result in our scenario. Let's look for example at the two following strings A and B:

A Ik wil u snel te zien

B Laten we leren kennen?

How do these two cluster together? When we look closely, we find that the following strings are also present in the cluster:

C =?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u snel te zien ..

D =?UTF-8?Q?=F3=BE=AC=8D_?= ... Hello! Laten we leren kennen?

This is the famous chaining effect. String A is very similar to string C, string C is very similar to D and string D is very similar to B. Generally speaking, chaining is an unwanted property that affects density-based clustering, but in our scenario, it actually seems to allow for the perfect kind of campaign analysis that we would expect a human, not a machine, to perform!

We have now performed clustering at the indicator level. Not only have we thus converted the raw data in categorical labels, but we also found out that clustering can already allow a form of campaign discovery thanks to the surprisingly desirable phenomenon of chaining. In the next phase, we define an algorithm to compare two kill chains by cross-checking these categorical labels, producing another similarity score (at the kill chain-level this time).

3.5.7. Phase 5: killchain level similarity computation

In phases 3 and 4 we transformed the raw data into categorical labels, for each indicator. Ultimately, though, campaign analysis should group incidents at the kill chain-level, not at the indicator level. In this phase, we develop a simple similarity metric to assess the similarity of two kill chains based on the result of the clustering at the previous phase. Figure 3.14 gives an overview of the input and the outputs of this process.

Intuitively, we have to compare every pair of kill chains by looking at the respective cluster labels of all their indicators. If an indicator ended in the same cluster for both kill chains, then this should be accounted as more similar. Once a kill chain-level similarity matrix is computed, it can be reversed

into a distance matrix, and clustering can be performed again. We develop a fairly simple algorithm to assess the similarity of two kill chains:

```
phase5(KC A, KC B, threat_module):
    similarity = 0
    LA = indicators_labels_A
    LB = indicators_labels_B
    for x in range 68:
        if LA[x] == LB[x]:
            similarity += weight(threat_module, x)
    return(similarity)
```

The most interesting component of the algorithm is the weight function. A weight function is necessary for two reasons. First of all, different indicators have different absolute importance. For example, the hash of a malicious payload is a far more important indicator than the language of the email. They both matter and they both should be used for campaign analysis, but the former is a much stronger link in forensic terms. Following this rationale, we assign an **impact** value to all the 68 features: low (1 point), medium (5 points), high (20 points). Secondly, we derive from domain knowledge that different kill chain components have different importance with different categories of threat actors. An example is useful to clarify. APTs and mass spam campaigns have some inherent differences to them, the most important being that the former are targeted (if not to a single individual at least to a group), whereas the latter generally are not. We should account for this in our system, allowing the analyst to select what we call a **threat profile** that will alter the weight of specific indicators to give a more realistic and coherent result. Finally, the weight function takes care of normalising the similarity score in the [0,1] interval. The 68 features were divided into four categories, trying to follow the original kill chain model:

- target: related to the victim (phase 1, 2 in LM kill chain)
- delivery: in this case all email metadata and headers (phase 2 in LM kill chain)
- static: metadata of the payload (phase 2, 4 in LM kill chain)
- dynamic: indicators collected by executing the payload (phases 4, 5, 6, 7 of LM kill chain)

An example of threat profile for APT would give much more weight to the victim and maybe less to the delivery as the effort of varying delivery indicators is far less than in the case of mass spam. Little weight is given to static because of obfuscation techniques, and more is given to dynamic as core behaviour of malware and action on objectives will be likely similar in a campaign (also require much effort to alter. expensive) Figure 3.15 shows the threat profile weights used for our spam/phishing dataset and a suggested threat profile weights for APT targeted malicious emails.

3.5.8. Phase 6: killchain level clustering (campaign labeling)

A similarity score is computed with the algorithm proposed at the previous phase. The scoring is done for all couple of kill chains, resulting in a kill chain-level similarity matrix for each time interval considered. Similarly to what done previously, this one can too be converted into a distance matrix by performing (the similarity score was normalised in [0, 1]):

$$DistMatrix = 1 - SimMatrix$$

The final working item in this project is to feed the kill chain similarity matrices to a clustering algorithm. The clusters found are going to be the campaigns identified by the system and will be its final output. Figure 3.16 shows a high-level diagram of this process

We use density-based clustering for the same reasons we used it in 3.5.6. We test both DBSCAN and HDBSCAN again. The same methodology of phase 4 was used to set DBSCAN parameters. The `min_samples` variable was set to 1 again, for the same reasons mentioned in section 3.5.6. The `eps` parameter was similarly fine-tuned by finding the optimal compromise between false positives and quality of results. The approach was again to minimise false positives and maximise utility, preferring a complete result at the price of some false positives. Again, fine-tuning DBSCAN parameters for

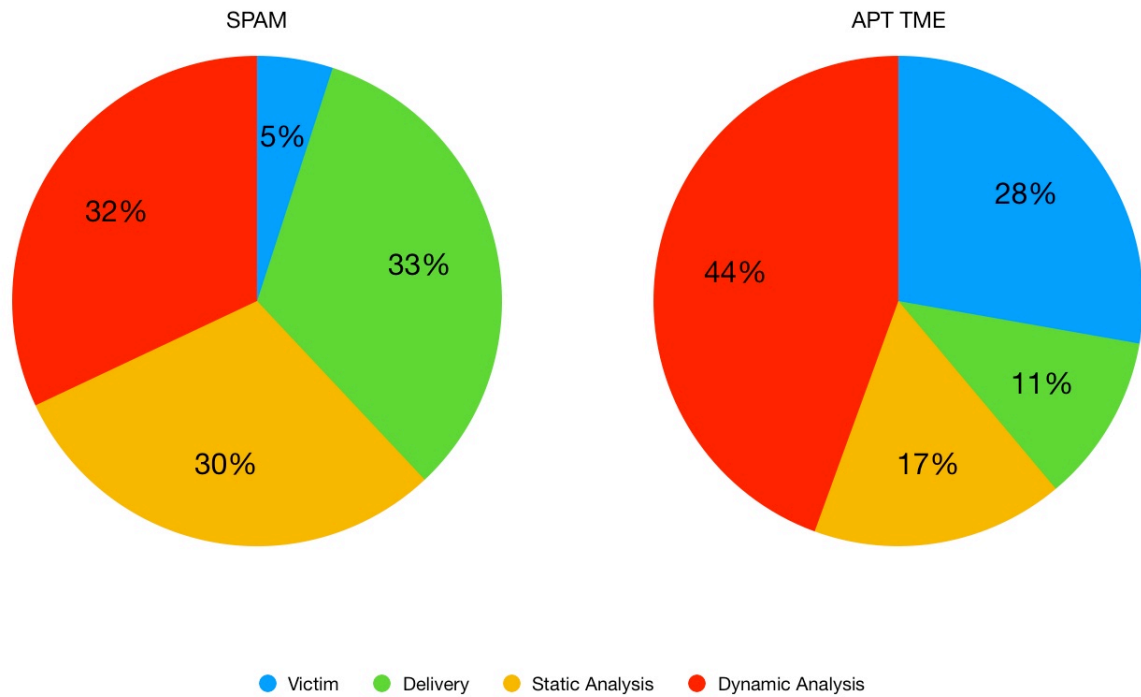


Figure 3.15: On the left: threat profile used for campaign analysis of spam/phishing campaigns. On the right: a suggested threat profile for APT targeted malicious emails. Notice how victim indicators weight is small in the case of spam/phishing campaigns (which usually are massive and untargeted) and higher in the case of advanced persistent threats' targeted malicious emails.

one week of data took many hours, and it looks like it is necessary to retune it for every other week manually. On the other hand, HDBSCAN is again far easier to use. The only parameter of HDBSCAN, `min_cluster_size`, is set again to 1 and the algorithm will take care of optimising the `eps` value.

This phase completes our proposed system for automated campaign analysis. In the next chapter, we evaluate the results of our implementation, and we assess the performance of the variable configurations we can test (with or without approximate match, DBSCAN/HDBSCAN).

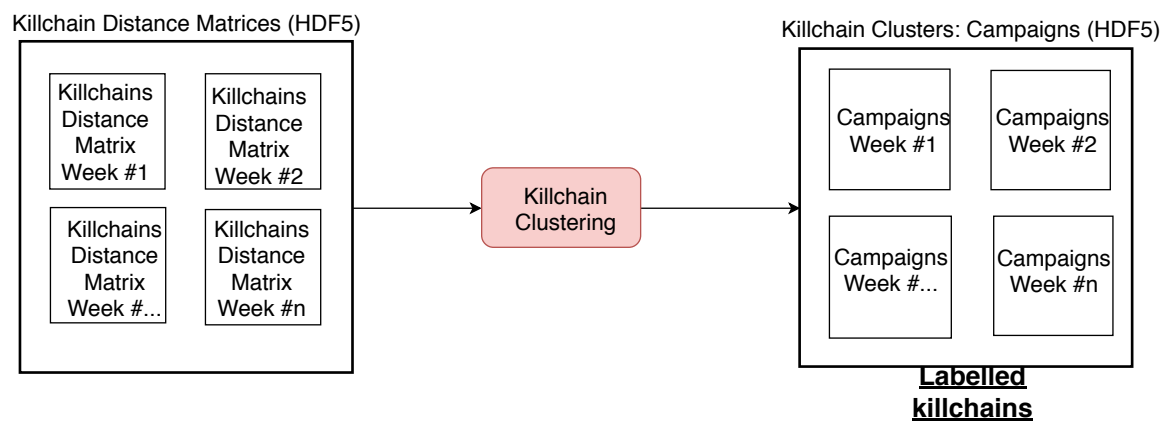
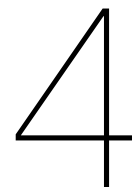


Figure 3.16: Sixth and last phase of our proposed automated campaign analysis system. Density based clustering is performed on the distance matrix computed at the previous phase. This process identifies the campaigns in the input dataset.



Results

In this chapter, we present the results of this thesis project. First, we briefly reintroduce the domain, the research gap and our research goals. Then we show the performance of the developed framework with various settings, introduced in the previous chapter. The performance analysis allows us to understand if the designed system is a significant contribution to the body of knowledge. We quantify precision in terms of false positives, and we quantify recall in terms of false negatives. We compare the performance of various versions of our automatic framework, showing what setup performs best and why. Comparing the performance enables us to quantify the benefit of automation in campaign analysis and also to assess the consequences of applying approximate string matching on some features. Next up, we address the limitation of our framework. Finally, in the last section, we showcase some notable findings discovered by the system.

