Network-based detection of malicious activities - a corporate network perspective

Kidmose, Egon

DOI (link to publication from Publisher):
10.54337/aau300041111

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Kidmose, E. (2019). *Network-based detection of malicious activities - a corporate network perspective*. Aalborg Universitetsforlag. https://doi.org/10.54337/aau300041111

NETWORK-BASED DETECTION OF MALICIOUS ACTIVITIES – A CORPORATE NETWORK PERSPECTIVE

BY
EGON KIDMOSE

DISSERTATION SUBMITTED 2018

AALBORG UNIVERSITY
DENMARK

Network-based detection of malicious activities – a corporate network perspective

PhD Dissertation

Egon Kidmose

Dissertation submitted November 9, 2018

Dissertation submitted:	November 9, 2018
PhD supervisor:	Assoc. Prof. Jens Myrup Pedersen Dept. of Electronic Systems, Aalborg University
PhD Co-Supervisor:	Infrastructure Engineer M.Sc. Søren Brandbyge LEGO System A/S
PhD committee:	Associate Professor René Rydhof Hansen (chairman) Aalborg University
	Dr. Cyril Onwubiko Recearch Series
	Professor Michal Choras UTP University of Science and Technology
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Electronic Systems

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-356-3

Published by:
Aalborg University Press
Langagervej 2
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2018

Abstract

This dissertation is concerned with exploring how corporations can mitigate security threats from the Internet. The described research delves into two distinct topics. First, we improve on correlation and filtering of alerts from Intrusion Detection Systems to make it feasible in practice. Second, we explore how to detect malicious and abusive domain names, in order to block their usage and disable threats depending on the Domain Name System.

Threats from the Internet are increasingly relevant as corporations continue to adopt processes that make use of the Internet, thereby increasing trust in the inherently insecure network of networks belonging to many different entities. This development adds to and multiplies the threats from malicious entities, as systems enabling business processes see increased Internet-connectivity and thereby exposure to e.g. cybercriminals.

Correlation and filtering is explored, in particular how it can be performed without depending on costly feature engineering and tuning, thereby offering a feasible solution to the problem of too many correlated and false alerts from Intrusion Detection Systems. By introducing a general approach that precludes feature engineering and requires that alerts are ingested as text without assumptions on the format, we argue that our methods achieve significant lower deployment costs than existing methods, which makes practical applications feasible. Two presented implementations, one based on Recurrent Neural Networks and one based on Latent Semantic Analysis, are evaluated on public data, and found relevant to consider for practical use.

Domain names, the Domain Names System, and abuse of both, are explored with a focus on the resulting threats. One finding in the analysis is that pre-registration detection is a promising approach for efficient prevention of many threats because it applies before the registration process for a domain is completed. Subject to accurate prediction of malicious intents, threats that rely on domain names can be mitigated efficiently by blocking registrations. A novel method for analysing domain name blacklists is proposed. The method applies over time and covers entire blacklists, as opposed to a sampled subset. It is demonstrated how lexical analysis on domain names can contribute to recognising malicious domain names. Finally, a method to develop heuristics based on analysis of cybercriminal schemes and techniques is presented and applied. It is found suitable for guiding an efficient manual effort to identify malicious domains from a subset of a Top-Level Domain.

The main contributions include: A proposal for and evaluation of a method for feasible correlation and filtering of alerts. A definition of pre-registration detection, and separate studies indicating that it is achievable through lexical analysis of domain names and heuristics developed from cybercriminal schemes and techniques. Methods to analyse blacklists and large sets of domains, which can help to establish a ground truth on abusive domains in future work on implementing and evaluating pre-registrations detection.

Resumé

Denne afhandling er en undersøgelse af hvorledes virksomheder kan håndtere sikkerhedstrusler fra Internettet. Den beskrevne forskning går i dybden med to specifikke emner. Som det første forbedres korrelering og filtrering af alarmer fra Intrusion Detection Systemer ved at gøre det praktisk muligt. Som det andet undersøges hvordan ondsindede og krænkende domænenavne kan detekteres. Dette gøres med henblik på at blokere deres brug og uskadeliggøre trusler der afhænger af Domain Name System.

Trusler fra Internettet har stigende relevans, da virksomheder fortsat inkorporerer processer som gør brug af Internettet. Dette implicerer en øget tiltro til dette usikre netværk af netværk, som ejes af mange forskellige aktører. I takt med at systemer som understøtter forretningsgange i øget grad forbindes til internettet, forøges og forstærkes eksponeringen til, og truslerne fra, ondsindede aktører såsom cyberkriminelle.

Det undersøges hvordan korrelering og filtrering kan udføres uden brug af omkostningstung Feature Engineering og systemtilpasning, for således at tilbyde en praktisk opnåelig løsning på det problem at Intrusion Detection Systemer genererer for mange korrelerede og falske alarmer. Ved at introducere en general tilgang, som udelukker Feature Engineering og kræver at alarmer behandles som tekst, uden antagelser om formatet, vurderes det at der opnås væsentligt lavere omkostninger ved udrulning, hvilket muliggør praktisk anvendelse. To implementeringer, en baseret på Rekurrerende Neurale Netværk og en baseret på Latent Semantisk Analyse, præsenteres og evalueres på offentlig tilgængelige data, og findes relevante at betragte til praktisk anvendelse.

Domænenavne, Domain Name Systemet og misbrug af begge undersøges, med fokus på de trusler der følger deraf. Et af analysens fund er at præ-registreringsdetektion er en lovende tilgang for at opnå effektiv afværgelse af mange trusler da detektion virker inder registreringen af et domæne fuldføres. Forudsat fejlfri detektion vil trusler der afhænger af domænenavne kunne uskadeliggøres ved at afvise registreringer af pågældende domænenavne, hvilket er yderst effektivt. En ny metode til at analysere sortlister over domænenavne præsenteres. Metoden betragter sortlisterne over tid og dækker alle sortlistede domæner fremfor blot et udsnit. Det demonstreres hvorledes leksikalsk analyse af domænenavne muliggør detektion af ondsindede domænenavne. Slutteligt præsenteres og anvendes en metode til at udvikle heuristik fra analyse af cyberkriminelles modus operandi og deres teknikker. Denne findes anvendelig til at styre en effektiv manuel indsats for at finde ondsindede domæner fra en delmængde af et Top-Level Domæne.

De primære bidrag omfatter: En foreslået metode til praktisk opnåelig korrelering og filtrering af alarmer, herunder evaluering af metoden. En definition af præ-registreringsdetektion, samt selvstændige studier som indikerer at dette kan opnås ved hjælp af leksikalsk analyse af domænenavne, samt heuristik udviklet fra cyberkriminelles modus operandi og teknikker. Metoder til at analysere sortlister og store mængder af domæner, hvilket kan bidrage til at etablere den underliggende sandhed i fremtidigt arbejde med at implementere og evaluere præ-registreringsdetektion.

Curriculum Vitae

Egon Kidmose

Egon Kidmose received the BScEng (Computer Engineering) in 2012 and the MScEng (Networks and Distributed Systems) in 2014, both from Aalborg University. His master thesis was a proposal on how Hidden Markov Models can be applied to reduce the number of Intrusion Detection System alerts pertaining to bot malware infections.

His academic experiences, prior to commencing the PhD study, includes a visit of four months, hosted by the Security & Machine Learning Research Group, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, in 2013, focusing on machine learning for network traffic classification, as well as a seven month employment with the Communication Systems group, Dept. of Engineering, Aarhus University, during 2014 and 2015, doing research on privacy and security in the Smart Grid.

In May 2015 Egon commenced his PhD study titled "Network-based detection of malicious activities – a corporate network perspective", under the Industrial PhD programme of Innovation Fund Denmark. He enrolled with the Technical Doctoral School of IT and Design at Aalborg University and in accordance with the Industrial PhD programme he also joined LEGO System A/S, as an infrastructure engineer within Corporate IT.

Egon is interested in IT security in general, including corporate and applied perspectives. He is particularly interested in network security, incident detection, machine learning and application of Big Data methods for security.

List of publications

The following is a list of my peer-reviewed publications, including one that predates the PhD project($*$), three that are published during the PhD project but not included in this dissertation(†), and one that is accepted but not published at the time of submission(‡). One manuscript currently in review and included in this dissertation is not listed, see Chapter 5.

1. E. Kidmose, E. S. M. Ebeid, and R. H. Jacobsen, "A framework for detecting and translating user behavior from smart meter data," in *Smart Systems, Devices and Technologies (Smart), International Conference on*. IARIA, 2015, pp. 71–74.*
2. E. Kidmose, M. Stevanovic, and J. M. Pedersen, "Correlating intrusion detection alerts on bot malware infections using neural network," in *International Conference on Cyber Security and Protection of Digital Services (Cyber Security 2016)*, C-MRIC. IEEE, 2016, pp. 195–211.
3. K. Shahid, E. Kidmose, R. L. Olsen, L. Petersen, and F. Iov, "On the impact of cyberattacks on voltage control coordination by regen plants in smart grids," in *2017 IEEE International Conference on Smart Grid Communications*, IEEE, 2017, pp. 23–26.†
4. E. Kidmose and J. M. Pedersen, "Security in internet of things," in *Cybersecurity and Privacy-Bridging the Gap*. River Publishers, 2017, pp. 99–118.†
5. E. Kidmose, K. Gausel, S. Brandbyge, and J. M. Pedersen, "Assessing usefulness of blacklists without the ground truth," in *10th International Conference on Image Processing and Communications*, 2018.
6. E. Kidmose, E. Lansing, S. Brandbyge, and J. M. Pedersen, "Detection of malicious and abusive domain names," in *Data Intelligence and Security (ICDIS), 2018 1st International Conference on*. IEEE, 2018, pp. 49–56.
7. E. Kidmose, M. Stevanovic, and J. M. Pedersen, "Detection of malicious domains through lexical analysis," in *Cyber Security And Protection Of Digital Services (Cyber Security), 2018 International Conference on*. IEEE, 2018, p. 64–56.
8. J. M. Pedersen and E. Kidmose, "Security in internet of things: Trends and challenges," in *BIR 2018 Short Papers, Workshops and Doctoral Consortium co-located with 17th International Conference on Perspectives in Business Informatics Research (BIR 2018)*, 2018, pp. 182–188.†
9. E. Kidmose, E. Lansing, S. Brandbyge, and J. M. Pedersen, "Heuristic methods for efficient identification of abusive domain names," accepted for *International Journal on Cyber Situational Awareness (IJCSA), Vol. 3, No. 1*, 2018.‡

Preface

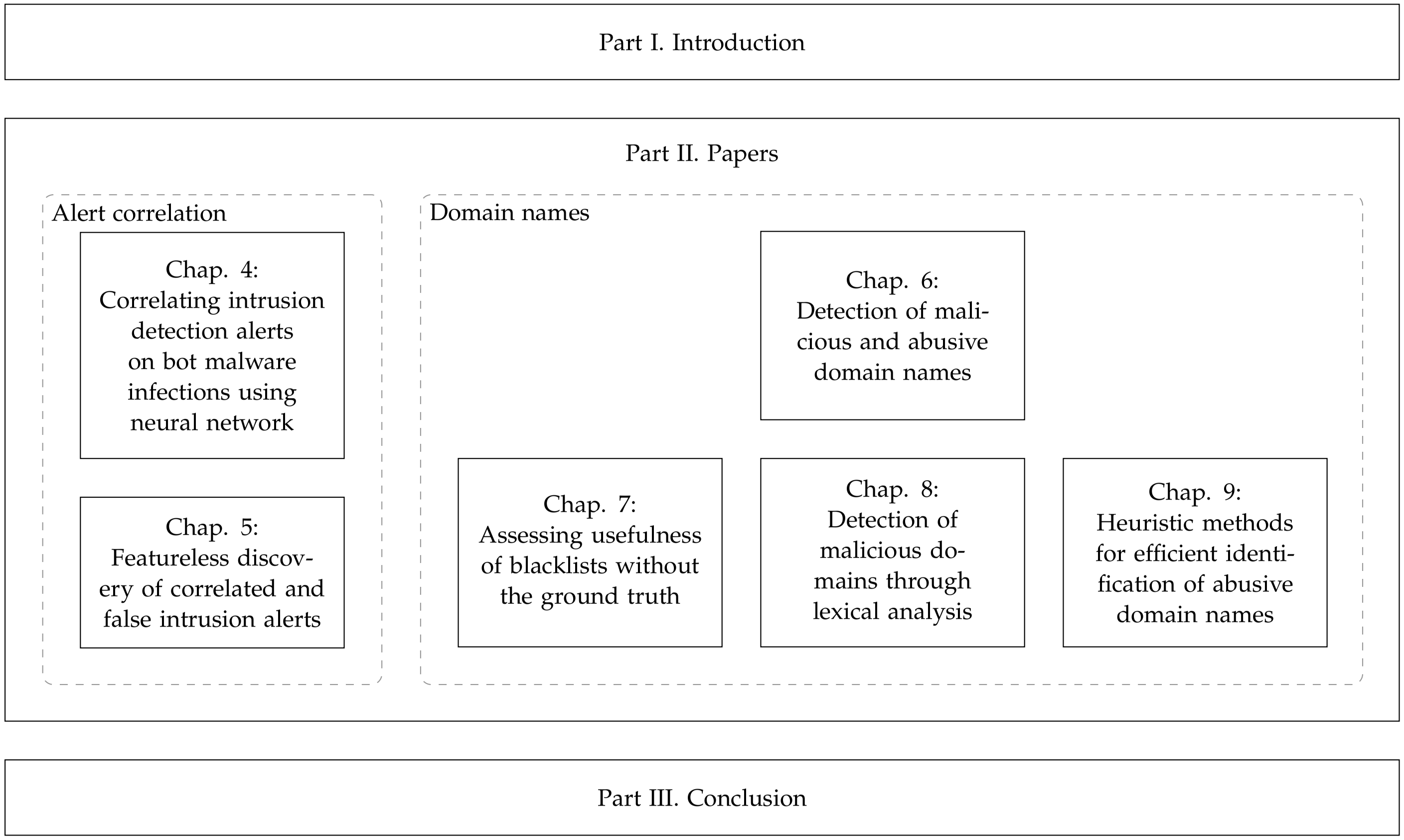
This PhD dissertation is organised as a collection of papers and consists of three parts. Part I outlines the background and motivation, leading up to a problem formulation, followed by a short description of how each paper contributes towards a solution. Part II is the main body, consisting of four peer-reviewed, published conference papers and two journal manuscripts. At the time of submission, one manuscript is in review and another is accepted for publication. Part III discusses the findings of the papers, outline future work, and concludes on the dissertation.

As a collection of papers, each paper in Part II can be seen as an independent contribution towards addressing the overall problem, but they also have some internal relations. As the first of two topics, Chapters 4 and 5 are concerned with correlation and filtering of alerts from Intrusion Detection Systems (IDSs). The first paper provides elaborate details of a novel method, while the second paper extends the first by generalising the concept, by introducing an alternative implementation, and by extending the evaluation.

The second topic is abuse of domain names and the Domain Name System (DNS). Chapter 6 can be seen as an introduction to this topic, with Chapters 7-9 going deeper into specific subtopics. The road map on the following page serves as a reference for this logical structure.

My contribution to each paper is outlined in the co-author statements that are signed by all co-authors, approved by The Technical Doctoral School of IT and Design, and made available to the assessment committee prior to assessment (See colophon for committee members).

The page of the chapter heading for each paper states (planned) publication venue and is followed on the next page by a copyright notice pertaining to the paper, which applies until the next chapter or part heading. Layout and formatting have been revised and obvious typos have been corrected in the papers. References are collected in the back of the dissertation. Attention is brought to Figure 4.1 vs. Figure 5.3 and Figure 4.2 vs. Figure 5.4. They carry similar messages and have some resemblance, as per above description of the first topic.



Part I. Introduction

Part II. Papers

Alert correlation

Chap. 4: Correlating intrusion detection alerts on bot malware infections using neural network

Chap. 5: Featureless discovery of correlated and false intrusion alerts

Domain names

Chap. 6: Detection of malicious and abusive domain names

Chap. 7: Assessing usefulness of blacklists without the ground truth

Chap. 8: Detection of malicious domains through lexical analysis

Chap. 9: Heuristic methods for efficient identification of abusive domain names

Part III. Conclusion

Acknowledgements

Being the greatest and most challenging endeavour in my career to date, this dissertation would not have come to be were it not for a great deal of people, which I would like to acknowledge here;

Jens Myrup Pedersen has been a great supervisor. He has especially proven invaluable by being able to take a step back, challenge what I saw as my challenge, and pose the magic question to help me gain a mental breakthrough when I really needed it. Thank you.

Søren Brandbyge has filled the role as co-supervisor with excellence, providing guidance on a deep technical level, while maintaining the grand picture, to an impressive degree, and that while also being a much-appreciated colleague. Thank you.

Matija Stevanovic and Mikael Andersen, have been very valuable to me by providing guidance, advice, and fruitful discussion. I am very happy for them to have followed great opportunities elsewhere after making many highly-valued contributions in the first parts of my PhD study. Thank you both.

Morten Tornbo came into the picture as I approached the home stretch, and have no doubt been instrumental in helping me stay focused and moving towards completion. Thank you.

Dorthe Sparre, Section Administrator of the Wireless Communication Networks section, have been a great help with many practical issues and formalities, not to forget that she does a great job of engaging in and facilitating social relations within the section. Thank you.

Thanks to Innovation Fund Denmark, for granting the funding for the Industrial PhD project. Clearly, a project like this would be something else without money, especially for the courses and conferences which I enjoyed a lot and found very valuable.

Likewise, I would like to thank LEGO for partnering in this project, providing both funding and a great work environment. I really appreciate how my managers have given me freedom to pursue my ideas in the project and freedom from operational tasks that are important when running a business, but sometimes less relevant to a PhD project. At the same time, they have contributed with very useful discussions and enabled me to engage in relevant activities. I think we have found the perfect balance, which has made my work truly relevant in practice. Thank you.

The engagement with DK-Hostmaster have evidently been fruitful and Erwin Lansing have contributed with a great deal of expertise, which have contributed substantially to shape the direction of the project. Thank you.

Thanks to all the great colleagues at both AAU and LEGO, who not only have served as expert references on many different topics, but also have provided pleasant, humorous human contact. I believe the last part is very crucial for me keeping my wit throughout the project and in my future career.

And finally, the two most important ones: Thank you Helene for agreeing to me taking on this project. Your support and patience have been essential for me. I do not believe that any of the hard sciences that I like to deal with can explain why a cup of coffee brought to me by you provides so much more of a boost than any other cup of coffee. Thank you, Ingrid, for being such a joy that I cannot describe it and for also energising me like nothing else.

Egon Kidmose
Aalborg University, November 9, 2018

Contents

Part I

Introduction

Chapter 1

Background

Over the last decades the Internet and the way it is used have developed rapidly, to a point where the Internet is now an essential part of how our society functions. This can be seen from the share of the adult Danish population that has not used the Internet at all in more than three months: Over the course of ten years this number decreased from 16% to 2%[1] [1]. Today practically every adult in Denmark is using the Internet regularly. Another sign of the trend is the passing of the Danish law on public digital mail, stating that in general citizens and businesses must receive mail from public authorities in digital format only [2]. This demonstrates how much we rely on the Internet for critical processes. For private Danish businesses, the impact of the trend can be observed as 54% of all businesses using advanced technology within IT[2] [3]. For large businesses the number is 87%[3]. This increased use of Internet dependent technology is likely fuelled by a potential for growth as well as increased competitiveness and productivity [4].

1 Internet Security

Unfortunately, the Internet that is key to this development has suffered from malicious activity virtually since the beginning, and it still is very much afflicted by this today. This is problematic as the increased dependence multiply the existing threats and introduce new ones, thereby adding significantly to the problem. Furthermore, the volume and speed have long surpassed the human and non-digital capacity in many places, which corresponds well with a motivation of improved efficiency, but it also means that humans are out of

[1] Adult population: 16-74 years old. Period: 2008-2018

[2] Advanced technology: Internet connected sensors, satellite-based services, Big Data analysis, robots, 3D-printing, and Artificial Intelligence (AI), in accordance with [3].

[3] Large businesses: More than 250 employees.

the loop, for better or worse. The consequence is a vast increase in how much we rely on, depend on, and trust in the Internet and the connected systems.

In this regard it is important to keep in mind that the fundamental structure of the Internet is a network of interconnected networks, primarily tied together by technology designed for connectivity. As evident from e.g. [5], the original design goals evolved around enabling communication, paying no attention to security. Ensuring that other parties can be trusted or that malicious parties can be policed are requirements that appeared later, and as they have not been thought into the design from the beginning, they now pose significant challenges. While much effort has been invested in improving upon this, the Internet remains a network of networks that reside in different jurisdictions across the globe and are owned by a wealth of nations, organisations, and private persons. Organisations, private persons, and nations using the former as proxies can be hard to identify on the Internet, especially if they seek to remain anonymous. Given the prevalence of maliciousness and abuse, I assert that securing the Internet is an ongoing effort, which might not be possible or feasible at all. In the meantime, corporations, like any other part of the Internet-connected society, are driven by the great benefits of the Internet, while the negative implications are accepted. With the benefits and gains in focus, it is imperative to be wary of implicitly or unknowingly accepting the drawbacks.

2 Cybercrime

Potential for criminals and a crime-based economy arise in this setting, where society increases trust, reliance, and dependence to unprecedented levels, while policing remains an unsolved problem. This has given rise to a well-organised cybercrime ecosystem [6], which has been estimated to cause damage worth 445-608 Billion USD [7] and generate an illicit revenue of 1.5 Trillion USD annually. These claims on revenue and damage are inherently uncertain and difficult to verify, because attackers are interested in operating undetected, victims might try to limit impact by not disclosing incidents, and the security industry making the commercial publications depends on the perceived size of threats. However, the Europol appears to trust such reports enough to cite them and echo that the damages amounts to hundreds of billions of euros annually [8].

In this dissertation, cybercrime is defined in a broad sense, such that it covers any activity involving the Internet, where applicable law for anyone involved or affected, accepted terms or agreements, or the non-conflicting interests of non-criminal Internet users are violated. Activity that aims to prepare for violating activity is also included, and so is unsuccessful activity where intentions are criminal. Finer-grained distinctions, such as cyber-warfare among nation-states and online fraud committed by organised crime gangs, does not appear relevant in this context, as threats and damages apply to corporations and society on the Internet regardless of the actor.

3 Existing solutions

As a result of the increased dependence on the Internet and the unfortunate growth in cybercrime, there are ongoing efforts to improve the security on the Internet, and the security for the connected organisations. One major direction that has received much attention recently is that of intelligence, where defenders seek to understand threats, thereby becoming capable of efficiently mitigating them before or during incidents. This is evident from a wide range of both commercial and open services available. Another major direction is detection, which enables organisations to recognise incidents as they start or unfold, rather than later when the full negative impact is a fact. This well-developed research field has matured, resulting in a wide selection of open and commercial IDSs being available. This includes variants that react automatically, such as Intrusion Prevention Systems (IPSs) and Web Application Firewalls (WAFs). With IDS technology being used extensively, it appears prudent to investigate it in relation to practical applications and the current threats posed to Internet-connected systems.

4 Summary

Internet security is an afterthought and an unsolved problem, yet society and businesses increase dependence to reap benefits. With increased dependence comes additional and larger threats, which for instance can be mitigated through detection with IDSs. The following chapter describes cases of real incidents, demonstrating the reality of the threats.

Chapter 2

Motivation

Being a part of the society, corporations are also subject to the conditions described in Chapter 1, i.e. both that of increased adoption of and trust in Internet connected systems, and the rise of cybercrime. Unfortunately, examples of how this can go wrong are abundant. In this chapter, three interesting cases are highlight and briefly analysed.

1 Case: The Target breach

In 2013, the department store Target was attacked by cybercriminals and suffered a serious data breach. Details of 40 million credit/debit-cards were stolen, along with personal information of 70 million customers, presumably all to be abused for fraud [9]. A contractor, who controlled refrigerator equipment on Target's network via the Internet, was used as a stepping stone by the attackers. It has been reported that Target had detection solutions in place and that alerts were raised on an early stage of the attack, but during a human processing step those alerts were deprioritised, allowing the attackers to continue and exfiltrate the data [10].

This case is notable for multiple reasons. First, it was an early example of the scale at which financially motivated cybercriminals can exploit a corporation, with repercussions extending to the CEO and CIO leaving the company [11]. Second, it shows how increased use of the Internet introduces new threats, that in this case turned out to be very real. Third, it shows that relying on human resource to process detection alerts can fail.

2 Case: WannaCry

In 2017, the WannaCry ransomware was used in a worldwide cybercrime campaign, where the data on tens of thousands of computers was encrypted, thereby denying owners access, and demanding ransoms. Impacted companies and organisations included hospitals, car factories, telephone providers, utility companies, logistics services, and schools [12].

The campaign was notable as it demonstrated how a simple criminal scheme can scale and how the severe impact hits diverse victims, who all have in common that they relied on Internet-connected systems for processing and storing valuable information.

3 Case: NotPetya

In 2017, the NotPetya campaign also appeared to be a large-scale ransomware attack, with widespread damage through encryption of data and systems. Being only one of the victims, the shipping company Maersk reported their damages to be in the range of 200-300 million USD [13]. Investigations showed that NotPetya was apparently masquerading as ransomware, as there was no means for decrypting data, should a victim decide to pay the ransom. It was believed to be an attack from one sovereign nation towards another, with collateral damage reaching unprecedented levels for cyber-warfare [14].

This incident was notable because it provides a firm figure on the direct losses incurred by a single company, and because it shows how collateral damage from cyber-warfare can hit corporations when processes are enabled by, and data is stored on, Internet-connected systems.

4 Summary

From the above cases it is clear that the use of Internet-connected systems introduces threats from malicious actors on the Internet. When corporations rely on such systems for important processes the threats can have significant impact. In the following chapter, a problem formulation is stated to define the scope of this dissertation.

Chapter 3

Problem formulation

1 Problem formulation

In a world where increased usage of the Internet offers many benefits the desire to remain competitive and relevant drives corporations towards more trust in, higher reliance on, and more dependencies towards the Internet. This introduces and extends significant threats, which corporations in general accept, either explicitly or implicitly. Therefore, I ask the question:

> *"How can corporations mitigate threats from*
> *the Internet, without impeding the business?"*

2 Contributions of papers

Each of the papers in the ensuing Part II contributes towards solving the problem. In the following, each paper is summarised and the relation to the problem is highlighted.

Chapter 4: Correlating intrusion detection alerts on bot malware infections using neural network

The sub-problem addressed is that some existing detection solutions, such as IDSs, raises so many false and correlated alerts that the alerts are unfeasible to process manually. The contribution is a proposal for and evaluation of a novel approach for handling this problem. The essential idea is to apply neural networks for learning how to interpret alerts without human involvement. This proposal is remarkable because it precludes feature engineering. Feature engineering represent substantial cost when systems needs to be tuned and tailored for each deployment – something that receives no attention in prior work on the problem. By avoiding feature engineering our proposals lowers the bar for when it is feasible to implement correlation and filtering. We conclude that it is possible to use supervised machine learning to extract useful information from IDS alerts on bot malware infections. This indicates that correlation and filtering can be done without the costly tuning and feature engineering.

Providing filtering and correlation capabilities, without a need to invest heavily in feature engineering and tuning, is promising for practical applications and prompts for further research. This has the potential to provide an efficiency gain in manual processing of alerts and a better return of investment when deploying IDSs.

Chapter 5: Featureless discovery of correlated and false intrusion alerts

This paper expands and generalises the one above, with multiple new contributions: We elaborately discuss the drawbacks of feature engineering. We present a general approach, as opposed to a specific implementation. We adapt an existing, unsupervised method and compare it to our previously proposed supervised method. Finally, we extend the evaluation from alerts on bot malware to also include benign traffic, and we introduce another public data set of diverse, contemporary attacks. Our conclusion is that our general approach provides for feasible implementations that are practically relevant, in particular because the general approach provides independence from costly feature engineering.

We explore the efficiency gain from correlating and filtering, under the constraint that implementations must be feasible, thereby making this work relevant for practical threat mitigation.

Chapter 6: Detection of malicious and abusive domain names

This paper signifies a new direction compared to the above, as it is focused on abuse of domain names. Serving as a precursor to the subsequent papers within this area, the paper holds and extensive review of prior work and a security-oriented analysis of the process for registering domain names, which has received little attention previously. The contribution is the outline of future directions, herein especially the finding that pre-registration detection is a promising, yet unexplored, potential for combating abuse, where criminal activity can be stopped efficiently.

This paper documents a study on how domain names and DNS can be used to mitigate threats, and pre-registration is found to be a particularly interesting approach.

Chapter 7: Assessing usefulness of blacklists without the ground truth

This paper presents a study on domain name blacklists, including the methodology. Our study is unique in that all blacklisted domains are considered. This differs from all known prior studies on the topic as they either rely on a sampled view of blacklists or on a ground truth to be available. We contribute by describing our methodology, demonstrating how it can be applied, and by relating it to the common approach of sampling blacklists.

Our method can be applied to analyse blacklist and gain understanding of their qualities and potential for mitigating threats.

Chapter 8: Detection of malicious domains through lexical analysis

In this paper, we present a classifier that can detect malicious domain names through lexical analysis of the domain names themselves. We contribute with an overview of useful lexical properties and with details on how these can aid in detecting malicious domain names. Our conclusion is that lexical analysis can be used to detect malicious domains and works particularly well for domains created with Domain Generating Algorithms (DGAs).

Lexical analysis can be applied to detect malicious domains, especially DGA domains, in order to mitigate threats that rely on those.

Chapter 9: Heuristic methods for efficient identification of abusive domain names

This paper presents an approach for finding malicious Second Level Domains (2LDs) within a given Top-Level domain (TLD). The contribution is a methodology where heuristics are developed and applied to guide a manual vetting effort, in order to efficiently find malicious domains. The heuristics are built on an understanding of techniques employed by criminals to enable their schemes. We conclude that heuristic enabled us to identify malicious domains with a low manual effort.

Our method is useful for identifying and reacting to abusive domains, thereby mitigating the threat they pose.

3 Conclusion on Part I

This concludes the introductory part, which has provided background (Chapter 1), motivation through cases (Chapter 2), and the problem formulation, along with an outline of contributions for each paper (Above). The following part is the main matter of the dissertation, namely the papers. This part can be read in full in the presented order, or select papers can be read on their own, as per the Preface and the accompanying road map.

Part II

Papers

Chapter 4

Correlating intrusion detection alerts on bot malware infections using neural network

Egon Kidmose, Matija Stevanovic, and Jens Myrup Pedersen

The paper has been published in the
Proceedings of The 2016 International Conference On Cyber Security And
Protection Of Digital Services (Cyber Security), June 2016.

Abstract

Millions of computers are infected with bot malware, form botnets and enable botmasters to perform malicious and criminal activities. Intrusion Detection Systems are deployed to detect infections, but they raise many correlated alerts for each infection, requiring a large manual investigation effort. This paper presents a novel method with a goal of determining which alerts are correlated, by applying Neural Networks and clustering, thus reducing the number of alerts to manually process. The main advantage of the method is that no domain knowledge is required for designing feature extraction or any other part, as such knowledge is inferred by Neural Networks. Evaluation has been performed with traffic traces of real bot binaries executed in a lab setup. The method is trained on labelled Intrusion Detection System alerts and is capable of correctly predicting which of seven incidents an alert pertains, 56.15% of the times. Based on the observed performance it is concluded that the task of understanding Intrusion Detection System alerts can be handled by a Neural Network, showing the potential for reducing the need for manual processing of alerts. Finally, it should be noted that, this is achieved without any feature engineering and with no use of domain specific knowledge.

1 Introduction

All those of our daily and critical tasks that rely on the Internet are threatened by botmasters who uses botnets to generate profit through various malicious and criminal schemes. Victim PCs become bots when botmasters infect them with bot malware. Botnets are formed by joining many bots and provide the platforms that enable the schemes. One scheme is theft of sensitive information, such as online banking credentials, where a botnet has been used to steal as much as 47 million USD [15]. Another common scheme is e-mail spamming, of which the majority has been attributed to botnets [16], making botmasters the very largest culprits behind a yearly loss of 100 billion USD [17]. Other common schemes are Distributed Denial of Service (DDoS) attacks, breaking targeted online services for days, and click fraud, where bots are used to fake user clicks on online ads, defrauding advertisers. The size of botnets has great impact on the success of the schemes, but due to the covert nature of botnets quantitative measurements are difficult. However, given opportune conditions the torpig botnet has been confirmed to consist of 180.000 bots [18].

In order to detect intrusions, including bot malware infections, Intrusion Detection Systems (IDSs) are commonly deployed. Some IDSs inspects network traffic from a point in the network and are known as Network-based Intrusion Detection System (NIDS), while others detect malicious activity from the host machine and are known as Host-based Intrusion Detection

System (HIDS). NIDS are considered less intrusive to deploy, as they require no change to hosts, which obviously is not the case for a HIDS. HIDS on the other hand can access network data (from the given host at least) and much more information that is not available to a NIDS. Examples of NIDSs include Snort [19], Bro [20] and Suricata[1]. Another discerning feature of an IDS is whether it relies on signatures of known malicious activity, i.e. signature based detection, or has some definition of "normal" in order to perform anomaly detection. Signature detection is best suited for detecting previously known malicious activity, while anomaly detection is better for malicious activity that is hitherto unknown and thereby cannot have known signatures.

While the name suggests that an IDS will raise alerts on intrusions this is not the entire truth. IDSs produce alerts whenever a suspicious event is observed, as a result, it is not guaranteed that alerts correspond one-to-one with intrusions. As an example, if an attacker scans for vulnerable service on a host that does not provide the service an IDS may still raise an alert. Alert with this one-to-none relation is referred to as false alerts. Another example is the case where an alert is raised when a host is scanned, and another alert is raised when the vulnerability is exploited. In this case, alerts correspond to infections as many-to-one and the alerts are said to be correlated. The impact of IDSs raising false and correlated alerts is described in [21]. The authors find that three instances of the Snort IDSs, deployed at a large financial institution, produce an average of 411,947.18 alerts pr. day. Only one in ten alerts is found to be interesting, and it seems reasonable to assume that the number of infections is even lower. Such a high ratio of irrelevant information is obviously a costly burden on security officers, potentially making the IDS deployments useless. IDSs might possibly be modified to raise less alerts, but that would increase the risk of missing an infection. Another well-established approach is to correlate or determine correlation between alerts, in order to fuse them, such that the result maps directly to one infection [22–26]. With the added information on alert correlation comes an improved potential for filtering out false alerts as utilised by [24–26]. When it comes to bot infections in particular, rather than general intrusions, [26] is particularly interesting, as the authors show that IDS technology can be used to detect bots. The authors of [26] support the claim that a single infection may have many alerts and their method will only alert on an infection when the underlying IDS has raised at least two correlated alerts.

Generally speaking three classes of methods for correlating alerts exists. The first class of methods is naive methods, which deems alerts to be correlated if selected features (e.g. Internet Protocol (IP) addresses or timestamp) are (almost) equal. Naive methods have the disadvantage that heuristic choice on feature set and thresholds affects performance [23]. The second

[1] http://suricata-ids.org/

class is model-based methods. These methods define a model for a bot life cycle or attack scenarios and find correlated alerts by matching alert sequences to the model [24, 26]. A noticeable member of this class is BotHunter [26], which achieves good results , but still has the drawback that a model must be defined heuristically by a human. The third and final class consist of methods based on Machine Learning (ML). Using ML can eliminate some of the need for human heuristics to define a model for bot malware infections, though still depending on expert knowledge to select and transform features, also known as feature engineering. The methods are trained on labelled data [22, 25]. Common for all existing methods is that they on some level rely on domain specific knowledge or heuristics about botnets.

In this work, we propose a method without any reliance on domain specific knowledge and without any feature engineering. To enable this, a method based on ML is proposed, which applies Neural Network (NN) to extract information from IDS alerts. A NN mimics a brain with neurons organised in layers. In its simplest form a layer operates by feeding an input sample to each neuron, which calculates a value and the values are concatenated to form the layer output. In particular, the presented method implements a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) neurons, where the input is a sequence [27]. The reason for using an RNN is that it processes variable length input, with a running time scaling linearly with the length, making it suitable for processing IDS alerts represented as human readable text strings. NNs can be used as a supervised machine learning method by training to a local optimum with the back-propagation algorithm and Stochastic Gradient Descent (SGD), on labelled training data.

Clustering refers to the task of grouping samples in clusters, such that samples in the same cluster are similar, under some notion of similarity. IDS alerts have successfully been clustered earlier by [22] and [28] has also had success with applying clustering in the context of botnets. While the above two examples are highly specialised for the relevant domain, many general clustering algorithms also exist. An example is Density-based spatial clustering of applications with noise (DBSCAN), which initially was proposed for spatial data, but is applicable to any clustering problem where a distance between two samples can be determined [29].

In this work, we present a novel method for reading IDS alerts in order to apply an existing clustering method and predict which alerts pertain to which incident. Initially a NN is trained on IDS alerts, labelled with information on which botnet infection the alert pertains. By training, a set of parameters for the NN is obtained, which can be used to map alerts into a vector space of fixed dimensions. Mapping alerts to the vector space enables application of a clustering algorithm to form clusters of alerts. The main contribution and novelty of our approach is the use of NN for reading string representations of IDS alerts on bot infections. As a result, neither method nor implementation is tied to any specific IDS (and possibly not even to IDSs as opposed to other monitoring tools). Furthermore, the methods do not require any domain knowledge or feature engineering, which is also a hitherto unseen trait. The gain from this is that as botnets continue to evolve, this method will remain relevant as long as some IDS is capable of raising alerts and training data is available.

The remainder of this paper is organised as follows. Section 2 explains how NNs can be used to read alerts, including how it is trained and how it is used to map alert strings to vectors. The section also describes the DBSCAN clustering algorithm, discusses how it can be used to cluster alerts and presents a proposal on how to use the clustering result to assign new alerts to known incidents. The data used for evaluation is described in Section 3, with the results of applying the methods following in Section 4. Results are discussed in Section 5 before conclusions are drawn in Section 6.

2 Method

In this section, the novel idea of applying NN to read IDS alerts is presented and a clustering algorithm is introduced. The key idea is to use NNs for reading alerts, while presumably conserving correlation information, and the first part of this section serves to explain this idea. Following this, the proposed approach to training the NN is explained and a simple method for detecting correlation between two alerts is proposed. Finally, the clustering algorithm DBSCAN is summarised and it is explained how it can be used to analyse the output of the NN and assign new alerts to known incidents.

2.1 Purpose of using a NN

Humans are able to read text and that is presumably, why all common IDSs can output alerts as text strings. IDSs can also output alerts in other formats but tying to a specific machine-readable format results in some degree of lock-in and prohibits future addition of other sources. To overcome this problem, the presented method reads alerts as text strings. For memory efficiency and to enable further ML, each alert is mapped to a feature vector of fixed size. The purpose of this first part of the method is to use NN to implement a mapping function from a string of any length to a vector:

$$\mathbb{A}\colon \text{IDS alerts – Strings of varying length} \tag{4.1}$$

$$\mathbb{M}\colon \mathbb{A} \to \mathbb{R}^n \tag{4.2}$$

Obviously, it is crucial that information that can be used to determine correlation between alerts is conserved. Importantly, this is all to be done without applying knowledge of how the alert strings are structured, and without domain specific knowledge of botnets, networks, or IDS technology.

LSTM RNN, a type of a NN, is used to implement $\mathbb{M}$. An LSTM RNN, capable of parsing alert strings to vectors is illustrated in Figure 4.1. An RNN reads an ordered sequence of vectors one vector at a time, while maintaining internal state, thus the output for the last vector is a result of the entire sequence. Each character in an alert string is encoded as a vector, with a one on the index of the character and with all other entries set to zero. While such a NN is trivial to implement it needs to be trained to find suitable parameters, called weights, for each neuron in the NN. Without training, the output cannot be expected to hold information about correlation. The NN can be trained efficiently on labelled data – samples of input data paired with desired output. Input data (alerts) are readily available, but without expert knowledge it is unknown what features are useful, and thus the corresponding output (vectors) is unknown. This calls for an alternative approach to training the NN of $\mathbb{M}$, which is presented in the following.

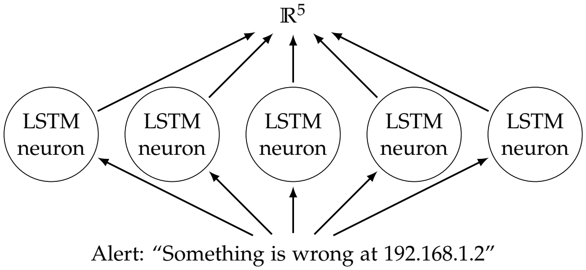


Fig. 4.1: An LSTM RNN capable of mapping an IDS alert string to a vector and an implementation of the mapping function $\mathbb{M}$ defined in Equation (4.1). This example consists of one layer of five neurons.

2.2 Training the LSTM RNN

Instead of alerts as input, pairs of alerts are considered. When two alerts are correlated (on the same infection) the mapping function must produce two similar vectors. When two alerts are uncorrelated (not on the same infection) the mapping function must produce dissimilar vectors. To describe this formally the function $\mathbb{I}$ is introduced, which for an alert returns the integer identifying the infection the alert pertains. Note that $\mathbb{I}(A_1) = \mathbb{I}(A_2)$ iff. alerts A_1 and A_2 are correlated, while $\mathbb{I}(A_1) \neq \mathbb{I}(A_2)$ iff. the two alerts are uncorrelated. For similarity we introduce the similarity function Sim, with an output much larger than some constant c for similar vectors and an output much smaller than c for dissimilar vectors.

$$\mathbb{I} : \mathbb{A} \rightarrow \mathbb{Z} \tag{4.3}$$

$$Sim : (\mathbb{R}^n, \mathbb{R}^n) \rightarrow \mathbb{R} \tag{4.4}$$

$$Sim(\mathbb{M}(A_1), \mathbb{M}(A_2)) \ll c \text{ iff. } \mathbb{I}(A_1) \neq \mathbb{I}(A_2) \tag{4.5}$$

$$Sim(\mathbb{M}(A_1), \mathbb{M}(A_2)) \gg c \text{ iff. } \mathbb{I}(A_1) = \mathbb{I}(A_2) \tag{4.6}$$

Alerts labelled with incidents are available, as will be described in Section 3, so for any pair of alerts it can easily be determined if a pair is correlated. Based on the assumptions that the directions of vectors are a meaningful way to represent correlation, cosine-similarity is used as the similarity function *Sim*. For vectors with the exact same directions, the cosine similarity is 1 and for perpendicular vectors it is 0. This is in accordance with Equations (4.5) and (4.6). For training purposes correlation is encoded accordingly (0 for uncorrelated, 1 for correlated). As already discussed, $\mathbb{M}$ can be implemented with a LSTM RNN, leaving only the problem of how to train, as $\mathbb{M}$ is not directly applicable to the training data (Alert to vector vs. Alert pair to correlation). An architecture solving this problem is already defined in Equations (4.5) and (4.6), but possibly more tractable as presented in Figure 4.2. While $\mathbb{M}$ maps an alert to a vector, this new architecture maps an alert pair to a scalar representing correlation of the two alerts. The architecture consists of two instances of $\mathbb{M}$ and the similarity function *Sim*. Note that the two instances of $\mathbb{M}$ are identical, meaning that implemented with LSTM RNN they must have the same weights in order to implement the same mapping.

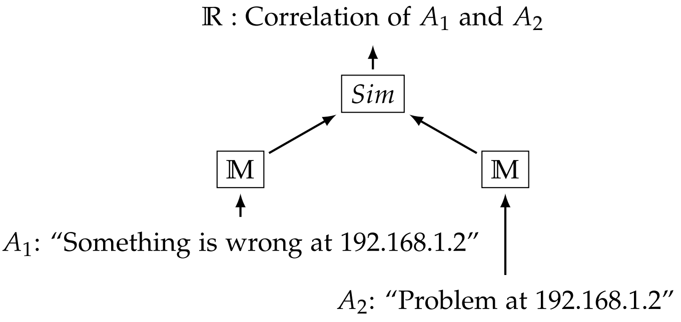

$\mathbb{R}$: Correlation of A_1 and A_2
Sim
$\mathbb{M}$
$\mathbb{M}$
A_1: "Something is wrong at 192.168.1.2"
A_2: "Problem at 192.168.1.2"

Fig. 4.2: Two instances of the mapping function $\mathbb{M}$ map an alert pair to a vector pair. The similarity function *Sim* of the two vectors is used as correlation.

By training the NN presented in Figure 4.2 on pairs of alerts to match the correlation, it is assumed that $\mathbb{M}$ is trained to map alerts to vectors in a way that conserves information such that correlation can still be determined. If this assumption holds, then a LSTM RNN can indeed be trained to read IDS alerts, without embedding domain specific knowledge in the method.

2.3 Detecting correlation with the LSTM RNN

A straightforward approach to evaluate if a suitable mapping function can be learned is to repurpose the architecture of Figure 4.2: Two alerts can be mapped to vectors through the use of $\mathbb{M}$ and the cosine-similarity of the two vectors will serve as an estimate of their correlation. Given the limited size of the model and of the data set, imperfections are expected to manifest as noise in the output, such that the result will not be in the set of $\{0,1\}$ but rather in the range of $[0,1]$. To handle this a threshold of 0.5 is applied, such that detection outcomes are formed according to the following:

$$\mathbb{I}(A_1) = \mathbb{I}(A_2) \text{ iff. } Sim(\mathbb{M}(A_1), \mathbb{M}(A_2)) > 0.5 \tag{4.7}$$

$$\mathbb{I}(A_1) \neq \mathbb{I}(A_2) \text{ otherwise} \tag{4.8}$$

Thus far an approach to obtain a mapping function (from alert strings to vectors) has been presented. A clustering algorithm is now introduced to cluster the vectors, thereby serving to solve the problem of many correlated alerts.

2.4 Clustering alerts with DBSCAN

The algorithm used for clustering is DBSCAN [29]. DBSCAN builds on a density-based notion of clusters: A cluster is an area with high sample density, fully surrounded and separated from other clusters by an area of low sample density. In further details, samples are put into three categories: 1) *Core samples* are in high density areas, 2) *Border samples* which are close to a core sample, 3) *Noise samples* which are far from core samples. Notions of high density, close to and far from are determined by the *min_samples* and *eps* parameters. Two points are close if their distance is less than *eps* and far if their distance is greater. High density is when a sample has at least *min_samples* within a distance of *eps*. A cluster is made up of all the core samples that are close to each other, resulting in high density, together with border samples that are close to the core samples. Noise is not part of any cluster. While DBSCAN originally was presented as a clustering algorithm for spatial data, the original work outright state that, the algorithm is applicable to "some high dimensional feature space" as well. For this purpose, the output space of $\mathbb{M}$ is considered such a feature space and the distance measure used is the cosine distance, which corresponds well with the cosine similarity used for training. The implication of this is that alerts presumably can be clustered according to which incidents they pertain. This is a significant contribution to solving the problem of IDSs raising many correlated alerts, but we go even further and will now propose a method for assigning new alerts to known incidents.

2.5 Incident prediction based on clustering results

In order to predict the class (incident) that samples (alerts) belong to, a set of training samples are clustered, and the resulting clusters are labelled with a class. Prediction is then performed by assigning test samples to a cluster, and the label of the cluster will be the predicted class.

For DBSCAN any sample used for clustering is unambiguously associated to exactly one cluster or classified as noise. This provides for a simple approach to labelling each cluster: Clusters are labelled with the class that has the most samples in the given cluster, weighted by the inverse of the number of samples in each class. The weighting serves to solve class imbalance, where a class with particularly many samples will dominate all clusters due to small but significant probabilities of samples being misplaced. The original work presenting DBSCAN does not present a method for assigning test samples to existing clusters, however the original definitions provide for a solution: If a sample is within *eps* of a core sample it will qualify as either core or border sample of the same cluster, according to the definitions of [29]. Based on the above observation, it is proposed to assign new samples to the same cluster as any core sample within a distance of *eps*. In the case that no core samples are within *eps* the sample is deemed noise. With a method for labelling clusters and a method for assigning test samples to clusters, prediction can be implemented as outlined in the previous paragraph.

In this section, we have proposed a novel method to obtain a function mapping from alert strings to vectors, without relying on any domain knowledge. An algorithm to cluster such vectors has also been described and additions needed for prediction has been proposed. When combined, the resulting method will group correlated IDS alerts and it will assign new alerts to the existing groups. This is a solution to the problem of IDSs raising many correlated alerts. The following sections serves to investigate how well the proposed method performs.

3 Data for evaluation

The CTU data set [30][2] used in the evaluation consist of traffic traces recorded in a lab network, while executing a bot malware binary on a virtual PC with Internet connection. Only traffic from infected PCs is considered and it is all considered as being malicious. One traffic trace is provided pr. infection, therefore the following four pre-processing steps are applied: *1)* The Snort IDS is applied to the traffic traces individually, producing alerts[3]. *2)* For each

[2] Available from: `https://mcfp.felk.cvut.cz/publicDatasets/`

[3] Using Snort version 2.9.7.6, DAQ version 2.0.6, built in rules and `https://snort.org/rules/snortrules-snapshot-2976.tar.gz`, accessed October 6th, 2015.

infection, a unique random IP address is generated and used to replace the IP address of the victim PC. *3)* All alerts are pooled into one set and alerts are then randomly split into three sets: Training, validation, and testing. *4)* Finally, within each set, all alerts are paired with itself and all other alerts. Informally, this pairing procedure can also be understood as the full outer join or the Cartesian product of the set of all alerts with itself. Figure 4.3 illustrates this procedure with two incidents and without splitting into separate training, validation, and test sets. Each pair is labelled with the correlation of the two alerts.

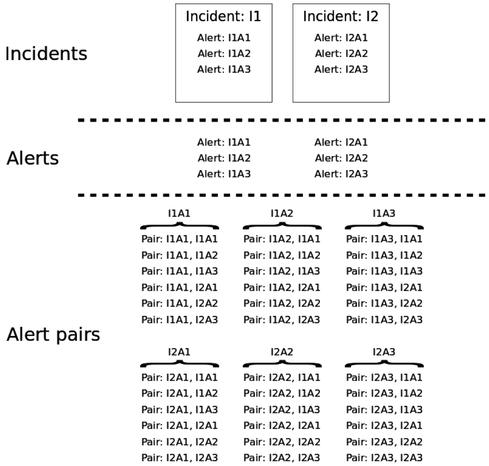


Fig. 4.3: The data used for evaluation consists of multiple traffic traces, two in this example. Each traffic trace corresponds to an infection and can be used to generate a set of IDS alerts. By pooling all alerts together and pairing all alerts with all alerts, a data set of pairs is obtained (Similar to a full outer join or Cartesian product).

From the CTU data set, infections with bot malware are selected based on the following criteria; *1)* Infections involving Command and Control (CnC) infrastructure controlled by the researchers are excluded, as this is deemed unrealistic. *2)* Infections involving multiple victim hosts, and thereby multiple infections, are excluded, as traffic cannot be labelled. *3)* Infections where the victim is already infected are excluded, as vulnerability exploitation is considered an essential aspect of the infection. Applying these three criteria results in 58 infections, as of January 25th, 2016. Infections with no alerts are excluded as this signifies that Snort fails to meet the condition that alerts must be raised by the IDS (12 infections). Infections with less than 100 alerts are excluded as they are less problematic than those with more alerts (28 infections). Infections with more than 500 alerts are excluded due to the number of training alerts severely affecting training time (30 infections). The resulting base data set consists of 7 incidents with a total of 2158 alerts, distributed as seen in Table 4.1.

Incident:	1	2	3	4	5	6	7	Total
Alerts:	100	184	317	328	390	395	444	2,158
Alerts (Pct.):	4.63%	8.53%	14.69%	15.20%	18.07%	18.30%	20.57%	100.00%

Table 4.1: Base data set used in evaluation, before splitting into train, validation, and test sets.

The base data set of 2,158 alerts is shuffled and split into a training set (60.00%), a validation set (20.00%) and a test set (20.00%). When pairs are created within each set it results in 1,674,436 pairs, 185,761 pairs and 185,761 pairs, respectively. The set of training pairs is down sampled without replacement to 600,000 pairs. The distribution between correlated and uncorrelated pairs varies a bit among the sets, with correlated pairs making up between 15.68% and 16.87%.

4 Results

In this section, two sets of results are presented. The first set is the results of applying the method presented in Section 2.3 to detect correlation between pairs of alerts. This is to investigate if this very simple method is able to detect correlation between alerts. The second set is the results of applying the learned mapping function and clustering as discussed in Section 2.4, including prediction of incidents for test alerts (Section 2.5. This serves to show if the mapping function is capable of extracting the information required for successfully clustering alerts. For the evaluation the mapping function, $\mathbb{M}$, is implemented with a single layer of 10 LSTM neurons and cosine-similarity is used as the similarity function, *Sim*. Training is done with back-propagation and SGD, a learning rate of 0.1 and mini batches of 10,000 alerts pairs for 10 epochs. Running time with the given data set is between one and two days on a shared server with 8 AMD Opteron 6272 processors (64 cores but using no more than 32 threads) and 512 GiB memory Ubuntu 12.04 LTS.

4.1 Detect correlation with LSTM RNN

A mapping function is trained on the training pairs, following the approach described in Section 2.2. The detection method presented in Section 2.3 is then applied to the validation pairs, resulting in the detection outcomes in Table 4.2[4].

[4] The following common abbreviations for detection outcomes are used throughout the paper: True Positive count (TP), True Negative count (TN), False Positive count (FP) and False Negative

Incident	TP	TN	FP	FN
1	282	6,592	5,888	168
2	1,808	12,486	13,050	240
3	6,272	25,424	16,576	0
4	13,196	36,628	20,608	252
5	12,438	39,042	17,658	684
6	6,290	30,848	23,668	5,568
7	13,368	40,508	19,860	2,120

Table 4.2: Detection outcome counts for predicting correlation of pairs. Both members of pairs are counted.

As seen in Table 4.3, the TPR is good for most incidents (Higher is better), while the TNR is not quite as good (Higher is better). This indicates that correlation is often detected when present, but also when not present. Thus, the detection method of Section 2.3 is not directly useful.

Incident	TPR	TNR	FPR	FNR
1	62.67%	52.82%	47.18%	37.33%
2	88.28%	48.90%	51.10%	11.72%
3	100.00%	60.53%	39.47%	0.00%
4	98.13%	63.99%	36.01%	1.87%
5	94.79%	68.86%	31.14%	5.21%
6	53.04%	56.59%	43.41%	46.96%
7	86.31%	67.10%	32.90%	13.69%
Avg.	83.32%	59.83%	40.17%	16.68%

Table 4.3: Detection outcome rates for predicting correlation of pairs by reusing the training architecture.

4.2 Clustering

For clustering, the mapping function is applied to the training alerts and the resulting vectors are then clustered. Combinations of the following parameter values are tried: $eps = \{0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1\}$ and $min\_samples = \{1, 3, 10, 30\}$. Parameters have been selected based on the following criteria: *1 – Too few clusters:* With fewer cluster than incidents, only some incidents can be predicted, which is undesirable. *2 – Too many clusters:* Obtaining almost as many small clusters as there are alerts is of

count (FN). Positive is correlated and negative is uncorrelated. R is rate as in True Positive Rate (TPR).

little use, so that is also undesirable. *3 – Homogeneity:* Clusters with few different incidents provides for fewer prediction errors, which is desirable. This is captured by the measure of homogeneity [31], which provides one scalar for the entire clustering, 0 being worst, 1 being best. The numbers are presented in Figure 4.4 and Figure 4.5. Very similar results are obtained when performing prediction on the validation alerts, but the plots are omitted here for brevity. The sweet spot for obtaining no less cluster than the number of incidents appears to be *min_samples* = 10, although it yields a few times as many clusters as incidents. For the highest homogeneity *eps* = 0.01 is used.

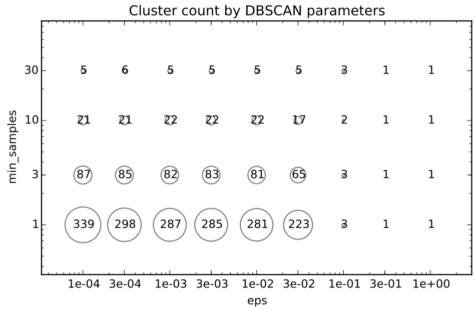


Fig. 4.4: Number of clusters.

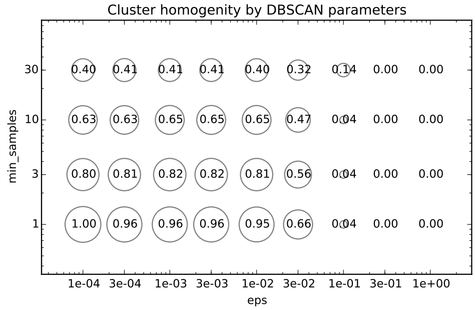


Fig. 4.5: Cluster homogeneity

The output from clustering the training alerts is labelled and kept for prediction, as proposed in Section 2.5. The incident of alerts from the test set are then predicted. Table 4.4 shows how many test alerts, grouped by incident, are assigned to each cluster. Ideally, all alerts from one incident are assigned to the same cluster and only alerts from one incident are found in each cluster. The first condition leads to a low number of clusters while the second leads to perfect homogeneity, resulting in perfect prediction performance. In the worst possible scenario, where the mapping functions fails, different incidents will be assigned to the clusters at random and with similar probabilities, leading to no predictive capabilities. Prediction on alerts found to be noise samples cannot succeed, so ideally no alert should be noise. However, 157 alerts out of 431 (36.43%) are indeed classified as noise. The consequence is that at least 36.43% of alerts will not be predicted correctly, or alternatively, no more than 63.37% of predictions can be correct. In particular, incidents 1, 6 and 7 have a high share of alerts labelled as noise. In addition to randomness, a possible explanation for incident 1 is the low number of alerts, leading to a class imbalance problem when training the mapping function. For all three incidents, it is also possible that the alerts are simply more diverse than for the other incidents, making it harder for the mapping function to produce good vectors. Most clusters only contain alerts from one incident, meaning that as long as the cluster has been labelled with the right incident, the prediction will be correct for these alerts.

Incident	Cluster																				
	noise	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	21	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	1	0	0
2	5	0	0	0	0	0	0	2	0	0	0	0	3	8	6	0	0	0	8	5	2	0
3	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	9	14	22	0	10	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
5	12	0	0	42	0	5	0	0	4	0	8	0	0	0	0	3	1	0	0	0	0	0
6	51	0	0	12	0	3	0	1	0	0	0	3	0	0	2	0	0	0	0	0	0	4
7	59	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	3	0	3	0	0

Table 4.4: Misclassification matrix showing how many alerts, by true incident, are assigned to each cluster.

True	Predicted							
	noise	1	2	3	4	5	6	7
1	21	6	0	0	0	0	0	1
2	5	6	21	2	0	0	0	5
3	0	0	0	78	0	0	0	0
4	9	0	0	0	48	0	0	0
5	12	0	0	0	0	63	0	0
6	51	2	0	1	0	15	7	0
7	59	0	0	0	0	0	0	19

Table 4.5: Misclassification matrix for incident prediction.

Table 4.5 shows the misclassification matrix for incident prediction. Ideally the correct incident is predicted for each alert, which will then be counted somewhere on the diagonal of the matrix (Ignoring the noise column when referring to the diagonal). Disregarding the noise column for an instance, the prediction is correct for 242 alerts out of 274 (88.32%). Of course, the noise samples are to be included, meaning that out of the total of 431 alerts, the correct incident is predicted 56.15% of the times. The largest off diagonal entry, disregarding noise samples, are 15 alerts from incident 6 predicted to be incident 5. The only other notable deviation is for incident 2, which has 21 correct predictions, but 6, 2 and 5 alerts are predicted to be incident 1, 3 and 7, respectively. Possible explanations are similar to those of the previous section: Class imbalance in training data leading to mapping function not “caring” about incident 2, high diversity between alerts of the incident or some random error.

Table 4.6 holds the following prediction metrics:

$$precision = \frac{TP}{TP + TN} \tag{4.9}$$

$$recall = \frac{TP}{TP + FN} \tag{4.10}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.11}$$

The table also contains the support, i.e. number of alerts. As to be expected, the worst F_1 score is seen for incidents with many samples discarded as noise (incidents 1, 6 and 7).

5 Discussion

In this section, the evaluation scenario, the achieved performance, and future plans are discussed.

Incident	Precision	Recall	F_1	Support
1	0.43	0.21	0.29	28
2	1.00	0.54	0.70	39
3	0.96	1.00	0.98	78
4	1.00	0.84	0.91	57
5	0.81	0.84	0.82	75
6	1.00	0.09	0.17	76
7	0.76	0.24	0.37	78
Average	0.88	0.56	0.62	
Total				431

Table 4.6: Performance of predicting the incident of test alerts. Average is weighted by support.

5.1 Evaluation scenario

It was chosen to perform evaluation on traffic traces recorded in a lab setup designed by researchers, generated by executing bot binaries selected by researchers. This involves choice with potential for affecting on the results, such that the performance seen in the evaluation will not reflect the performance in some real-life setting. The is a commonly accepted risk and is mitigated by using public data and by clearly stating the selection criteria used in this work. As a consequence, other researchers are able to reproduce the results or evaluate other methods under the same conditions.

Only using malicious traffic can lead to overly optimistic results, as the methods ability to handle noise in the form of false alerts, is not evaluated. On the other hand, it is simply assumed that all traffic from bots is malicious. This might not be the case and there might be false alerts in the used data, meaning that the evaluation conditions might be harder than they appear. Solving these problems or including labelled traffic that is certainly benign, requires solving the very challenging problem of classifying traffic as (non-)malicious. Alternatively, some errors in the labelling of data must be tolerated. Unlabelled data can be used for testing, as others have done [26, 32, 33], where a hopefully low number of alerts can be validated manually.

When pairing all alerts with all other alerts in a set, the result is n_{alerts}^2 pairs. In the setup used for evaluation this leads to problematically long running times for the training set, while the validation, and test sets was manageable. In the presented evaluation, this is handled by using a subset of all possible pairs. An interesting area for future research is to investigate how the size of the training set impacts the performance of the mapping function. When it comes to clustering and prediction the problem can be solved by considering alerts in temporal order and apply the "atom model" of [22].

In the current evaluation, all alerts are pooled together, split into training, validation, and test sets, and finally pairs are constructed. This is sound in so far that test is not performed on training samples, but the split can be performed in other ways, e.g. by incident before pooling alerts or after constructing pairs. Sorting alerts in temporal order and splitting at certain times is most similar to how the method would be applied in real life. The impact can be studied by performing evaluation in all the different ways.

5.2 Performance

When detecting correlated pairs using a threshold on the similarity of their vector representations the TPR is found to be 83.32%. This shows that the assumption about the mapping function providing similar vectors for correlated alerts (Equation (4.6)) is somewhat correct for the trained function. The TNR is found to be 59.83%, which suggest that the obtained mapping function also is reasonable at mapping uncorrelated alerts to dissimilar vectors (Equation (4.5)).

When predicting incidents, 88.32% of the alerts that the method provides a prediction for are assigned to the correct among seven incidents. The 157 alerts (36.43% of all) that are categorised as noise must of course be included, when discussing the performance, but there is an important difference between the algorithm making a false prediction, and outputting that prediction was not possible for the given alert. In a real-life deployment, only processing the noise alerts corresponds to a reduction in manual workload by 63.57%, which is highly valuable. Furthermore, manual inspection of the 157 noise alerts shows that only 13 alerts are raised on obviously malicious activities, namely Snort recognising a certain malware types or sensitive information being transferred. The remaining majority of 144 alerts are raised on atypical or wrong protocol usage (TCP or HTTP), which is not a clear sign of malware. This suggest that the majority is truly noise with no information, i.e. false alerts, or at least hard to extract information from. The implication of this is that it might be a wrong assumption that no alerts from a bot infected PC are false. If this is the case, the mapping function is trained with incorrect labels, which will severely impede the ability to extract useful information, and lead to degraded performance in the evaluation results. Possible solutions to this include validating and correcting the labels, which is cumbersome and hard, or adding benign traffic to the evaluation, which is easier.

DBSCAN defines clusters by density, as discussed in Section 2.4, but in further details, the used definitions are associative. An effect of this is that two samples that are not close will still be clustered together if they are close to a core sample, or even a chain of core samples that are close. In cases where the mapping function erroneously maps two correlated alerts to points that are not close, this associative behaviour might fix the error. On the other hand, if two uncorrelated alerts are erroneously mapped to similar vectors, the association can lead to at least one alert being mistaken for belonging to a wrong incident. Given that the clustering results are good, with few samples being misplaced, it is assumed that the beneficial effect is outweighing the impeding.

Overall, the demonstrated detection, clustering and prediction can only be achieved if the mapping function is capable of extracting relevant information. Based on this it is concluded that a useful mapping function can be trained, without feature engineering or expert knowledge.

5.3 Future work

In the current evaluation, a single layer of ten LSTM neurons is used. The common approach to determining the required size and number of layers for a given problem is to naively try. This leaves a theoretically infinite parameter space, from which only a single point was considered. For the neuron implementation, only LSTM has been considered, while multiple alternatives can replace or be combined with LSTM. For the similarity measures, only cosine similarity has been used, but it can be substituted with any similarity measure. Exploration of these possibilities and their impact on performance and training time is planned to be the next step.

Another interesting question is what information is actually extracted by the mapping function. Previous work suggests IP addresses and timestamps are valuable features. Some also rely on IDS specific features such as Snorts signature ID, which is assumed have strong ties to the part of IDS alerts that explains the alert in human language. In this work, no effort has been made to understand the meaning of the vector space that alerts are mapped to. A future study will focus on these questions.

6 Conclusion

In this paper, we tackle the problem of Intrusion Detection Systems (IDSs) raising many correlated alerts on bot infections by proposing a novel method for reading IDS alerts using Neural Networks (NNs). It is demonstrated that the proposed method enables clustering of correlated alerts and prediction of cluster affiliation for new alerts, which reduce the manual effort required to handle the vast number of correlated alerts. The key contribution is that the mapping function, which reads IDS alerts as text strings and produces feature vectors, is learned from labelled training data. The mapping function is implemented with a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN). No feature engineering is performed, and no domain knowledge is used in the method. The assumption is that a mapping function can be learned from labelled training data, such that alert strings can be mapped to vectors of fixed size, without losing the information required to determine correlation. The mapping function is applied in combination with a similarity function to detect pairs of correlated alerts, with some success. A clustering algorithm is also applied to a set of alerts, represented as vectors through the mapping function, to obtain a set of clusters. The clusters clearly show that useful information is extracted by the learned mapping function, as alerts from different incidents are assigned to different clusters, and alerts from the same incident are assigned to the same clusters. Furthermore, clusters appeared to generalise, as they are found to enable prediction. The conclusion is that a mapping function can be trained to read IDS alerts, contributing to solving the problem, without the use of domain knowledge.

Chapter 5

Featureless discovery of correlated and false intrusion alerts

Egon Kidmose, Matija Stevanovic, Søren Brandbyge, and Jens Myrup Pedersen

The paper is in review for being published in
The International Journal of Computer and Telecommunications Networking (Computer Networks).

The layout has been revised.