4.1. Campaign Analysis, Research Gap and Research Goals of this project

In 2011 researchers from Lockheed Martin Corporation released a whitepaper [10] that marked a paradigm shift in how cyber **threat intelligence** is performed. In this paper, Hutchins et al. introduced the concepts of cyber kill chain and campaign analysis. The kill chain is a concept adapted from the military domain to the cyber domain. A kill chain is a multi-stage representation of an intrusion attempt. The stages are seven and allow to map an attack from its ideation to its completion. Indicators are atomical pieces of information belonging to the kill chains (e.g. the hash of malware, the sender email, the IP that malware connects to). Clearly, enough indicators can be used to fingerprint attackers.

Campaign analysis is the process of linking together repeated attempts from a malicious entity by searching for repeating indicators in the respective kill chains. The state of the art in campaign analysis has two significant problems: lack of automation and strictness in indicator overlapping. Lack of automated implementations makes the task of looking for overlapping indicators manual, something unfeasible in most scenarios, and exact match of indicators may not be the ideal way to perform campaign analysis for all indicators.

In this project, we have set two research goals to overcome the gaps in the state of the art. We have designed a system to automatically extract features from malicious emails, enhancing the produced kill chain with indicators from post-installation phases with a malware lab and established a methodology to cluster related kill chains in campaigns. Furthermore, we have introduced approximate string matching for some of the indicators.

4.2. Experimental results

In this section, we present a performance analysis of the system developed in section 3. First, we briefly discuss the spamtrap training data. Then we explain the experimental setup, and finally, we show the actual results and their analysis.

4.2.1. Dataset

To build and test our system we relied on a dataset of malicious emails generously provided by Dutchsec, an Utrecht-based cybersecurity firm. In 2016 Dutchsec developed a spamtrap, a honeypot dedicated to collect malicious emails for threat intelligence purposes. The spamtrap has been running and collecting emails since then. For our study we focused on a collection of emails spanning 16 months (February 2016 - May 2017), the spamtrap was modified afterwards, and we wanted our data to be as gapless and consistent as possible.

In this period, the spamtrap collected around half a million emails. From data exploration, we can state with a certain degree of confidence that the dataset is entirely malicious, although a portion of emails is “legitimate” commercial spam, which may not be considered malicious universally. Regardless, the most substantial majority of these emails concern fraudulent spam (e.g. sex enhancing drugs) and spear phishing attacks.

Following the kill chain framework, we mapped the spamtrap features to kill chain indicators. We found the original data to lack indicators of phases post delivery (everything revolving around the installation of malware and its execution). This lack is quite logic, as the spamtrap was developed to study email related metadata. To fill this gap in the kill chain model, we extracted roughly 38000 different attachments, and we run them into a custom-made cuckoo malware lab, obtaining dozens of new indicators for phases exploitation, installation and command&control. We finally selected 68 indicators to be used in our system; the full list can be found in Appendix A at Table A.1. We selected eight of these 68 features for approximate indicator match. To do this, in Section 3.4.4 we developed a methodology based on the entropy of the cosine distance in each indicator space.

Finally, to make the computation quicker (this is a proof of concept, after all), we split the dataset in weeks. We found the emails to be roughly equally distributed across the 16 months period (with some exceptions), with an average load of 10000 emails per week.

4.2.2. Experimental Setup

In the previous chapter, we have proposed and implemented a system to solve the research gaps identified. To evaluate the system as well as some implementation options, we need to come up with a formal experimental setup.

The primary goal of this project is to automate the process of campaign analysis, improving the state of the art of manual exact match performed by a security analyst. Thus, we need to not only evaluate if our system is capable of performing campaign analysis (and thus if density-based clustering is a suitable choice for automation) but also if it does so *better* than the state of the art. Secondly, we want to establish if approximate indicator match of suitable indicators is a valuable addition to the original model. Finally, we want to assess if the density-based clustering is better when the density is assumed to be fixed (DBSCAN) or variable (HDBSCAN) in the dataset.

We design five different setups to test for all these conditions:

1. Manual exact (only) indicator match: performed manually. This is the state of the art in campaign analysis as described by Lockheed Martin [10].
2. Automated exact (only) indicator match, clustering with DBSCAN
3. Automated exact (only) indicator match, clustering with HDBSCAN
4. Automated mixed indicator match (exact + approximate), clustering with DBSCAN
5. Automated mixed indicator match (exact + approximate), clustering with HDBSCAN

Comparing setup 1 with the others will let us understand if the automated setup is working and if it performs better or worse than the state of the art. Comparing setup 2 and 3 with 4 and 5 will let us understand the consequences in terms of performance of introducing approximate match of indicators. Finally, we can compare the result of DBSCAN and HDBSCAN. This last condition can make us understand if our dataset is better suited by an algorithm that allows for variable density regions or by one that requires a fixed density throughout the dataset.

Our approach to testing these five setups is to evaluate their **performance**. We have not yet discussed precisely what we mean by performance. To do so, we first establish that the most critical metric at our disposal is the quality of the campaigns discovered. Therefore, our approach is two-fold.

First, we measure the false negatives over a dataset of synthetic campaigns: this allows to quantify the **recall** of the system. Recall measures the percentage of relevant instances retrieved over the total amount of relevant instances. This metric gives us an idea of how *complete* our campaign analysis is. We then measure the false positives: this gives us a measure of **precision**. Precision measures the percentage of relevant instances over the total retrieved instances. This metric gives us an idea of how many irrelevant items will be in the clusters of actual campaigns.

4.2.3. False Negatives and Recall

Recall is defined as follows:

$$recall = \frac{TP}{TP + FN}$$

To assess recall we need to know what is the true size of a campaign (the denominator, meaning all the relevant items). Obtaining knowledge is problematic when performing unsupervised learning as prelabelled data is missing. Furthermore, in this specific scenario, we are dealing with a dataset not only large but also highly multidimensional. Moreover, the features are often non-categorical complex strings.

The most appropriate way to assess recall is to search and label campaigns in the dataset indeed manually. Unfortunately, manual labelling is unfeasible in this case due to the multidimensionality mentioned above and the limited amount of time and human resources.

Therefore, we designed an alternative approach mixing synthetic data and highly distinguishable campaigns encountered in the data during its exploration.

We designed four synthetic spear phishing campaigns ($S1, S2, S3, S4$), each composed by ten emails; they can be seen in detail in the Appendix, at Figures B.1, B.2, B.3, B.4. To evaluate the limits of our system we designed the four campaigns to be increasingly complex. We envisioned an increasingly skilled attacker that, aware of the traditional campaign analysis methodology, tries to avoid repeating indicators.

In the first campaign $S1$, we imagine an attacker merely spamming the same kill chain ten times. In the second campaign $S2$, the attacker is more sophisticated. He tries to avoid campaign linkage by using ten different sender email addresses. Furthermore, he randomises a six digits number at the end of his subject and filename. Finally, he uses obfuscation techniques to make sure his malware (hidden within a seemingly benign attachment) produces a different hash every time. To make his life easier, he crafts the message in a way that any reply will always go to the same return address. He is not aware that his x-mailer will remain the same if he uses the same email client. The third attacker is more proficient in avoiding campaign analysis. No indicator is ever repeated. He uses ten different email addresses registered at the same domain to send the messages, making sure that the return addresses are all different. The subjects he uses are always different, although the overall meaning remains the same. Finally, the x-mailer and the attachment name are partially randomised by appending a random six digits code at the end of the string. Finally, the fourth campaign is crafted by an attacker that is making sure every single indicator is never identical. He uses entirely randomly generated email addresses and crafts the subjects by hand to resemble real ones. Only the filename and the x-mailer maintain a structure, with a word followed by a six digits random number.

Ideally, we would expect state of the art exact-only indicator matching to link the first two synthetic campaign together. Indicators in campaigns $S3$ and $S4$ are never repeated. We crafted $S4$ especially to understand where and if our fuzzy system breaks. It is also important to note that to craft these synthetic campaigns we took inspiration from the real campaigns we found in the dataset while exploring it.

We found several real campaigns while performing data exploration. Among them, three stood out significantly from the rest. Because manual evaluation of every single message is required to establish the campaign size formally, we were able to find all related message by looking at five highly relevant indicators that are easy to compare (in the case of spear phishing): hashes of parts and attachments, subjects, sender emails, payload sizes, filenames. We can then establish with a high degree of confidence the size of these three campaigns. Thus we can use them for counting false negatives by counting the missing items. Excerpt of these three campaigns can also be found in the Appendix at figures C.1, C.2, C.3

Table 4.1 shows the percentage of false negatives found in the five setups mentioned above. More interestingly, figure 4.1 shows the average recall of the five setups over the seven campaigns analysed.

	S1	S2	S3	S4	R1	R2	R3	Average
Manual Exact	0%	0%	90%	90%	0%	0%	0%	26%
Automated Exact + DBSCAN	0%	90%	90%	90%	0%	0%	0%	39%
Automated Exact + HDBSCAN	0%	0%	0%	0%	0%	0%	0%	0%
Automated Mixed + DBSCAN	0%	0%	80%	90%	31%	77%	0%	40%
Automated Mixed + HDBSCAN	0%	0%	0%	40%	30%	77%	37%	26%

Table 4.1: Percentage of false negatives over the seven campaigns studied. S1 to S4 are synthetic campaigns, the other three are real campaigns found in the data set. S1 to S4 have increasing levels of indicator entropy to try to circumvent traditional campaign analysis.

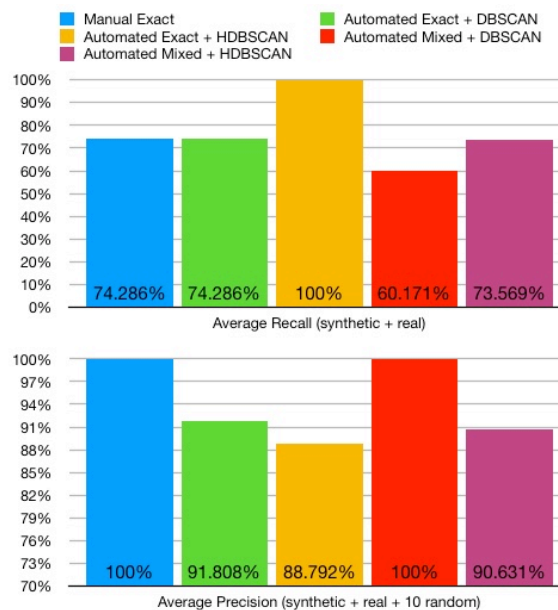


Figure 4.1: Average recall (top) and precision (bottom) of the five setups tested. Campaign analysis automated by HDBSCAN clustering of exact exclusive match of indicators has the best recall with a perfect score. The recall was assessed on four synthetic campaigns and on three real campaigns found while exploring the dataset. All systems performed well enough concerning precision (bottom), with values never lower than 88%. The best performers are, unsurprisingly, the exact manual match and, surprisingly, the automated mixed indicator match clustering with DBSCAN. Further analysis, see 4.3 shows that this high value of precision is due to really low recall performance of this setup.