Abstract

Malware and cyber-attacks cause substantial damage and loss to many corporations, who all rely heavily on the security of IT systems. A commonly deployed countermeasure is Intrusion Detection Systems (IDSs). Unfortunately, IDSs typically raise many alerts on a single incident, with redundant information, as well as false alerts that are only noise to security analysts. Considering the out-of-the-box performance, the impact of these problems is so large that the produced alerts are often of limited practical use. Existing solutions rely heavily on domain expertise, as evident from how the expertise is embedded in feature engineering procedures, and directly in methods and algorithms. This has substantial negative impact on the costs of development, deployment, and maintenance, as well as on the adaptability of methods and implementations. The use of feature engineering, while generally acknowledged to boost classification metrics, further creates a need for substantial investment of expensive and scarce data science expertise. We find that reliance on domain expertise and feature engineering severely inhibits the feasibility of applying existing correlation and filtering methods in practice. To address this, we propose a novel approach to correlating and filtering, based on two constraints: Methods must be without feature engineering and methods must consume alerts as text strings. Two implementations are presented and evaluated on a partly private and on a public data set. We find that the implementations perform adequately to be considered in a practical setting. We also confirm that the method can be applied without feature engineering or embedding of domain expertise, which is a significant improvement compared to existing methods. Measured by known classification metrics the implementations cannot compete with the existing methods, supporting the conjecture that feature engineering and domain expertise can boost these metrics. We conclude that it is possible to correlate and filter IDS alerts, with implementations that are feasible and relevant in practice.

Keywords

Alert Filtering, Alert Correlation, Intrusion Detection System, Malware Detection, Neural Network, Latent Semantic Analysis, Clustering

1 Introduction

Over the past decades we have seen tremendous advances in information technology, coupled with a widespread adoption. While bringing many benefits, this has also made society and corporations vulnerable to a multitude of attacks. Criminals have identified a large range of illicit but financially rewarding schemes based on this, of which many involve infecting victims with malware or otherwise intruding on networks. Such schemes include harvesting credentials from victims, and abusing resources for sending spam e-mails, or launching Distributed Denial of Service (DDoS) attacks, and often rely on running malware on victim hosts. Malware often implements a Command and Control (CnC) channel back to infrastructure controlled by the criminals, and the victim host is then referred to as a bot, which is part of a botnet. A single botnet has been measured to encompass 180, 000 victims [18], while estimates on their size range in millions [34, 35]. An estimate of 500 million computers are infected and enrolled in botnets every year, causing a global loss of 110 billion US dollars/year [35]. The cost of cybercrime in general have been estimated as high as 445 to 608 billion USD [7].

A prerequisite for efficiently applying countermeasures is the ability to detect intrusions. Detection capabilities can be provided by Intrusion Detection Systems (IDSs), which in essence observe activity and raises alerts on malicious activity. We discern between Host-based Intrusion Detection System (HIDS) and Network-based Intrusion Detection System (NIDS), with the former having access to more detailed information internal to the host, and the latter being less intrusive, and with a potential to cover multiple hosts. Prior work suggests that the information available in the network is sufficient, and thus NIDSs are preferred [26, 28, 33, 36]. Examples of NIDSs that have had commercial breakthroughs are Snort [19], Bro IDS [20], and Suricata [37]. However, the usefulness of IDSs is limited by poor quality of alerts. From a theoretical point of view this is unsurprising, as perfect malware detection is impossible [38]. In practice, this problem has shown to be substantial, [21] reporting that three instances of snort deployed in large financial institution on average produce 411, 947.18 alerts per day. It is obviously unfeasible to process so many alerts manually, and the vast number of low-quality alerts causes alert fatigue as well as waste of resources.

In order to address the problem of poor alert quality, we observe that the problem is compounded by two underlying problems. A substantial share of alerts are false alerts, where no malicious activity occurred, but imperfections in the IDS still lead to an alert being raised. As an example, nine in ten alerts are found to be irrelevant in [21]. To security analysts this is pure noise, and false alerts should be filtered out prior to any manual investigation. We refer to the problem of false alerts being present in IDS output as the **filtering problem**. The other problem is that alerts with correlated information are raised as separate alerts. This leads to waste, when multiple investigations are initiated independently, and also when the analyst has to search for correlated alerts in order to gain understanding of a multistage intrusion. Numbers reported by [21] tells that they received roughly 40, 000 relevant alerts per day. Even with a conservative (high) estimate of hundreds of incidents per day, the number of alerts per incident is in the hundreds. Ideally, links between correlated alerts should be identified for the analyst, and only one alert should be raised per incident. We refer to the problem of correlated alerts being present in IDS output as the **correlation problem**.

A naïve approach to address the problems is to assume that IDSs are overly sensitive and tune them accordingly. This conflicts with the requirement that IDSs should not miss any incident, and the need for detailed information in post-detection analysis. Furthermore, it leads to expert time being wasted on tuning at setup and during daily operations. It seems a more prudent approach to let IDSs be overly sensitive, and then address the filtering and correlation problems in a pre-processing step. This is supported by such approaches achieving good results [21–26, 28, 33, 39–41], and by existence of commercial products such as Cisco Security MARS and FireEye.

Existing methods for correlation and filtering that use Machine Learning (ML) all rely on feature engineering, which is the art of transforming data before applying ML algorithms. It is a generally proven method for improving performance, but it comes with some drawbacks, which needs to be considered. Severyn and Moschitti states that feature engineering is a tedious task, and it reduces adaptability by requiring substantial re-engineering [42]. Anderson et al. refer to it as a pain point in building trained systems, which require, yet challenge, computer scientists with PhD-level training, and note that it requires dramatically more iteration and adjustment than what is immediately apparent [43]. Khurana et al. is a third example of similar position, stating that feature engineering is largely manual, a complex exercise, iterative, based on trial and error, and often the most time-consuming step in a data science workflow [44]. Gauging the exact cost of feature engineering is hard, and likely impossible to do in general, but there are examples where the effort or cost related to ML and feature engineering has been reported:

The Netflix Prize was an open competition to improve performance on a movie recommendation problem, based on a fixed data set [45]. It shows that a 10% incremental improvement over a working system was worth at least a million dollars, and that a lead time of years was acceptable. The effort made by contestants during the first third of the competition (11 months) is indicated by numbers reported in [45]: 20.000 teams registered, 2.000 teams submitted results, and 13.000 different results were submitted[1]. We judge that a large effort must have been made, also considering that the contest continued for a total of 33 months. The IBM DeepQA project [46] had the goal of creating a system named Watson, capable beating human grand masters at the Jeopardy quiz show. The project had a core team of 20 scientist and engineers and lasted 3 years. Watson applies *more than 100 different techniques* and a principle of *many experts* to find answers, indicating that feature engineering is a key aspect. 5.500 experiments were conducted, consuming a staggering 30.117 CPU-years, which supports the claim that many iterations are required. The man-hours and compute resources spent must have been

[1] We use the date of publication for [45] as a conservative (Late) estimate of when included numbers were recorded.

substantial.

Like feature engineering, expertise from the networking and security domains can aid in correlating and filtering alerts, but again the drawbacks prompt for some attention. Applying heuristics and domain expertise can potentially improve performance, but adaptability and maintainability might see significant negative impact. Consider as an example a correlation and filtering method that relies on comparing source and destination IP addresses. This reflects the domain knowledge that network intrusions have a source and destination, which might boost performance under certain conditions, and appears to be sensible. Consider then an attack where the source can be spoofed, such that it can be chosen freely by the adversary. This would prompt for redesigning the method, redoing feature engineering, and changing the implementation, as the assumption is broken. Similar arguments can be made about swapping NIDS with HIDS, changing or updating IDS, different attackers having different methods of operation, etc. The essence of the issue is that domain experts make assumptions about the conditions, so the inevitable changes in conditions can invalidate the assumption, the method and, the implementation.

We believe that feature engineering and domain expertise can provide a performance improvement, but the price to pay is substantial resources and loss of adaptability. The loss of adaptability means that the returns of invested resources is limited, which can make methods unfeasible and explain limited adoption by practitioners. Or in other words, it is not worth it for corporations to invest in developing a method and doing feature engineering when adaptability inhibits reuse. We propose a shift from seeking excellent performance on classification metrics with methods that are unfeasible in practice, towards sufficient practical performance with methods that are feasible due to significantly decrease implementation costs and improved adaptability.

Our contributions are 1) to question if feature engineering and embedded domain expertise is the right approach to the filtering and correlation problem, 2) to present a novel, general approach that is free from feature engineering and embedded domain expertise, 3) to present two implementations of the general approach, and 4) to evaluate the methods on two different data sets.

The paper is structured as follows: We survey related work in Section 2. Our general approach is presented in Section 3, along with our own method based on Neural Networks and our adaption of Latent Semantic Analysis (LSA) to the present problem. In Section 4 the two data sets and relevant procedures are described, along with a proposal of metrics to capture performance for practical purposes. Results of applying the two methods to the two data sets are presented in Section 5, and discussed in Section 6, before we conclude on this work in Section 7.

2 Related work

In this section we survey existing work on solving the correlation and filtering problems. First, we outline the idea of submitting IDS alerts to a pre-processing step, highlighting relevant prior art. Then we survey approaches and techniques that have been used, and we pay particular attention to feature engineering. As data for evaluation poses some interesting challenges, we summarise the options, reasoning, and choices found in related work. As this work builds on the claim that there is a mismatch between practical application and commonly used metrics, we finally highlight prior work supporting this.

2.1 Pre-processing

Leaving out the details of correlating and filtering for a start, we introduce the notion of pre-processing. As discussed in Section 1, and as evident from the extent of existing work in Table 5.1, IDSs raise too many alerts for manual processing. One solution is to introduce a pre-processing step prior to manual processing. A pre-processing step takes alerts as input, and outputs **hyper alerts**. A hyper alert represents one or more lower level alerts, as illustrated with Figure 5.1. If done correctly, pre-processing solves the filtering and correlation problem. In the related work, hyper alerts are also referred to as meta alerts or reports.

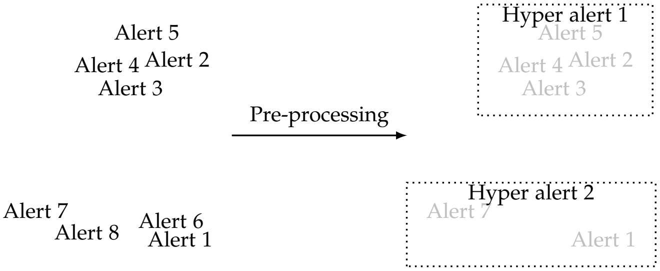


Fig. 5.1: Pre-processing alerts into hyper alerts.

Paper	[47]	[23]	[22]	[48]	[24]	[25]	[26]	[39]	[21]	[49]	[30]	[50]	[51]
Correlating	x	x	x	x	x	x	x	x		x	x	x	
Filtering			x	x	x		x	x	x	x		x	x
Similarity		x	x			x				x	x	x	
Rules	x			x	x		x						
Graph-based				x		x						x	
Machine Learning			x			x	[a]	x	x		x		x
Prototype available	x						x						
Qualitative	x	x		x		x	x	x		x			
Quantitative			x	x				x	x			x	x
New metrics				x							x		x
Real world traffic		x					x	x	[b]	[b]	[b]		x
Controlled attack		x	2000 [52]	2000 [53]		2000 [53]		1999 [54]	2010	2013		2012	
Controlled Malware							x				[30]		

Table 5.1: Overview of related work. Columns are publications, sorted by increasing year of publication. The first two rows, (Correlating and Filtering), show which problems are addressed. The following four rows, (Similarity, Rule, Graph-based, Machine Learning), indicate the techniques used. The next four rows, (Prototype available, Qualitative, Quantitative, New metrics), pertain to prototype availability and evaluation method. The last three rows, (Real world data, Controlled attack, Controlled malware), describe the data used for evaluation, where a reference indicates that the data is available, and the year indicate when the data was created or recorded.

[a] Machine Learning is not used for filtering or correlating (But in the two IDS plug-ins SLADE and SCADE)

[b] Background traffic only

A simple pre-processing method is to fuse alerts by applying the *atom model*, where incoming alerts are compared to existing hyper alerts; Alerts are fused with the hyper alerts that they most probably belong to, unless the highest probability is below a given threshold; then the alert forms a new hyper alert [22]. In [22] the hyper alert is simply represented by the most recent alert, while [23] applies a complex structure that enables heterogeneous information sources. Fusing multiple correlated alerts into fewer hyper alerts, addresses the correlation problem, and brings down the number of items requiring manual investigation. Fusing alerts into graphs has also been proposed, providing some insights to how attacks develop [25].

The filtering problem can be addressed when fusing, by filtering out alerts that fail to meet some criteria for being fused. This relies on the assumptions that false alerts fail to fuse, while true alerts are fused. An evaluation of this approach, using 10 incidents in a fully controlled lab network, was found to produce hyper alerts on all incidents [26]. Filtering can also be done at the hyper alert level, i.e. after fusing, as suggested by [22]. The assumption behind this is that false alerts are fused together, and that false hyper alerts are easier to filter than individual false alerts. Approaches for filtering only have also been proposed, and could be combined with subsequent correlating [21, 51].

Having clarified the ideas of pre-processing, correlating, and filtering, we now move on to a study of the approaches found in prior work.

2.2 Existing approaches

A commonly used technique in existing methods is to implement a function for estimating similarity of alerts, or alternatively estimating the probability that two alerts are correlated. Naïve implementations, relying on human heuristics to estimate correlating probability, have been demonstrated by [22, 23, 25]. Applying human heuristics, both [22] and [23] constructed functions to estimate correlation probability. Formal parameter optimisation was applied by [22]. Probability of correlation was estimated with two supervised ML algorithms in [25]. Rather than labelling real samples for training, this work relies on manually crafted training samples, and is therefore fully dependent on human heuristics.

Another approach to handling the correlation problem, is to rely on rules that describe relations between different attacks steps and assume that alerts represent distinct steps. Generally, correlating is implemented by associating necessary prerequisites, and potential consequences to alert classes. The rules can be manually defined, as suggested by [24], or mined from data as in [25, 48]. The latter two also encompasses varying degrees of correlation, as opposed to binary, and [25] involves continuously adjusting probabilities when in operation. Hyper alerts are formed according to the given rules, with associativity enabling more complex scenarios to be captured. The procedure of [26] also relies on predefined rules, describing that certain types of alerts must have been raised if a hyper alert is to be produced. With rules, and in particular with a notion of varying degrees of correlation between alerts, hyper alerts are naturally expressed as graphs, and some work specifically apply graph-based methods or modelling [25, 48, 50].

While some of the earlier discussed methods include ML, they also rely heavily on heuristics based on domain expertise. However, correlating and filtering can be left to ML to a large extent. Clustering corresponds well with the correlation problem, and classification with filtering. An example of how ML can learn to solve the correlation and filtering problem from data, without embedding domain expertise, is presented by [39]. The method is composed of Self-Organising Map (SOM) and k-means clustering. SOM is an unsupervised application of Neural Network (NN) that learns to map samples into a space of lower dimensionality, so that similar samples are close, and dissimilar samples are disparate in the output space. The k-means clustering algorithm is used to obtain well defined clusters in the output space. Each of two stages applies SOM and then k-means, first to group alerts by correlation, and second to filter out groups of false alerts.

2.3 Feature engineering

Some feature engineering methods applies to ML problems in general, while others embed domain expertise and are thus limited to the given domain. Feature selection and transformations are examples of general feature engineering. Selection is widely used, but we highlight [21], who select features that only applies to network traffic, and a feature that does not exist for anomaly-based IDSs, leading to a method that only applies to rule-based NIDS. Mean/variance normalisation is an example of a general feature transformation applied by [39], which mitigates that some ML methods have bias towards features with high variance or mean. The use of domain expertise for feature engineering appears to be most impacted by the challenges outlined in the introduction, yet all found examples of correlation and filtering with ML make use of this. A common example is a feature transformation that produces a distance between two Internet Protocol (IP) addresses. The distance is computed as the number of most significant bits that the two addresses have in common [22, 23, 25]. This method only applies when source and destination is known, meaning that it fails for HIDS alerts on activity not involving the IP layer, or if a source IP cannot be obtained reliably. A more specific example is that [22] recognised that source, and destination IPs were swapped for a certain class of alerts. Part of their feature engineering was to compensate for this IDS error, by swapping source and destination IPs back, when an alert was of the given class. The implications of this is that time has been wasted on debugging the IDS, and the method is tailored to the specific situation. BClus, the ML method proposed by [30], relies on complex aggregations over time and low-level network features. Finally, all existing methods rely on fixed attributes to be extracted from alerts, which again create strong ties to specific IDSs and their output format.

2.4 Data sets and evaluation

Evaluation of filtering and correlation methods is commonly carried out on obsolete or private data sets. As attacks, malicious traffic, and the noise from benign traffic, evolves over the years, evaluations with the DARPA 1999 and 2000 data sets [53, 54], or the DEFCON 8 CTF data set [52], becomes obsolete. The malicious activity in these data sets is bluntly obvious by current standards. The malicious activity includes well known attacks launched over plain text protocols, DDoS attacks and attacks during a CTF game, where the actors have limited motivation to act stealthy. Current attacks in the real world are expected to be significantly stealthier and more difficult to detect. One example is that malware is now often trying to hide among legitimate user activity, such as web browsing and e-mailing, with both malicious and benign traffic being encrypted. While the data set of [52–54] are obsolete, they have the benefit of being publicly available, which allows for verifying and comparing results. At the time when [22] and [23] was published, their used data sets where likely still relevant, but noting that [25], [39] and [49] used 6-, 11- and 13-year-old data sets, suggests a challenge with data availability within our field. This is supported by the observation that all surveyed evaluations with real world traffic, include no references to the data sets, suggesting that it is not available to the public. We suspect that these highly valuable data sets are kept private to avoid privacy issues. The impact of this is evident from work such as [26]. The used data sets appear substantial and realistic, but others are hindered in assessing the data, reproducing the results, or reusing the data for comparison, as the data is not available. Furthermore, part of the evaluation ignores false positives, as it is not feasible to thoroughly inspect the data in order to label it. The return of investing in labelling data increases with reuse, so sharing data could potentially make it feasible.

Only one example of recent, publicly available, and substantial data on real malware is known to the authors [30]. This particular data set contains traffic traces of real malware executed in a controlled environment. The omission of real benign user activity eliminates privacy concerns in relation to publishing the data. However, in reality benign activity will be present, and an evaluation should reflect this to provide a reasonable representation of real conditions.

An approach to generate current data sets that can be shared without compromising privacy is presented in [55]. The idea is to model benign activity, so that it can be emulated in a lab, along with controlled attacks, or execution of malware, such as that of [30]. The reasoning is that model descriptions and data generated from models are not problematic with regards to privacy, hence data generated in the lab can be shared freely. It is unclear exactly how the method can guarantee that sensitive information is not captured by the model and made evident in the generated data, and the method appears to be susceptible to bias through design choices, as is the case for other methods relying on synthesising data. A recent data set has been produced according to the approach, including raw traffic capture and details about the attacks that enable labelling [56].

Another solution is to record data on real activity and then remove the sensitive information before sharing. This can be implemented by e.g. removing network traffic payloads, pseudonymisation by randomising IP addresses, and by truncating traffic to flows. We are not aware of any substantiated claim that this can be done without losing information that can be of use to some methods, and indeed this would seem counter-intuitive. It remains unclear, what notion of privacy is guaranteed, and data labelling remains an extensive task.

Yet another solution is to evaluate existing methods on a private data set, but this requires implementation to be available or to be re-implemented. We are only aware of two examples of work where publicly available implementations are referenced [26, 47], and re-implementing requires substantial effort, without much assurance that it will match the original implementation sufficiently well. This would still not enable reproducing results, as performance likely depends on the data.

2.5 Performance evaluation

As evident from Table 5.1 there are examples of both quantitative and qualitative evaluation in the related work. Qualitative evaluations provide some interesting details of results, but without quantitative metrics it is challenging to compare performance of different methods or variation thereof. For quantitative evaluations, it is common to apply metrics for detection and classification, such as accuracy, precision, recall, F1-score, etc. A gap between these established metrics and the application domain is noted in [30], which also includes a proposal for a set of new metrics to address this. The new metrics include timeliness of detection and correct detection of infections, rather than correct detection of samples. Also focusing on real-world applicability, [51] takes the view that an organisation has a finite bandwidth, or *daily investigation budget*, and measures performances in terms of bandwidth versus recall. This suggests that the classification/detection metrics are mostly of academic interest, and less relevant for evaluating the feasibility and relevance of a method in practice. None of the related work reports on the effort invested in feature engineering, the dependence of domain expertise, or the adaptability.

In this section, we have surveyed existing methods for correlating and filtering. One important finding is that feature engineering and domain expertise are fundamental to all the existing methods. Another important point is that there is support for rethinking how performance is measured, as the classification and detection metrics are somewhat disconnected from the application domain.

3 Method

In this section we present a novel approach to how correlating and filtering of IDS alerts can be automated. The novelty of the approach is that there are no use of feature engineering and no use of domain expertise. The approach relies on data for learning how textual alerts can be read. In addition to the general approach, two examples of how it can be implemented are also presented. The first implementation applies Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs), and the second LSA.

3.1 General approach

Having established in the Introduction that using feature engineering and embedding domain expertise can yield methods that are not feasible in practice, we strictly avoid using such. Feature engineering and embedding of domain expertise are examples of making assumptions about the problem, which are translated into optimisation in implementations. Problems arise when these assumptions are broken. We allow only two assumptions about the problem and build our general approach on that. First, we assume that alerts carry sufficient information to determine which are false and which are correlated. We expect no loss of generality from this, as it follows naturally from the assumption that the filtering and correlation problems for a set of alerts can be mitigated. As for the second assumption, we observe that alerts must be consumed by a machine so a general machine-readable representation for alerts is required. Applying a schema implies feature selection and is out of the question. Clearly, it would degrade adaptability to only be compatible with IDSs producing alerts that can fill the schema and by requiring a parser to be maintained. We note that all IDSs familiar to us are capable of presenting alerts as human readable text strings. Also, all representations of alerts that we know of are a subset of all text strings. Consequently, the second assumption is that alerts can be represented as human readable text strings.

To only build upon these two assumptions, we propose an approach, consisting of three phases, as outlined in Figure 5.2. In the first phase, a **mapping function** is learned from alerts. The mapping function must be capable of mapping alerts, represented as text strings, into a vector space, which we refer as the **abstract feature space**. In the seconds phase, the mapping function is used to map alerts into the abstract features space. Finally, in the third phase, clustering is applied in the abstract feature space to obtain **hyper alerts** that represent incidents. This approach requires that a method is capable of learning a discriminative mapping function in the first phase, such that applying it in the second phase yields a representation where alerts can be discriminated by incident, such that the clustering in the third phase is meaningful. If a discriminative mapping function can be learned from data, it will be able to replace feature engineering and domain expertise as known from existing work. If possible, this eliminates the need for investing in feature engineering and eliminates the issues of adaptability.

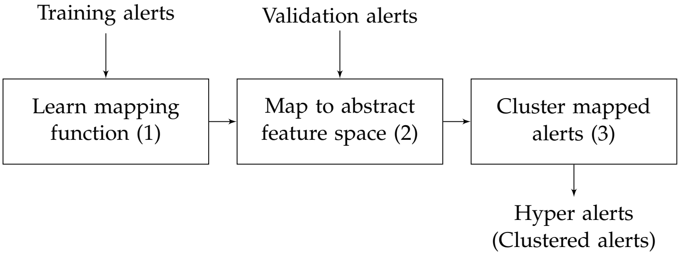

Training alerts
Validation alerts
Learn mapping function (1)
Map to abstract feature space (2)
Cluster mapped alerts (3)
Hyper alerts (Clustered alerts)

Fig. 5.2: Overview of how the methods are applied.

Having described the overall proposal, two examples of how a mapping function can be obtained will follow: One using LSTM RNN and one using LSA.

3.2 LSTM RNN: Mapping function

LSTM RNN is a well-regarded method in the area of Natural Language Processing, and therefore a good candidate for being able to produce a discriminative mapping function. As a neural network-based method, it is computationally efficient and benefits from recent developments in hardware, including utilisation of GPUs for computation. We propose to implement a mapping function with a LSTM RNN as depicted in Figure 5.3:

$$\mathcal{M}\colon \mathbb{A} \to \mathbb{R}^{10} \tag{5.1}$$

The NN can be trained efficiently on data consisting of input/output pairs, using the Backpropagation algorithm. However, given that no secu-

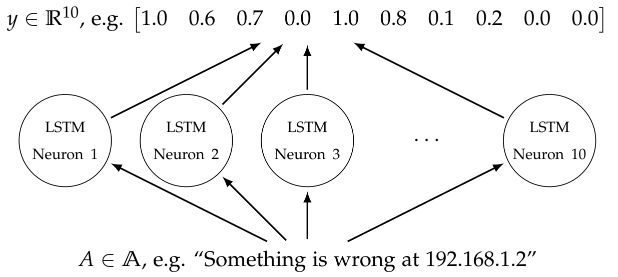

$y \in \mathbb{R}^{10}$, e.g. $\begin{bmatrix}1.0 & 0.6 & 0.7 & 0.0 & 1.0 & 0.8 & 0.1 & 0.2 & 0.0 & 0.0\end{bmatrix}$
LSTM Neuron 1
LSTM Neuron 2
LSTM Neuron 3
...
LSTM Neuron 10
$A \in \mathbb{A}$, e.g. "Something is wrong at 192.168.1.2"

Fig. 5.3: The Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) implementation of the function ($\mathcal{M}$), mapping Intrusion Detection System (IDS) alerts into an abstract feature space. [57, Fig. 1]

rity domain expertise can be embedded in the method, the output part of training data is not available. Taking alerts (Input) and creating corresponding points in the abstract feature space (Output), would produce the required data, but would also be a clear violation of the independence from domain expertise and feature engineering. Consequently, the mapping function cannot be trained directly.

3.3 LSTM RNN: Training a mapping function

To obtain an approach for learning a mapping function from data, we elaborate on the meaning of discriminative. To this end, we introduce an indicator function for correlation ($\mathcal{I}$), and a similarity function ($\mathcal{S}$):

$$\mathcal{I}\colon (\mathbb{A}_i, \mathbb{A}_j) \to \{0,1\} \tag{5.2}$$

$$\mathcal{S}\colon (\mathbb{R}^{10}, \mathbb{R}^{10}) \to \mathbb{R} \tag{5.3}$$

The mapping function and the corresponding abstract feature space is said to be discriminative iff. uncorrelated alerts are very dissimilar (Equation (5.4)), and correlated alerts are very similar (Equation (5.5)). Equation (5.4) states that the problem of detecting false alerts is equivalent to finding outliers in the abstract features space. Correspondingly, Equation (5.5) states that the problem of grouping correlated alerts is equivalent to clustering in the abstract feature space.

$$\mathcal{S}(\mathcal{M}(A_1), \mathcal{M}(A_2)) \ll c \qquad \text{iff. } \mathcal{I}(A_1, A_2) = 0 \tag{5.4}$$

$$\mathcal{S}(\mathcal{M}(A_1), \mathcal{M}(A_2)) \gg c \qquad \text{iff. } \mathcal{I}(A_1, A_2) = 1 \tag{5.5}$$

An NN for calculating the similarity of two alerts, as expressed in the left-hand side of Equations (5.4) and (5.5), can be implemented as illustrated in

Figure 5.4. Assuming that similarity in the abstract feature space can be approximated by the output of the indicator function, this architecture enables us to learn a mapping function. For a data set of alerts labelled with incident IDs, it is trivial to create pairs of alerts, for which correlation is known. The NN can then be trained, with Backpropagation, using the alert pairs as input, and $\mathcal{I}(A_1, A_2)$ as target output. NN weights make up all the trainable parameters, and they are all within the two instances $\mathcal{M}$. By tying the parameters of the two instances together during training, the result is a set of parameters that can be read out and reused in a single instance of $\mathcal{M}$. Thus, a mapping function can be learned from data, based on these very general assumptions and the proposed training architecture. The evaluation will show if the learned mapping function is discriminative.

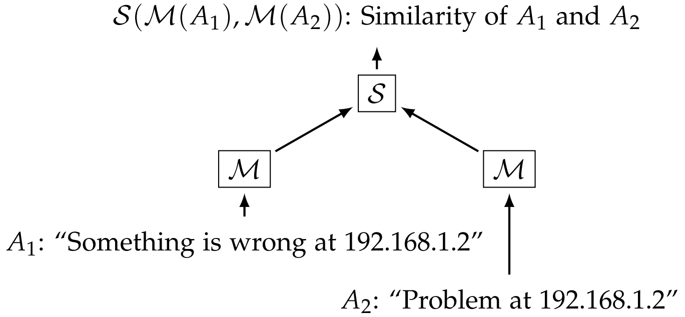

$\mathcal{S}(\mathcal{M}(A_1), \mathcal{M}(A_2))$: Similarity of A_1 and A_2
$\mathcal{S}$
$\mathcal{M}$
$\mathcal{M}$
A_1: "Something is wrong at 192.168.1.2"
A_2: "Problem at 192.168.1.2"

Fig. 5.4: LSTM RNN for estimating similarity of two alerts in an abstract feature space. Two alerts are mapped by two instances of the mapping function ($\mathcal{M}$), to two points in the abstract feature space. The two points are compared by the similarity function ($\mathcal{S}$). [57, Fig. 2]

3.4 LSTM RNN: Details of the mapping function

The mapping function is implemented with a single layer of 10 LSTM neurons with *tanh* non-linearities. In theory, a single hidden layer of sufficient size is enough to estimate any functional mapping [58, pp. 130-131]. Multiple sources suggest limiting layer size to what yields adequate performance, and in [57] 10 was found sufficient [59, pp. 22.11-22.12] [60, part 3] [61, p. 158]. Input is One-Hot encoded as encouraged by [59, p. 22.5]. Forward recurrence is used rather than bi-directional, for the improved performance. Finally, cosine similarity is used as the similarity function $\mathcal{S}$.

Alerts are interpreted as sequences of letters, hence $\mathcal{M}$ is implemented with an RNN. Figure 5.5 presents the same mapping function as in Figure 5.3, but with the recurrence made explicit. Letters are input one by one, with forward connections carrying over internal state encoding the preceding sub-sequence. The final output encodes the entire sequence in $\mathbb{R}^{10}$. Each

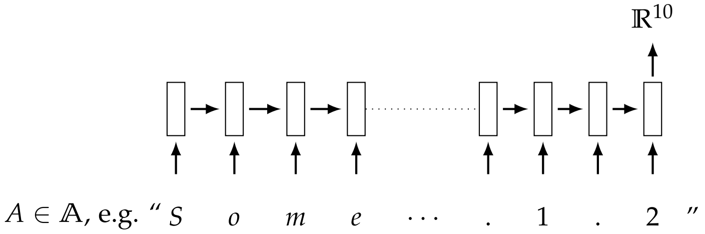

$\mathbb{R}^{10}$
$A \in \mathbb{A}$, e.g. "S o m e $\cdots$. 1 . 2 "

Fig. 5.5: Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) reading an alert, letter by letter. Letters at the bottom: Input example. Vertical arrows: Input and output connections. Horizontal arrows: Recurrent connections across elements in the input. Rectangles: Network as seen in Figure 5.3, repeated per element in the input, all having the same parameters.

recurrence is a fixed set of matrix multiplications and additions, and the alert length is limited, thus the computation of $\mathcal{M}$ scales well.

Pairs of alerts are built by making all possible combinations of the relevant sets of alerts, (Cartesian product in Set Theory or cross-join with self on a constant attribute in Relational Algebra). It is noted that if n is the size of the training alerts set, then the set of training pairs will be of size n^2, thus adding training alerts has a big impact on the memory and computation needed for training. Training is repeated for 10 epochs, each epoch utilising the entire data set, and each incrementally improving performance. For each epoch, training is done with randomly sampled mini-batches of 10000 samples, in order to add noise and to match computations to hardware. Experience is that noise aids in avoiding bad local optima. Training on a mini-batch is done with the Back-Propagation algorithm, meaning Stochastic Gradient Descent (SGD) towards a locally optimal loss (Binary cross entropy), controlled by a learning rate of 0.003. Back-Propagation with SGD is generally not guaranteed to find a global optimum.

3.5 LSA

LSA is an Information Retrieval method for learning a transformation from unlabelled data. Put shortly, LSA is to count word occurrences per document to obtain a Term Frequency (TF) matrix and apply Singular Value Decomposition (SVD) to it. From the SVD, one can construct a transformation that maps a vector of word counts into a space of lower dimension, where the basis vectors are those that account for the most variance in the TF matrix. The assumption is that this compression preserves and extracts the most useful information, while removing noise. The output of the transformation is typically much smaller size than the input. We are not aware of any previous examples of LSA being applied to IDS alerts, but it appears obvious

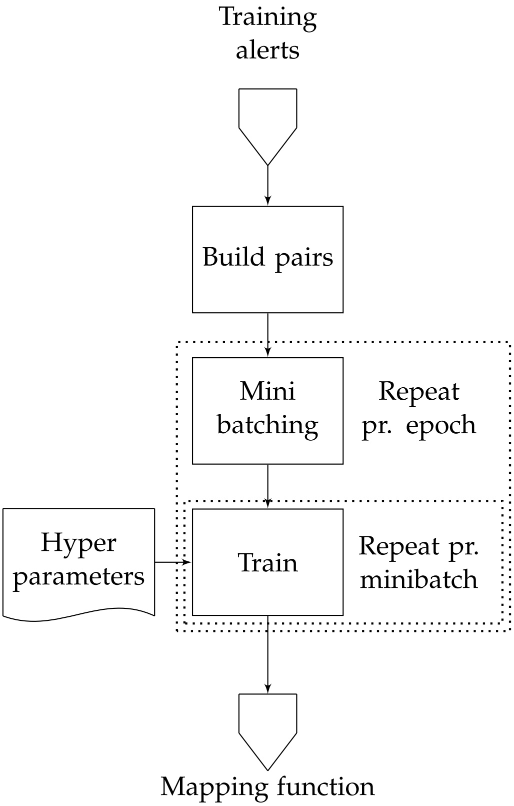


Fig. 5.6: Procedure for training the network depicted by Figure 5.4.

to consider alerts as documents. Furthermore, the derived transformation corresponds perfectly to our mapping function, and the output space to our abstract feature space, making LSA appear as a viable implementation of our general method.