In terms of average recall, our automated exact indicator match using HDBSCAN is the best performer with a perfect 100% average recall over our campaigns.

4.2.4. False Positives and Precision

False positive assessment is less problematic because it is not required to know the size of the relevant items set (true positives plus false negatives). Indeed, it is sufficient to evaluate only the returned items and count which ones do not belong to the cluster. We randomly select ten samples from our kill chains and evaluate the cluster they end up after the campaign clustering stage. We count as false positives all items that should not be in the cluster with our sample because they are not part of the same campaign. We do the same also for the three real campaigns described in the prior section.

After counting false positives for these thirteen items, we can compute the precision as follows:

$$precision = \frac{TP}{TP + FP}$$

Precision is a measure of how many relevant results are returned by the system. We can use it to understand how much irrelevant items (noise) are returned additionally.

Figure 4.1 shows that the average precision of our five setups is satisfactory, with the lowest score being 88% for automated exact match using HDBSCAN.

4.3. Analysis

We now proceed to analyse the results of our evaluation.

The goal of our system is to aid a security analyst by correlating related intrusion attempts in campaigns. In this scenario, we believe that recall is more important than precision. The reason for this is that if noise is not overwhelming an analyst can easily ignore it, whereas making sure that all relevant items are clustered together is a much harder task without prior knowledge.

Our automated exact-only system that uses HDBSCAN for clustering is the best performer regarding recall. It is the only system that was capable of perfectly recalling all items from our synthetic and real campaigns. Being recall our most crucial parameter for evaluation, and being precision not overwhelmingly bad (88%), we can confidently state that this is the best methodology for campaign analysis among those that we tested. Figure 4.4 shows the visual result of HDBSCAN over the exact-only kill chain distance matrix.

When comparing this solution to the state of the art manual exact-only campaign linkage, we can see that unsurprisingly precision is worse: obviously, there are no false positives when manual campaign analysis is performed. On the other hand, though, recall is significantly better (25% better on average). For an analyst exploring the data in search of a pattern, this has a critical impact.

We can thus affirm that the system designed can not only perform automated campaign analysis, but it does it better (with higher recall but also higher noise) than the state of the art methodology.

Quite surprisingly, our automated mixed system using DBSCAN performs flawlessly concerning precision. The recall is more than 10% better than the average of the three other automated setups. This anomaly is easily explained when looking at the average recall of this setup: it is by far the lowest at 60%. This system returned mostly singleton clusters, failing at clustering together related relevant items. Singleton clusters have zero false positives, hence the high precision value.

Another interesting finding is that HDBSCAN is less precise than DBSCAN and has a higher recall for both mixed and exact exclusive indicator matching. This most likely reason behind this difference is that HDBSCAN allows for variable density clusters, whereas DBSCAN requires a fixed value of density. Furthermore, if extensive testing is necessary to find the optimal parameters of DBSCAN, HDBSCAN automatically select the best at runtime. With better performance and far easier usability, HDBSCAN is a preferable choice to DBSCAN in this context.

Finally, we see that both of our mixed approaches perform worse in terms of recall than the equivalent exact-exclusive. In the mixed tests, we use exact string matching for most features and approximate string matching for eight suitable others. This means that we are contributing in adding more noise in our dataset. Indeed the average distance between kill chains is 0.974 for exact exclusive indicator match is performed and 0.871 for mixed exact and approximate indicator match. This can be seen in figures 4.3 and 4.2. They show a tSNE projection of the two distance matrices produced by the system before clustering. The exact-only projection is noticeably less noisy than the mixed one. This ulterior noise most likely is the culprit of the worse performance of both density-based clustering algorithm. Regardless, the conclusion is that when automating campaign analysis through density-based clustering using approximate string matching of indicators worsen recall and is, therefore, less desirable than exclusively exact indicator matching. We can say that density based clustering itself ends up performing a form of approximate campaign analysis, making approximate matching of indicators redundant and even excessive.

4.4. Limitations

Although promising, our methodology comes with four significant limitations.

Training In order to find new campaigns, the system needs to be retrained over the entire dataset. Retraining could be done every hour, day, week depending on the scenario.

Computational complexity Computing the kill chain level similarity (section 3.5.7) is resource intense because a recursive operation has to be performed on all combinations of kill chains. Combinations grow quadratically over the number of items considered. In this project, the computation was spread among 64 cores and still required several days to finish. Code revision and implementations on faster languages than python could help speed up the process (pypy, c). On the other hand, campaign analysis should be performed only on emails that have already been flagged as malicious. This should realistically be a rather small subset of the total emails within a company.

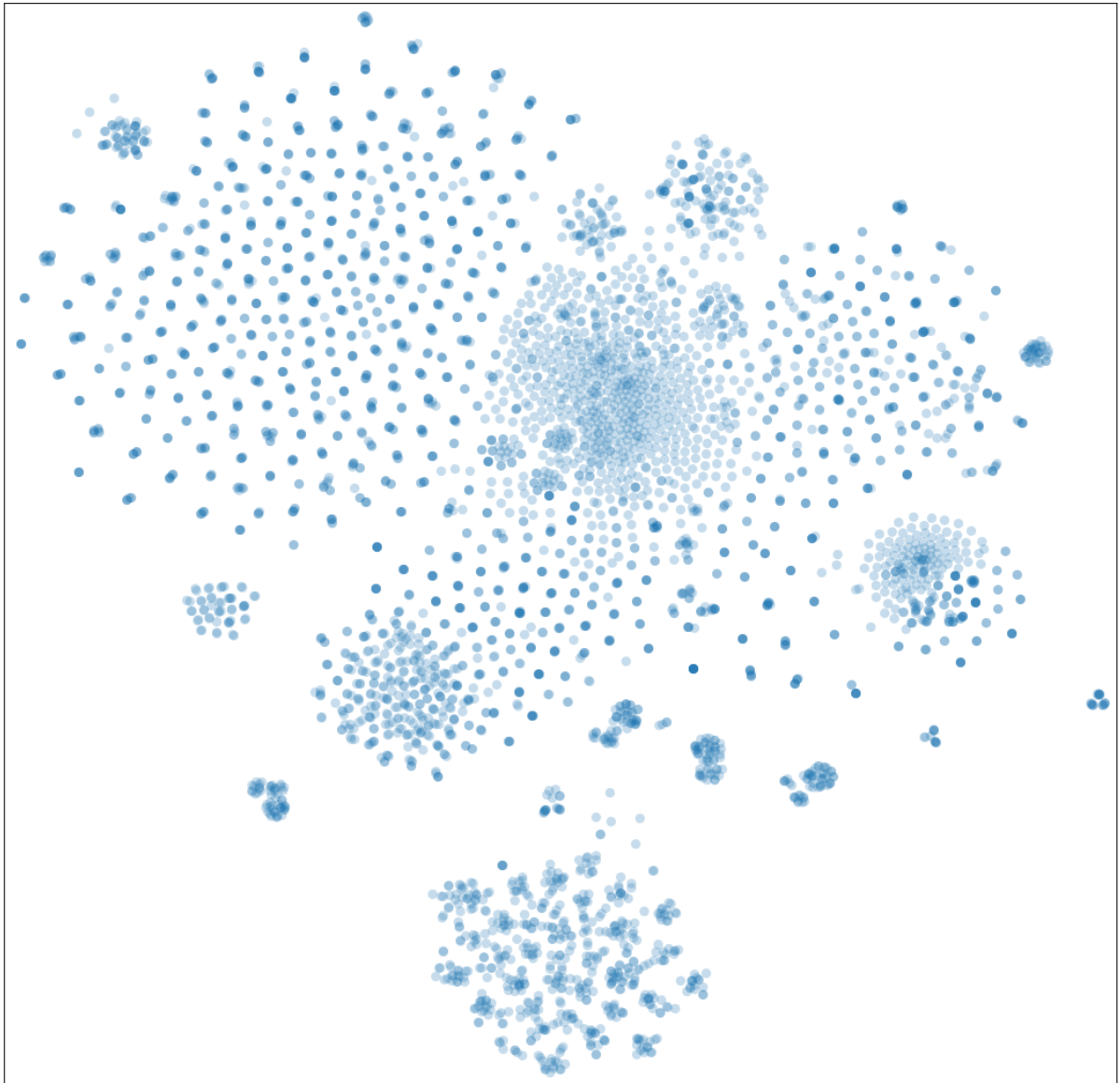


Figure 4.2: t-SNE projection of the killchain distance matrix obtained when both exact and approximate indicator matching is performed. Every dot represents a killchain. This is noticeably noisier than when exact-only match is performed, see Figure 4.3

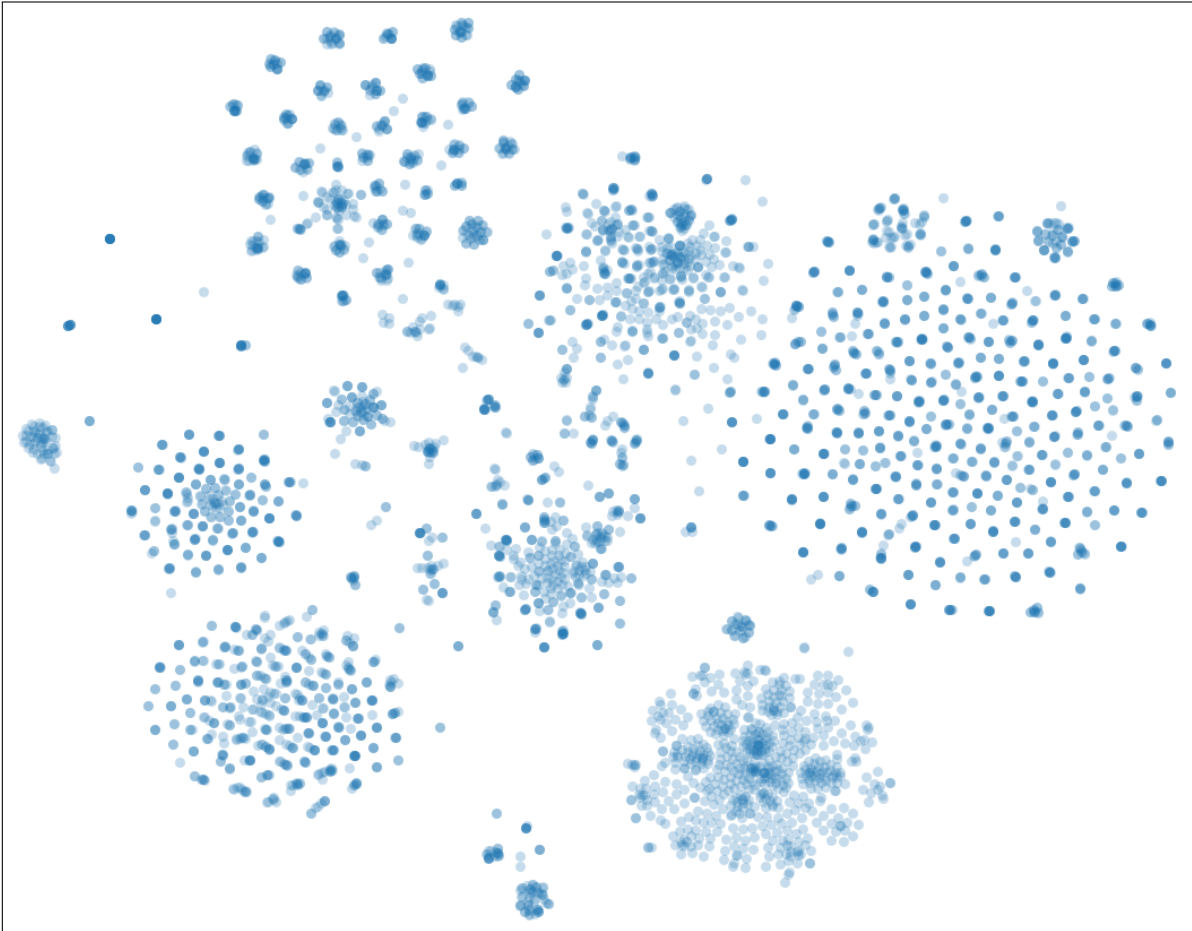


Figure 4.3: t-SNE projection of the killchain distance matrix obtained when exact-only indicator matching is performed. Every dot represents a killchain.

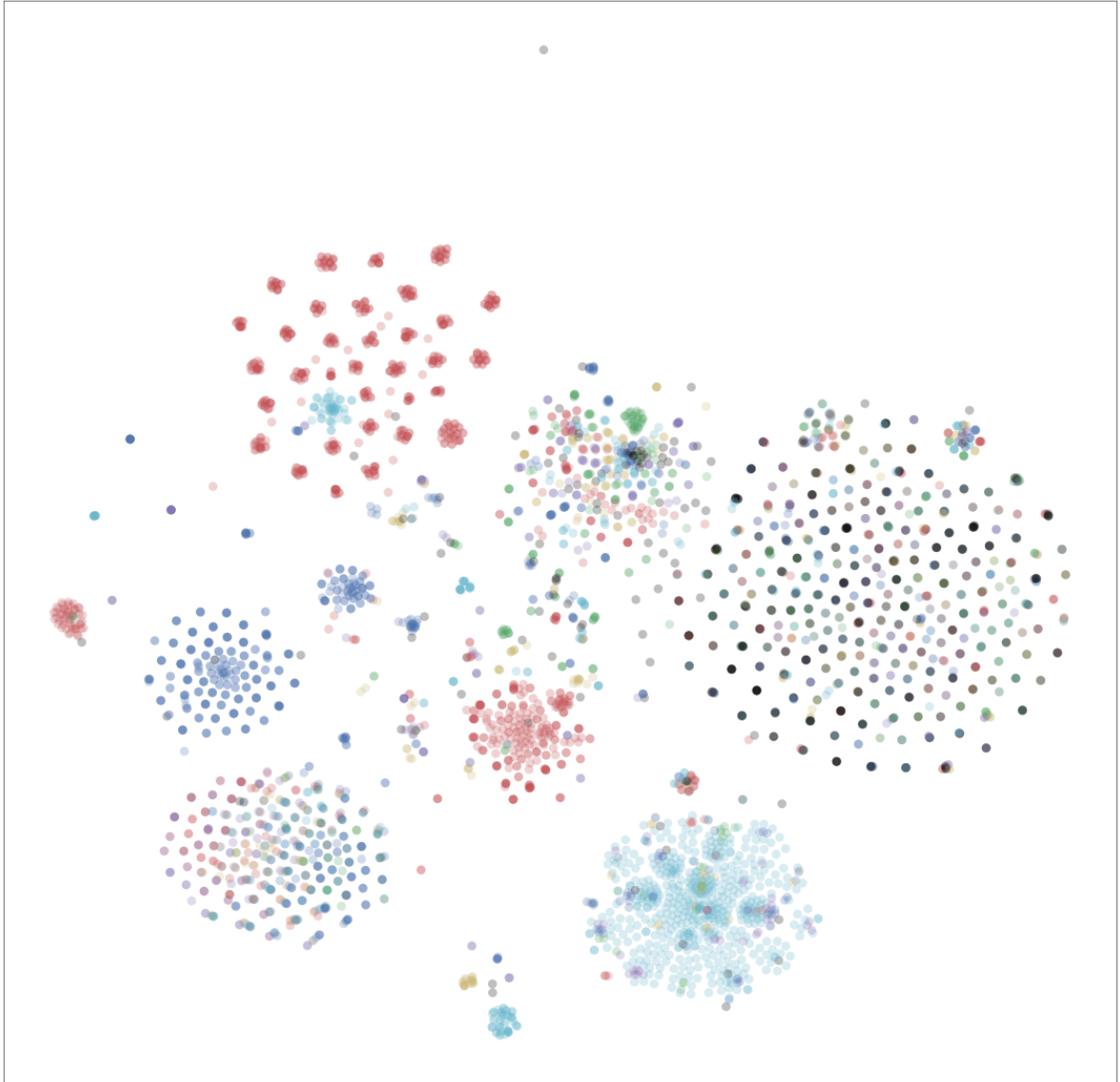


Figure 4.4: HDBSCAN clustering of the killchain distance matrix obtained when exact-only indicator matching is performed. Every dot represents a killchain.

Privacy Applying this methodology organisation-wide means that all emails metadata and content would have to be accessible, which has a privacy problem. This is a common issue in network monitoring. A solution here could be to anonymise the most privacy-sensitive features (like email body, subject and sender) for example by hashing them. As hashes are non-deterministic, this would still allow for an exact match of indicators.

Malignity blindness System is blind to malignity, meaning it clusters related emails regardless of them being harmful or benign. The emails should be prefiltered based on malignity by some indicator to perform valuable threat intelligence.

4.5. Notable Findings

The system developed was capable of successfully discovering real campaigns in the dataset. In this section, we present some notable findings to showcase the insight that can be obtained with campaign analysis. We selected a random week of data and ran the best set up (exact match only + HDBSCAN). The system thus identified 581 campaigns (out of 4617 emails). After examining the content of these campaigns, we identified several aspects worthy of interest.

One of the most intriguing characteristics of the campaigns discovered is the different degree of randomisation of specific indicators. We hypothesise that there may be a correlation between the amount and kind of randomisation and the skill level of an attacker (or at least the amount of time that went into crafting the campaign). For example, almost all the campaigns discovered present randomisation of subjects and attachment filenames, but only a minority has different attachment hashes and x-mailers. Whereas a less skilled attacker will randomise the filename, a more expert attacker, aware of hash-based filtering, will also modify the file itself. This alteration entails extra effort, but dramatically increases the chances of a successful hack, as the most common antiviruses banally cross-check the hash of a file with a database of known malicious hashes. The most basic form of randomisation consists of the attacker appending a long random number to a string. An example of this is the classic filename `Factuur #xxxxxxx` used in spear phishing attacks to trick users into opening a seemingly important file. Campaign 8, table 4.2 shows examples of this kind of randomisation in subject and filenames. A slightly more sophisticated form of randomisation can be seen in the sender email addresses, where typically a name is combined to a surname by a special character like dash, underscore or dot, and a number may or may not be added at the end. We observed this too in campaign 8. Names, surnames and domains are probably selected from a list and randomly put together. Campaign 158, table 4.3, shows an example of a campaign with seemingly randomly generated usernames. Lastly, we have observed the randomisation of whole sentences sharing a common meaning, see campaign 188, table 4.4.

Another interesting discovery is the difference in tooling between more or less advanced threat actors. Less sophisticated attackers seem to use general purpose email clients like Windows Outlook or Apple Mail, whereas more sophisticated attackers are likely using a program to craft custom emails. Whereas stock email clients do not allow to modify fields like the x-mailer, custom tools allow all sort of changes to email header fields. We have discovered two example of campaigns where the x-mailer was likely modified. In campaign 245, table 4.5, the attacker is using a plethora of seemingly regular x-mailers to send spam. The most likely explanation for this is that he is using an email crafting program always to vary the x-mailer (it seems quite unlikely each of this is sent from a different program or machine when automated solutions are available). Another example, even more peculiar, can be observed in campaign 135, table 4.6. Here the x-mailer was by the attacker to avoid giving away any information about his set up. What is interesting though, is that as usual randomisation is not exactly perfect because a pattern, a structure, is maintained by the randomising function. What is interesting here is we can now use the attacker highly specific hiding technique against him to identify and track him over time in his future attacks.

The campaigns discovered in the spamtrap seem to belong to two kinds of attack categories: mass (probably fraudulent) spam and mass spear phishing (seemingly not targeted). The majority of campaigns falls under mass spam, where the goal is to socially engineer the user into buying sex enhancing drugs or dating related services. Although we did not investigate further, this kind of mass spam cam-

Table 4.2: Campaign #8

from	subject	xmailer	hashes	filenames	length
Christine Graves <GravesChristine58@iwkd.net>	FW: Payment #502728		[d0ee28..., 21c...]	wire_payment_502728.zip	11339
Rochelle Grimes <GrimesRochelle8033@likeland.hu>	FW: Payment #034118		[488063..., 21c...]	details_payment_034118.zip	11339
Leonardo Berg <BergLeonardo61@smolbti.ru>	FW: Payment #386003		[8d65c8..., 996...]	details_payment_386003.zip	11233
Kermit Berger <BergerKermit12@radom.vectranet.pl>	FW: Payment #702831		[66134b..., d00...]	urgent_payment_702831.zip	11695
Jenna Castillo <CastilloJenna6824@axtel.net.mx>	FW: Payment #471779		[8a0be3..., 741...]	urgent_payment_471779.zip	12167
Martha Hodge <HodgeMartha9584@vtr.net>	FW: Payment #406089		[a0c556..., f76...]	payment_urgent_406089.zip	12446
Guy Nelson <NelsonGuy964@aroundinsound.com>	FW: Payment #468850		[7ed91e..., 703...]	details_payment_468850.zip	11951
Luz Bruce <BruceLuz31@flaglerdocimage.com>	FW: Payment #210439		[7ed91e..., b1e...]	urgent_payment_210439.zip	11951
Brianna Wilder <WilderBrianna6557@wanadoo.fr>	FW: Payment #813248		[04c828..., 940...]	payment_info_813248.zip	11696
Bobbi Welch <WelchBobbi40@ronansys.com>	FW: Payment #858510		[600fd3..., 5be...]	wire_payment_858510.zip	13233

Table 4.3: Campaign #158

from	subject
"Rocky" <Jellec@oz-on-control.com>	Best World source of your medical needs
"Nora" <Emielqaufi@franchise-az.com>	Re: World Original remedies for treating ED
"Jefferson" <Erwinvyjo@dtechbiz.com>	Re: World Licensed anti-ED drugs
"Lucinda" <Wouterycu@telefontamiri.com>	Re: Viagra World store from Pfizer
"Dianna" <Casparvura@nt-air.com>	Best prices on pharm pills
"Darius" <Frankheso@hobartours.com>	Re: World FDA approved ED pills
"Kurtis" <Patrickomiza@forum-erotyczne.com>	Re: Online World Drug Store, Big Discounts
"Margret" <Jorisvrwhl@itdprose.com>	Re: World Branded pilules for Sale
"Cynthia" <Derkvui@tapone.com>	World BrandStore of Pharmacy
"Josue" <Bertzunjw@vemorrerconnosco.com>	Re: World Viagra is real only here
"Barbara" <Corneliazfyai@christchurchtours.com>	Generics Vs Brand Name Drugs

paings is often fraudulent too, or at least it involves the illegal purchase of drugs. See for example campaign 181, table 4.7. A small minority of campaigns falls under what is called spear phishing. Usually, the attacker crafts a seemingly reputable email either linking to a malicious payload or having it as an attachment. Social engineering is used here too. The subjects and attachment filenames are often either urgent or relating to a serious matter (like an invoice). The attachment itself is rarely the payload, as email clients may scan these. Instead, they are word, excel or pdf documents that contain a link to the malicious payload or download it in the background if macros are enabled. See for example campaign 8, table 4.2.