The details of our LSA implementation, and corresponding considerations are as follows: Alerts often contain more than words, e.g. IP addresses, timestamps, or rule IDs, hence we count N-grams in place of words. To limit computation and memory usage, only 1-, 2-, and 3-grams are considered. N-grams found in only one or more than half of the alerts are expected to convey little information, so they are discarded. N-grams that are found in few alerts are assumed to be particularly relevant for describing those alerts, and conversely those N-grams found in many alerts are expected to convey little information. To implement this, the Term Frequency weighted by Inverse Document Frequency (TF-IDF) is used in place of plain TF. To limit computation and memory usage, the top 10.000 N-grams by term frequency across the corpus are used, the rest are discarded. Finally, the top 100 largest components of the SVD are used, both to limit computation and memory usage, as well as the size of the model for transformation, and to remove noise.

3.6 Clustering procedure

Second and third phases (Recall Figure 5.2) are captured by Figure 5.7. The second phase applies the learned mapping function to alerts to obtain their representation in the abstract feature space.

The third phase implements the DBSCAN clustering algorithm to the alert representations. In accordance with $\mathcal{S}$ being the cosine similarity, DBSCAN uses cosine as distance.

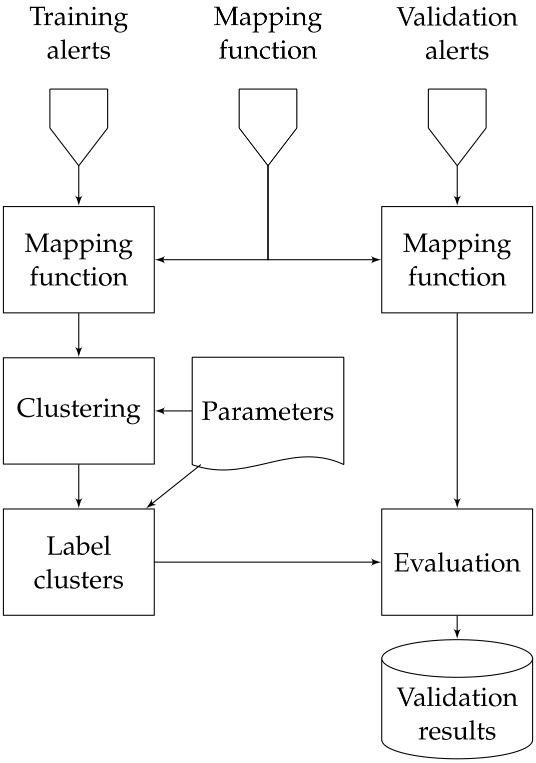


Fig. 5.7: Data flow for clustering.

To gain an understanding of the partitioning of the abstract feature space in a systematic way, clusters are labelled with incident IDs in the following way: Core points are used to represent clusters, and thereby the partitioning.

The most frequent incident ID for alerts in a cluster is taken to be the incident ID of the cluster. To handle varying frequency of alerts from different incidents, the majority vote is weighted by the inverse frequency of alerts from each incident in the whole data set. By imperfections in the learned mapping function, or in the used clustering algorithm, some clusters can be labelled as holding mostly false alerts. This is accepted, as grouping false alerts in a cluster still removes redundancy in the final result, although filtering them out altogether is preferred. Using the partitioning and the concepts of closeness from DBSCAN, it can be predicted to which incident in the training data a new alert belongs to, or if it is a false alert. Validation alerts are mapped into the abstract feature space and compared to the core points found in the training data set. If a point is within *eps* of a labelled core point, the validation point is classified as belonging to the same incident as the core point. If a point is not within *eps* of a labelled core point, it is an outlier, and the alert is predicted to be a false alert.

3.7 Section summary

To summarise, we have presented a general approach for extracting useful information from IDS alerts, to automate correlation and filtering without any feature engineering or expert knowledge, and with no ties to particular IDSs. Additionally, we have proposed two methods for learning mapping functions. The first employs a LSTM RNN architecture and learns from labelled alerts. The second is LSA applied to unlabelled alerts. Note that the implementations are available as open source[2]. We now turn to the methodology used for evaluating the two methods.

4 Evaluation

In this section, we describe how we evaluate our proposal. Our null-hypothesis is that correlation and filtering methods only can perform adequate for practical application if feature engineering, domain expertise, or both are used in the methods. To test this, the two methods presented earlier are applied to two data sets and scored by metrics capturing practical performance. In this way we seek to demonstrate that our proposed approach and implementations are counterexamples, leading us to reject the null-hypothesis, and conclude that adequate filtering and correlation can be achieved without feature engineering and embedding of domain expertise. To this end, we here introduce the two data sets, a view on practical alert processing, and a set of metrics that capture practical performance.

[2] `https://github.com/kidmose/lstm-rnn-correlation`.

As already discussed in Section 2, data for evaluation poses a challenge when working with correlation and filtering. The training procedure for the LSTM RNN and the need for hard evaluation metrics further constrains the possible data set to those where labels are available. We have identified two usable data sets, one from the Malware Capture Facility Project (MCFP) [30] and the other being the CIC IDS 2017 data set [56].

4.1 Data Set 1: MCFP Bot traffic merged with benign

Complete traffic traces of bot malware operating in a lab is provided by the MCFP[3] [30]. This data set is interesting because holds data on real bot infection, that have been allowed to execute in a lab, to simulate an incident. As traffic traces are provided per execution of individual bot malware sample and without benign traffic, labels can be applied efficiently to each incident before merging. On the other hand, the absence of benign activity and the resulting absence of false alerts pose a challenge to the representativeness of the data. To overcome this, we propose a procedure for obtaining false alerts and for merging all the alerts, as outlined in Figure 5.8. The procedure can be reused by others, resulting in a data set that is available to anyone for the parts where privacy allows it, and where the private part can be created according to the following description. This offers a not previously described point in the trade-off between representativeness, privacy, and availability of data.

The per bot traffic traces are processed with the Snort IDS[4] resulting in alerts, which are labelled with incident. Table 5.3 provides an overview of the data. For further details, see our previous work [57].

Bot	Alerts	Alerts (%)
1	100	4.36 %
2	184	8.53 %
3	317	14.69 %
4	328	15.20 %
5	390	18.07 %
6	395	18.30 %
7	444	20.57 %
Total	2158	100.00 %

Table 5.2: Overview of alerts raised on the MCFP bot traffic (Part of Data Set 1).

[3] Available from: `https://mcfp.felk.cvut.cz/publicDatasets/`

[4] Using Snort version 2.9.7.6, DAQ version 2.0.6, built in rules and `https://snort.org/rules/snortrules-snapshot-2976.tar.gz`, accessed October 6th, 2015.

False alerts are obtained by monitoring the traffic of tens of office user PCs for 35 days, using a Snort instance configured as above. It must be asserted that the alerts are indeed false, which is done by asserting three independent conditions. The first condition is that the monitored hosts/network is part of a well-managed corporate infrastructure, with fundamental information security mechanisms in place, such as user rights management, firewalls, antivirus, patch management, etc. This is believed to decrease the likelihood of a machine being infected. The second condition is that typical corporate methods and procedures, for identifying infected hosts are in place. This involves various solutions in the categories of both HIDS and NIDS, beyond what is part of the data collection setup described herein. The third condition is that heuristics are applied to screen for alerts that are likely to be true. Manual inspection is applied to the heuristically selected subset of alerts, with the goal of determining if the alert is false. Any host identified by an alert that cannot be confirmed to be false, is handled as if it was infected. The first heuristic is based on infections likely causing at least one priority 1 alert, thus all priority 1 alerts are inspected. The second heuristic is that any mention of the keywords `malware`, `malicious`, `blacklist`, `trojan`, and `bot` in the alert is a strong indicator of a true alert, thus alerts matching any of these keywords are inspected. The manual inspection involves the interpretation of alerts, the collection of relevant information, (exploit description, host name look-up, the presence of exploited service and patch level, other alerts on either source or destination host, interview with system owner, blacklists), and finally a conclusion on whether there is reason to suspect that any host was infected. In cases where suspicion remains, both source and destination hosts implied by the alert are considered infected. Consequently, alerts involving confirmed or potentially infected hosts are discarded.

During the 35 days, 5.548.539 alerts were raised, involving 10.907 different IP addresses, of which 1.552 IP addresses belong to unique hosts in the corporate domain. The existing corporate methods and processes had detected infections leading to 251.904 of the alerts being excluded (0 priority 1 alerts, 251.880 priority 2 alerts, and 24 priority 3 alerts). No alerts matched the keywords. 104 alerts of priority 1 were raised and inspected manually. Among these, 71 alerts were confirmed to be false alerts. The remaining 33 suspected true alerts lead to 713.737 alerts being excluded, (37 priority 1 alerts, 607.293 priority 2 alerts and 106.407 priority 3 alerts). Combining the above different grounds for discarding alerts, produces the used set of false alerts. As evident from Table 5.3, the alerts discarded, are distributed across the different priorities.

Simply pooling the true and false alerts into a single data set will introduce artefacts, where it is trivial to discern intrusions in time and in IP space. To avoid this, the alerts are rewritten to fulfil the following statements:

	Alerts	Prio. 1	Prio. 2	Prio. 3	Hosts	Corp. hosts
Recorded data set	5.548.539	104	4.028.411	1.520.024	10.907	1.552
Discard list 1	251.904	0	251.880	24	-	-
Discard list 2	713.737	37	607.293	106.407	-	-
Used data set	4.582.898	67	3.169.238	1.413.593	9.333	1.236

Table 5.3: False alert data set in numbers, with details of what was discarded. **Discard list 1**: Alerts with source or destination IP flagged by typical corporate methods and procedures. **Discard list 2**: Alerts with source or destination IP that was found in a priority 1 alert which could not be rejected as false. Note that alerts can be counted in both discard lists, hence summing across rows will not add up.

1. First alert of each incident is after first false alert.
2. Last alert of each incident is before last false alert.
3. Alerts of the same incident maintain their relative difference in time.
4. IP address of each incident victim is replaced with one appearing in the false alerts.
5. Resulting IP address must be unique for each incident victim.

The ambiguity that remains from the above is handled by randomising, using independent continuous uniform distributions across the possible time span, and independent uniform discrete distributions, across the set of possible IP addresses. To control the training time, $(O(n_{alerts}^2))$ the data set is stratified by discarding false alerts, so that they make up 50% of the data.

4.2 Data set 2: CIC IDS 2017

The CIC IDS 2017 data set[5] [56] stands out from other recent data set because it includes traffic traces, which can be passed to Snort, and because the accompanying homepage describes the incidents with sufficient details to label alerts by incident. This enables use of the data set for evaluating our filtering and correlation methods. Furthermore, the data is presumable available to anyone, it is not subject to privacy constraints, and it appears representative with contemporary types of incidents.

Alerts are again obtained by processing the individual traces with Snort. Labels are applied to alerts by comparing the timestamp of the alert to the

[5] Homepage: `http://www.unb.ca/cic/datasets/ids-2017.html`. Access kindly provided to the authors upon request.

time interval of incidents and by comparing the IPs of both. An alert is labelled with a given incident when both of two conditions are met: First, the alert timestamp must match the time interval of the incident. Second, either source or destination IP of the alert must match either the attacker or victim of the incident. A network device performed Network Address Translation (NAT) in the data collection setup, which makes is meaningless to match both source and destination to attacker and victim. Lack of comprehensive details on port usage for incidents and details of the NATing process makes it impossible to reconstruct the translations. This also leads to the exception that both IP addresses of the NATing device are ignored when matching. There are no cases where multiple labels match an alert. If an alert fails to match on either or both of the time and IP conditions, it is labelled as a false alert.

The number of total alerts amounts to 431860, which compares poorly with the 4361 alerts in the final Data Set 1. A stratified down sampling of Data Set 2 to approximately one hundredth of the previous size will mean that some incidents are going to have only one or zero alerts, which is problematic. Instead, only incidents with more than 200 alerts are down-sampled to 200 alerts, and then false alerts are down-sampled to match the total size of Data Set 1. A summary of the alerts and their labels is shown in Table 5.4.

4.3 Metrics

A set of commonly used metrics for detection and classification problems, are also used for evaluating the performance of the proposed method. We use accuracy, precision, recall, and F1 score as defined by Equations (5.6)-(5.9):

$$\text{accuracy} = \frac{TP + TN}{FN + FP} \tag{5.6}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{5.7}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{5.8}$$

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.9}$$

True Positive count (TP) refers to the count of correct positive outcomes/class assignments, False Negative count (FN) to the count of incorrect negative outcomes/assignments to other classes, False Positive count (FP) to the count of incorrect positive outcomes/class assignments and, True Negative count (TN) to the count of correct negative outcomes/assignments to other classes.

The use of these metrics enables comparison with other works within the field, and they are well suited to describe performance on detection and clas-

	Alerts		Alerts (Stratified)	
Incident	Count	%	Count	%
False alerts	172447	39.93	1894	43.88
Bot	2812	0.65	200	4.63
DDoS	23111	5.35	200	4.63
DoS GoldenEye	2200	0.51	200	4.63
DoS Hulk	223760	51.81	200	4.63
DoS Slowhttptest	2172	0.50	200	4.63
DoS slowloris	162	0.04	162	3.75
FTP-Patator	441	0.10	200	4.63
Heartbleed	92	0.02	92	2.13
Infiltration	3312	0.77	200	4.63
PortScan	596	0.14	200	4.63
SSH-Patator	387	0.09	200	4.63
Web Attack BF	195	0.04	195	4.52
Web Attack SQLi	15	0.00	15	0.35
Web Attack XSS	158	0.04	158	3.66
Total	431860	100.00	4316	100.00

Table 5.4: Alerts raised on the CIC IDS 2017 data set. Break-down by labels. The second and third columns are before stratification. The fourth and fifth columns are after (Data Set 2).

sification problems in general. However, they fail to capture the actual value gained from applying correlation and filtering methods, as also mentioned in Section 2. In the following, we propose a set of metrics to address this, motivated by a model of operation and by general costs associated with a Security Operations Center (SOC).

Any corporation or other large organisation is expected to have some unit handling (hyper) alerts and incidents, which we refer to as a SOC. Automatically reacting to every (hyper) alert, including the false, will clearly cause too many disruptions, hence it is assumed that (hyper) alerts are processed manually, before any action is taken. The processing of an (hyper) alert, by an analyst, is assumed to have a unit cost. In reality, the required time and level of expertise varies, and thereby so does cost, but with large volumes this simplification is reasonable. Given a hyper alert, we assume that one alert is picked at random from the hyper alert, and used to determine what has happened, and what action is to be taken. The analyst is assumed to be perfect, meaning that when analysing an alert, the analyst correctly identifies the victim, the incident, and whether it is a false alert. Assuming that the alerts of a hyper alert are sufficiently homogeneous, i.e. hyper alerts mostly hold alerts of the same incident, this can be expected to work well. Given this model of a SOC, the ratio of alerts to hyper alerts is also the ratio between cost of analyst workload, with and without filtering and correlating. In the following, this will be formalised as the Normalised Alert Reduction Factor (ARF).

The other side of operating a SOC, is the risk of missing incidents altogether. Setting the cost of this is extremely hard, as incidents might be stopped from causing harm through other mechanisms than those initiated by the SOC, or in worst case they can potentially be the end of the corporation. Acknowledging that the cost of an incident is impossible to describe in general, we propose to consider the rate at which incidents are missed. This provides insights as to the probability that a random incident will be missed and can be used as an input to risk management, where uncertainties already need to be handled. A metric named Incident Miss Rate (IMR) is defined to capture this.

For use in the formal definitions, the following sets of alerts, hyper alerts and incidents are defined:

$$\mathbb{A}_{in} : \text{Set of all alerts, before pre-process} \tag{5.10}$$

$$\mathbb{A}_{out} : \text{Set of all alerts, after filtering by pre-process} \tag{5.11}$$

$$\mathbb{I}_{in} : \text{Set of all incidents, before pre-process} \tag{5.12}$$

$$\mathbb{I}_{out} : \text{Set of hyper alerts, produced by pre-process} \tag{5.13}$$

$$|\mathbb{X}| : \text{Number of elements in set } \mathbb{X}$$

As already stated, ARF describes ratio from all input alerts to the number of hyper alerts:

$$\begin{aligned} ARF\colon \mathbb{A}_{in}, \mathbb{I}_{out} &\to \mathbb{R} \\ \mathbb{A}_{in}, \mathbb{I}_{out} &\mapsto \frac{|\mathbb{A}_{in}|}{|\mathbb{I}_{out}|} \end{aligned} \tag{5.14}$$

$$\begin{aligned} &ARF \leq |\mathbb{A}_{in}| && ARF = |\mathbb{A}_{in}| \iff |\mathbb{I}_{out}| = 1 \\ &ARF \geq 1 && ARF = 1 \iff |\mathbb{A}_{in}| = |\mathbb{I}_{out}| \end{aligned}$$

IMR captures how likely it is that an incident will be missed altogether. By comparing how many incidents are in the input data, against how many are represented in the output, it can be measured how often the method makes an error leading to an incident being missed altogether. Ideally this metric is zero, as that means no incidents are missed, while a value of one, means all incidents are missed.

$$\begin{aligned} IMR\colon \mathbb{I}_{in}, \mathbb{I}_{out} &\to \mathbb{R} \\ \mathbb{I}_{in}, \mathbb{I}_{out} &\mapsto \frac{|\mathbb{I}_{in} \setminus \mathbb{I}_{out}|}{|\mathbb{I}_{in}|} \end{aligned} \tag{5.15}$$

$$\tag{5.16}$$

$$IMR \leq 1 \qquad IMR = 1 \iff |\mathbb{I}_{in} \setminus \mathbb{I}_{out}| = |\mathbb{I}_{in}| \tag{5.17}$$

$$IMR \geq 0 \qquad IMR = 0 \iff |\mathbb{I}_{in} \setminus \mathbb{I}_{out}| = 0 \tag{5.18}$$

The above definition of IMR in Equation (5.15) provides for the understanding of how the metric is relevant, but the notion of an incident not being represented in the output hyper alerts ($\mathbb{I}_{in} \setminus \mathbb{I}_{out}$) is impractical. Instead, we recall that the analyst picks an alert at random, correctly identifies the incident of that alert, and associates the hyper alert with that incident. The probability that the ID of a given incident, i, comes out as the ID of hyper alert o, is equal to the count of alerts in o that have the given ID, divided by the number of alerts in the hyper alert (Equation (5.19)). The likelihood that a certain incident ID is not present in any of the produced hyper alert, is the product of the probabilities of not being in each of the individual hyper alerts (Equation (5.20)). Based on this, IMR is calculated as the expected rate at which an incident is not being found by the described "oracle analyst" (Equation (5.21)).

$$P(ID|o) = \sum_{a \in o} \mathbb{1}(a \in i)/|o| \tag{5.19}$$

$$P(\neg ID|\mathbb{I}_{out}) = \prod_{o \in \mathbb{I}_{out}} 1 - P(ID|o) \tag{5.20}$$

$$\begin{aligned} IMR &= \sum_{i \in \mathbb{I}_{in}} P(\neg ID|\mathbb{I}_{out})/|\mathbb{I}_{in}| \\ &= \sum_{i \in \mathbb{I}_{in}} \prod_{o \in \mathbb{I}_{out}} \left(1 - \sum_{a \in o} 1(a \in i)/|o|\right)/|\mathbb{I}_{in}| \end{aligned} \tag{5.21}$$

5 Results

Both of the two implementations (LSTM RNN and LSA) has been applied to both of the two data sets (Data Set 1, MCFP merged with false alerts, and Data Set 2, CIC IDS 2017). Each of the four combinations of implementation and data set was executed ten times, each time learning a mapping function and finding clusters from training cuts that consists of $9/10^{th}$ of the data.

The remaining $1/10^{th}$ made up the non-overlapping, left-out validation cut for each execution.

To enable comparison with prior work, classification metrics for using the labelled clusters to predict incidents of are presented in Table 5.5. Numbers are the mean over the ten executions.

Implementation	Data set	Acc.	Prec.	Rec.	F1
LSTM RNN	1	0.737	0.731	0.737	0.720
LSTM RNN	2	0.403	0.401	0.403	0.381
LSA	1	0.826	0.865	0.826	0.793
LSA	2	0.694	0.748	0.694	0.649

Table 5.5: Classification metrics (Accuracy, Precision, Recall and F1-score) for predicting incidents of validation alerts based on clustering of training alerts. Broken down by implementation applied and by data set. The reported numbers are the mean of 10 executions with non-overlapping validation cuts of the data. Higher is better.

Table 5.6 holds the domain specific metrics that we have proposed, also for both implementations and both data set. As this is the measuring point that we argue is relevant for practical purposes, worst- and best-case performance is included in addition to the mean. Figure 5.9 holds a scatter plot of IMR and ARF for all four combinations and the ten executions of each, i.e. the data aggregated in Table 5.6. Data set is encoded by the shape using dots for Data Set 1 and squares for Data Set 2. Implementation is encoded by the colour using Blue for LSTM RNN and red for LSA. Performance increases when moving up (Higher ARF) and to the left (Lower IMR).

As a mean to identify and understand systematic errors confusion matrices are included for each combination of implementations and data sets with Figures 5.10, 5.12, 5.11, and 5.13. Note that the numbers are normalised according to support for each label to avoid suppressing the incidents with few alerts. Ideally, diagonal values are one and off-diagonal values are zero, as this would indicate no confusion among incidents.

		IMR			ARF		
Implement.	Data set	min	mean	max	min	mean	max
LSTM RNN	1	$0.00e^{0}$	$6.18e^{-3}$	$1.79e^{-2}$	$8.31e^{0}$	$9.25e^{0}$	$9.82e^{0}$
LSTM RNN	2	$0.00e^{0}$	$6.12e^{-2}$	$1.09e^{-1}$	$2.44e^{0}$	$3.00e^{0}$	$3.39e^{0}$
LSA	1	$0.00e^{0}$	$7.26e^{-17}$	$1.33e^{-16}$	$2.16e^{1}$	$2.33e^{1}$	$2.54e^{1}$
LSA	2	$8.93e^{-2}$	$9.93e^{-2}$	$1.64e^{-1}$	$4.19e^{0}$	$4.55e^{0}$	$5.07e^{0}$

Table 5.6: Application domain metrics for performance of LSTM RNN and LSA implementations on Data Set 1 and 2. Aggregation spans 10 non-overlapping validation cuts of the data. Incident Miss Rate (IMR) describes how likely it is that an incident is missed, so lower is better. Normalised Alert Reduction Factor (ARF) describes the reduction in manual effort, so higher is better.

6 Discussion

Two implementations have been presented and evaluated, which calls for a comparison. By the commonly used classification metrics of Table 5.5 LSA beats LSTM RNN on all metrics (Fixing data sets for a fair comparison). The picture is similar for domain specific metrics of Table 5.6 with the exception of IMR on Data Set 2. For IMR on Data Set 2, LSTM RNN beats LSA, but the values are less than an order of magnitude apart. So, classification metrics clearly ranks LSA over LSTM RNN, while the domain specific metrics largely agree, but on one point they disagree or at least turns out inconclusive. One possible interpretation is that classification metrics are better suited for capturing performance differences, while the domain specific metrics fail to separate the two implementations. Another is that the domain specific metrics show the right picture in the light of practical application value, while the conclusion drawn on classification metrics is wrong due to a disconnect from the application domain. The missing link between classification metrics, the correlation and filtering problems, and the values and risks of the application domain is an interesting topic to explore even further.

To better understand the difference in performance between LSTM RNN and LSA we inspect the per execution performance as plotted in Figure 5.9. It is noted that for Data Set 2 (Squares), which unanimously is the most difficult of the two, the implementation appears to behave different. While LSA (Red) consistently outperforms LSTM RNN (Blue) in terms of ARF (High is better), the picture for IMR is more complex (Left is better). LSA is most consistent with a single outlier, while the performance of LSTM RNN spans a larger range in what appears as four different steps. The single outlier for LSA can be explained by inspecting the confusion matrices of each execution, and their aggregate which is presented in Figure 5.13. The aggregated and nine of the ten confusion matrices are similar in that they have non-zero diagonal values, except for *DoS Hulk* alerts. This indicates that in general the procedure of learning a mapping function with LSA, applying it, clustering the result, and making prediction using the clusters will capture at least some alerts from all incidents, except *DoS Hulk*. The one exceptional execution, represented by the outlier red square in Figure 5.9, differs in that the diagonal value for *SQL Injection* is also empty, indicating a failure to capture this incident. So where nine executions of LSA on Data Set 2 failed completely to capture exactly one specific incident, the tenth execution stands out by failing for an additional incident, which explains the increased IMR of the outlier. We note that 58% of the errors for *DoS Hulk* are due to confusion with another volumetric DoS attack happening two days later. It is unsurprising that the *SQL Injection* incident triggers the second failure, as it is by far the incident with the lowest number of alerts (15 alerts or 0.35% of Data Set 2, as per Table 5.4). With 13 or 14 alerts in the training data and only one or two in the validation data we find it somewhat surprising that our approach was able to capture this very imbalanced class in nine of ten executions.

Returning to the topic of classification metrics vs. the proposed domain specific metric, one noteworthy point is that LSTM RNN only achieves 40.3% accuracy on Data Set 2. As per Table 5.4 43.88% of the alerts are false, so crude “Filter out all alerts” or “Label all alerts by most common label” implementations would beat LSTM, and it is therefore tempting to state that LSTM RNN fails for Data Set 2. However, the domain specific metrics in Table 5.6 shows that LSTM RNN is still capable of reducing the number of alerts to a third. We claim that any practitioner or manager responsible for a SOC would find it adequate to cut a large workload down to a third, which is a point that the classification metrics misses completely. The results also indicate that the implementation can be expected to miss 6.12% of incidents. This has to be weighed against the benefits, in particular the limited cost of deployment and maintenance. In settings where it is important to lower the resources invested, and where there is willingness to accept a higher risk, this could very well be acceptable. Where it is crucial to minimise risk, and where large investments are of no concern, this might not be sufficient. In the end, finder the optimal point in this trade-of requires known costs which are not general, and we leave it to practitioners to pass judgement on this. Again, the classification metrics of Table 5.5 fail to capture the above, while the proposed domain specific metrics describes the gain (ARF) and risk (IMR), which can be used in a practical setting to determine if the implementation is relevant. The domain specific metrics proposed are therefore the best way to capture correlation and filtering performance for practical purposes.

Both domain specific metrics (Table 5.6) and classification metrics (Table 5.5), are remarkably better on Data Set 1 compared to Data Set 2. Figure 5.9 generally confirms that executions running on Data Set 1 (Dots) exhibit better performance than executions with Data Set 2 (Squares). The one exception is one of the ten executions of LSTM RNN on Data Set 2 which exhibits an IMR of 0, which is on par with the best executions on Data Set 1. This suggests that Data Set 1 is significantly less challenging than Data Set 2. To explore this, one could evaluate more methods on the same data to understand if it is only less challenging for the implementations of our approach or if it is generally easier. Being able to observe this difference in the first place highlights the value of evaluation on multiple data sets, as the common practice of using 10 validation cuts does not identify this issue. Possible reasons that Data Set 2 is harder than Data Set 1 includes: It holds more incidents (14 vs 7), it spans a shorter period of time (5 days vs 35 days), the incidents are more diverse, the incidents are more contemporary, and the incidents are perhaps more random as they are driven by human rather than software (bot malware).

Inspecting the confusion matrices for the two implementations and the two data sets (Figures 5.10, 5.11, 5.12, and 5.13), provides for some interesting observations. We find three to be particularly interesting. First, most of the outcomes are on the diagonal, meaning that ground truth label matches the predicted label. If the output of the mapping function, which is the input for clustering and classification, is random and without relevant information this pattern is extremely unlikely. Therefore, this proves success on learning a mapping function without feature engineering or embedding of domain expertise. If the performance is adequate is discussed below. The second observation is that the first column, which indicates a prediction of "false alert", is also substantial. This indicates that a true alert was mistakenly classified as false, i.e. filtered out. Keep in mind that due to the nature of the filtering and correlation problems such an error is not equivalent to missing an incident, but it contributes to the risk of that happening. It does however show that the mapping function, clustering, and classification in combination are less than perfect solutions, which makes it relevant for further exploration. A specific direction could be to explore if it is possible to adjust the aggressiveness for filtering and correlating, in order balance the trade-of between IMR and ARF in practice. It is also possible that some of this is due to bad labelling. False alerts might have occurred on the malicious parts of Data Set 1 due to e.g. background activities of the operating system, and malicious activity in the benign part might have slipped through our procedure described in Section 4.1. For Data Set 2 we did observe some inconsistencies in timestamps between the descriptions and the labelled flow data that is distributed along with the raw traffic. The flows were not used for this work, but it indicates that errors are present, in this particular case perhaps due to human involvement. Thirdly, there are some significant deviations from the above two types of observations. In Figure 5.10 there are few errors ($70\% - 99\%$ are correct) for incidents 2-5, and the errors made are not confusions towards false alerts. This can be due to the LSTM RNN implementation being well suited for capturing the relevant information, as the LSA implementation (Figure 5.11) are much more prone to filter out incidents 2 and 4. Both implementations have a tendency to confuse $12\% - 14\%$ of the alerts on incident 2 and $20\% - 24\%$ of alerts on incident 6 as being incident 5. This might be explained by similarity in the raised alerts, which is possible as it is two examples of bot malware. In Figure 5.12 we see many deviations from the diagonal and the first columns. The most significant is 67% of alerts on *Web Attack – Sql Injection* (11) being confused as *DoS Slowhttptest* (5). As both incidents are using the HTTP protocol to deliver application level attacks this confusion is not very surprising, and with only 15 total alerts on (11) it is expected that the bias is towards (5), cf. Table 5.3. Other significant deviations are other DoS attacks being confused as *DoS Slowhttptest* (5). In Figure 5.13 the only substantial deviation from the diagonal and the first column is that 58% and 56% of *DoS Hulk* (6) and *DoS GoldenEye* (7) are confused as being *DDoS* (15). This is can be explained by all three being DoS attacks, and in particular by DoS Hulk also spoofing random sources thereby mimicking the distributed nature of DDoS. Furthermore, both DoS Hulk and DoS GoldenEye use randomly generated data (User-Agent and Referred in HTTP) and similar techniques (HTTP Keep-Alive and no-cache also in HTTP). In summary, many outcomes are on the diagonal, meaning that they are correct, many are in the first row, meaning they are incorrectly filtered out, and the remainder of errors have possible explanations based on deeper understanding of the data.

Having discussed the implementations, the metrics, and the data sets, we now turn to the most important question: Can the proposed general approach be used to filter and correlated alerts with adequate real-world performance without investing time and resources in feature engineering and without depending on domain expertise? Our general approach, and thereby the LSTM and RNN implementation are done without any feature engineering or domain expertise, so the question can be rephrased to: Do the evaluation results indicate that the implementations perform adequately to be of practical relevance? The factor of eliminated manual analysis effort can be expected to be in the range from 3.00 to 23.27 (Mean ARF, Table 5.6). As already argued, we are confident that cutting a substantial task to a third is indeed valuable and worth doing. Especially as the proposed approach provide implementations that are easy to apply compared to other methods that demand substantial investments in feature engineering and domain expertise which can make them unfeasible in practice. Incident miss rates are ranging from $7.26e - 17$ to $9.93e - 02$. In the best case, the risk is practically none. In the worst case, missing one in ten incidents might appear concerning but let us consider the alternatives. Manually processing all alerts is practically impossible for large network, and should one choose this approach it will still be subject to risk of missing incidents due to error and alert fatigue Existing methods have poor adaptability and require insurmountable investments for feature engineering and domain expertise, making them unfeasible. Even if the investments should be deemed acceptable existing methods are still not flawless, but for obvious reasons no prior work have reported their IMR for comparison. Taking a step back, the IDSs are not flawless either as is the case for many other applied security mechanisms. This is why practitioners apply the principle of layered security such that multiple mechanism can complement each other. As IDSs together with filtering and correlation methods are intended to fit in such a setting, we believe the worst IMR is acceptable.

Our future effort will focus on exploring other variations under the general approach that we have proposed and evaluated here, seeking to improve the performance. As always, more extensive evaluation on more data will contribute to our understanding, thus we will pay attention to such possibilities. Experience shows that the need for raw traffic and the labelled alerts is difficult to fulfil. One possible way to overcome this is to relax the condition that the same IDS must be used for all data sets, effectively requiring raw traffic. The drawback of this is that the IDS then becomes another unknown when comparing performance on different data sets. It would also be very interesting reproduce some of the existing methods and apply them to the same data as ours, but preliminary efforts have proven that this is difficult. Curiously, this is very much in line with our reasoning that feature engineering and embedded domain expertise degrades adaptability.

7 Conclusion

In this work we have described the correlation and filtering problems, which are that Intrusion Detection Systems (IDSs) raise false alerts and alerts that carry redundant information. The implication is that alerts needs to be preprocessed to be of any practical use for detection. Existing solutions apply feature engineering and embed domain expertise. We argue that this approach breaks adaptability and cause a need for substantial investments in expert time for each deployment, resulting in the approaches currently being unfeasible in practice. To overcome this, we propose a novel approach that is free of feature engineering and embedded domain expertise. In our approach a mapping function is learned from data, such that any alert can be mapped into an abstract feature space of limited dimensions. In this space, alerts can be clustered according to the incident, removing redundant information, and filtering out false alerts. We propose a set of metrics that measure the value of applying a filtering and correlation in practice. The evaluation results show that correlation can be done without feature engineering or domain expertise embedded in the method. The key contribution is an approach that is feasible for widespread practical use and provide adequate performance, as opposed existing methods that require substantial investments for each deployment, making them unfeasible. We conclude that this method is relevant for practical purposes and will pursue further development.

Acknowledgements

This work was supported by Innovation Fund Denmark, Industrial PhD Programme, grant no. 5016-00018.

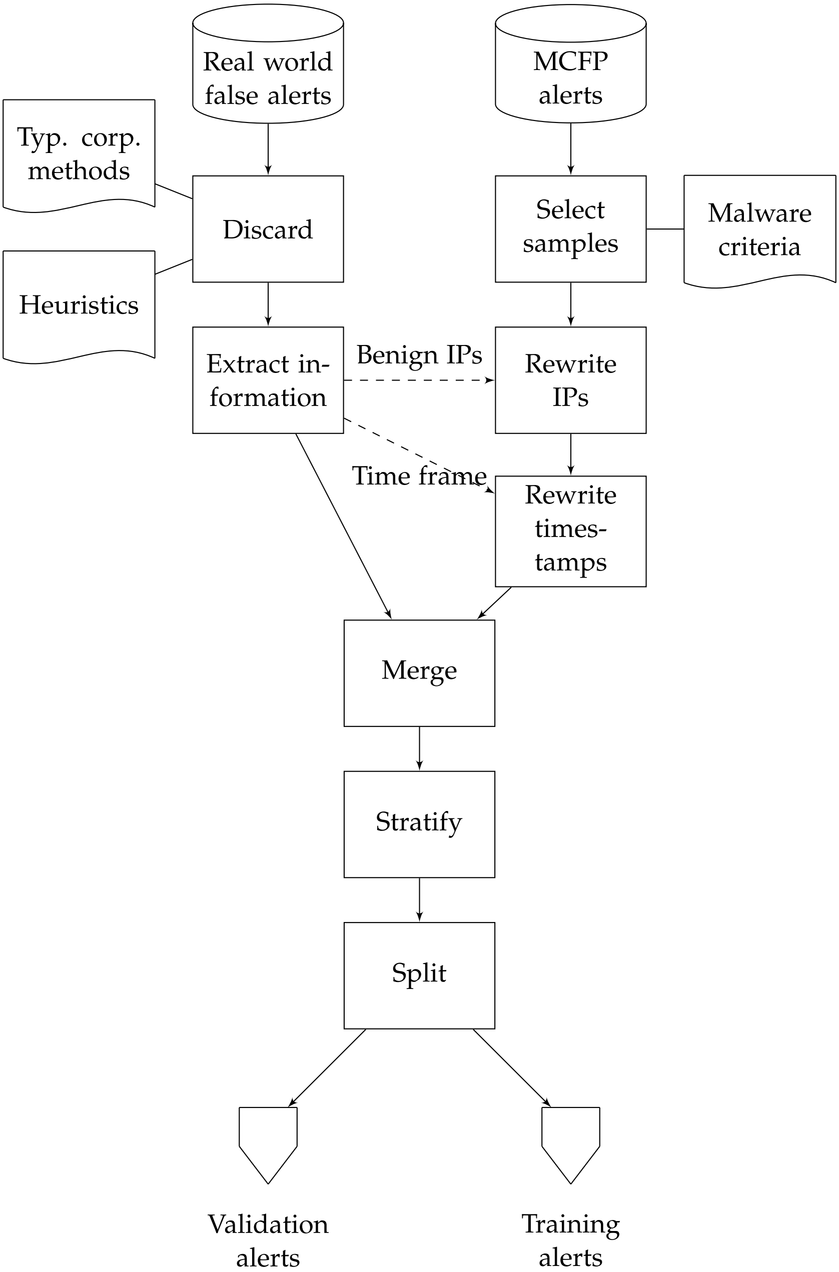


Fig. 5.8: Data flow graph for data preparation.

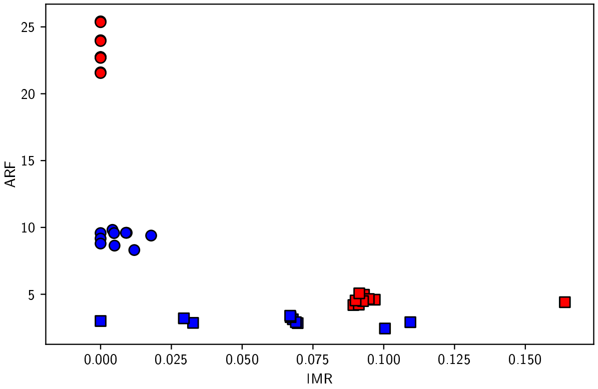
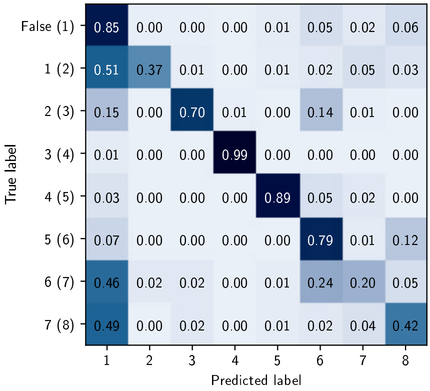


Fig. 5.9: Application domain metrics plotted for LSTM RNN (Blue) and LSA (Red) implementations, applied to Data Set 1 (Dots) and Data Set 2 (Squares). Each point represents one of 10 non-overlapping validation cuts of the data (10 cuts for each of the four combinations, 40 points total). Incident Miss Rate (IMR) describes how likely it is that an incident is missed, so left is better. Normalised Alert Reduction Factor (ARF) describes the reduction in manual effort, so higher is better.



	1	2	3	4	5	6	7	8
False (1)	0.85	0.00	0.00	0.00	0.01	0.05	0.02	0.06
1 (2)	0.51	0.37	0.01	0.00	0.01	0.02	0.05	0.03
2 (3)	0.15	0.00	0.70	0.01	0.00	0.14	0.01	0.00
3 (4)	0.01	0.00	0.00	0.99	0.00	0.00	0.00	0.00
4 (5)	0.03	0.00	0.00	0.00	0.89	0.05	0.02	0.00
5 (6)	0.07	0.00	0.00	0.00	0.00	0.79	0.01	0.12
6 (7)	0.46	0.02	0.02	0.00	0.01	0.24	0.20	0.05
7 (8)	0.49	0.00	0.02	0.00	0.01	0.02	0.04	0.42

Fig. 5.10: Normalised confusion matrix for detection based on mapping function learned with LSTM RNN on Data Set 1. Diagonal values of 1 and off-diagonal values of 0 is ideal.

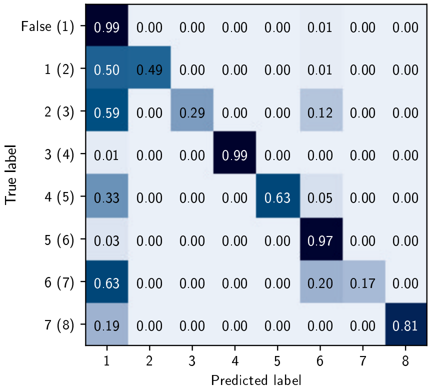


Fig. 5.11: Normalised confusion matrix for detection based on mapping function learned with LSA on Data Set 1. Diagonal values of 1 and off-diagonal values of 0 is ideal.

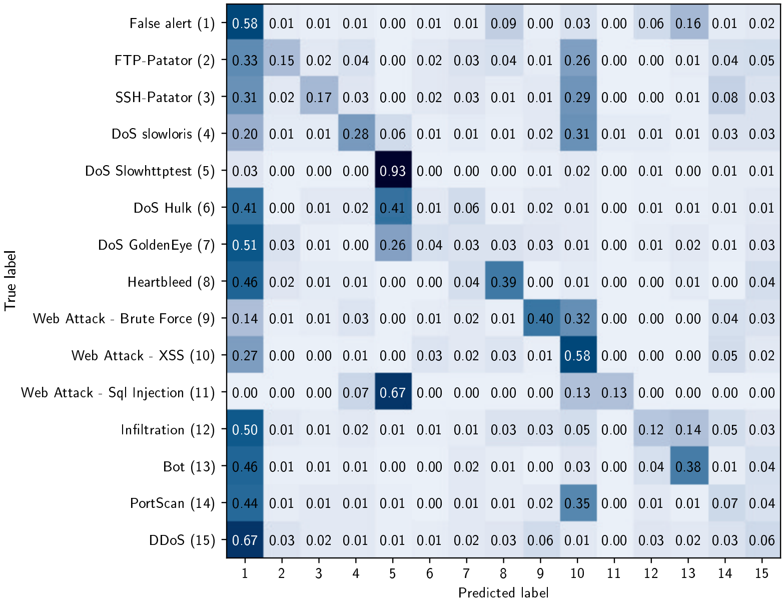


Fig. 5.12: Normalised confusion matrix for detection based on mapping function learned with LSTM RNN on Data Set 2. Diagonal values of 1 and off-diagonal values of 0 is ideal.

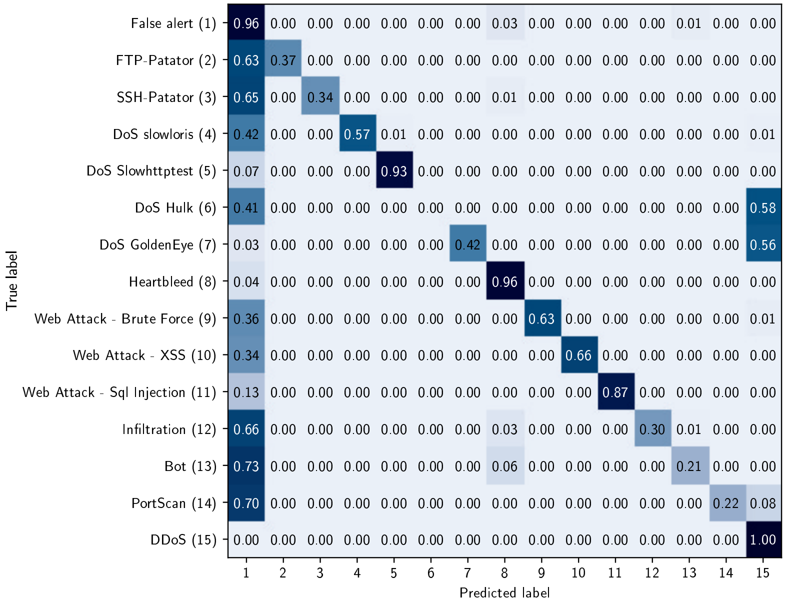


Fig. 5.13: Normalised confusion matrix for detection based on mapping function learned with LSA on Data Set 2. Diagonal values of 1 and off-diagonal values of 0 is ideal.

Chapter 6

Detection of malicious and abusive domain names

Egon Kidmose, Erwin Lansing, Søren Brandbyge, and Jens Myrup Pedersen

The paper has been published in the
Proceedings of The 2018 1st International Conference on Data Intelligence and Security (ICDIS), April 2018.

Abstract

The Domain Name System (DNS) is a critical component of the Internet, and as such it is widely relied upon by a large part of the world. Consequently, it can be abused for multiple purposes, with financial gain being perhaps the most obvious, and important. An important countermeasure to such criminal and malicious activity is to identify involved domains, in order to blacklist or otherwise disable them. In this paper we provide the results of studying existing work on detecting malicious domains and analyse the findings. We identify an approach which is promising but has received surprisingly little attention; Pre-registration detection. We identify the following gaps between the problem of domain abuse, and the described state-of-the-art: Existing work on Pre-registration is strictly focused on a single form of abuse, spam, hence it must be explored if Pre-registration detection can be applied to other forms of abuse as well. Existing work, on both Pre- and Post-registration detection, is focused on a few Top-Level domains (TLDs) and Registries, prompting for studies with other TLDs and Registries. There is relevant information, including Registrant-based features, that has not yet been used for Pre-registration detection – which also calls for investigation. Finally, a study of a real-world deployment of Pre-registration detection at a Registry has not yet been presented, despite the potential of the approach. We contribute with an analysis of existing work, by identifying the state-of-the-art, and by identifying important areas of future work.

1 Introduction

The Domain Name System (DNS) makes it simple for Internet users to refer to machines, or services, with human readable domain names, rather than machine readable Internet Protocol (IP) addresses. This feature, along with some more technical features relating to robustness and availability, makes DNS an essential component of the Internet. With the rise of malicious and criminal activity on the Internet, cyber criminals have naturally included DNS in their operations as well. Criminals can both use DNS as a component in their abusive infrastructure – like it was designed for, but to support a malicious scheme – or they can use it as a vector for attacks. As a trivial example of the former, malware can be configured to exfiltrate stolen credentials through Command and Control (CnC) infrastructure, identified by a domain name, as opposed to an IP address. This has the following three benefits for the criminals; 1) Management and configuration is simpler when dealing with domain names, rather than IP addresses. 2) Domain-to-IP mappings can be changed rapidly, providing flexibility, and resilience, to law enforcement take-downs of hosts. 3) Domain-to-IP mappings can be one-to-many, and many-to-one, providing resilience to take-downs, some anonymity for criminals, and robustness to host unavailability. An example of the latter use case – using DNS as an attack vector – can be observed in some e-mail phishing attacks. The attacker's goal is to lure victims into performing some action, e.g. follow a link, or to disclose information, enabling some malicious scheme. For instance, the attacker can register a domain name resembling that of a legitimate bank, send e-mails to victims regarding urgent issues with their accounts, and provide a hyperlink with the closely-resembling, but malicious, domain. When a victim fails to recognise the difference, follows the link, and logs in to the attacker's fake banking site, the attacker has gained the victim's credentials. A third class of abuse, which is not considered in the present work, is to exploit security deficiencies in the protocol itself, e.g. for Distributed Denial of Service (DDoS) amplification or cache poisoning.

In our study of prior work on detecting malicious domains (Section 3), we note that most proposals rely on information about DNS usage. This implies that malicious domains are allowed to operate, at least for some time, and provides a window of opportunity for cyber criminals. The distributed nature of DNS, including caching at various levels, makes it difficult to reason about the possibilities for preventing abuse, in this setting. Obviously, detecting and mitigating abusive domains, while attacks might be pending, or ongoing, is better than nothing, but less than ideal.

We instead propose, to detect abusive domains based only on information available before domain registration applications are completed, which we refer to as Pre-registration detection. This makes it possible to implement the detection at the Registrar or Registry, where it is possible to reject registrations, intervening before any abuse can occur. This approach differs from the above-mentioned existing work in multiple ways:

1. It is guaranteed that the detected and rejected domains are never used in criminal and malicious schemes.

2. The time requirements at registration time are much more relaxed, than at query time.

3. The detection can only rely on information available before registrations are completed.

In Section 2 this paper provides some background on DNS, including the highly relevant registration process, and on DNS abuse. Building on the background, and some familiarity with DNS in general, we summarise the findings of our extensive study of existing work in Section 3. Gaps found in the existing work, and the resulting potential for improvements are discussed in Section 4. We make the primary contribution of identifying multiple, highly interesting areas of future work, before concluding in Section 5.

2 Background

Studying the detection of malicious domains is aided by an understanding of DNS, as well as an understanding of malicious activity. For a basic introduction to how DNS resolves domain names to IP addresses, we refer the reader to existing work, such as [62]. The essential domain registration process appears to receive less attention, as we only found it described in [63–65]. In the following, we elaborate on the DNS usage, and the registration process, by putting it in a temporal context, and by including consideration of malicious usage, i.e. abuse.

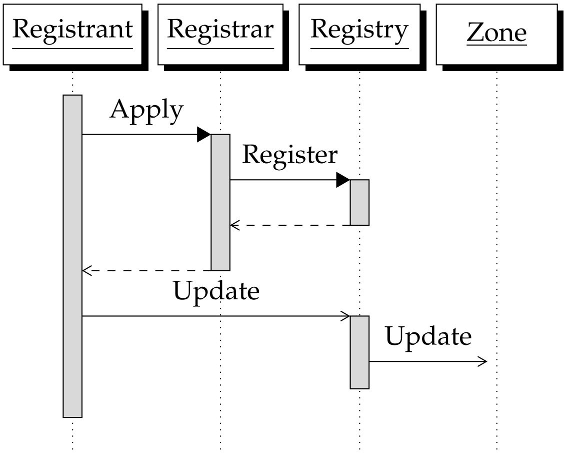


Fig. 6.1: Sequence diagram for the domain registration process.

The process to register a Second Level Domain (2LD) is outlined in Figure 6.1. A **Registrant**, who would like to register a 2LD, such as `example.com`, initiates the process. The Registrant applies with a **Registrar**, who must be accredited by the **Registry** operating the given Top-Level domain (TLD). The Registrar passes the application to the Registry. If the domain is available, and the application meets some given requirements, the Registry grants the Registrant the right to use the 2LD, under some agreed conditions, which for instance can defined what constitutes abuse. The registration is now formally completed, but the domain cannot be resolved until the TLD Authoritative Name Server (ANS) zone is updated. Hence, the domain cannot be abused yet. The Registrant provides details of the 2LD ANS to the Registry, who updates the TLD zone, to delegate authority of the 2LD. After the domain registration and the first zone update, the domain can be resolved and used, as illustrated in Figure 6.2. It is important to note that the events included have a strict ordering, allowing us to define the following periods:

Pre-registration is the time before the first update to the zone, where it can be guaranteed that the domain has not been abused on the Internet, as it has not yet been published in the TLD zone[1]. The Pre-registration information available to the Registrar includes the 2LD name, the Registrants payment information (including billing address), and the physical address of the Registrant. As for the Registry, the information available Pre-registration includes: 2LD name to be registered, Zone update (including ANS for the 2LD), and the identity of the Registrar. In some cases, such as for Country Code Top-Level Domains (ccTLDs), but not for old Generic Top-Level Domains (gTLDs) like `.com` and `.net`, the Registry must also maintain a record of the Registrants contact details, and address. Individual Registries might impose, or be imposed, requirements for further information. In the specific case of `.dk`, the Danish ccTLD, national Registrants are also required to validate with a national digital authentication scheme.

Post-registration is defined as the time after the first zone update. As the 2LD can be resolved at this time, information relating to actual usage becomes available; The TLD ANS, operated by the Registry, will see samples of queries to the domain[2]. Should the Registrant desire to change the ANS for the 2LD, this will be known by the Registry, who implements this, by updating the TLD zone. Internet Service Providers (ISPs), and others with special access to Internet traffic, can observe usage of the plain-text DNS protocol, e.g. to establish a passive DNS database. Anyone can query the Registry's `whois` information, for information about a specific domain. The type of information in `whois` can vary with TLDs. For gTLDs, like `.com`, and `.net`, the Registries are also obliged to provide daily updated copies of their zone files upon request.

With some reluctance, we further divide Post-registration into a **Pre-abuse** period, and a subsequent **Post-abuse** period. The separation between the two is the point in time where the given domain is involved in some abuse or malicious activity. Hence, Pre-abuse information only reflects benign usage, where Post-abuse also reflects malicious usage. On a conceptual level, it is difficult to unambiguously determine if a certain domain resolution is malicious or benign. On the technical level, the distributed nature of DNS, and caching, also blurs the lines. As an example, a Recursive Name Server (RNS) will cache the answer to an assumed benign request, and use that much later, when answering a subsequent, assumed malicious, requests. It is unclear if the exchange between RNS and ANS belongs to Pre- or post-abuse. We maintain the distinction here, as much related work claims to be able detect

[1] Expired registrations of the same domain that have been removed from the zone, are considered separately.

[2] Due to caching at lower levels in the DNS server hierarchy, only some requests will reach the TLD ANS. In general, the TLD ANS will get to know the Fully Qualified Domain Name (FQDN) requested, and the IP address of the Recursive Name Server (RNS). An exception to this is when an RNS implements Query Minimisation, by removing FQDN, and only including the necessary 2LD [66].

Pre-abuse, while relying on the domain being actively used [62, 64, 67–72].

Based on the preceding description of DNS usage, we see only one approach that can guarantee the prevention of abuse, given adequate detection accuracy: Pre-registration detection of malicious domains, followed by effective reactions, such as rejecting applications to register domains.

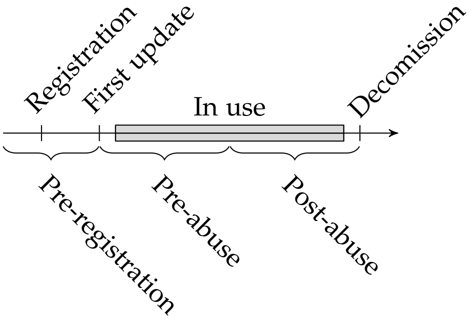


Fig. 6.2: Timeline for a domain, with different time-frames for detection.

With an understanding of the domain life-cycle in place, we now briefly describe different forms of DNS abuse. As mentioned in Section 1, DNS is useful for cyber criminals to operate **malicious infrastructure**, providing simpler host management, resilience, and some obscurity. **Fast flux** is a variation of this, where low Time-To-Live (TTL) values for **A records** enables rapidly changing, or agile, domain-to-IP mappings[3]. This obscures investigation and provides resilience to host take-downs. **Double flux** is another variation, where the same technique is also applied to the **NS records**, providing another layer of resilience, and obscurity[4]. **Domain flux** is a related technique, where malware with Domain Generating Algorithms (DGAs) generate many candidate domains for CnC infrastructure, of which only one, or a few, are used, and only for a short time. This provides resilience to blacklisting, and to Registries reacting to abuse. Among the user-centric forms of abuse, **phishing** has been described in the introduction, where domains are registered to mimic benign domains, and trick users. **Typosquatting** is similar, as abused domains resemble popular domains, but differ slightly, in order to land users that make typos. **Scam web shops** are operated to defraud victim shoppers, or to sell contraband goods, and constitute domain abuse as well. Such operations often rely on phishing and typosquatting as a means to land victims. Much of the user-centric abuse also relies on

[3] An A record maps a domain name to an IP address. A records are used to resolve domain names, among other things.

[4] An NS record maps a subdomain to the domain name of an ANS, thereby delegating authority of that subdomain to the ANS.

a domain for sending **spam** e-mail, which is also abuse. When registering domains for e.g. spamming, or scam web shops, criminals can attempt to "inherit" a good reputation through **re-registration** of an expired, previously benign domain. If the malicious re-registration is immediate after expiry, it is referred to as **drop-catch**, while **retread** registrations happens after some time. Overall, we refer to domains that serve to enable any abuse, such as the above, as malicious.

Having provided some background on DNS, and on malicious activity utilising DNS, we now move on to describe and analyse existing work.

3 Related work

In this section we describe the results of studying existing work. Understanding existing approaches to the problem is essential for the subsequent analysis, seeking to identify what can be done to further the state-of-the-art. To this end we have summarised the existing work in Table 6.1, for a high-level overview, with a more detailed description and analysis in the following section.

System	Paper	Year	Pre-reg.	Pre-abuse	Post-abuse	ML	Data	Detection object
Notos	[67]	2010		[a]	x	x	ISP RNS, honeypots, blacklists.	Malware-related domains.
-	[64]	2010		x	x	x	Registry, Registrar, WHOIS.	Malicious domains
Kopis	[62]	2011		[a]	x	x	TLD ANS, TLD ANS, IP reputation.	Malware-related domains.
-	[73]	2011			[b]		TLD ANS, WHOIS, blacklist, mail server.	None. Analysis of domains in Spam URLs.
BotGAD	[68]	2012		[a]	x		RNS, blacklists.	Botnet related domains.
FluxBuster	[74]	2012			x	x	RNS.	Flux domains.
-	[65]	2013	[b]				Registrar, Registry.	None. Analysis of spam domain registration.
EXPOSURE	[69]	2014		x	x	x	RNS, reverse DNS, Google.	Fast flux domains.
Phoenix	[70]	2014		x	x	x	RNS.	DGA domains among malicious domains.
-	[71]	2015		x	x	x	Client DNS traffic at ISP.	Classification: Look-up failure patterns for domains. Detection of infected clients.
DFBotKiller	[72]	2015		x	x		Client DNS traffic at ISP.	Clients infected with DGA malware.
PREDATOR	[75]	2016	x			x	Registry.	Spam domains.
-	[76]	2017	[b]			x	Registrant, Registrar, Registry.	Clustering: Campaigns of malicious domains registrations.

Table 6.1: Overview of existing work on detection of malicious domains. Entries are categorised by point in time where they can be applied; Before the domain is added to the TLD zone (Pre-registration, **Pre-reg.**), before the domain is actually abused/used for any attack (**Pre-abuse**), or after abuse/attack has occurred (**Post-abuse**). It is indicated if Machine Learning (ML) is used. The information used for detection, which can be a subset of that used for training, is summarised under **Data**. **Detection object** describes exactly what the proposed method is seeking to detect. Note that some entries are proposals for classification methods or studies without explicit method proposals.

[a] Not strictly Pre-abuse. Method can potentially detect malicious domains before abuse occurs but considers available Post-abuse data in detection.

[b] Not detection. Analysis of data that potentially can enable detection.

Antonakakis et al. have proposed the Notos system to dynamically assign reputation scores to domains [67]. The system is focused on agile, malicious domains, such as those used for domain flux. Using RNS traffic, aggregated per domain, *network-based features* are calculated from known historical IP-to-domain mappings, lexicology is applied to calculate *zone-based features*, and *evidence-based features* are obtained from malware caught in honeypots. Notos appears complex, as it includes a network of traffic data collection points, a passive DNS database, and honeypot analysis of malware samples.

Antonakakis et al. have also proposed Kopis for detecting malware related domains, by passively sniffing DNS traffic at TLD and 2LD ANS [62]. Analysing the data in windows of time, *requester diversity* describes the geographic distribution of those trying to resolve the domain, *requester profile* describe if the requester is an RNS, or single user, and *resolved-IPs reputation* describes how the resolved IP relates to reputation of Autonomous Systems (ASs), Classless Inter-Domain Routing (CIDR) prefixes, and countries. Through supervised Machine Learning (ML), applied to the extracted features, Kopis is capable of recognising domains used by malware.

Felegyhazi et al. used information from a Registrar, and Registries, together with knowledge of already abused domains, to infer other newly registered domains that are likely to be abused [64]. The inference uses clustering analysis on *name server features*, and *registration features*, which is intended to capture double fluxing and bulk registration. Evaluation, using domains found in spam mail, shows that the proposed approach can flag domains earlier than the blacklist considered in the study. The method requires to observe some domain abuse before the likely-to-be-abused domains can be inferred, and at the same time the goal of the proposal is to lessen the window of opportunity for abuse, rather than to eliminate it. This implies that some abuse is tolerated.

Hao et al. also studied domains from Uniform Resource Locators (URLs) found in e-mail spam, received in a spam trap [73]. Information relating to registration is used, along with traffic from a TLD ANS, to understand registration behaviour and infrastructure used by spammers. It is found that roughly half of the malicious domains are registered less than a day before they are used in spamming campaigns. The authors appear to take the optimistic view that, as long as blacklists are not lagging behind malicious domain registration, by more than a day, then half of the domains are captured. A complimentary observation appears to be that half of the domains are abused within 24 hours of registration, meaning that there is limited time to detect, and react. Further exploration is required to conclude on this, but assuming that attackers can uphold their operation, with only a fraction of the registered domains slipping through, this situation is concerning. The study is extended in [65], also focused on spam, where it is recognised that there is a gain in being able to reject applications to register domains for abuse. The important findings include: 1) A few Registrars account for the majority of spam related domains registered. 2) Spam related registrations are over represented at times where Registrars see spikes in overall registrations. 3) Spam registrations tend to include previously expired domains, presumably to benefit from their good reputation. PREDATOR, a machine learning based detector for spam domains, is proposed in [75]. A highly relevant property of PREDATOR is that it applies at time-of-registration, which corresponds to Pre-registration. While the motivation suggests that malicious domains, in general, pose a problem, the features appear to be focused on spam, with evaluation only using spam domains. The applied machine learning methodology stands out, as it respects practical and temporal constraints. For instance, the methodology is realistic in that data used for training the detector is strictly older than that used for testing. As evident from Table 6.1, PREDATOR is the only Pre-registration detection method found in our study.

Vissers et al. have analysed registration campaigns, carried out by criminals, where large amounts of domains are registered over time, to be abused only briefly, before being discarded [76]. The existence of such campaigns emphasises that Post-registration detection is a losing strategy, while Pre-registration detection is a solution. This work is highly relevant because it is one of the few examples that rely only on Pre-registration information, and because it is the only example where Registrant features are considered. The stated purpose is to analyse registration campaigns, and to cluster registrations according to such campaigns, while no detection method is presented or discussed. Even so, it is demonstrated that Pre-registration information can be used to discriminate abusive domains by registration campaign, and this speaks to the feasibility of Pre-registration detection.

BotGAD detects domains used by botnets, based on *group activity*, i.e. correlated DNS behaviour, among infected bot machines [68]. Linear algebra is applied to identify correlation, and periodicity, for both clients, and domains. As techniques such as domain flux and DGA thwart the above method, the authors propose to add a complementary clustering step. This clustering is performed on information extracted from RNS traffic; *DNS lexicology features*, *DNS query features* and *DNS answer features*, where the last also includes reputation in the form of blacklist information. Finally, malicious clusters are detected using Sequential Probability Ratio Testing, a hypothesis testing method, applied to the information described above. BotGAD is shown to perform well on the problem of detecting botnet related domains.

FluxBuster is a method to detect malicious domains that make use of flux techniques [74]. This is very much like Notos [67], as they both detect flux/agile domains, and rely on RNS traffic, collected at multiple points. FluxBuster stands out by not relying on existing blacklists. The proposed procedure is similar to BotGAD [68], in that it consists of a (pre-)filtering step, followed by clustering, and a final step for detecting clusters that represent flux/agile domains.

Bilge et al. proposed, and deployed, EXPOSURE, to detect malicious domains [69]. The detection relies on RNS traffic, enriched with results retrieved from reverse DNS look-ups, and queries to the Google search engine, both for the domain and for the IP resulting from reverse DNS look-up. The work targets fast flux domains, through use of specific features capturing low TTL values in DNS responses, and domains with poor human-readability. As Google possibly has a well-performing detection method in place, to eliminate malicious domains in search results, there is a concern of how much detection power is inherited from Google. An interesting feature selection method, based on a genetic algorithm, is applied. Another interesting aspect is that EXPOSURE was used to provide a public blacklist service.

Phoenix, proposed by Schiavoni et al. [70], clusters known bad domains, using *linguistic features* and *IP-based features*. Centroids are selected to represent different DGA implementations and used to detect the same algorithms in DNS traffic. Phoenix is presented as a method to gain *Intelligence and Insights*, and not explicitly as a detection approach.

Luo et al. have analysed DNS traffic from clients, as seen in an ISP network, focusing on failing queries. A manual analysis of observed failures is used to derive four failure patterns; *Highly Random*, *Partially Random with Limited Character Set*, *Mutated String*, and *Substring*. Four corresponding similarity functions are presented, and each is used independently for hierarchical agglomerative clustering. Similarities used are: Log-likelihood of 2-grams, Jaccard Similarity of used character sets, Levenshtein distance, and a custom measure based on common substrings. Clustering is applied continuously over time, for each similarity function, with rules for pruning, merging, and raising alerts. The reported metrics include the ability to detect botnet victims but are found lacking details of how malicious domains are detected.

DFBotKiller inspects DNS traffic from clients, to establish a client reputation, indicating if the client is infected with DGA bot malware [72]. DFBotKiller is similar to BotGAD [68], as they both target group activity, i.e. correlated client behaviour. It is also similar to the work of Luo et al. [71], as they both consider failed queries for detecting DGA botnet activity. In addition to group activity, and query failures, three lexicology features are considered, and combined into a time signal for domain reputation: *Jensen-Shannon divergence* over N-grams, *Spearman's rank correlation coefficient* over N-grams, and the average *Levenshtein distance* among all domains in a group of domains.

While much work has already been done, there are some significant gaps in the area of detecting malicious domains. We highlight the following;

1. Some of the existing work relies on prior abuse and preceding analysis, to establish known bad domains [62, 67, 74]. This requires time to be spent on curating black lists and implies that some abuse is tolerated.

2. Much of the existing work relies on DNS traffic [62,64,67,71]. This also implies that some abuse is tolerated.

3. Most of the work only addresses a subset of abuse, such as spamming, phishing, or Fast Flux [62,67,68,70–74]. This leads to a need for complex suite of methods to cover all forms of abuse.

As will be substantiated in Section 4.1, Pre-registration appears very promising, thus the work of Hao et al. [75] is noted as the most prominent. However, it is limited to domains involved in spam, hence there are multiple scenarios of abuse that are not covered, when comparing to the different forms of abuse outlined in Section 2, or the taxonomy of Moura et al. [63]. Other studies, where Pre-registration data is analysed, although without proposing detection approaches, are found to support the feasibility of Pre-registration detection [65,76].

This concludes the study of related work, which leads to the following discussion on identified topics of relevance to future work within the field.

4 Discussion

In this section we discuss relevant topics identified in the previous sections, in order to extract the most important points. This includes considerations on when, in a domain's life-cycle, abuse can be detected, which entities can detect abuse, the features used for detection, and the aspects of real-world applications. Finally, we outline the future work.

4.1 Time of detection

Three non-overlapping time-frames for detection were established in Section 2, and illustrated with Figure 6.2; Pre-registration, Pre-abuse, and Post-abuse. As evident from Table 6.1, only one proposal for Pre-registration detection was found during our study, and it is focused only on spam domains [75]. This is surprising, as Pre-registration detection is a powerful preventative tool [63, 65, 75]. Furthermore, Pre-registration detection can be used to fully prevent abuse of detected domains. An argument against Pre-registration is the limited information available, compared to Pre- and post-abuse, but this can be overcome [75,76]. Another challenge for the Pre-registration detection is the ethical problem of addressing perpetration, that has not yet been committed – especially in presence of uncertainty and errors. This challenge is not unique to detection of malicious domains and can for instance be addressed by adjusting the actions taken.

The alternatives to Pre-registration, are Pre- and Post-abuse. As argued in Section 2, it can be difficult to separate the two, thus it makes sense to consider them together as Post-registration. Post-registration detection implies that some usage is tolerated, and it is hard to argue that abuse can be prevented. However small a window of opportunity the cyber criminals are given, they seem to be capable of exploiting it, suggesting that Pre-registration has superior potential, over Post-registration. It remains a highly relevant question if the limited information available at Pre-registration is sufficient for adequate detection performance.

4.2 Features

All the studied proposals for detection can be modelled as an algorithm applied to data, which consists of features. One difference among the studied proposals is the set of features used, which is believed to significantly affect detection performance. Consequently, feature selection and feature engineering can have significant impact on detection performance.