Finally, we showcase three more oddities encountered:

- Campaign 184, table 4.8, shows a very creative social engineering campaign revolving around facebook. Ironically, the attacker set the x-mailer to `ZuckMail [version 1.00]`, a subtle reference to Facebook creator Mark Zuckerberg.
- Campaign 121, table 4.9, shows a campaign where the sender email is randomised, but there is a mismatch between the Address Book style name and the actual email address.
- Campaigns 83, table 4.10, presents machine made artifacts in the subjects. These artefacts not only allow us to understand something about the attacker tooling but also allows us to further fingerprint this specific attacker in the future.

Table 4.4: Campaign #188

from	subject
"Bruno Coley" <Lynn_Louis@nnawheel.com>	Want that position get the Degree
"Maxwell Hoffman" <Leblanc_Jan@arkas.com>	Look what you are up against...without a Degre...
Latoya Rodgers <Grant_Billie@laudicom.com>	You have the on job experience now get the Deg...
"Alberto Gibbons" <Plummer_Mercedes@sanibeland...>	Look what you are up against...without a Degre...
"Troy Ramirez" <Leach_Sam@imprimerie-remy.com>	You have the training now get the Degree to m...
Jon Vinson <Calhoun_Roberta@nurturehq.com>	Be the best...Get that Degree!
"Nicholas Erwin" <Funk_Robby@zenners.com>	Be the best out there with our Degree
"Becky Newsome" <Aldridge_Audrey@restaurantdul...>	Everyone has a degree but not everyone has exp...
"Haley Wooten" <Estrada_Tyree@chillhouseadrasa...>	Separate yourself from all the rest with a Degree

Table 4.5: Campaign #245

from	subject	xmailer	hashes
"Jordyn" <starosta@teleneuritec.googleboobs162...>	veel van dames uit badoo.com in uw Burgh wachten.	'Microsoft Windows Live Mail 15.4.3550.950'	92fe6a...
"Jessie" <cleistogamousc@abjectivea.googleboob...>	Sometimes, life has a cruel sense of humor, gi...	'Microsoft Windows Live Mail 15.4.3543.943	610fea...
"Simone" <atlantav@transuraniumx.googleboobs16...>	The one you love and the one who loves you are...	Microsoft Windows Live Mail 15.4.3500.900	e5deac...
"Geneva" <coelenteratel@feodatorym.googleboobs...>	her scent fades from the pillows, and even fro...	Microsoft Windows Live Mail 15.4.3579.979	29a65f...
"Katelin" <whitingsm@dissentiousz.googleboobs1...>	veel van kutjes uit badoo.com in uw stad wachten.	Microsoft Windows Live Mail 15.4.3599.999	6163ee...
"Sasha" <amentaln@acyrologyd.googleboobs162.net>	If you gave someone your heart and they died, ...	Microsoft Windows Live Mail 15.4.3596.996	54d990...
"Nina" <supportivelyn@taliacotianb.googleboobs...>	zoveel van dames uit badoo.com in uw plek wach...	Microsoft Windows Live Mail 15.4.3571.971	9d1221...

Table 4.6: Campaign #135

from	subject	xmailer
"zx" <helpsofthelpsoft@helpsoft.com.cn>	MAKE YOUR SPACE SHINE: handmade oil painting !	'Lbmk 6'
"fgc" <cyjidoka@cyjidoka.com>	MAKE YOUR SPACE SHINE: handmade oil painting !	'Vdryldnvh 1'

Table 4.7: Campaign #181

from	subject	hashes
Winston Heard <Starnes_Eugenio@telefontamiri.com>	Superior Quality male pills	6b9059...
Ellen Bledsoe <Estes_Katy@investmenttrustforum...>	Classic orgasm volumizer	178be0...
Roderick Jimenez <Lowry_Emery@studiodeko.com>	Genuine erectile vitamins	4d3526...

Table 4.8: Campaign #184

from	subject	xmailer	hashes
"Facebook" <notification+zj4ozz926stc@facebook...>	Ken je Jeanine Triezenberg, Fabienne Mok en 8 ...	ZuckMail [version 1.00]	[897c2b..., 85f...
"Facebook" <notification+zj4ozz926stc@facebook...>	Peter V Beusekom, Paul Molenaar en 4 anderen w...	ZuckMail [version 1.00]	[535f16..., a5d...
"Facebook" <notification+zj4ozz926stc@facebook...>	Je hebt meer vrienden op Facebook dan je denkt	ZuckMail [version 1.00]	[2b0a49..., ecd...

Table 4.9: Campaign #121

from	subject	hashes
"Rhoda Robles" <Whaley_Porfirio@erenkoysutesis...>	I'm ready for you baby and I know what you want.	3d8ed4...
"Rosemary Cruz" <Champion_Allie@aumod.com>	Buy Male Sexual Enhancement Pills	fc48b6...
"Salvador Dow" <Compton_Patsy@visualdbsee.com>	Then our pilules are of great usefulness for you!	ab67b2...
"Jeremy Root" <Murphy_Sam@lumenique.com>	Special RX from Canada for men	74684f...

Table 4.10: Campaign #83

from	subject	hashes
Nat Calhoun <uniglobularj@uncordedn.dating-8.bid>	=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik wil u te ont...	c9d90e..., d13...
Robyn <notabilityp@hydroterpeneq.dating-49.bid>	=?UTF-8?Q?=F3=BE=AC=8D_?= ... Niet bekend te ...	e14fd0..., c9d...
Brant <endosclerited@katharinat.dating-35.bid>	=?UTF-8?Q?=F3=BE=AC=8D_?= ... Ik heb soms wil...	c9d90e..., 97a...
Krista <wateragesp@boreadesv.luckyd-88.bid>	=?UTF-8?Q?=F3=BE=AC=8D_?= ... Niet bekend te ...	2cd7ad..., c9d...
Vicky <algometryp@herdshipp.dating-49.bid>	=?UTF-8?Q?=F3=BE=AC=8D_?= ... Mijn favoriete ...	96e0ae..., c9d...
Ali <sterilizationsl@tablemaidq.dating-48.bid>	=?UTF-8?Q?=F3=BE=AC=8D_?= ... We houden met e...	c9d90e..., e8e...

5

Conclusion and Future Work

We now present our conclusions and suggests the possible future work, both concerning the improvement of our framework and the new research that can be performed with the output of our framework.

In this thesis project, we have addressed the task of performing campaign analysis. Campaign analysis is the Threat Intelligence process of grouping together intrusion attempts that are orchestrated by the same threat actors against the same victims with a specific goal. Lockheed Martin first introduced this process in its 2011 seminal white paper “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains”. The researchers of Lockheed Martin came up with campaign analysis to cope with Advanced Persistent Threats, highly motivated, resourceful and skilled groups that started to become increasingly problematic at the beginning of the decade.

State-sponsored APTs typically engage in acts of cyberwarfare (like the Stuxnet hack) and espionage (both industrial and traditional). To do so, they develop bleeding edge tools and techniques. Unfortunately, these tend to eventually become available to less skilled groups, including traditional criminal groups, when they are disclosed irresponsibly to the public. Irresponsible disclosures vastly increased the number of entities capable of performing complex hacking: if before the rise of the APTs and their techniques, for most organisations the eggshell model of security (composed of a single hardened layer of defences at its border) sufficed, now this is just not good enough. As the threshold for advanced hacking lowered, it became evident that a new model of security, multi-layered and pervasive, was necessary. Threat Intelligence emerged as a valuable process to improve defences according to this new strategy. Threat Intelligence has the goal of studying threats techniques, tools, motives and by allowing to track them through time. Lockheed Martin campaign analysis is a process to perform data-driven threat intelligence by linking together related intrusion attempts (called kill chains in the model) by looking for repeating elements. This methodology is similar to a police investigation discovering a serial killer by finding peculiarities that repeat across a multitude of seemingly unrelated murders.

Unfortunately, campaign analysis is a process rarely performed in practice. There are several reasons behind this, one of them is that there is a lack of automated framework to perform it. A medium-large organisation would likely need hundreds of security experts to establish connections between all incidents manually. Needless to say, this makes only large organisations where security is an absolute necessity (like Lockheed Martin) capable and willing to perform it manually.

Having highlighted the research gap, lack of automation in campaign analysis, we established three research questions:

1. Can campaign analysis be performed with an existent clustering technique?
2. Does approximate match of suitable indicators improve recall performance of campaign analysis?
3. Can a methodology be developed to assess what indicators are most suitable (and what are not) for approximate match?

We collaborated with Dutchsec, a cybersecurity company based in Utrecht that provided us with an extensive collection of malicious emails. We first extracted interesting features from these emails and

further found more by setting up a malware lab where we executed all the malicious attachments of the emails, for a total of 68 features per kill chain.

We then designed a framework to perform automatic clustering of campaigns using two steps of density-based clustering. We tried to select the best methodology by assessing the performance of two density-based clustering algorithms: DBSCAN and HDBSCAN. Furthermore, to answer our third research question, we tested the two algorithms on two versions of our framework: one allowing only exact match of features and one allowing approximate string matching for some features. Only exact match of features is performed in the state of the art. We selected the features more appropriate for approximate string matching using both domain knowledge and by performing a statistical analysis of their entropy.