It is important to note that access to information on certain features is not a matter of course. As described in Section 2, different entities have access to different information. Sources of information include: The Registrar, the Registry, the zone file, the ANSs from various levels, the ISP network, and clients. For a proposal to be relevant in practice, the used features must naturally be available. Relying on multiple sources adds undesired complexity, which must be weighed against the gains of having more information available.

When considering the registration process, the Registrant is a crucial actor. Surprisingly, none of the studied work, that proposes methods for detecting malicious domains, considers features describing the Registrant, despite such information being available to the Registrar, and in some cases also to the Registry. Reasons for this can be, that legal conditions constrain the Registrars from sharing this information, or that the redundancy of Registrars impedes the motivation to detect malicious domains at this point. Payment information is highly sensitive, and cannot leave the Registrar, but in the case of ccTLDs, and new generic TLDs, the contact details of the Registrant must be passed to the Registry, which is to keep it on record, and provide it through WHOIS look-ups. We see a great, unexplored potential for Registrars or Registries to detect malicious domains by novel Registrant-based features.

4.3 Feature selection and engineering

Existing work proposes many interesting features, and we have identified additional ones describing the Registrant. A high number of features leads to a need for feature selection, as excess features cause additional costs for compute, and storage, and increases complexity. Some ML algorithms also suffer a performance loss, when the input holds irrelevant, correlated, or noisy features. Bilge et al. proposes a genetic algorithm to automatically select well-performing feature sets [69]. Other possibilities include Principal Component Analysis (PCA), which can help to identify the most, and least, significant features. Empirical Distribution Function (EDF) plots and histograms can be used to analyse how individual features can contribute to detection. There is a need to explore feature selection for Pre-registration detection of malicious domains.

In order to exploit knowledge about certain features, or to normalise features, matching them to an ML algorithm, it is common to apply feature engineering. The work of Hao et al. is an interesting example of this, where the work preceding PREDATOR is used to establish a model for registration patterns, allowing them to compute a probabilistic feature, describing if a registration is part of a burst [65, 73, 75]. As feature engineering is about transforming the available data into more relevant features, more generic examples include all uses of N-grams, distance measures, and statistics aggregated over the available data. Feature engineering is believed to be important for achieving high detection performance.

4.4 Diversity of conditions

Existing work that involves specific TLDs are largely focused on the `.com` and `.net` gTLDs, which are both operated by the same Registry. This can be due to the general popularity of said TLDs, because the Registry is forthcoming in providing information from the zone files, or for some other reasons. However, only considering (Pre-registration) detection for a narrow set of Registries and TLDs might fail to highlight relevant aspects, as conditions for (ab)using domains vary. One seemingly important difference lies in the choice of Registrars. Some Registrars might offer pricing models that attract, or deter, certain types of abuse, and some can be more efficient in their response to abuse, than others. Another interesting aspect is the difference in what measures are already implemented to counter abuse, both at Registries and Registrars. It seems likely that cyber criminals will prefer the Registrars and Registries that provide some anonymity, e.g. allowing for proxy contacts in WHOIS, over those that require extensive validation, and provide full contact details. Differences among Registries can cause different compositions of abuse. Consequently, it is highly relevant to explore how detection can be achieved for a diverse set of Registries and TLDs.

4.5 Real world application

As already discussed, access to information is essential for practical application. Another very important aspect is temporal constraints, which must be respected if a method, and the evaluation thereof, is to be of practical relevance. A naive approach to evaluating performance is to batch all data, and cut it for test and training, and for cross-validation, completely at random. This is not realistic, as the model used for detection is trained on information, which has not been observed at the time of detection. Consider, as an example, the case where all malicious domain names that are registered as part of a campaign are configured to point to some ANS, and where each ANS is only used exactly once within a time window of 10 days. A method that captures this pattern can probably detect some malicious domains in a naive evaluation, but it will fail completely for the first 10 days of a campaign, when applied in reality. Consequently, the performance observed in such naive evaluations cannot be expected to reflect reality. Hao et al. evaluate their proposal in a more suitable manner, also including a time window for training the model [75].

Considering the benefits of Pre-registration detection, data accessibility, temporal constraints, and the ability to effectively prevent abuse, it appears that Registries are very suitable vantage points for combating domain abuse. We hence see a great potential in developing a method for Pre-registration detection of malicious domains, and in evaluating it at Registries.

Considering practical operational use of a detection method, we highlight some shortcomings with well-known statistical performance metrics, (accuracy, precision, and recall). Accuracy describes overall performance well but fails in the presence of class-skew data, where high error rates for small classes will be hidden. Precision and recall provide good insights to performance in a theoretical context, or when false detection can be tolerated, or caught by manually vetting all positive detection. However, they fall short in a practical context, if fully automated reaction to false detection is intolerable, and where manual capacity inhibits vetting all positive detection outcomes. In such a context, it is more prudent to consider an organisations capacity for vetting and processing positive detection outcomes, which is referred to as the **Daily investigative budget**. We propose a metric that counts the number of true positive detection outcomes within the daily investigative budget. This will describe the value of adopting a method better than precision and recall, as it incorporates the impact of need for manual vetting, and a limited capacity for doing so. If the Daily investigative budget is known, this metric estimates the actual effect. Alternatively, return of investment sweet spots can be identified by varying the Daily investigative budget. In this setting it is highly relevant to rank domains, e.g. according to likelihood of being malicious, rather than provide a simple, binary detection outcome.

4.6 Data and ground truth

Another aspect relating to the real-world application is how ground truth is obtained. In security, when using real data, it is generally hard to obtain the ground truth, as cyber criminals desire to avoid detection. When using real data, a common practice is to rely on blacklists, other available detectors, and sometimes on some degree of manual investigation, to correctly identify positive samples. The Alexa Top-N (`https://www.alexa.com/`) of popular domains have been used as a whitelist, but leaving many domains unlabelled, or grey-listed. These methods are practical but have some noteworthy biases: Blacklists and other detectors can be based on similar principles, with similar shortcomings, as the method under evaluation. They can also be really good at detecting the simpler and more obvious cases, because this is easier. Manual investigation is error prone, and subject to a human factor. Whitelisting Alexa Top-N domains is very biased towards popular domains, as it fails to whitelist all benign domains not among the N most popular.

In addition to involvement in registration and ordinary use, Registries are also responsible for decommissioning domains, and for handling abuse complaints. When processing abuse complaints for domains, Registries invest manual effort in establishing if the domain is malicious. The outcome of this is highly relevant as ground truth, and we are surprised to not find it being used in any of the studied work. Such data will reflect cases that are relevant to Registries, but there are still potential biases. For instance, it is quite likely that certain forms of abuse are more likely to be reported than others. The number of reported domains involved with fraudulent web shops might be driven up by consumer and merchant interests. On the other hand, it can appear futile to report DGA domains, leading to suspected domains not being reported. It appears that this bias will be towards the cases that are most relevant to the daily operations of a Registry. We do not see this as problematic, but rather a benefit for the Registry combating abuse.

Considering abuse cases, and the fact that the Registry is required to investigate them in their own right, highlights the potential of applying **active learning**; The idea of active learning is to, not only retrain the detection system repeatedly, but to also include information about past errors as feedback. This fits well with the notion that positive detection is to be investigated further manually, before the Registry removes a domain. Active learning has proven to provide significant gains, as the detector improves over time [51]. As criminals have great freedom in selecting Registrars and TLDs, active learning might be particularly relevant.

4.7 Future work

When comparing the problem with the state-of-the art discussed in this paper, it is clear that there is a gap between the abuse of domains today, and the current capabilities to prevent, and detect this abuse.

We found it especially interesting to observe how little attention has been paid to Pre-registration detection: The limited work carried out so far has focused on spam domain detection, and while results are promising, spam is by no means the only form of domain abuse. Thus, it is interesting to explore whether other types of malicious domains could be detected Pre-registration.

The information used for Pre-registration detection of spam in existing work is sufficient to solve the concrete problem, but there is also Pre-registration information which has not been considered. This represents untapped potential, so when exploring Pre-registration detection of malicious domains in general, it is thus relevant to consider all of the following information:

1. Lexicology analysis of domain name.
2. Registration history of domain name.
3. Registrant information.
4. Contents of first zone update.
5. Reputation of Registrar, and any other entities available in the above.

The Registrant information is particularly interesting, as it has not been used for detection previously, neither for Pre- or Post-registration detection.

In the existing work on Pre-registration detection, a single Registry and two TLDs have been considered. As Registries and TLDs are different, also from the perspective of cyber criminals, it is relevant to explore the potential of Pre-registration detection with other Registries and TLDs.

Pre-registration detection is especially interesting because it can be applied before domain registrations are completed. This can be used to reject application for registration of domains, and thereby avoid all abuse of the detected malicious domains. We find it highly relevant to explore detection in an applied context, at a Registry, as domain registrations can be rejected here. Additionally, the Registry has a unique access to data, as well as ground truth, in the form of abuse cases. Furthermore, a deployment at a Registry can yield perfectly realistic conditions and allow for assessing the actual benefit and value gained.

5 Conclusion

In this paper we have analysed the state-of-the-art for detecting malicious domains and found that it largely relies on domain names being in use, and resolvable on the Internet. This provides cyber criminals with a window of opportunity, spanning from registration completes, and until detection/blacklisting catches up. This implies that some abuse is tolerated within the window. Cyber criminals have adjusted their modus operandi accordingly, such that they are able to maintain their criminal operations, by only abusing a domain for a few days, or even hours. We identify Pre-registration detection as an efficient solution to this, where accurate detection can be used to reject registrations of malicious domains, guaranteeing that no abuse occurs.

Pre-registration differs from the state-of-the-art (Post-registration detection), on the following three important points: No abuse is tolerated, real-time requirements for reaching detection verdicts are relaxed, and only Pre-registration information is available.

Top-Level domain (TLD) Registries are in a unique position, as they have access to exclusive information, on which Pre-registration detection can be based. Furthermore, Registries control the TLDs, providing a single point where all applications for registrations can be scrutinised, and potentially rejected. Finally, Registries have historic data on abuse, with potential value for training, and evaluation – a potential that has not yet been explored.

Our future plan is to explore the possibilities of Pre-registration detection, for more abuse in general, using hitherto unused features, while also considering a real-world deployment.

Chapter 7

Assessing usefulness of blacklists without the ground truth

Egon Kidmose, Kristian Gausel, Søren Brandbyge, and Jens Myrup Pedersen

The paper has been published in the
Proceedings of The 10th International Conference on Image Processing & Communications (IPC), November 2018.

Abstract

Domain name blacklists are used to detect malicious activity on the Internet. Unfortunately, no set of blacklists is known to encompass all malicious domains, reflecting an ongoing struggle for defenders to keep up with attackers, who are often motivated by either criminal financial gain or strategic goals. The result is that practitioners struggle to assess the value of using blacklists, and researchers introduce errors when using blacklists as ground truth. We define the ground truth for blacklists to be the set of all currently malicious domains and explore the problem of assessing the accuracy and coverage. Where existing work depends on an oracle or some ground truth, this work describes how blacklists can be analysed without this dependency. Another common approach is to implicitly sample blacklists, where our analysis covers all entries found in the blacklists. To evaluate the proposed method 31 blacklists have been collected every hour for 56 days, containing a total of 1,006,266 unique blacklisted domain names. The results show that blacklists are very different when considering changes over time. We conclude that it is important to consider the aspect of time when assessing the usefulness of a blacklist.

Keywords

Domain names, blacklists, Domain Name System

1 Introduction

Domain names are crucial for a lot of Internet activity, including malicious activities of cyber criminals. Examples of such use include criminals sending SPAM and phishing e-mails, where domains are used for linking to malicious resources, or it can be observed when malware successfully infects a victim machine and needs to establish a Command and Control (CnC) channel with the attacker. This is often achieved through the Domain Name System (DNS).

If malicious domains are known, queries to resolve them through DNS provides for simple detection. Blocking the responses can prevent or inhibit malicious activity. Consequently, practitioners employ domain name blacklists to defend systems. Blacklists also finds use in research on detecting malicious domain names, and malicious activity in general. While the research area is very much active and goes beyond using blacklists for detection, blacklists are still used as a ground truth and for validation [77].

Unfortunately, the attackers hold the initiative and choose which domain they register and which they abuse, making it hard to determine if a domain is malicious or not, and even harder to identify all malicious domains. This leads to a problem of lacking coverage, i.e. the fact that the known malicious domains only cover a subset of the ground truth of all malicious domains [78].

In addition to this, criminals have adopted their practices to exploit any time lag until a domain is blacklisted, making timeliness a challenge with further negative impact on accuracy [65] [76]. In summary, the accuracy of blacklists is expected to be impeded by insufficient timeliness and coverage, while the ground truth is not available for measuring the accuracy directly.

This paper contributes by presenting a method for assessing the coverage and timeliness of blacklists. An important, novel point is that this is done without assuming any ground truth. Instead, blacklists are analysed according to a set of proposed metrics, that covers all entries of the lists. Ultimately, this paper improves our understanding of benefits and shortcomings of blacklists, considering complete blacklists and without relying on a ground truth.

2 Related work

Sinha et al. have studied four IP blacklists and their spam detection accuracy. The SpamAssassin spam detector is validated against manually curated spam and non-spam e-mails and used as an oracle. Results include non-negligible false negative rates (never less than 35%) and false positive rates (as high as 11%). These errors can either be due to lacking blacklist accuracy or due to errors by the oracle. Our work differs by having domain blacklists as subjects and by eliminating the risk of errors by a non-perfect oracle.

A study of blacklists accuracy and timeliness has been conducted by Sheng et al., addressing phishing URLs [78]. 191 newly received phishing URLs are submitted to eight different web browsers/browser plugins, mainly relying on URL blacklisting. Results show that when the phishes are first received, coverage is typically 10% and after two hours most blacklists still cover less than 50%. We expand these insights by observing the full blacklists, doing so for 56 days, thereby processing 1,006,266 unique malicious domains.

There is ample of work on detecting malicious domains that relies on blacklists, either as a data source or as a ground truth, of which we now describe a select few. Antonakakis et al. presents Notos, a system for detecting malicious domains, based on DNS traffic. It relies on three blacklists for ground truth [67]. The nDews system has a similar scope to Notos but identify suspicious domains [79], relying on three blacklists to qualify suspicion in a second phase. FluxBuster is a proposal for detecting malicious domains that exhibit fluxing – a technique applied by criminals to improve resilience of their CnC infrastructure [74]. It is found that malicious domains can be detected days or weeks before appearing in any of 13 blacklists, highlighting the problem of timeliness. BotGAD is a system for detecting CnC domains based on DNS traffic combined with three blacklists [68]. The state of the art for detecting malicious domains is perhaps best captured by PREDATOR, which aims to detect SPAM domains before they are registered, and therefore before they can be abused in any way [75]. The authors behind PREDATOR state that there is a lack of ground truth on malicious domains but find that blacklists are the best available solution for evaluation. Two blacklists make up the ground truth used for evaluation.

Studies on domain abuse also rely on blacklists. Felegyhazi et al. use a blacklist as seed for inferring new malicious domains [64]. Again, the issues of timeliness and lacking coverage are identified. Vissers et al. analyse the modus operandi of cyber criminals, in particular how they register domain names, using blacklists to validate maliciousness [76]. They too confirm challenges regarding timeliness and find that 18.23% of the malicious domains they identify are never captured by the three blacklists in use.

Overall, we find that blacklists are used both for operational purposes and in research, making the problems relating to coverage and accuracy highly relevant. The existing work on assessing blacklists either rely on an oracle being available to provide the ground truth or rely on a source that samples the ground truth for a subset.

In this work we seek to improve the state of the art on two important points: 1) We present a method to analyse entire blacklists, rather than assuming a limited view based on sampling. This provides insights that would otherwise be impossible and eliminates any bias due to sampling. 2) The method does not rely on a ground truth or an oracle to be available. Consequently, errors made by non-perfect ground truth/oracles are eliminated, and so is any need for manually vetting ground truth/oracles.

3 Methods

This section holds a description of the data collection procedure, followed by a description of the proposed metrics.

3.1 Data collection

A common practice in prior work is to analyse blacklists by sampling them according to some feed of positive samples (e.g. domains found in received spam). The motivation for this choice is unknown but could be due to a preference for Block Lists served through DNS (DNSBL) or other *query-only blacklists*, which has some operational benefits. Drawbacks of using query-only blacklists include that timeliness only can be explored for the time after a domain is produced by the feed of positive samples, and that any bias from the feed will propagate to how the blacklists are sampled. In order to eliminate the sampled view of blacklists, and instead gain full insights into all the blacklisted domains, we focus our study on *retrievable blacklists*. This allows us to analyse all blacklisted domains and not only those hit by sampling. It will also allow us to establish when a domain is blacklisted.

Some blacklist feeds are offered as paid services, likely to cover for the effort of curating blacklists and justified by the value of accurate detection. At the same time, there are communities that have an interest in blacklists, and possibly for that reason there are also free and public community curated blacklists. This work is based on freely available blacklists from both companies and communities, subject to agreeable usage terms.

With our focus being on domain names, we consider domain and URL blacklists, where domains can be extracted from the URLs. Blacklists can be targeted at certain types of maliciousness, but we see no reason to exclude certain types.

Based on the above criteria, a manual search of the Internet has been conducted, with an outset in the blacklists found in related work. The result is a set of 31 retrievable blacklists, targeting either malicious domain names or URLs. The blacklists are retrieved hourly with a system based on MineMeld [80]. Table 7.1 holds the URLs for each blacklist.

BID	Data feed URL	BID	Data feed URL
1	<RWT>/CW_C2_DOMBL.txt	17	<HF>/grm.txt
2	<RWT>/CW_C2_URLBL.txt	18	<HF>/hfs.txt
3	<RWT>/LY_C2_DOMBL.txt	19	<HF>/hjk.txt
4	<RWT>/LY_DS_URLBL.txt	20	<HF>/mmt.txt
5	<RWT>/LY_PS_DOMBL.txt	21	<HF>/pha.txt
6	<RWT>/TC_C2_DOMBL.txt	22	<HF>/psh.txt
7	<RWT>/TC_C2_URLBL.txt	23	<HF>/pup.txt
8	<RWT>/TC_DS_URLBL.txt	24	<HF>/wrz.txt
9	<RWT>/TC_PS_DOMBL.txt	25	http://malc0de.com/bl/ZONES
10	<RWT>/TL_C2_DOMBL.txt	26	<MDL>
11	<RWT>/TL_PS_DOMBL.txt	27	<MD>/db.blacklist.zip
12	<ZEUS>	28	<MD>/immortal_domains.zip
13	<HF>/ad_servers.txt	29	<MD>/justdomains.zip
14	<HF>/emd.txt	30	<MD>/malwaredomains.zones.zip
15	<HF>/exp.txt	31	<MD>/spywaredomains.zones.zip
16	<HF>/fsa.txt		

Table 7.1: Overview of blacklists included in the analysis. BID: Blacklist Identifier. Abbreviations used in in URLs: |https://ransomwaretracker.abuse.ch/downloads| (|<RWT>|), |https://zeustracker.abuse.ch/blocklist.php?download=baddomains| (|<ZEUS>|), |https://hosts-file.net| (|<HF>|), |http://malware-domains.com/files| (|<MD>|) and |http://www.malwaredomainlist.com/mdlcsv.php| (|<MDL>|).

3.2 Metrics

With a set of 31 blacklists, containing a total of 1,006,266 unique domains, it is evident that some statistics or metrics are required to gain insights. The following is our proposals for metrics to use for such analysis.

As a component to coverage, we consider the **size** as the number of domains found in a blacklist. With malicious domains appearing rapidly, entries must be added to maintain coverage, thus we identify **appearances**. To avoid false positive for once-bad-now-benign domains, domains must be removed so we count **removals**. Finally, removing a malicious domain by mistake is problematic, so we look for **reappearances**, where domains are removed and reappear on the same blacklist, although this also can occur for other reasons.

4 Results

Data collection started on January 25th, 2018, and up to March 22th (56 days) all blacklists found in Table 7.1 were retrieve every hour. A total of 1,006,266 malicious domains was recorded. During the first day, across all lists, 731,118 domains appeared, while the number was 208,413 for the second day. This makes them the two days with the most appearances, while the 21st day of our collection was the third busiest day, with only 22,267. We assume this to be an effect of a backlog for ingesting already blacklisted domains into our data collection system and define our epoch to start with the third day, March 27th. Figure 7.1 presents the appearances after the epoch.

A breakdown of size, appearances, removals, and reappearances per blacklist is shown in Table 7.2. Only 13 reappearances occurred, and they all occurred for BID 13 on April 14th. Out of the 31 blacklists, 19 saw no changes.

Domain appearances/removals per list are shown in Figure 7.2 and Figure 7.3.

5 Discussion

This work is, to the best of our knowledge, the most comprehensive study of domain blacklists in terms of number of blacklists and the duration of the observation. It has indeed provided relevant insights into blacklists and their usefulness. In the following we will highlight the most relevant insights.

One important observation is that 19 of 31 blacklists did not change during the 56 days. These 19 blacklists account for only 43,451 of all the 1,006,266 unique domains, meaning the static lists are small in size. Some of the largest static list (BID 15 with 12,306 domains, BID 20 with 5,521 and BID 24 with 3,638) can be summarised as domains where malware and exploits are peddled, or where misleading marketing practices are applied. As such threats

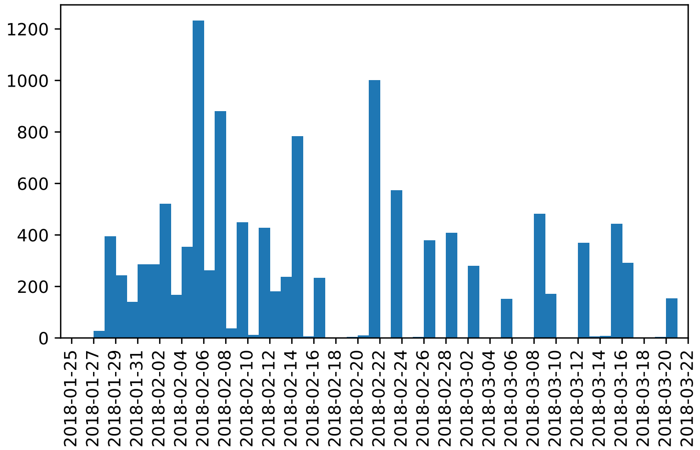


Fig. 7.1: Number of new domains appearing in a blacklist per day.

BID	Size	App.	Rem.	Ch.
31	4528	4528	4528	9056
23	26122	2675	2674	5349
14	226309	1544	1539	3083
29	22920	1165	1157	2322
22	268991	837	836	1673
16	345246	560	558	1118
21	40501	285	285	570
30	284	284	284	568
25	175	8	8	16
1	130	2	2	4
13	49620	2	2	4
19	189	1	1	2

Table 7.2: Size, appearances (App.) and removals (Rem.) by blacklist. Changes (Ch.) is the sum of App., Rem., and Reappearances. Sorted by number of changes. 19 blacklists with no changes are omitted.

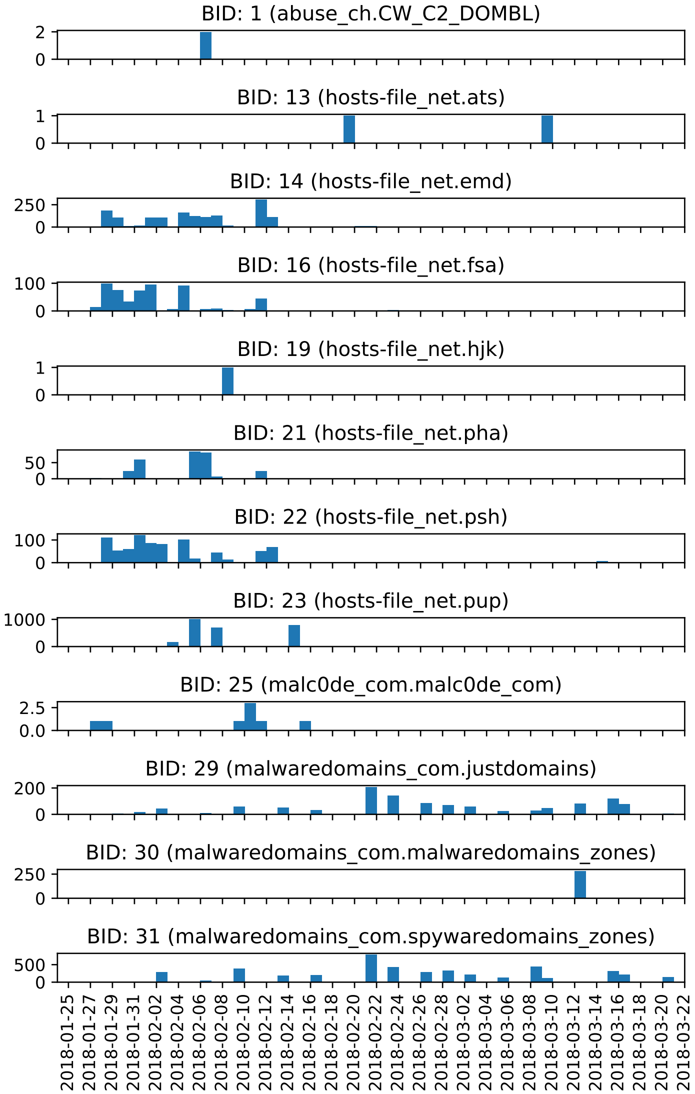


Fig. 7.2: Appearances per list over time.

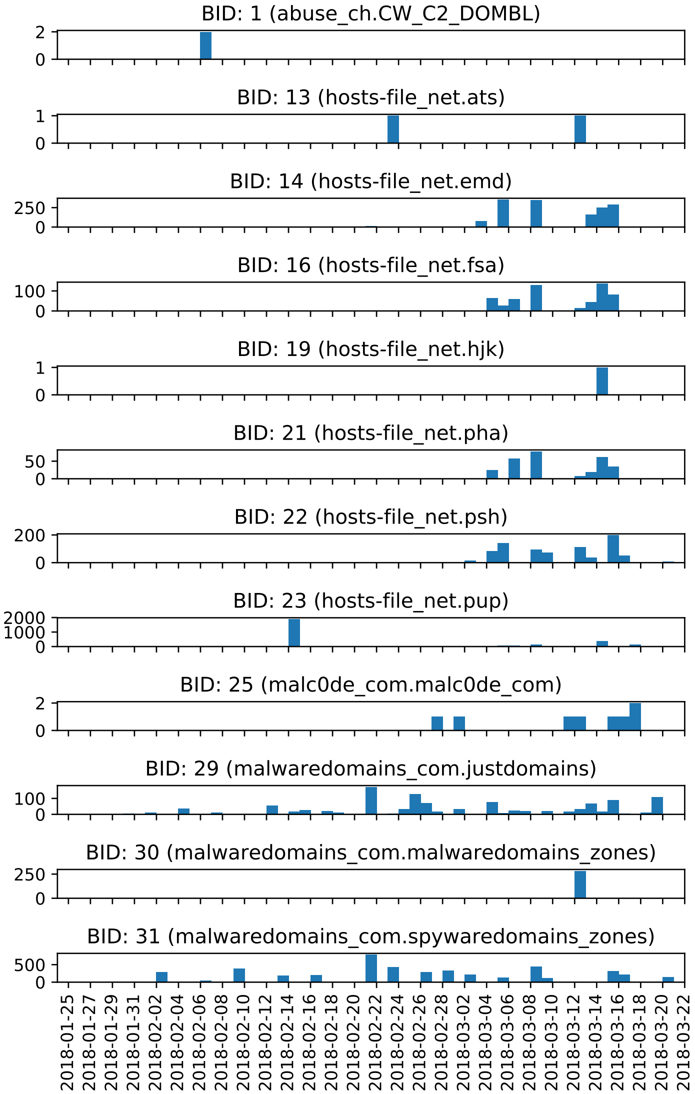


Fig. 7.3: Removals per list over time.

are harder to prosecute than direct attacks, it is to be expected that they only change infrequently, and these lists could very well be useful. As for the multiple static malware tracking lists, it seems more questionable if they are useful countermeasures, keeping in mind that criminals have high agility when it comes to using domains for CnC infrastructure. For the static lists targeting specific malware families, an explanation for the lack of changes can be that the criminals have moved on to use other malware.

From Table 7.2 we see that the number of appearances and removals are approximately equal for each list, hence all the list are of approximately constant size. It can also be seen that the numbers of appearances and removals equals the size of the blacklist for BIDs 30 and 31, meaning that the lists potentially have changed completely during the observation period. This is to be compared with BIDs 13 and 16, where only 0.004% and 0.162% of the lists changed. An explanation for this could be that BIDs 13 and 16 addresses domains involved with fraud, for which take-downs actions might be slower.

Figures 7.2 and 7.3 show traits that are apparent when maintaining the time dimension. Take for instance BID 13, where both appearances are followed by a removal a few days later, suggesting a relatively quick response, both in addressing the underlying problem and from the list maintainer. For BIDs 14-22 the pattern is instead that we first see some weeks with only appearances, followed by weeks of only removals. Hence one can expect relatively high time lags. Blacklists 29 and 31 show a different pattern, with more evenly distributed changes, indicating more active maintenance. The different types of maliciousness addressed by different lists can also be an explanation.

As future work, we would like to expand our study, with a larger set of blacklists and with a longer period of observation. The large share of static entries might be explained by the lists not seeing sufficiently active maintenance, or by the combination of a limited time span and the nature of the malicious use. If it is the case that domains typically are abused, and therefore blacklisted, for e.g. months then expanding the observation to e.g. a year will improve the analysis. Expanding with more blacklists that target the same types of maliciousness can allow us to determine a typical rate of changes, providing for judging what lists appear to be remarkably above or below the norm. When considering the domains appearing after the epoch, there was too small an intersection between the lists for any meaningful analysis to be made. With a longer period of observation is possible that we can identify interesting relationships between blacklists, such as overlaps, aggregations and typical lag times. One way to expand the set of blacklists is to include query-only blacklists. This would of course raise the issue of how retrievable and query-only blacklists can be compared. Finally, it appears relevant to evaluate blacklists in practice and relate that to our metrics.

6 Conclusion

In this paper we describe how it is a problem to assess the usefulness of blacklists in the absence of a ground truth. We address the problem by proposing a method for analysing blacklists in order to understand how they change over time. We find large difference among blacklists. Based on this it is apparent that the size of a blacklist alone does not signify its usefulness. Observing blacklists for weeks provides a better picture of which potentially are up to date and therefore useful. Longer running observations and more blacklists are required to understand relationships between blacklists, and we see this as the future direction of this work.

Chapter 8

Detection of malicious domains through lexical analysis

Egon Kidmose, Matija Stevanovic, and Jens Myrup Pedersen

The paper has been published in the
Proceedings of The 2018 International Conference On Cyber Security And
Protection Of Digital Services (Cyber Security), June 2018.

Abstract

Malicious domains play an important role for many malicious operations: For example, botnets use them for avoiding hard-coded IP addresses when connecting to command-and-control servers, and they are heavily used by criminals when distributing spam and phishing e-mails. Being able to identify malicious domains and block the harmful traffic is therefore one of the keys to create a more secure cyber environment. In this paper we demonstrate how the lexical analysis of domain names can contribute to increasing precision and decreasing the number of false positives when combined with other basic domain features.

1 Introduction

The Domain Name System (DNS) plays an important role in the operation of some of the most important malware and cyber threats that exists today: For example, when a machine is infected to become part of a botnet it will need to contact the command-and-control infrastructure. Since hard-coded IP addresses have numerous drawbacks in terms of resilience, the IP address is resolved by looking up domain names which are often randomly or semi-randomly generated. Another example is phishing e-mails, where the victim is encouraged to click on a link leading to a website with malicious content. This website is usually hosted using some domain name. While it may not be randomly generated, our thesis is that the name will often have different characteristics than a benign domain name. Yet another example is that of malicious websites in general, which can be used for various malicious purposes such phishing or drive-by downloads of malware. Unless using hard-coded IP-addresses, visiting such websites also require the look-up of a domain name. Being able to detect malicious domains is therefore a cornerstone in fighting a large set of diverse cyber threats, since once the domains are detected it is easy to block traffic to/from these domains, or even use the detection of such traffic to identify machines compromised by malware. The approach is not new: Blacklisting of domains has been around for decades, and there exists free and commercially available blacklists with good reputations such as Spamhaus, ivmSIP and Barracuda. There are also commercially available services such as Secure DNS by CSIS [81], which indeed blocks traffic based on DNS requests containing domain names known to be malicious. Other researchers have also shown interest in early detection of malicious domains, e.g. [75], which aims at identifying abuse domains already at the time of registration. However, existing methods are largely based on history and intelligence, and it takes time from the malicious activity is started and until a domain is blacklisted/blocked. This is problematic as many of the domains involved in malicious activities are only used for a short duration of time (hours or days), before the activity moves on to new domains. The paper contributes to solving this problem by looking at domain features that do not require history or intelligence: In particular, we look at domain names and their lexical features, inspired by the work of e.g. [82] and [83] combined with basic domain features and demonstrate how the lexical features can be used to improve the precision of machine learning algorithms to detect malicious domains, while at the same time decreasing the false positive rates. In our work we provide a comprehensive overview of lexical properties of malicious domain names and we explore how malicious domain names can be identified using lexical analysis.

2 Methods

The foundation of the paper is to train machine learning algorithms to distinguish between malicious and benign domain names. To initiate the research, we need to go through the following steps:

- Obtain a data set with Ground Truth.
- Extract relevant features for each data point (in our case for each domain name).
- Perform experiments with selected machine learning algorithms: Train the algorithm on training data, and test on testing data.

2.1 Ground Truth data

Obtaining the ground truth is always challenging [84], especially in a case like this where the definition of malicious domains is not set in stone.

For this study, we selected a number of domains we consider to be malicious, as well as a number of domains we consider to be benign. In order to make sure we did not bias the data a particular effort was done to identify both malicious and non-malicious domains created by Domain Generation Algorithms (DGA). The malicious domains were collected using eight different publicly available domain name blacklists, combined with a number of domains captured by our own malware testing setup, which was used to test out 300.000 samples of malware each for a 2-minute duration, and monitor their domain look-ups [85] [86]. A complete overview of these domains is provided in Table 8.1.

For the benign domains, we combined the most popular domains from alexa.org (top one million) with DGA domains from `www.datadrivensecurity.info`, as listed in Table 8.2.

Data sets		Number of domains	
		Non-DGA	DGA
abuse.ch	ZeuS Tracker	437	-
	Palevo Tracker	14	-
	Ransomware Tracker	1248	-
malware-domains.com	Just domains	13073	-
	Zeus Gameover	-	190033
	Conficker	-	106101
	Pushdo	-	10951
	GOZ	-	7348
	Microsoft Botnet	22036	-
host-file.net	Ad/tracking servers (ATS)	47960	-
	Malware distribution (EMD)	137237	-
	Exploits sites (EXP)	17282	-
	Fraud sites (FSA)	134501	-
	Spamming sites (GRM)	674	-
	Spamming sites (HFS)	573	-
	Hijack sites (HJK)	74	-
	Misleading marketing (MMT)	5533	-
	Pharmacy activities (PHA)	23143	-
	Phishing sites (PSH)	133913	-
	Warez distribution (WRZ)	3231	-
datadrivensecurity .info	Cryptolocker	-	34319
	Goz	-	7347
	New Goz	-	10999
malwaredomainlist .com	Malware-related domains	1253	-
malc0de.com	Malware-related domains	208	-
malwarepatrol.net	Malicious URLs	35518	-
phistank.com	Phishing URLs	14807	-
AAU-STAR	Domains from malw.testing	27778	-

Table 8.1: Data sets of malicious domain names.

Data sets		Number of domains	
		Non-DGA	DGA
alexa.org	Most popular domains	971424	-
datadriven security.info	Legitimate DGA domains	-	133927

Table 8.2: Data sets of benign domain names.

2.2 Features

We divide the features into three sets, where the first is general and the two others are lexical features. The lexical features are again split into two sets, namely simple lexical features and advanced lexical features.

The following gives an overview of the features used:

Basic Domain Features

- Number of domain levels (n-LD).
- Top level domain (TLD).
- Length of Fully Qualified Domain Name (FQDN).

Simple Lexical Features

- Length of 2nd Level Domain (2-LD).
- Ratio of consonants in the 2-LD.
- Number of vowels in 2-LD.
- Number of numeric characters in 2-LD.
- Number of special characters in 2-LD.
- Ratio of special characters in 2-LD.

Advanced Lexical Features

- Language indicator
- Number of English words in 2-LD.
- Entropy of 2-LD.
- N-gram analysis of 2-LD (www.alexa.org).
- N-gram analysis of 2-LD (English dictionary).

2.3 Machine Learning Algorithms

We chose to use supervised machine learning algorithms, more specifically the Random Forest Classifier. The evaluation is performed using the 10-fold cross validation scheme: Data is partitioned into 10 folds, each in turn left out for testing in 10 separate executions, with the remaining nine tenths used for training. Each fold is repeated 10 times. All methods are implemented in Python.

Moreover, three different scenarios are chosen, where each scenario represents a subset of domains from the data set listed above. This is done in order to obtain a better understanding of how DGA-domains affect the results.

- In Scenario I, we use the full data-set, i.e. all the malicious and non-malicious domains.
- In Scenario II, we study only non-DGA domains, implying that we omit both malicious and non-malicious DGA domains. As a consequence, the only non-malicious domains are then the Alexa-top-1M.
- In Scenario III we study only malicious DGA domains. For the benign domains both Alexa-top-1M and benign DGA domains are used.

3 Results

In the following, the results are presented for each of the three scenarios. To evaluate the features, we use the following commonly used measures:

- True Positive Rate (TPR/recall)
- False Positive Rate (FPR)
- Precision
- F1-score

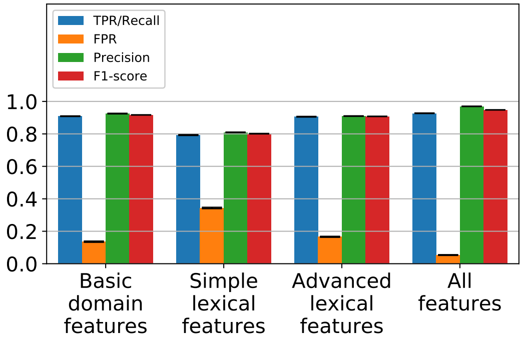


Fig. 8.1: Detection performance metrics for scenario I: Full data-set. Bars signify mean values across 10 cross validation folds, with non-overlapping, stratified batches of the data-set, and across 10 repetitions. Error bars indicate maximum and minimum values observer across 10 folds and 10 repetitions, (Note that the variance is low, hence the bars are very close the mean value).

Figure 8.1 shows the results of Scenario I, which includes all malicious and non-malicious domains. It is clear that even just using the basic domain features, we achieve relatively good results with a precision of 0.92. However, the FPR of 0.14 is quite high, which in practice would turn into many false alarms. Using the Simple lexical features alone does not give very good results, with the precision down to 0.81, and an FPR of 0.34. The Advanced lexical features alone improve the results and become comparable to using the Basic domain features, with a precision of 0.91 and an FPR of 0.17. However, the best results are achieved when combining all the lexical features and the basic domain features: It now gives a precision of 0.97, and an FPR of only 0.05.

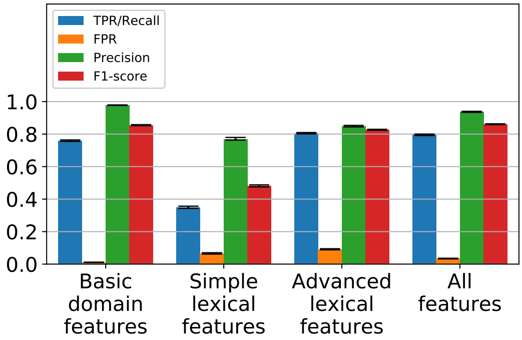


Fig. 8.2: Detection performance metrics for scenario II: Non-DGA. Including malicious non-DGA domains, and Alexa-top-1M. Excluding malicious and benign DGA domains. See Figure 8.1 for explanation.

Figure 8.2 shows the results of Scenario II, where both malicious and non-malicious DGA domains are left out. In this case the malicious domains seem much easier to detect accurately, and in fact the basic domain features alone produces very good results with a precision of 0.98 and an FPR of just 0.01. This is better than using simple lexical features alone (precision 0.77, FPR 0.07) or Advanced lexical features alone (precision 0.85, FPR 0.09). When all features are used this is actually worse than using Basic domain features alone, with a precision of 0.94 and an FPR of 0.03. However, TPR/recall and F1-scores are marginally improved (from 0.76 to 0.79 and from 0.85 to 0.86 respectively).

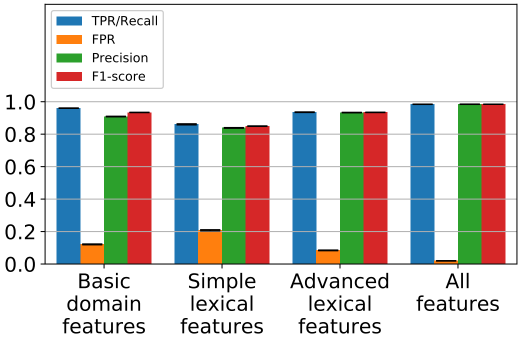


Fig. 8.3: Detection performance metrics for scenario III: DGA. Including malicious DGA domains, Alexa-top-1M, and benign DGA domains. Excluding malicious non-DGA domains. See Figure 8.1 for explanation.

The results of Scenario III, which include only the DGA-domains together with the benign domains from alexa.org are presented in Table 8.3. In this case, using the Basic domain features gives a precision of 0.91 and an FPR of 0.12. As was the case in the previous scenarios, using the Simple lexical features only does not give very good results: The Simple lexical features give a precision of 0.84 and an FPR of 0.21, while the Advanced lexical features give a precision of 0.93 and an FPR of 0.08 – actually better than the Basic domain features alone. However, combining all the features improves the results significantly with a precision of 0.98 and an FPR of 0.02.

A more detailed presentation of the results, including means, standard deviations, maximum and minimum are found in Tables 8.3-8.6.

4 Discussion

The results demonstrate that using lexical features can in many cases improve the detection performance of malicious domains compared to using only Basic domain features. However, the results are highly dependent on which data sets are used, in our case demonstrated in the different scenarios: Identification of DGA-based domain names shows the most promising results for now, whereas identification of non DGA-based domain names requires further work. It is also clear that while malicious domain names can be identified by performing purely lexical analysis of domain names (especially for DGA-based domain names), the most promising use seems to be in combination with Basic domain features.

One of the main challenges in studies like this is to find ground truth data, which is correctly classified. Moreover, for the results to be directly applicable in a real operating network, the traffic should also be representative of this particular network.

First of all, the data sets representing malicious data are all from well-respected sources which gives a good certainty that they are correctly classified. Also, the mix of data from different sources covering different kinds of malicious activities, provides us with a good and diverse coverage of many different kinds of malicious domains – however, it is impossible to guarantee that we have covered all possible patterns of malicious domains. With respect to the benign data, the Alexa-top-1M domains are all popular domains with many visitors, and as such should be benign. However, it cannot be ruled out that some of the domains have been abused for shorter or longer duration of time. This leads again back to the challenge that it is hard to classify domains as malicious and non-malicious without a clear definition combined with the methods to test each domain towards this. Whether the Alexa-top-1M domains are representative of benign domains is an open question – it could well be that the most popular domains have different characteristics in e.g. length and language that less popular but still benign domains.

Second, the results will always depend on which domains are present in a given data set, including the distribution of malicious/non-malicious domains, and as evident from our results also the distribution of DGA-based domain names and non DGA-based domain names. Therefore, while the results give a nice indication of the methods, they are not directly transferable to a production network.

Overall, we would say that while the data sets used in the study are suitable for a preliminary study of whether the inclusion of lexical features is feasible, a further study should make use of more substantial data sets, especially with respect to the benign data. It should also be considered to include more advanced lexical features that could encompass the "malicious nature" of the domains used by cybercriminals.

5 Conclusion

Detecting malicious domains is an important cornerstone in the battle against cybercrime, as the Domain Name System plays an important role in many malicious operations. These include botnets and their contact to command-and-control servers, phishing websites, and websites with malicious content and code.

This paper introduced a novel way of identifying malicious domains, which is not based on intelligence or historical information about the domains, but instead features associated with the domain name.

The feasibility of the approach was demonstrated by extracting the features from a large number of malicious and benign domains. Based on supervised machine learning using Random Forest Classifier,

The results we obtain demonstrate that the approach of using lexical features is promising especially for data sets including DGA-based domain names, and especially when combined with Basic Domain Features.

Future work will be based on both further refinement of the methods, with more advanced features, and a more sophisticated analysis using more substantial data sets in particular when it comes to representation of benign domains.

		Mean	Std. means	Max.	Min.
Scenario	Feature set				
I	1	9.25e-01	1.52e-04	9.27e-01	9.23e-01
	2	8.10e-01	2.75e-04	8.11e-01	8.08e-01
	3	9.10e-01	1.41e-04	9.11e-01	9.08e-01
	All	**9.70e-01**	2.03e-04	9.71e-01	9.68e-01
II	1	**9.78e-01**	1.88e-04	9.79e-01	9.76e-01
	2	7.69e-01	1.08e-03	7.79e-01	7.63e-01
	3	8.49e-01	5.33e-04	8.53e-01	8.44e-01
	All	9.37e-01	3.58e-04	9.40e-01	9.34e-01
III	1	9.09e-01	1.47e-04	9.10e-01	9.07e-01
	2	8.38e-01	2.64e-04	8.41e-01	8.36e-01
	3	9.33e-01	2.57e-04	9.35e-01	9.32e-01
	All	**9.84e-01**	1.24e-04	9.86e-01	9.83e-01
Over all	Over all	9.01e-01	1.08e-03	9.86e-01	7.63e-01

Table 8.3: Summary of precision across 10 cross validation folds. Standard deviation of means (Std. means) is the standard deviation of the means of each of 10 cross validation folds. Maximum (Max.) and minimum (Min.) are for all 10 repetitions of 10 folds. "over all", aggregates the table vertically, i.e. across scenarios and data sets. "Over all Std. means" is the maximum value found in the columns, i.e. worst case.

		Mean	Std. means	Max.	Min.
Scenario	Feature set				
I	1	9.09e-01	2.18e-04	9.11e-01	9.07e-01
	2	7.93e-01	3.22e-04	7.95e-01	7.90e-01
	3	9.06e-01	2.84e-04	9.08e-01	9.04e-01
	All	**9.27e-01**	1.71e-04	9.29e-01	9.25e-01
II	1	7.59e-01	4.56e-04	7.64e-01	7.55e-01
	2	3.50e-01	9.34e-04	3.56e-01	3.43e-01
	3	**8.05e-01**	5.48e-04	8.09e-01	8.01e-01
	All	7.95e-01	4.62e-04	8.00e-01	7.91e-01
III	1	9.60e-01	1.58e-04	9.62e-01	9.59e-01
	2	8.61e-01	5.20e-04	8.64e-01	8.57e-01
	3	9.35e-01	2.51e-04	9.37e-01	9.34e-01
	All	**9.84e-01**	6.79e-05	9.85e-01	9.83e-01
Over all	Over all	8.32e-01	9.34e-04	9.85e-01	3.43e-01

Table 8.4: Summary of TPR/Recall across cross validation runs. See Table 8.3 for explanation.

		Mean	Std. means	Max.	Min.
Scenario	Feature set				
I	1	1.36e-01	3.22e-04	1.40e-01	1.32e-01
	2	3.44e-01	6.12e-04	3.48e-01	3.39e-01
	3	1.66e-01	2.67e-04	1.69e-01	1.63e-01
	All	**5.35e-02**	3.74e-04	5.58e-02	5.12e-02
II	1	**1.10e-02**	9.36e-05	1.19e-02	1.03e-02
	2	6.69e-02	4.59e-04	7.00e-02	6.19e-02
	3	9.16e-02	3.87e-04	9.46e-02	8.81e-02
	All	3.41e-02	1.98e-04	3.61e-02	3.23e-02
III	1	1.21e-01	2.21e-04	1.24e-01	1.19e-01
	2	2.08e-01	4.99e-04	2.11e-01	2.04e-01
	3	8.40e-02	3.49e-04	8.62e-02	8.21e-02
	All	**1.98e-02**	1.58e-04	2.08e-02	1.81e-02
Over all	Over all	1.11e-01	6.12e-04	3.48e-01	1.03e-02

Table 8.5: Summary of FPR across cross validation runs. See Table 8.3 for explanation.

		Mean	Std. means	Max.	Min.
Scenario	Feature set				
I	1	9.17e-01	7.62e-05	9.18e-01	9.16e-01
	2	8.01e-01	2.26e-04	8.03e-01	8.00e-01
	3	9.08e-01	1.84e-04	9.09e-01	9.07e-01
	All	**9.48e-01**	9.41e-05	9.49e-01	9.47e-01
II	1	8.55e-01	3.16e-04	8.58e-01	8.52e-01
	2	4.81e-01	8.83e-04	4.87e-01	4.75e-01
	3	8.26e-01	3.77e-04	8.29e-01	8.24e-01
	All	**8.60e-01**	3.63e-04	8.63e-01	8.58e-01
III	1	9.34e-01	8.92e-05	9.35e-01	9.33e-01
	2	8.50e-01	1.89e-04	8.51e-01	8.48e-01
	3	9.34e-01	1.74e-04	9.35e-01	9.33e-01
	All	**9.84e-01**	6.11e-05	9.85e-01	9.84e-01
Over all	Over all	8.58e-01	8.83e-04	9.85e-01	4.75e-01

Table 8.6: Summary of F1-measure across cross validation runs. See Table 8.3 for explanation.

Chapter 9

Heuristic methods for efficient identification of abusive domain names

Egon Kidmose, Erwin Lansing, Søren Brandbyge, and Jens Myrup Pedersen

The paper is accepted for publication in
International Journal on Cyber Situational Awareness (IJCSA), Vol. 3, No. 1, 2018.

Abstract

Domain names and the Domain Name System (DNS) are essential to the Internet, but unfortunately cybercriminals also make use of these to fulfil their nefarious agenda and gain illicit profit. In this work we survey known forms of domain and DNS abuse from the criminal business point of view. This is related to abusive techniques, which we also survey. Based on the theoretical understanding of the abusive techniques, we devise a set of practical heuristics for recognising said techniques. This enables a focused and efficient manual analysis of heuristically ranked domains, with the goal of identifying abusive domains. As the .dk Country Code Top-Level Domain has received little scrutiny in the past, but is believed to see only limited abuse, it represents a relevant and presumably challenging case for identifying abuse, and we therefore use it for evaluation. A sampled set of 10.000 second level domains are monitored for 66 days, heuristics are applied, and the resulting rankings guides a manual vetting. Our findings are that with automated heuristics we can limit the manual investigative effort to hours, but still identify 5 domains which was actively abused during our observation period.

Keywords

DNS, Domain Name, Abuse, Heuristics, Top-Level Domain.

1 Introduction

As links between the cyber and physical realms have grown in numbers and strength, the potential profit and impact of cybercrime has grown too. Domains names are a prime example of this, as businesses, organisations, and private persons use domain names not only for technical administrative purposes, but also very much for branding themselves on the Internet. As domain names and the Domain Name System (DNS) are ubiquitous, cyber criminals have naturally found it useful too, both for technical purposes and especially due to it being trusted by users and organisations.

One example of cybercrime involving domains is phishing, where attackers pretend to be a trustworthy third party and lure the victim into disclosing confidential information such a credentials, which can be exploited for profit. In phishing schemes, it is common to use domain names mimicking the third party to gain the trust of the victim. This type of attack targets the discrepancies between human fuzzy interpretation of domain names and the exact mapping provided by DNS. It can be combined with similar techniques, such as mimicking visual identity and logos, which are not pertaining to DNS and domain names. In addition to these human-targeted techniques there are also examples of abuse that are purely technical. Just like legit organisations rely on the DNS infrastructure, so do the criminals. This can be observed when bots or other malware attempt to establish a Command and Control (CnC) channel from compromised victim machines to the attacker, when spam e-mails are sent, or when scam web shops present themselves like ordinary, legit web shops. There are many more examples of how domain names and DNS can be abused, but we defer a more extensive survey to Section 2.

The global losses caused by cybercrime in 2017 have been estimated by Lewis [7] to be between 445 and 608 billion USD, while McGuire [87] estimate the annual global revenue for cybercrime to be 1.5 trillion USD. Given the size of the cybercriminal underground economy, it is clear that it is beneficial for the criminals to specialise in different roles. Examples of such roles include finding exploits, writing malware, infecting victims, operating CnC infrastructure, and laundering money [6]. While the individuals of a group of criminals can specialise accordingly, the reality is that criminals even specialise to the extent that they provide their services on well-established underground markets. Pay Per Install (PPI) is an example of this, where the operators of botnets sell or rent the victims as a service. Another example is how credentials and personal information are traded on well-organised, underground, online marketplaces, where customer satisfaction is ensured by providing merchant ratings and escrow services. As a final example, malware and exploit kits come nicely packaged with 24/7 phone support.

These phenomena are clearly undesirable for the general, law abiding society, so luckily there are means to counter them. As already exemplified, cybercrime relies extensively on domain names and DNS, which therefore is an interesting means for solving the problem. Victims can defend themselves with blacklists of bad domains and with detection methods based on both heuristic algorithms and data driven machine learning. An alternative, complementary approach is for the registries operating Top-Level Domains (TLD), such as *.com* or *.dk*, and related DNS infrastructure, to block queries or reject registrations for abusive domains. The registries are not necessarily impacted directly by abuse, but if a TLD is widely and commonly abused, the reputation is impacted negatively, and thereby the value too. Furthermore, there are legal aspects that motivate registries. Registries are thus both capable and motivated to combat abuse.

Our primary contributions are a method to heuristically rank domains, such that manual effort invested towards identifying abusive domains can be used efficiently, and the first published scientific study on abuse in the *.dk* TLD. We also contribute with an overview of malicious techniques, with heuristics motivated by an understanding of the techniques, and with the detailed results of applying the heuristics.

In Section 2 we survey different types of abuse, with an outset in the business models and the criminal economy in. Based on these types of abuse we identify malicious techniques employed by the criminals in relation to domain names and DNS. In Section 3 we employ the insights into the techniques to develop a set of heuristics for identifying domains where the techniques are applied. We collect data for a subset of the *.dk* country code TLD and apply the heuristics to search for abusive second-level domains (2LD), with Section 4 describing the outcome. We offer our interpretation of the results in Section 5, describe the future direction in Section 6, and conclude on the current study in Section 7.

2 Background

In this Section we survey abuse from a criminal business perspective, identifying different schemes, with clear links to how domain names are abused, and how the illicit profit is generated. This is followed by a survey of techniques that enables or improves the schemes. The distinction between schemes and techniques is important, because schemes allow us to understand the motivation of the criminals, while understanding of the techniques enables us to look for technical artefacts that can be searched for at a large scale.

2.1 Abuse Schemes

Phishing has already been described in the Introduction as a scheme where criminals rely on luring victims into disclosing confidential information. In this scheme the business model can be as simple as: Register a domain similar to the trusted third party, e-mail victim(s) misleading instructions, and receive credentials from the victims that succumb to the attack. Obtaining victim e-mail addresses, sending large amounts of phishing e-mails, and exploiting the results can be delegated, hence the criminal value added in phishing comes from tricking users. It is relevant to discern between 0-day phishing domains registered with intentions of abuse and compromised phishing domains that are registered with good intentions, but later compromised by criminals and abused for phishing [63]. 0-day phishing domains should ideally be detected before registration, or briefly thereafter, as criminals register, exploit, and discard domains rapidly [65]. Registries are challenged when it comes to addressing abuse with compromised domains without involving the registrant owning the domain, as any action by the registry will likely interfere with legitimate use of the domain. On the other hand, involving the rightful, exploited registrant can be slow and time consuming.

E-mail spam is the well-known malpractice of sending bulk, unsolicited e-mails, where profit can be made, for instance, by infecting victims (PPI), or through ads and referrals to dubious web shops [88]. While the problem is well known, there is plenty of evidence that the abuse of domain names for spamming has not been handled yet [65].

Scam web shops use domain names for landing customers, just like legit web shops, but the shipped goods might be counterfeit, if it is even shipped at all, and the customer credit card details might be stored and abused [89]. If the criminals accept payments but never ship the goods, or ship cheaper counterfeits, they profit. If the consumer is knowingly buying counterfeit goods, the criminals are profiting from facilitating the illicit trading of counterfeit goods. In all cases, domain name abuse enables illicit profit.

Domain parking is the practice of registering a domain without developing it and without providing genuine content, but rather redirecting traffic to a parking service, which generates generic content, typically advertisements, in order to monetise from users who, by mistake, point their web browser to the domain [90]. Parking is clearly not aligned with the users' intent, but as stated by Moura et al., it is not necessarily illegal, hence it can make sense to distinguish between legal parking with ads and illegal, malicious parking, where users are diverted to scams, exploits, or other attacks [63]. In either case the revenue stems from selling redirections of users, regardless of the user's intentions.

Web spam is a scheme that lends itself to e-mail spamming, as it uses a bulk of useless or misleading content, but instead of distributing this via e-mail it presents itself as web pages [91]. Abusing multiple domains to host content that refers to each other only, the criminals seek to entrap search engine web crawlers and boost their own malicious content into search results, which is why this is also referred to as Blackhat Search Engine Optimisation [63]. Profit comes from serving victims with ads or malicious content. While this form of abuse is fully dependent on domain name abuse, it is obvious that search providers also have motivation and options to combat this.

Botnet CnC can abuse domain names as rendezvous points, where victim machines infected with bot malware can reach the bot master's infrastructure. With an established CnC channel, the bot master obtains scalable remote control and data exfiltration capabilities, which can enable other schemes, including harvesting of banking credentials or credit card details, sending of e-mail spam, or PPI.

2.2 Abuse Techniques

Mimicking is a technique intended to make the human victim confuse a malicious domain for a legit and trusted third party. This can be achieved with slight alterations from the third-party domain name, e.g. barely noticeable spelling errors, minor edits, insertion of hyphens, substitution with homoglyphs, and more. Attackers can both use mimicking domains actively, e.g. when sending targets phishing e-mails, and passively, such as with typosquatting where attackers rely on victims to mistype a domain name, so they end up at a parked domain.

Malicious re-registration, also known as drop catching, is when an attacker registers an expired domain for abuse. Browser bookmarks, hyperlinks, user-remembered domain names, residual search engine results, based on the previous content, and general system configurations cannot reflect that a domain has expired. Instead trust in a domain persists if it re-registered by a criminal. Users browsing the web can be redirected to parking services, bot-infected victims can be re-enrolled by a new botmaster, and DNS infrastructure can be hijacked just like any account recoverable through an e-mail address in the domain [92].

Bulk registration refers to the practice of registering many domain names in bulk. While Hao et al. [65] described how e-mail spammers employ this technique, it is clearly relevant for web spammers too, and also when employing Domain Flux (See below). The motivation for criminals to do bulk registrations lies in the convenience and scalability, and in the discounts offered by registrars.

Fluxing refers to a collection of techniques that abuse the capabilities of DNS to make the criminal's infrastructure more resilient to take-downs and/or harder to track, investigate, and block. Fast Flux is the first variant, where a domain name maps to many IP addresses that all provide identical service or content, possibly by proxying, with the mappings changing rapidly [93]. The benefit for the attacker is that forensic analysis on the victim only provides one of the many redundant IPs, that IP might only point to another victim unknowingly proxying for the attacker, and all this information is rapidly outdated. This thwart blacklisting and take-down efforts.

Double Flux extends Fast Flux by also applying the same approach to how authoritative nameservers are found (Fast Flux applies to A records in DNS. Double Flux applies to NS records.) [94].

Domain Fluxing can be seen as the inverse of the above: Fast Flux and Double Flux enables a domain name to point to many IP addresses. With Domain Flux, a large set of domain names can be used to point to a single IP. This is achieved with Domain Generation Algorithms (DGAs) that produce large sets of pseudorandom domain names, of which the attacker can chose to register any one to establish a CnC channel [95], [70]. This makes it unfeasible to block or sink-hole all the domains.

2.3 Related work

An empirical study of spammers advertising for scam web shops, focusing on the value chain for the criminal operation has been conducted by Levchenko et al. [96], providing the insight that payment processor represents a bottleneck and thereby an interesting point for disrupting the illicit business. This study is different to ours in that it goes towards the physical realm, studying the criminal business operations extensively. This clearly has benefits, but also some drawbacks, such as the ethical issue of completing business with the criminals, the inertia of operations (e.g. shipping), and poorer scalability compared to the cyber realm. Our proposal does not require business interactions, can be conducted with the speed and scale common to the cyber realm, and spans more forms of abuse.

3 Methods

With a solid understanding of abuse from the preceding surveys of schemes and techniques, we now move on, towards a method for identifying abuse. The goal is to efficiently identify abusive domain names. First, we reason for an approach of applying heuristics to domain names, as they can be applied at scale. We then present a set of concrete heuristics that can be used to rank domains by how likely it is that they are employing specific techniques. Finally, we describe an approach for manual vetting, which can be applied to the domains that are ranked the highest by the heuristics. By relying on heuristics to focus the manual effort where it is most likely to provide positive identification of abuse, our method optimises the number of identified domains that require action, with manual capacity as the constraining resource.

We observe that domain names are a frequent component among the surveyed schemes, and a component of all the techniques. This implies that domains have substantial potential for abuse, but also that analysis of domain names can provide insights to multiple forms of abuse, and that mitigation based on domain names can have serious impact on criminal activity. At the same time domain names can be processed as purely digital entities at a scale and speed that is significantly higher than if the analysis was to also encompass the criminal activities in the physical world.

When analysing abusive domains, a known problem is that the set of abusive domains is not clearly isolated in the set of all domains. This is due to the criminals having the liberty to register any free domain, while they have an interest in not disclosing which they register or abuse. A common practice is to rely on blacklist as a source of abusive domains, without including any non-abusive domains. This has the benefit of being a practical solution to the problem, but it relies on the blacklists to be correct, and it does not provide for identifying abusive domains that are not blacklisted. Furthermore, different blacklists often target different types of abuse, based on vaguely defined or unspecified criteria, making them difficult to understand. An additional concern is the extensive discrepancies between blacklists, which adds to the ambiguity. In this work, we define domain abuse as any use that violates accepted terms, applicable law, or the non-conflicting interest of non-criminal users of the Internet.

We analyse domains from blacklists as well as domains sampled randomly from the relevant zone-file, thereby providing insights on the zone and not just on the blacklisted part of it.

We hypothesise that given a solid understanding of the abusive techniques, and of the technical aspects of DNS and domain names, it is possible to describe artefacts of the techniques, which in turn enables us to define heuristics that can be applied automatically and at scale to highlight domains that are likely being abused, such that they can be subjected to deeper manual vetting process to identify abusive domains.

3.1 Data collection

This study is enabled by a unique access to information from the *.dk* zone, specifically a list of all 2LDs. Previous studies on abuse are largely focused on *.com* and *.net*, with some other TLDs also receiving some attention, but this is to the best of our knowledge the first published study of different forms of abuse in the *.dk* TLD.

To gather sufficient details for identifying of as many forms of abuse as possible the list of domain names is enriched with a data from the public WHOIS service[1]. The *.dk* WHOIS service applies a rate limit of 1 query per second, meaning that collecting data for all *.dk* 2LDs would take weeks. As abuse for a given domain might be limited to hours or days this poses a problem. We solve this by limiting our study to 10.000 domains. First, to increase the likelihood of finding abuse, we use all *.dk* 2LDs found in a set of 31 retrievable, public blacklists [97], as we have higher expectancy of these being abused. Second, we sample the list of all *.dk* 2LDs up to 10.000 total domains. For our subset of *.dk* domains we collect WHOIS data once a day. This strikes a balance between coverage of the zone and frequency of updates, given the boundaries of the rate limit.

3.2 Heuristics

Mimicking domains are intended to look similar to other domains, hence our heuristic for this is targeting similarity. Similarity can be expressed in many ways, but as domain names are essentially text strings it is obvious to apply the Levenshtein distance, which describes the minimum number of edits that transforms one string to another [98]. Clearly the similarity measure needs to be applied to a potentially mimicking domain and a potential target domain. For target domains we expect that criminal focus on popular domains, as they are more likely to impact a larger group of victims. Consequently, our evaluation uses the Alexa top 1 million of popular domain names as targets. We analyse both 2LD labels, such as the example part of example.*dk* and the

[1] `https://github.com/DK-Hostmaster/whois-service-specification`

Fully Qualified Domain Name (FQDN) of the *.dk* domains against the 2LD label and FQDN, as the TLD might and might not be part of the mimicry. We expect small editing distances for longer labels/domains to be less likely to occur naturally, meaning they are more interesting if they occur, so we normalise editing distance by label/domain length.

Malicious re-registration occurs very rapidly according to [99]. While their study is subject to the specific conditions for the *.com* domain, the logic reasoning is generally applicable. Hence, our heuristic for malicious re-registrations is to select the re-registrations that follows the closest after deletion for further inspection.

Double flux is defined as a domain having rapidly changing nameservers, so our heuristic is to rank domains by how frequent the set of nameserver hostnames change in the WHOIS data.

Domain Flux relies on DGAs to generate domains that appear pseudorandom, as seen in the examples provided by Schiavoni et al. [70]. Our heuristic is the entropy of the distribution of letters within the 2LD labels, which is expected to be high when the letters are pseudo-randomly distributed.

3.3 Manual vetting

For each of the above heuristics we obtain a ranked list of most likely abusive domains (Based on the heuristic). Given the lack of ground truth, the top ranked domains are vetted manually, based on the following procedure.

Additional information on the domain is retrieved. This includes any homepage hosted via HTTP(S). The domain is checked against a more extensive set of blacklists[2], including some that are query-only and not retrievable in full. If the registrant appears to be a Danish company, the official Central Business Registry (CVR) is queried[3], as for instance missing records are highly suspicious, while long-lived companies are assumed to be less prone to register a domain for abuse. The Google search engine is queried for the 2LD and the first 10 results are inspected for any obvious relations to abuse. Finally, the history of WHOIS data is inspected for any content or change that could be relevant to understand if the domain is abused.

4 Results

The following describes the details of the data collection operation, the results of applying heuristics, and the outcome the subsequent manual vetting. In summary, the heuristics based on editing distance extracted 78 domains which were vetted manually, leading to the conclusion that 5 was actively

[2] `https://www.urlvoid.com`

[3] `https://datacvr.virk.dk/data/`

abused during our observation period, while 22 were cases of defensive registrations. The manual vetting took approximately 4 hours in total.

4.1 Data collection

In accordance with the data collection procedure described in the previous section, 269 *.dk* domains were found on the monitored blacklists, and the remaining 9.731 domains were selected at random among all *.dk* domains. The WHOIS information for these 10.000 domain was retrieved once a day from March 23rd, 2018 to May 29th, 2018 (66 days).

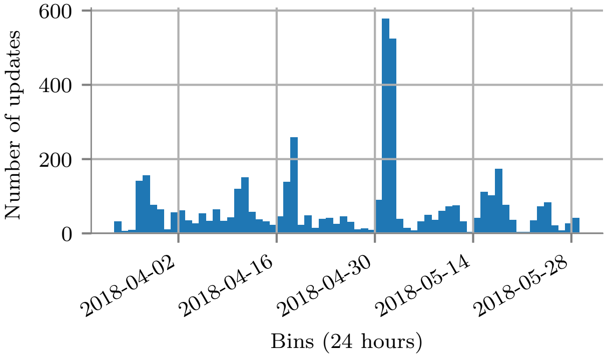


Fig. 9.1: Number of daily updated records.