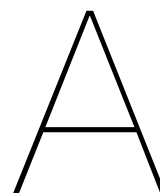
Finally, we obtained the following results:

- Campaign analysis can be automated and improved (concerning recall) by using density-based clustering.
- Density-based clustering adds a layer of fuzziness that allows improving recall compared to the traditional manual approach. In particular, it allows discovering campaigns even when they are characterised by randomisation of features to avoid classic campaign analysis.
- Density-based campaign analysis comes at the price of a loss in precision when compared to the traditional manual approach.
- Approximate string matching does not help in automating campaign analysis when density-based clustering is performed. Indeed, recall seems to worsen compared to when exact only string matching of features is performed.
- HDBSCAN performs better than DBSCAN for this task in terms of recall, which is more important than precision as finding complete campaigns is harder than dealing with noise. Furthermore, HDBSCAN is far superior in ease of use as the parameter tuning is minimal compared to DBSCAN, which require extensive trials and error.

We stress the importance of understanding the scenario where we imagine this framework to be utilised. This framework does not replace a security analyst. A security analyst is necessary to evaluate the found campaigns and eventually reject them. Our system should be seen as an aid in performing exploratory threat intelligence. We automated the expensive and tedious process of linking possibly related incidents, allowing analysts to be able to perform campaign analysis even in a large organisation dealing with a lot of data. In other words, we believe that our framework can scale down significantly the number of human resources needed to perform this process. Hopefully, lowering this threshold will allow small and medium organisations that do not have security as a critical requirement to finally perform their own threat intelligence.

We believe this framework is only a good starting point, as there is room for improvement. First of all, it would be interesting to understand if other clustering techniques could be used. Density-based clustering seems to do a good job overall, but it is possible that other techniques perform even better, especially in terms of precision. Likewise, it would also be interesting to study the impact of using different similarity metrics than the ones we developed to assess the similarity of intrusion attempts before the final round of clustering. We fine-tuned our system to cluster spear phishing campaigns because this kind of campaign was often occurring in our dataset. It would be interesting to also work with real APTs crafted emails. Unfortunately, these emails tend to be redacted when released to the public, so the best way to do this would be to work with an organisation that experienced APT hacking directly. The malware found in our dataset is old, and for the most part, the command and control server with which the malware was designed to communicate with is not online anymore. This means that when we tested the malware in our malware lab, we were only able to retrieve static features (belonging to the code itself) and behavioural (its actions), but not network communications. It would be very interesting to study the performance of our system on fresh malware. This has to be done in the first days, as the malware lifecycle tends to be rather short. The malware lab component itself can be improved by switching to spender-sandbox, a highly praised fork of the cuckoo sandbox we set up and used.

Moreover, we would like to highlight the future work that can be performed now that automatic campaign analysis is possible. Our system performs unsupervised learning by assigning campaign labels to malicious emails. Once an analyst evaluates these campaign labels, they could be used for a multitude of supervised learning tasks, for example, automatic classifications of emails in a live system.



Kill chain Features

	indicator name	h5 filename	DBSCAN eps	weight
1	additional_info.exiftool.AppVersion	VirusTotal	0.02	MEDIUM
2	additional_info.exiftool.Application	VirusTotal	0.02	LOW
3	additional_info.exiftool.CreateDate	VirusTotal	0.02	LOW
4	additional_info.exiftool.Creator	VirusTotal	0.02	LOW
5	additional_info.exiftool.DocSecurity	VirusTotal	0.02	LOW
6	additional_info.exiftool.FileType	VirusTotal	0.02	LOW
7	additional_info.exiftool.LastModifiedBy	VirusTotal	0.02	LOW
8	additional_info.exiftool.LinksUpToDate	VirusTotal	0.02	LOW
9	additional_info.exiftool.ModifyDate	VirusTotal	0.02	LOW
10	additional_info.exiftool.Pages	VirusTotal	0.02	LOW
11	additional_info.exiftool.Words	VirusTotal	0.02	LOW
12	additional_info.exiftool.ZipBitFlag	VirusTotal	0.02	LOW
13	additional_info.exiftool.ZipCRC	VirusTotal	0.02	LOW
14	additional_info.exiftool.ZipCompressedSize	VirusTotal	0.02	LOW
15	additional_info.exiftool.ZipFileName	VirusTotal	0.55	HIGH
16	additional_info.exiftool.ZipModifyDate	VirusTotal	0.02	LOW
17	additional_info.exiftool.ZipRequiredVersion	VirusTotal	0.02	LOW
18	additional_info.magic	VirusTotal	0.02	LOW
19	additional_info.openxmlchecker.ole.num_macros	VirusTotal	0.02	LOW
20	body	Email	0.35	HIGH
21	content_type	Email	0.02	LOW
22	date	Email	0.02	LOW
23	domain	Email	0.02	LOW
24	file.crc32	Cuckoo	0.02	LOW
25	file.name	Cuckoo	0.55	MEDIUM
26	file.type	Cuckoo	0.02	LOW
27	filename	Email	0.55	MEDIUM
28	first seen	VirusTotal	0.02	LOW
29	from	Email	0.45	HIGH
30	geo.city	Email	0.02	LOW
31	geo.country.name	Email	0.02	LOW
32	hash.sha1	Email	0.02	20
33	headers.Content-Type	Email	0.02	LOW
34	headers.Date	Email	0.02	LOW
35	headers.Priority	Email	0.02	LOW
36	headers.Return-Path	Email	0.45	MEDIUM
37	headers.X-Mailer	Email	0.02	MEDIUM

38	info.category	Cuckoo	0.02	LOW
39	length	Email	0.02	HIGH
40	marks.category	Cuckoo	0.02	LOW
41	marks.ioc	Cuckoo	0.02	LOW
42	marks.type	Cuckoo	0.02	LOW
43	meta.Application-Name	Email	0.02	LOW
44	meta.Author	Email	0.02	MEDIUM
45	meta.Creation-Date	Email	0.02	LOW
46	meta.Keywords	Email	0.02	LOW
47	meta.Last-Author	Email	0.02	LOW
48	meta.Last-Modified	Email	0.02	LOW
49	meta.Last-Save-Date	Email	0.02	LOW
50	received_date	Email	0.02	LOW
51	remote_addr	Email	0.02	LOW
52	reverse	Email	0.5	MEDIUM
53	scans.Avast.detected	VirusTotal	0.02	LOW
54	scans.Avast.result	VirusTotal	0.02	LOW
55	scans.Avira.detected	VirusTotal	0.02	LOW
56	scans.Avira.result	VirusTotal	0.02	LOW
57	scans.BitDefender.result	VirusTotal	0.02	LOW
58	scans.Kaspersky.detected	VirusTotal	0.02	LOW
59	scans.Kaspersky.result	VirusTotal	0.02	LOW
60	scans.Symantec.detected	VirusTotal	0.02	LOW
61	scans.Symantec.result	VirusTotal	0.02	LOW
62	sig.description	Cuckoo	0.02	LOW
63	sig.name	Cuckoo	0.02	HIGH
64	subject	Email	0.475	HIGH
65	target.category	Cuckoo	0.02	LOW
66	to	Email	0.02	LOW
67	type	VirusTotal	0.02	LOW
68	unique_sources	VirusTotal	0.02	LOW

Table A.1: FLT: Feature List Table, with source h5 file used, DBSCAN eps parameter and weight used to compute killchain level similarity

	average	standard deviation
additional_info.exiftool.AppVersion	0.999512	0.019505
additional_info.exiftool.Application	1.000000	0.009562
additional_info.exiftool.CreateDate	1.000000	0.001055
additional_info.exiftool.Creator	1.000000	0.007673
additional_info.exiftool.DocSecurity	1.000000	0.001293
additional_info.exiftool.FileType	1.000000	0.001056
additional_info.exiftool.LastModifiedBy	0.883789	0.320369
additional_info.exiftool.LinksUpToDate	1.000000	0.001055
additional_info.exiftool.ModifyDate	0.879883	0.320892
additional_info.exiftool.Pages	1.000000	0.007234
additional_info.exiftool.Words	1.000000	0.010360
additional_info.exiftool.ZipCRC	0.997559	0.048365
additional_info.exiftool.ZipCompressedSize	0.999023	0.015339
additional_info.exiftool.ZipFileName	0.999023	0.024797
additional_info.exiftool.ZipModifyDate	0.996094	0.052454
additional_info.exiftool.ZipRequiredVersion	0.966797	0.115283
additional_info.magic	0.999023	0.020724
additional_info.openxmlchecker.ole.num_macros	1.000000	0.000528
body	0.946289	0.151886

content_type	0.766113	0.274613
date	0.921875	0.076555
domain	0.989746	0.054091
file.crc32	0.997559	0.036955
file.name	0.974609	0.061529
file.sha1	0.990234	0.037078
file.type	0.413330	0.461046
filename	0.999023	0.015367
first seen	0.753906	0.339417
from	0.987305	0.049153
geo.city	0.968262	0.173706
geo.country.name	0.932129	0.247122
hash.sha1	0.980469	0.106910
headers.Content-Type	0.881348	0.219213
headers.Date	0.884277	0.102068
headers.Priority	1.000000	0.000707
headers.Return-Path	0.998047	0.011546
headers.X-Mailer	0.999023	0.027572
info.category	0.000000	0.000000
length	0.995117	0.066848
marks.category	0.126709	0.332604
marks.ioc	0.915039	0.225819
marks.type	0.000000	0.000000
meta.Application-Name	1.000000	0.000421
meta.Author	1.000000	0.000085
meta.Creation-Date	1.000000	0.000397
meta.Last-Author	1.000000	0.000060
meta.Last-Modified	1.000000	0.000379
meta.Last-Save-Date	1.000000	0.000379
received_date	0.965820	0.039554
remote_addr	0.982910	0.061133
reverse	0.991699	0.058219
scans.Avast.detected	0.875977	0.317107
scans.Avast.result	0.264160	0.440917
scans.Avira.detected	0.767578	0.278314
scans.Avira.result	0.296875	0.456917
scans.BitDefender.result	0.239014	0.426450
scans.Kaspersky.detected	0.756836	0.331052
scans.Kaspersky.result	0.275635	0.446763
scans.Symantec.detected	0.826172	0.340801
scans.Symantec.result	0.507812	0.499828
sig.description	0.593262	0.453618
sig.name	0.524902	0.426821
ssdeep	0.994629	0.057752
subject	0.967773	0.121797
target.category	0.000000	0.000000
to	0.961914	0.112798
type	0.998535	0.023553
unique_sources	0.582520	0.493161

Table A.2: Average and Standard Deviation of indicators cosine distance. Week 12 2016 (March 21st to March 27th)

B

Synthetic campaigns

Table B.1: Synthetic Campaign S1

from	subject	xmailer	filename	sha1
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb
George Lucas <george.lucas@scriptkiddo.com>	free movies download now	BOMBERMAILLLLL 99.9585B	popcornimeisback.exe	63425468248833699c63caa17d29068fbc7662fb

Table B.2: Synthetic Campaign S2

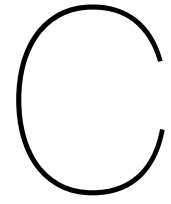
from	subject	xmailer	filename	sha1
Steve Steve <steve.steve@banzaimail-camo.com>	Your trip to Brussels invoice#384823	WONDERMAIL 7.777.6	INVOICE-688920.pdf	63425468343792699c63caa17d29068fbc7662fb
Steven Lucas <steven.lucas@banzaimail-camo.com>	Your trip to Brussels invoice#630571	WONDERMAIL 7.777.6	INVOICE-343792.pdf	63425468662073699c63caa17d29068fbc7662fb
Steven Bolak <steven.bolak@banzaimail-camo.com>	Your trip to Brussels invoice#790210	WONDERMAIL 7.777.6	INVOICE-662073.pdf	63425468747793699c63caa17d29068fbc7662fb
Monster Luas <monster.luas@banzaimail-camo.com>	Your trip to Brussels invoice#837581	WONDERMAIL 7.777.6	INVOICE-747793.pdf	63425468948467699c63caa17d29068fbc7662fb
Randy Bolak <Randy.bolak@banzaimail-camo.com>	Your trip to Brussels invoice#572893	WONDERMAIL 7.777.6	INVOICE-948467.pdf	63425468764343699c63caa17d29068fbc7662fb
Cptain Jordan <cptain.jordan@banzaimail-camo.com>	Your trip to Brussels invoice#894844	WONDERMAIL 7.777.6	INVOICE-764343.pdf	63425468203023699c63caa17d29068fbc7662fb
Randy Jordan <randy.jordan@banzaimail-camo.com>	Your trip to Brussels invoice#880807	WONDERMAIL 7.777.6	INVOICE-203023.pdf	63425468355612699c63caa17d29068fbc7662fb
Monster Jordan <monster.jordan@banzaimail-camo...>	Your trip to Brussels invoice#327095	WONDERMAIL 7.777.6	INVOICE-355612.pdf	63425468311790699c63caa17d29068fbc7662fb
Ellie Luas <ellie.luas@banzaimail-camo.com>	Your trip to Brussels invoice#765173	WONDERMAIL 7.777.6	INVOICE-311790.pdf	63425468283755699c63caa17d29068fbc7662fb
Steven Jordan <steven.jordan@banzaimail-camo.com>	Your trip to Brussels invoice#688920	WONDERMAIL 7.777.6	INVOICE-283755.pdf	63425468213588699c63caa17d29068fbc7662fb

Table B.3: Synthetic Campaign S3

from	subject	xmailer	filename	sha1
George Lucas <george.lucas@starmail.com>	You are 10 weeks late for your star wars merch...	Force Mail client 117345	STARWARS-MERCH-DELAY-132337.pdf	63425468132337699c63caa17d29068fbc7662fb
Steven Lucas <steven.lucas@starmail.com>	10 weeks delay star wars merch	Force Mail client 885020	STARWARS-MERCH-DELAY-178165.pdf	63425468178165699c63caa17d29068fbc7662fb
Steven Bolak <steven.bolak@starzone.com>	You are 10 weeks late!!! Collect your merch!	Force Mail client 378391	STARWARS-MERCH-DELAY-827478.pdf	63425468827478699c63caa17d29068fbc7662fb
Monster Luas <monster.luas@starmail.com>	Star wars: waiting for your payment 10 weeks	Force Mail client 684614	STARWARS-MERCH-DELAY-799221.pdf	63425468799221699c63caa17d29068fbc7662fb
Randy Bolak <Randy.bolak@starmail.com>	10 weeks delay star wars merch	Force Mail client 983811	STARWARS-MERCH-DELAY-580233.pdf	63425468580233699c63caa17d29068fbc7662fb
Cptain Jordan <cptain.jordan@starzone.com>	Star wars: waiting for your payment 10 weeks	Force Mail client 610174	STARWARS-MERCH-DELAY-305531.pdf	63425468305531699c63caa17d29068fbc7662fb
Randy Jordan <randy.jordan@starmail.com>	NEW MERCH Star Wars order, pay!	Force Mail client 754355	STARWARS-MERCH-DELAY-896682.pdf	63425468896682699c63caa17d29068fbc7662fb
Monster Jordan <monster.jordan@starzone.com>	You forgot to pay us for 10 weeks, star wars m...	Force Mail client 605285	STARWARS-MERCH-DELAY-634752.pdf	63425468634752699c63caa17d29068fbc7662fb
Ellie Luas <ellie.luas@starmail.com>	10 weeks delay in payment, star wars merch inv...	Force Mail client 896582	STARWARS-MERCH-DELAY-434338.pdf	63425468434338699c63caa17d29068fbc7662fb
Steven Jordan <steven.jordan@starzone.com>	fine for 10 weeks delay in collect...	Force Mail client 937950	STARWARS-MERCH-DELAY-989545.pdf	63425468989545699c63caa17d29068fbc7662fb

Table B.4: Synthetic Campaign S4

from	subject	xmailer	filename	sha1
Rebekah Fletcher : ronewalloby-7907@yopmail.fr	AIAA 2018 Technical Committees invite	callforpapers-806	callforpapers-6740217.pdf	634254686740217699c63caa17d29068fbc7662fb
Maritza Schwartz : icoddari-8105@conaaail.com	IEEE 2018 Technical Conference	callforpapers-922	callforpapers-5956651.pdf	634254685956651699c63caa17d29068fbc7662fb
Valerie Stephens : lyja-3857@slimail.com	Invite 2018 NIPS Conference invite	callforpapers-459	callforpapers-1580009.pdf	634254681580009699c63caa17d29068fbc7662fb
Deirdre Benjamin : errarraxi-3256@nowaia.com	IEEE 2018 Technical Committee invite	callforpapers-434	callforpapers-3293285.pdf	634254683293285699c63caa17d29068fbc7662fb
Bernard Petersen : loleniqytti-6771@lebbrai.it	ACM 2018 conference call	callforpapers-118	callforpapers-7937423.pdf	634254687937423699c63caa17d29068fbc7662fb
Stewart Mcknight : uzovaxi-4848@chinaxx.com	ACM 2018 technical conference invite	callforpapers-195	callforpapers-6468500.pdf	634254686468500699c63caa17d29068fbc7662fb
Malinda Galloway : uzovaxi-4848@peoples.hk	ACM computing survey 2018	callforpapers-408	callforpapers-4742059.pdf	634254684742059699c63caa17d29068fbc7662fb
Norbert Chambers : ceffotaffe-1404@banzaii.com	AIAA 2018 call for papers invite	callforpapers-611	callforpapers-6120838.pdf	634254686120838699c63caa17d29068fbc7662fb
Domenic Caldwell : memminajatt-4145@libero.ch	IEEE 2018 conference call papers	callforpapers-385	callforpapers-4702158.pdf	634254684702158699c63caa17d29068fbc7662fb
Stanton Roberson : ywelillette-6696@libero.com	NIPS invite to conference 2018	callforpapers-901	callforpapers-3346929.pdf	634254683346929699c63caa17d29068fbc7662fb



Real Campaigns discovered manually

117

	m-from	m-subject	m-xmaler	p-hashes	p-filenames	P-length
0	Selena Ruiz <RuizSelena445@predatorcheck.com>	FW: Payment #239446	[e393bd1cd878ce3606a55ad93df6e515221cfbd3,5cb...	[urgent_payment_239446.zip,	[12446,]	
1	Gonzalo Mendez <MendezGonzalo89@ultranet.inf.br>	FW: Payment #035137	[59ddd32d5411de8ca0f0a2b91e5a0ee27b1f164f,5ae...	[, payment_info_035137.zip]	[, 11792]	
2	Barbara Perry <PerryBarbara2526@dhcp-089-098-1...>	FW: Payment #394763	[5ae0a84432e34ce23e0a02cd2f36fcc6d911fa9f,7b8...	[urgent_payment_394763.zip,	[11792,]	
3	Shelia Sargent <SargentShelia85@chello.nl>	FW: Payment #352372	[d9f8669cd7812db941733771286f0cab4d419b07,e39...	[, details_payment_352372.zip]	[, 12446]	
4	Rachelle Holmes <HolmesRachelle7609@chello.pl>	FW: Payment #336635	[50d911baaa2c1a12fd5a713962b2c1973148a7b1,e39...	[, payment_urgent_336635.zip]	[, 12446]	
5	Marilyn Hendrix <HendrixMarilyn512@stephenrain...>	FW: Payment #533935	[e393bd1cd878ce3606a55ad93df6e515221cfbd3,010...	[details_payment_533935.zip,	[12446,]	
6	Karia Chapman <ChapmanKaria81958@ono.com>	FW: Payment #790221	[cb80804900f8b91cf94aef6aadd3840e610ace41,e39...	[, details_payment_790221.zip]	[, 12446]	
7	Merlin Cardenas <CardenasMerlin7552@impsat.net...>	FW: Payment #346417	[b7cc7d47d56e3b1e28de3d3fa8646a33e03fe86c,e39...	[, payment_urgent_346417.zip]	[, 12446]	
8	Anibal Acosta <AcostaAnibal67@gtinternet.com>	FW: Payment #368845	[a7ac9fa7b9a82d64da8409198976fcb408522cc2,e39...	[, wire_payment_368845.zip]	[, 12446]	
9	Bryon Nguyen <NguyenBryon41@51-151.wind.it>	FW: Payment #211300	[b57c465d17affd45a167b6b473929b188f147a2d,6bb...	[urgent_payment_211300.zip,	[11863,]	
10	Shelley Crosby <CrosbyShelley52985@plus.pl>	FW: Payment #386144	[a5b61968a3ef34c76f3d04bc11890da87f3c2e44,9e2...	[, urgent_payment_386144.zip]	[, 12923]	
11	Ivy Noel <Noellivy25038@propulsestudio.com>	FW: Payment #931560	[db39e9d11f67f3358b13d5a2365ab1978f029dd0,db0...	[, payment_info_931560.zip]	[, 11989]	

Figure C.1: Excerpt of real campaign R1 observed in the dataset

161

	m-from	m-subject	m-xmaler	p-hashes	p-filenames	p-length
0	Lessie Long <LongLessie8646@chello.pl>	FW: Payment #837021	[ae3c0458599136c9a4fa787c64be2d9e23825f3f,1e9...	[,]	[4463, 4609]	
1	Sonja Meadows <MeadowsSonja795@semis.cz>	FW: Payment #391987	[889b01ea994363692f616db6acc21af92f0d9b80,5c3...	[,]	[4456, 4383]	
2	Carmen Meadows <MeadowsCarmen93@fibertel.com.ar>	FW: Payment #309953	[7efc47becfe5d6c3c55996ef7b197c17928ca970,83d...	[,]	[4288, 4271]	
3	Angelia Eaton <EatonAngelia326@static-190-181-...>	FW: Payment #992005	[dc92e9c062e41f9493321c06ad0f17f5bdc65820,79f...	[,]	[4600, 4425]	
4	Jodie Webster <WebsterJodie52640@ct.co.cr>	FW: Payment #943606	[57790a0ef85fce551d8e67c396fa423b7dc0d853,cfc...	[,]	[4558, 4771]	
5	Marcellus Sweeney <SweeneyMarcellus84@kolibriu...>	FW: Payment #682202	[489a776d9a8ce02e6ad43d30bfb36b325e674b4,c5b...	[,]	[5011, 4978]	
6	Lane Petersen <PetersenLane00@cmodem-244-154.t...>	FW: Payment #962261	[79fb635c8925e877515c8864c7a41155eab52e9a,dc9...	[,]	[4425, 4600]	
7	Loraine Jacobs <JacobsLoraine4288@ivaipisos.co...>	FW: Payment #198447	[e881bbdbd4e35848ab2e1b4214a11c7feaad22d,d26...	[,]	[4572, 4712]	
8	Alyson Higgins <HigginsAlyson927@vallow.com>	FW: Payment #736244	[890fd7ab231e138d0a18225b97152ba63d073160,14f...	[,]	[4850, 4487]	
9	Tonia Mcpherson <McphersonTonia822@orange.es>	FW: Payment #848369	[0f9678b5405cc59ab6a1360624095fe06e866a8a,0bb...	[,]	[4619, 4823]	
10	Marta Spencer <SpencerMarta35615@telesatelit.ro>	FW: Payment #638195	[c94be983eac3a8b8b7045a14fd5028873a2f8618,6d1...	[,]	[4662, 4861]	
11	Lila Joyner <JoynerLila03@ono.com>	FW: Payment #538423	[c59234ea9d40c37d91039aaf4d8ab7f35a70c447,3a4...	[,]	[4547, 4590]	
12	Opal Garcia <GarciaOpal73@dsdevice.home>	FW: Payment #014696	[c0261586afa3d586e44ea9286efea8a592598dec,d8e...	[,]	[, 4833, 4726]	
13	Nina Morgan <MorganNina992@cable.net.co>	FW: Payment #472127	[392068d2de953284de84ee5bdd59e3e343b1452b,990...	[,]	[, 4654, 4589]	
14	Aaron Berger <BergerAaron973@axtel.net>	FW: Payment #302850	[7efc47becfe5d6c3c55996ef7b197c17928ca970,c0d...	[,]	[4288, 4271]	
15	Tasha Bailey <BaileyTasha99448@olsztyr.vecfran...>	FW: Payment #906371	[22986b02ece714827598ff9ab13993d2ca310aed,38e...	[,]	[4300, 4232]	
16	Dariusz Bauer <BauerDariusz512@kita.pl>	FW: Payment #302850	[766e0d6fe04d6003dbd3215dba7f4a117fb2378e,	[,]	[, 4492]	

Figure C.2: Excerpt of real campaign R2 observed in the dataset

197

	m-from	m-subject	m-xmailer	p-hashes	p-filenames	P-length
0	<Scott.Tabitha48@dsldevice.lan>	Invoice NSINV37969 from Tip Top Delivery	[Everest CRM Studio']	[3814bcb4dc256456b190086e9cd76a48848d9a9b, d50...	Invoice_NSINV37969_from_tip_top_delivery.rt...	[, 14550,]
1	<Zimmerman.Cathy8@ropeney.com>	Invoice COINV95447 from Tip Top Delivery	[Everest CRM Studio']	[b1cece2b2dcad0ca7e50978965c973cf6a8459f1, de8...	Invoice_COINV95447_from_tip_top_delivery.rt...	[, 14551,]
2	<McCoy.Alfreda53@une.net.co>	Invoice OFINV76309 from Tip Top Delivery	[Everest CRM Studio']	[381d3197e218c5ec690224972e8c98f51132b38b, 887...	[, Invoice_OFINV76309_from_tip_top_delivery.rt...	[, 14551,]
3	<Morrison.Arthur12@edatel.net.co>	Invoice QXINV89673 from Tip Top Delivery	[Everest CRM Studio']	[ef4360b36e18372f8ee77c2ba1f4a0f5ea83eba, 887...	Invoice_QXINV89673_from_tip_top_delivery.rt...	[, 14551,]
4	<Finley.Jordan6@itpcosolutions.com>	Invoice OFINV15682 from Tip Top Delivery	[Everest CRM Studio']	[887c8caa2b7781ef29aaeda4833af4fa554aced7, 292...	[Invoice_OFINV15682_from_tip_top_delivery.rtf,...	[14551, ,]
5	<Carney.Penelope64@une.net.co>	Invoice EAINV74322 from Tip Top Delivery	[Everest CRM Studio']	[745f519e41610bd5a89edb1359ced486474cca7f, fa3...	[Invoice_EAINV74322_from_tip_top_delivery.rtf,...	[14546, ,]
6	<Potts.Sterling7@adsl.viettel.vn>	Invoice RFINV46097 from Tip Top Delivery	[Everest CRM Studio']	[e0a3a0f84f6348568bd8d874aa28dc2734874e7a, f95...	[, Invoice_RFINV46097_from_tip_top_delivery....	[, 14546]
7	<Mcintosh.Ezra06@superonline.net>	Invoice PNINV44855 from Tip Top Delivery	[Everest CRM Studio']	[de877ac5cf74ee9a735714ef6d1aada80991955b, ee2...	[, Invoice_PNINV44855_from_tip_top_delivery....	[, 14551]
8	<Whitley.Marina5@turktelekom.com.tr>	Invoice MOINV05908 from Tip Top Delivery	[Everest CRM Studio']	[0fdc751b6e4d3c806917aa51099de4394a448773, 0e8...	Invoice_MOINV05908_from_tip_top_delivery....	[, 14546]
9	<Sargent.Philip6@premiericcione.com>	Invoice JXINV22664 from Tip Top Delivery	[Everest CRM Studio']	[051071612032542d5903984cdff6e1bdabda8653, 5ca...	[, Invoice_JXINV22664_from_tip_top_delivery....	[, 14551]
10	<Lester.Luke1@plexusmoz.net>	Invoice FPINV33692 from Tip Top Delivery	[Everest CRM Studio']	[745f519e41610bd5a89edb1359ced486474cca7f, 953...	[Invoice_FPINV33692_from_tip_top_delivery.rtf,...	[14546, ,]
11	<Becker.Eugene5@yourmacdoctor.com>	Invoice THINV13305 from Tip Top Delivery	[Everest CRM Studio']	[67907044c1b4668844abfabe281a0106e6f73fb8, 194...	[, Invoice_THINV13305_from_tip_top_delivery....	[, 14546]

Figure C.3: Excerpt of real campaign R3 observed in the dataset

Bibliography

- [1] Rohan M. Amin, Julie J. C. H. Ryan, and Johan van Dorp. Detecting targeted malicious email. *IEEE Security & Privacy*, 10(3):64–71, 2012. doi: 10.1109/MSP.2011.154. URL <https://doi.org/10.1109/MSP.2011.154>.
- [2] Parth Bhatt, Edgar Toshiro Yano, and Per M. Gustavsson. Towards a framework to detect multi-stage advanced persistent threats attacks. In *8th IEEE International Symposium on Service Oriented System Engineering, SOSE 2014, Oxford, United Kingdom, April 7-11, 2014*, pages 390–395, 2014. doi: 10.1109/SOSE.2014.53. URL <https://doi.org/10.1109/SOSE.2014.53>.
- [3] Blake D. Bryant and Hossein Saiedian. A novel kill-chain framework for remote security log analysis with SIEM software. *Computers & Security*, 67:198–210, 2017. doi: 10.1016/j.cose.2017.03.003. URL <https://doi.org/10.1016/j.cose.2017.03.003>.
- [4] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.
- [5] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *TKDD*, 10(1):5:1–5:51, 2015. doi: 10.1145/2733381. URL <http://doi.acm.org/10.1145/2733381>.
- [6] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico*, pages 73–78, 2003. URL <http://www.isi.edu/info-agents/workshops/ijcai03/papers/Cohen-p.pdf>.
- [7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.
- [8] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969. doi: 10.1080/01621459.1969.10501049. URL <https://www.tandfonline.com/doi/abs/10.1080/01621459.1969.10501049>.
- [9] Dan Guido. A case study of intelligence-driven defense. *IEEE Security & Privacy*, 9(6):67–70, 2011. doi: 10.1109/MSP.2011.158. URL <https://doi.org/10.1109/MSP.2011.158>.
- [10] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.
- [11] Karen Spärck Jones. IDF term weighting and IR research lessons. *Journal of Documentation*, 60(5):521–523, 2004. doi: 10.1108/00220410410560591. URL <https://doi.org/10.1108/00220410410560591>.
- [12] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240. doi: 10.1002/widm.30. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.30>.
- [13] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 296–304, 1998.

- [14] William J Lynn III. Defending a new domain: The pentagon's cyberstrategy. *Cyberwar Resources Guide*, (121), 2010.
- [15] James Rutherford and Gregory B. White. Using an improved cybersecurity kill chain to develop an improved honey community. In *49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016*, pages 2624–2632, 2016. doi: 10.1109/HICSS.2016.329. URL <https://doi.org/10.1109/HICSS.2016.329>.
- [16] Charles Smutz and Angelos Stavrou. Malicious PDF detection using metadata and structural features. In *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*, pages 239–248, 2012. doi: 10.1145/2420950.2420987. URL <http://doi.acm.org/10.1145/2420950.2420987>.
- [17] Peter P. Stavroulakis and Mark Stamp, editors. *Handbook of Information and Communication Security*. Springer, 2010. ISBN 978-3-642-04116-7. doi: 10.1007/978-3-642-04117-4. URL <https://doi.org/10.1007/978-3-642-04117-4>.
- [18] Konstantin Tretyakov. Machine learning techniques in spam filtering. In *Data Mining Problem-oriented Seminar, MTAT*, volume 3, pages 60–79, 2004.
- [19] Kiri Wagstaff. Machine learning that matters. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012. URL <http://icml.cc/2012/papers/298.pdf>.
- [20] Daniel S Wilks. Cluster analysis. In *International geophysics*, volume 100, pages 603–616. Elsevier, 2011.
- [21] Rui Xu and Donald C. Wunsch II. Survey of clustering algorithms. *IEEE Trans. Neural Networks*, 16(3):645–678, 2005. doi: 10.1109/TNN.2005.845141. URL <https://doi.org/10.1109/TNN.2005.845141>.