Figure 9.1 shows the number of domains that change per day. The spike around the end of April and the beginning of May is caused by a failure to retrieve the data, i.e. the records for some domains are flapping to and from empty. Figure 9.2 represents the same information but cleaned for this error. The cause has not been identified, but as evident from Figure 9.2 most records resumed the same value after said incident. The WHOIS data for 9.051 domains where unaffected, of the 949 affected domains, 846 domains saw exactly one failure, and six was the highest number of failures for any domain.

4.2 Editing distance: *.dk* 2LD labels against Alexa 2LD labels

The editing distance between the 10.000 2LD labels from *.dk* and the 888.876 unique 2LD labels from Alexa is naturally 0 for the 554 labels that are present in both. As lower is more interesting, we prioritise to analyse these. The 2LD in *.dk* corresponding to 12 of these labels was never active during our observation period, so they are ignored. Reasons for this inactivity are that

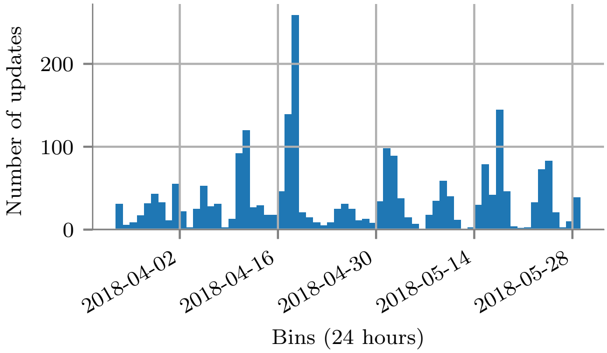


Fig. 9.2: Number of daily updated records with empty results removed.

they expired before our observation, meaning they was observed as "Deactivated" in WHOIS for a while before disappearing, or that the registration never completed, leaving them in a "Reserved" state. In either case, they were not in the *.dk* zone, were not resolvable via DNS, could not have been abused, and therefore they are ignored. Of the remaining 542 labels, the corresponding *.dk* 2LD for 506 labels had been registered for more than a year, which is the minimum registration period. This contradicts the expectation that the period of active abuse is short, so these are ignored, in order to focus on the remaining 36 domains. For 34 domains the manual vetting procedure yielded no indications of abuse. For one label, which coincides with a British menswear brand, the *<LABEL>.com* 2LD is found in Alexa and is registered to company behind the brand, but *<LABEL>.dk* is registered to an individual with no apparent affiliation to the brand, and the domain is parked (The authoritative nameservers are at *sedoparking.com*). This appears to be a parked, cybersquatted domain. The remaining one label relates to online marketing, and a marketing company owns the Alexa-listed *<LABEL>.com*. The *<LABEL>.dk* domain was to expire between the end of observations and the time of writing, is currently not listed in WHOIS, but is still reported as a phishing domain by Fortinet[4]. The Internet Archive[5] has a single capture during the suspicious registration period, which shows minimal front page consisting of nothing by some JavaScript and an iframe. It appears likely that this was a domain abused for phishing.

[4] `https://datacvr.virk.dk/data/`

[5] `http://web.archive.org/`

	# 2LD labels
Never active (Filtered)	12
Active for +1 year (Filtered)	506
Manually vetted	36

Table 9.1: Overview of the 554 2LD labels found in both the 10.000 *.dk* domains and in the Alexa top 1 million.

In summary, comparing 2LD labels, ignoring inactive and long-lived domains, we manually analysed 36 domains, and found two domains that were likely seeing active abuse. One is a parked cybersquatting of a domain related to a menswear brand. The other had suspicious content and was blacklisted as a phishing site.

	# 2LDs
Cybersquatting brand/web shop, parked	1
Phishing	1

Table 9.2: Overview of abusive domain found by comparing 2LD labels and subsequent manual vetting.

4.3 Editing distance: *.dk* 2LD FQDNs against Alexa FQDNs

The normalised editing distance between FQDNs highlighted 31 *.dk* domains as they were also found in Alexa, resulting in a distance of zero. This is of course not abuse and these are filtered out. Subsequently, we manually analysed the 50 pairs of *.dk* and Alexa FQDNs that were closest, without being exact matches. Seven of the top 50 pairs were the legit domain *triumphmotorcycles.dk* paired with the same 2LD label in *.de*, *.be*, *.ch*, *.es*, *.fr*, *.in*, and *.it* TLDs. Similarly, three pairs were the legit *blogspot.dk* paired the *.de*, *.mk*, and *.sk* variants.

For all 50 pairs, we subjected the 2LD from *.dk* to our manual vetting process, with the following results. Data on the Alexa domain from the same sources was also considered where relevant. 20 domains appeared to be used for hosting legit websites of enterprises, smaller companies, and private persons, with no suspicion raised through the manual vetting process. 22 domains appeared to be defensive registrations by the owners of the corresponding Alexa domain, either redirecting accordingly or parked with an apparently conscientious, add-free provider. Of these 22, 14 where defensive registrations for other *.dk* domains, six where for domains in the *.de* TLD, and remaining two are those pertaining to the *blogspot* label as mentioned above.

For three of the 50 pairs, the *.dk* 2LD is registered, is not found to be related to abuse, nor did we find any indication that it is actively used, e.g. with web page or Google results that include an e-mail address at the domain. These domains appear to be unused.

A summary of the non-abusive domains captured by the normalised editing distance between FQDNs is as follows: 20 domains seeing ordinary use, 22 cases of defensive registrations, and three passive domains.

One of the 42 *.dk* 2LD domains (50 pairs, 42 unique *.dk* domains) was found to be close to a service for price comparison service also in *.dk*, but with no apparent affiliation, and with the domain parked with *parkingcrew.net*. We strongly suspect this to be a typosquatting domains that was active in the period of our study, and still is at the time of writing. Two other pairs were cases of the *.dk* FQDN being similar to a *.de*, with 2LD labels matching, while the *.dk* domains redirect to parking services. We believe these two are typosquatted domains, that monetise through parking. Finally, we also found one domain that was close to domain with outdoor lifestyle content and a web shop. The suspected domain had been seized by authorities prior to our observations, the Internet Archive has records of a web shop which appeared highly suspicious, and it was blacklisted by Web of Trust[6] as a scam/counterfeit web shop.

	# Unique 2LDs (pairs)
Manually vetted	42 (50)
Legit use (excluding defensive)	20
Defensive registrations	14 (22)
- Defensive, paired with Alexa domain from **.dk*	8 (8)
- Defensive, paired with Alexa domain from **.de*	6 (6)
- Defensive, paired with Alexa domain from **.mk* or **.sk*	1 (2)
Not in use	3
Typosquatting of price comparison service, parked	1
Typosquatting (*.de* vs. *.dk*)	2

Table 9.3: Overview of pairs with low editing distance (Levenshtein) between one of 10.000 *.dk* 2Ld domains and a FQDN from the Alexa Top 1 million. Right-hand column list count of unique 2LDs from *.dk* where relevant number of unique pairs are listed in parenthesis. Note that a *.dk* domain can be in multiple pairs.

[6] `https://www.mywot.com`

4.4 Re-registration and High Entropy

Among the 10.000 domains that was observed, three was successfully re-registered during our 66 days observation period, with lags of 14, 15, and 152 days respectively. We found no evidence of abuse and assume all cases to be legit registrations.

For the 50 domains that had the highest entropy, we found no indications of active abuse. One domain had previously been seized by the authorities, but before our observations started, and due to trademark infringements and scams, which appears unrelated to the entropy. All of the 50 domains appeared human readable.

5 Discussion

We were able to apply theory about how criminals operate their business and what techniques they use to devise heuristics that automatically can prioritise domains for manual scrutiny. In the case of the Levenshtein editing distance as a heuristic for identifying the mimicking technique the automated procedures processed 10.000 domains and prioritised 78 domains[7]. Among these we are confident that five domains were being actively abused during our observation period: Four cases of apparent typosquatting with redirection to parking services, and one case of phishing. Additionally, we identified 22 cases of defensive registrations, which either have been abused previously and then seized, or the brand owner have deemed is so likely to be registered for abuse that it is worth acquiring. In either case, it supports that our heuristic is suitable for identifying relevant domains.

Considering that manual analysis of a domain takes a few minutes, we are able to identify five cases of abuse (and 22 defensive registrations) by investing about a half a working day (78 domains at 3 minutes per domains $\approx$ 4 hours). We hypothesise that if the entire procedure is applied to a larger set of domains, while the number of top-ranked domains subjected to manual vetting is kept fixed, the number of identified abusive domains is expected to be even better. This is based on the assumption that the expected prevalence of abuse increases with higher relative ranks, and the fact that with a larger set of domains the top-N (with fixed N) will correspond to a relatively smaller part of all the domains. This naturally prompts for validation through studies on larger scale, such as the entire *.dk* domain, in order to support or dismiss the hypothesis.

In our study, the heuristic for abusive re-registrations fails to capture any abuse. This can be because the manual vetting process fails to discover

[7] 36 2LD labels found in both *.dk* and Alexa. 42 unique *.dk* FQDNS that was in the top 50 of similar FQDN pairs.

present abuse, or because abusive re-registration does not occur for the *.dk* zone. Another more likely explanation is that the data set is too small and therefore does not contain abuse. Among the 10.000 domains observed for 66 days, we only observed 3 re-registrations. No abusive re-registrations can be expected from such a small set. To evaluate this heuristic the data set must be extended, which can be done by including more domains and by observing for a longer period of time. Extrapolating the observed frequency of re-registrations to the 1.3 million domains in *.dk*, 390 re-registrations can be expected to occur in a 66 days period. While the ratio of abusive re-registrations is not known, 390 re-registrations still appear to be a low count, so expanding the observation period also appears necessary. Alternatively, evaluation can be extended to more or larger TLDs.

The heuristic for entropy also fails to yield any domains applying domain flux in the top-50, with one apparently irrelevant exception (A domain abused and seized prior to our observations, with a human readable label that did not appear pseudo random, i.e. apparently not from a DGA). Like for re-registrations, this can be explained by errors in the manual vetting process or by the evaluation data not containing examples of DGA domain. We are inclined to rule out errors in the vetting process, as DGA domains are expected to clearly stand out simply by the pseudo randomness apparent from the label, see for instance Schiavoni et al. [70]. It is possible that there are no cases of domain flux in the data, or perhaps even in the *.dk* zone. The price of registering a *.dk* domain, the registration process, which involves a process for validation registrant identity, the legal requirement for public WHOIS, and other specifics of the *.dk* might divert certain forms of abuse to other TLDs. It is also possible that criminals have improved their DGAs to be more stealth. This appears to be possible by generating domains that are closer to legit domains in some lexical sense, such as character or N-gram distributions, or simply by combining random words. In this case, a new heuristic must be devised. As per Section 3 this prompt for a study of the novel techniques, if such exist, which would merely be guesswork without examples.

In Section 2 we discussed more techniques than Section 3 present heuristics for. This is due to limitation in the current available data. As described, Fast Flux exploits rapid changing A records on the authoritative nameserver, which is chosen by the registrant and not operated by the registry, therefore we do not have access to the master data. The nameserver operators are likely bound by confidentiality to their client, the registrant, and are perhaps also accomplices to abuse. Possible approaches to overcome this are passive DNS traffic monitoring and active probing. In the case of bulk registrations, practical limitations lead us to select a subset of domains for observation, as described in Section 3, which lead to our data not providing insights on this technique. The solution, which is in development, is to monitor the entire *.dk* zone, including new registrations.

6 Future research directions

Having demonstrated that the approach of analysing abuse techniques, devising heuristics, ranking domains, and manually analysing the domains ranked as most likely to be abusive is valid, at least for the used data set and some of the techniques considered, we would like to expand the study to obtain more general results. The most obvious first step is to analyse the entire *.dk* zone, as the limited number of domains is a recurring issue, as evident from the above discussion. This implies some practical challenges that we are currently working on. Similarly, expanding the duration of the observation is relevant, and this is more straightforward. Extending to other TLDs is also a possibility, but this is subject to the details available in the data. As registries and TLDs differ, the current heuristics might not apply, but this only means that it is a possibility to evaluate our entire approach, including the analysis of abuse techniques.

Some techniques are not presumably not evident in the WHOIS data, as discussed above, so passive DNS traffic monitoring and active probing of recursive nameservers are under consideration as data sources for further studies. Specifically, we are currently investigating the OpenINTEL framework[8], which can enable heuristics for the Fast Flux techniques and more.

In general, we still see domain names as an essential link between the physical and cyber realms, which is enabling for the majority of both legit and criminal activity involving cyber, and therefore a key point for attacking the criminal activity which evidently persists.

7 Conclusion

We have described the extent of cybercriminal business and surveyed the abusive schemes and techniques that employ domain names to enable the crime, but also make domain names and the Domain Name System a choke point for combating cybercrime. We have proposed an approach of defining heuristics for abusive domains names, based on the abusive techniques, and present a set of such heuristics. We have demonstrated that our heuristics can be applied to focus manual effort, allowing us to identify five abusive domains among 10.000, with four hours of manual effort. We contribute by detailing this approach, and by providing the first scientific study on abuse in the *.dk* country code Top-Level Domain.

[8] `https://www.openintel.nl`

Part III

Conclusion

Chapter 10

Discussion: Correlation and filtering

This chapter extends the discussions of Chapters 4 and 5, covering broader and more general perspectives within the topic of correlation and filtering.

1 An unsolved problem

Correlation and filtering of Intrusion Detection System (IDS) alerts was identified as research problem no later than 2001, and has continued to receive attention up to the time of our work in Chapters 4 and 5, as evident from the references in Table 5.1. Based on the results and conclusions of the surveyed work, the problem appears to have been studied extensively, and well-performing solutions have been implemented. This appears to conflict with occurrences like the Target case (Section 1 in Chapter 2) and real-life experiences of not having practical solutions available. There are at least two possible explanations for this: First, it is possible that methods are not adopted because they are not feasible in practice. Refer to Chapter 5 for elaborations on this. Second, methods might perform particularly well under the specific evaluation conditions, but not in general. In this case, the evaluation results are biased.

2 Evaluation bias

One possible source of bias in evaluation results is the evaluation procedures. A textbook example of this is to train and test Machine Learning (ML) methods in a way that is only is possible in an artificial setting where an entire sequence of observations is available. The results of such evaluation does not necessarily represent a practical setting where real-time constraints and the ordering of observations is enforced. No concrete examples of biased evaluation procedures are evident from the most recent of the work surveyed in Chapter 5, but on the other hand details are often insufficient to rule it out. Redoing evaluations of the prior work is one way to address this, but it is expected to require significant time, and in cases where tuning and training involves human expertise the re-evaluation is inherently subjective. Another solution is to focus on future proposals, ensuring that both methods and evaluation procedures are available, ideally implemented in code or alternatively documented so thoroughly that re-implementation is feasible.

Evaluation data is another potential source of bias in evaluation results. As some evaluation data must be selected, this is hard to avoid, but steps can be taken to address the issue. Our work in Chapter 4 is evaluated on traffic from bot malware, collected in a lab and without benign traffic. In the context of general correlation and filtering, these results would be considered biased. This is addressed by being explicit about the above details, and by expanding the evaluation data used in Chapter 5 with benign traffic and by introducing a new and more diverse data set.

A third potential source of bias in evaluation results is evaluation environment, i.e. everything except the procedure and the data. The IDS is a general example of this as it is part of the problem, and as different IDS are expected to produce different alerts, with different content, in different volumes, etc. This is complicated further with configuration and adjustment options, and especially with different rule feeds from communities and vendors. For comparable results, this naturally has to be fixed across evaluations, which can be done by clearly stating versions for both IDS and rule sets. Additionally, we use the recently emerged container technology to have a fixed, reproducible, lightweight, virtual IDS appliance[1], which is available to anyone.

3 Siamese Recurrent Network

The essence of Chapter 4 is the method for correlating and filtering alerts, using two identical Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs) with tied weights. This architecture is used because it proved capable of learning to estimate relatedness for pairs of alerts. At around the same time Mueller and Thyagarajan proposed a highly similar structure, which they call a Siamese Recurrent Architecture [100]. Their method is capable of learning to predict relatedness between pairs of natural language sentences and achieves good performance compared to the state of the art. This is interesting as the two architectures of neural networks are very similar, which could be due to a shared notion of similarity in pairs of text.

[1] Containerised IDS: `https://hub.docker.com/r/kidmose/snort/`

At the same time, the two problems are also presumed to differ significantly given different latent structure in the data, for instance because the alerts are machine generated, while the sentences are natural language. Another significant difference is that in our work the ability to estimate relatedness is a mean to solve the problem of correlation and filtering, whereas for Mueller and Thyagarajan such estimation capabilities appears to be the overall goal. The conditions of the two pieces of work also differs as labelled pairs of human sentences appears to be scarce, while our method is more likely to encounter problems due to too many training samples resulting from our data pre-processing procedure (See Figure 4.3).

Due to two different problems being addressed, the metrics differ, and performance does not compare directly. However, considering the architecture, it is obvious that the ideas underpinning the two solutions are highly similar, so comparing method performance on the same problem could be interesting. This can be done by extending the method of Mueller and Thyagarajan to do filtering and correlation, i.e. add clustering and incident prediction steps, or by trimming our proposal to only estimate relatedness, i.e. not do clustering and incident prediction. Mueller and Thyagarajan apply a more complicated procedure, involving pre-training with a separate data set, a step for encoding words as vectors with another neural network, and other details, which has potential to impact performance. A practical comparison could yield insights as to what is gained and lost from the added complexity and from the different choices on procedures and architectures. Relevant aspects to consider includes performance on solving the problem, needs for tailoring methods to the specific problem, and computational costs.

4 Hetero- and homogeneous alerts

One of our motivations for exploring filtering and correlation in a setting where alerts are represented as text is an interest in correlation and filtering for heterogeneous alerts [22, 23], as opposed to homogeneous alerts, which has received much more attention[2]. All known IDSs have the capability to represent alerts with lines in log files, thus heterogeneous alerts can presumably always be represented as text strings. Two alerts represented as text are considered homogeneous despite any differences in the content of the text. This means that our methods in Chapters 4 and 5 can be applied to heterogeneous alerts, without any additional steps.

Depending on the implementation, either the Latent Semantic Analy-

[2] Homogeneous alerts are here defined as a set of alerts that all have the same schema, i.e. the same fields and the same possible values for those fields. Conversely, heterogeneous alerts are defined as a set of alerts where some alerts have a different schema, i.e. different fields and/or different possible values in those fields.

sis (LSA) or the LSTM RNN is tasked with learning to extract semantic meaning, or as put in the papers, to learn a useful mapping function from alerts into the abstract feature space. Although the schema is the same when applying the method to heterogeneous and to homogeneous alerts, the latent structure in the text, and thereby the difficulty of the task, is expected to be different. For this reason, our previous evaluation results on homogeneous alerts cannot be generalised to heterogeneous alerts, so further studies into this are required. This can bring us a better understanding of how the methods perform in scenarios where the latent structure of the data is expected to be more complex. Bringing heterogeneous correlation and filtering to a level of maturity where it can easily be applied in practice is particularly interesting as corporations are believed to have many sources of security related event information (not just alerts) that represents unused potential for better detection coverage of the infrastructure and for improved performance.

5 Semantic clustering for security in general

Correlation and filtering can be seen as a special case of semantic clustering, where filtered alerts correspond to outliers. The LSA implementation introduced in Chapter 5 is an example of this. It is an algorithm for extracting semantic meaning of texts, which we combine with the DBSCAN clustering algorithm to build correlation capabilities. Semantic clustering can be applied to other problems within security, such as for instance phishing, where it is relevant to understand the campaigns of criminal groups, as this can guide proactive efforts and direct attention to the most important threats.

One approach could be to manually derive criteria for each campaign, but this scales poorly. Semantic clustering, if successful, can extract semantics from the emails to automatically find structure in a large body of phishing emails. This can for instance be implemented to analyse mail received by a spam trap on a large scale, thereby providing valuable insights into criminal campaigns which again can guide mitigation efforts. Filtering capabilities are relevant if data contains errors, which for instance could be the case with user reports of phishing. Turning the concept around and focusing on outliers, it is also interesting to explore if semantic clustering could be applied all emails, in order to highlight malicious cases such as CEO Fraud by marking it as outliers. With emails used as an example, it is important to note that semantic clustering and related methods also has potential for many other topics within security, including web content, WHOIS information for domain names, Domain Name System (DNS) records etc.

Chapter 11

Discussion: Domain names and DNS

Domain names and the protocol enabling their use, DNS, is the second topic covered in this dissertation. Domains names are intended as human-readable handles for infrastructure and offers a hierarchical structure of administrative responsibility on the Internet. Being widely deployed and used, domain names and DNS are also abused for cybercrime in many different ways. Some of these have been explored in Chapters 6-9, which also are concerned with detection of the abuse, specifically the malicious domains. The following extends on the discussions in these chapters and discuss the topic in general.

1 Pre-registration Detection

The essential point of pre-registration detection, as introduced in Chapter 6, is to identify domain names that are sought registered with intent of abuse, in order to prevent that they are activated and become resolvable through DNS. Given successful detection, this can completely prevent the intended abuse of detected domains, which is a huge win for security on the Internet.

As discussed in Chapter 6, the topic has however received little attention in the past. This could be because less information is available, compared to post-registration or post-abuse, which make accurate detection harder. In Chapter 8 we evaluate lexical analysis methods for detecting some abusive domain names, and in Chapter 9 we explore the use of heuristics for other forms of abuse. Both the lexical features and the heuristics can be used pre-registration, so it is highly interesting to combine these into a solution for pre-registration detection and evaluate if it can work in practice.

Challenge for evaluations, generally relevant for abusive domains, is how the ground truth on maliciousness evolves over time. Domains might be compromised and abused, they might expire and be re-registered with different intentions, and domain owners might change their intentions from good to bad and vice versa. All this, and the fact that criminals often seeks to remain undetected, contributes to making it hard to establish the ground truth. In Chapter 9 we use WHOIS information as data source, and find that for this source (*.dk* WHOIS) a daily update appears sufficient as records appears to change much slower.

When developing or deploying pre-registration detection methods, it essential to keep in mind that the ability and right to use the Internet, including registering a domain, is tightly coupled to the freedom of expression in a digital society. Consequently, it is important to avoid the risk of false positive detection interfering with legitimate registrations. Similar concerns can be raised in other detection and prevention settings, but in the pre-registration setting it is particular important because it implements restrictions before any criminal or abusive act is carried out.

2 DNS Security Extension (DNSSEC)

As mentioned in Chapter 1, there are ongoing efforts to improve security on the Internet, and some of them relates to domain names and our work on mitigating domain abuse.

Domain Names System Security Extension (DNSSEC) is an example of this. While DNSSEC has been deployed for a while, it is considered emerging in this context as the current adoption is limited [101]. DNSSEC provides integrity guarantees for responses by cryptographically signing and validating them [102]. This signing provides some authenticity guarantees for responses, making them more trustworthy. However, it does not protect against malicious or compromised domain owners, who are free to serve any malicious content. Furthermore, it does not provide confidentiality for the domain name being resolved. While problematic in other perspectives, the lack of confidentiality means that even with DNSSEC it possible to analyse DNS traffic in the network. This can for instance be done by applying the methods presented in Chapters 7-9.

3 DNS over HTTPS

HTTPS represents a similar improvement for the long-standing HTTP, but also provides confidentiality for content, and it is seeing much greater adoption [103]. HTTP(S) rely on DNS to resolve host names (Assuming the URL contains a host name and not an IP), thereby leaking the host name through DNS. This means that HTTPS by itself does not affect the ability to study domain name lookups in the network as mentioned above. However, as the ability to do so also is perceived as a security threat, and probably rightly so, there are efforts to provide confidentiality of DNS queries. One example of this is DNS over HTTPS (DoH), which in essence tunnels DNS traffic through HTTPS, from the client to a server on the Internet [104].

For corporations, the ability to inspect DNS traffic and to direct DNS traffic to trusted recursive resolvers are important security mechanisms. These mechanisms will fail if DoH becomes available on corporate clients as name resolution traffic will disappear into the vast amount of HTTPS traffic. Given the support of organisations known for setting the trend on the Internet this scenario appears likely [105,106].

This is a case of a common conflict between security and privacy of individuals or users, versus the desire of organisations to monitor and control. In the case of corporations and employees, the intent with traffic inspection is presumably legitimate, well-meant, and in line with the interests of the employees. However, the same Internet and protocols are also used by dissidents living under totalitarian regimes who direly needs confidentiality, and by ordinary citizens who are interested in preserving their privacy in the presence of commercial interests in the Internet. The choice of corporation to use the Internet, means that such other interests also affect the trends. It seems corporations need to consider alternative or additional mechanism to handle the threat of losing visibility into and control with DNS traffic.

Tight management of clients is one option, but with the trend towards web applications, which can implement DoH in the application code, this does not appear promising. In this setting, security mechanism would need to tap into application code, which is expected to be complex, cumbersome, expensive and scale poorly, i.e. this is not a relevant option. Another option is to apply SSL-stripping for traffic leaving the corporate network towards the Internet. The essential idea of SSL-stripping is to intercept traffic and terminate the secure tunnel, before proxying traffic onto the Internet, optionally with a new secure tunnel to the intended host. This is relatively trivial for network administrators on corporate networks to implement. The result is that the network administrators (or malicious attackers) get access to decrypted traffic at the proxying node, thereby solving the problem. Again, this poses a threat to general Internet users so HTTP Strict Transport Security (HSTS) was implemented to prevent SSL-stripping, and it is efficient as long as hosts on the web are preloaded, like many major sites, or have been visited before [107]. It appears that if clients are tightly managed, such that HSTS can be tampered with by system administrators, and if network administrators employ SSL-stripping, then insights into DNS traffic can be maintained in the future. However, this clearly have significant costs in terms of maintenance, deviation from Internet standards, ethics, circumventing security, etc. It seems unlikely that going against general goals and development of the

Internet is prudent in the long run. However, on short term and depending on how a corporation value the above costs, it is not impossible that this is still be relevant in some cases.

Another complication to the above problem appears with the introduction of recent concepts for corporate IT solutions, such as Open Platform, Software as a Service, Bring Your Own Device, and mobility. These ideas are replacing the more traditional and conflicting concepts like the idea of defending a perimeter and maintaining a Walled Garden. In the context of the recent or upcoming concepts, the above approach of maintaining access to DNS traffic can fail, as it is challenging to manage employees' devices tightly and to always keep them within the corporate network. VPN technologies can solve part of this, and corporations are not required to adopt the new concepts, but as described in Chapter I, they are driven towards the Internet by prospects like growth-potential, increased competitiveness, and productivity.

All of the above is concerned with how the companies can improve their stance individually and from an introspective point of view. The problems appear hard to solve, and where solutions exist, they come with significant drawbacks. This supports that can there can be a potential in also turning things around to look out onto the Internet instead of only into the infrastructure of corporations. From our work and my experiences, it is evident that the organisations behind important infrastructure, like DNS and the business of domain names, can have an interest in improving Internet security, while they also have the position to implement and take action. For society in general it appears that the most rewarding approach is to work together with these organisations towards a more secure Internet. While the above covers corporations too, they might also benefit from, and see a need to, implement defensive mechanism minded specifically on their network, keeping in mind that Internet trends appears to work against relying on such approaches alone.

4 Certificate Transparency logs

Another recently emerged technology related to HTTPS with implications for security of domains names is Certificate Transparency (CT) logs [108]. With the adoption of HTTPS, web servers need certificates from a trusted Certificate Authority (CA), which is expected to record each issued certificate to the public CT logs. As browser vendors employ increasingly explicit and disrupting security warnings to push for deprecation of HTTP and HTTPS with non-logged server certificates, CT logs are expected to cover a significant share of legitimate servers and traffic within a few years. As certificates include domain names, the logs can then be seen as a global view of legitimate domains of web servers. Clearly, criminals will also have to adopt

HTTPS and consequently the domains of their web servers will appear in the public logs. Alternatively, malicious web servers will be rejected in modern browsers and will clearly stand out by not using a logged certificate. This is promising because techniques like those presented in Chapters 7-9 then can be applied at a global scale, using the domains found it the CT logs.

Unfortunately, certificate issuance and domain registration are not strictly coupled. Criminals, and others, have the option to register and abuse a domain in ways that does not require a web server, meaning they will not have to get a certificate that ends up in the logs. This will however have large impact as all the many forms of abuse that involves malicious web services behind domain names would be eliminated. In addition to the decoupling of domain registration and certificate issuance, a domain can also expire and be re-registered without the CT reflecting the fact. As CT logs are expected to provide a very good coverage of domain names present on the web, it is highly relevant to gain a better understanding of this relation, e.g. through a data driven analysis. Independently of the former, it is also highly relevant to explore the potential for identifying malicious domains using CT logs. This has already been done for a combo-squatting, a specific form of abuse where the 2LD label of the targeted domain is combined with another word [109], and a free online service offers to monitor CT logs for potential phishing variant of given domains [110]. There is no apparent reason that this only can be done for phishing and combo-squatting, and the potential for the many other forms of abuse should be explored.

Chapter 12

Conclusion

As this dissertation is a collection of papers, Chapters 4-9 are self-contained and the reader is referred to the full conclusions in the last section of those chapters. The following is a summary of the two topics and of the individual papers, summarising primary contributions and the most important conclusions only.

Motivated by business benefits, corporations increase their use of and reliance on Internet-connected systems for business processes and information storage, in alignment with a trend observed throughout society. This leads to increasing risks from organised cybercrime and other threats that apply due to trust in the inherently insecure Internet. Therefore, I ask: *"How can corporations mitigate threats from the Internet, without impeding the business?"* The detailed answers are found in the papers, but in brief we have identified two topics that offer different opportunities.

1 Filtering and correlation

Deploying Intrusion Detection Systems (IDSs) can enable corporations to detect and react on incidents. To avoid wasting human resources on excess analysis and to avoid costly false negative errors due to alert fatigue, filtering and correlation is required to reduce the number of false and correlated alerts. We find that existing methods for correlation and filtering generally suffer from a high cost of deployment. This is due to reliance on feature engineering and tuning according to human expertise. We have been unable to identify any widely adopted practical methods and suggest that this can be due to the high deployment cost and/or due to existing methods not generalising well. To address this, we propose to solve the problem of correlation and filtering using Machine Learning, with the novel take that we preclude any feature engineering and ingest alerts as text without assuming anything about the format. We present two implementations, a supervised method based on Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs) and an unsupervised method based on Latent Semantic Analysis (LSA). Both implementations are evaluated on two different data sets. The conclusion on the papers in Chapters 4 and 5, is that our novel featureless approach to correlation and filtering is a promising direction for developing methods that are feasible to deploy in practical settings, where cost of deployment and maintenance must be considered.

- Chapter 4 introduces the problem, presents the LSTM RNN implementation, and concludes that it is promising for incidents of bot malware infections.
- Chapter 5 generalises the above to a general approach, introduces the LSA implementation, extends the previous data set with false alerts, includes a new publicly available data set in the evaluation, and holds a proposal for metrics that capture practical performance. The conclusion is that the proposed general approach, which precludes feature engineering and tuning, can provide value in practice, while also being feasible due to low deployment costs.

2 Domain names and DNS

On the topic of domain names and the Domain Name System (DNS) we have explored multiple directions for mitigating threats. We have contributed by formalising the concept of pre-registration detection and conducting studies that contribute with understanding of how domains can be detected under in this setting, where some notable constrains apply, while the potential gain – fully efficient prevention – can make it worthwhile. Specifically, our studies on lexical analysis of domain names in Chapter 8 and our study on technically motivated heuristics in Chapter 9 can be applied for pre-registration detection. The methods proposed in Chapters 7 and 9 are useful for establishing and evaluating ground truth data sets of malicious and benign domains names. Overall, our work within this topic has brought us closer to understand if and how we can achieve pre-registration detection of abusive domains in general, which would have significant impact on many Internet-based threats.

- Chapter 6 is a preliminary study of the problems related to the DNS and domain name abuse. We find that pre-registration detection to be a promising approach for efficient mitigation of threats involving domain name abuse. However, it has received little prior attention. We note that pre-registration detection only can rely on a subset of the data available for active domains, but has more relaxed timing requirements, posing a slightly different problem.

- Chapter 7 presents a novel method for analysing domain name blacklists. The method distinguishes itself from prior work by including all blacklisted domains in the analysis, as opposed to the common approach of relying on some oracle to provide examples of malicious domains that are used to sample blacklists. The results show that among the analysed blacklists many are static on a timescale of months, while others see some change. The conclusion is that our proposed method provided insights on the usefulness of domain name blacklist by covering complete blacklists and the temporal dimension.

- Chapter 8 presents a method for doing lexical analysis of domain names and explains how the resulting features can be combined with supervised Machine Learning methods to detect malicious domains. The conclusion is that lexical analysis of domain names is useful for detecting malicious domain names, in particular those generated by Domain Generating Algorithms (DGAs).

- Chapter 9 holds a survey of schemes and techniques used by cybercriminals, from which a set of heuristics are developed, and used to focus a manual vetting process. Using the proposed method summarised above, five malicious domains are identified by spending four man-hours on manual vetting. We conclude that analysis of criminal techniques is a valid approach to obtain heuristics that can guide manual effort to be more efficient.

The overall conclusion is that corporations have multiple options for mitigating threats, of which we explored some. Correlation and filtering can address shortcomings in existing technology, and with our proposal this appears to have become practically feasible. Many threats stem from or make use of domain names, and there is unexploited potential in detecting and reacting to abusive domains. Both topics have ample of opportunities for further improvement, as the following chapter will summarise.

Chapter 13

Future work

Each of the Chapters 4-9 provides ideas for future work and discuss them. The most relevant and interesting are summarised here together with extracts of the discussions in Chapters 10 and 11. In no particular order, some interesting directions for future work is to . . .

- Explore how performance of our approach for featureless correlation and filtering can be improved, in order to make it even more relevant in more environments. This can involve improvements on the current implementations or completely rethinking how featureless, IDS-agnostic filtering and correlation can be done.
- Explore how evaluation conditions, the training data, and IDSs affects featureless alert correlation and filtering, as this has impact on the practical relevance.
- Explore how featureless correlation and filtering applies to heterogeneous alerts, as the benefits of avoiding feature engineering and tuning appears highly relevant in an environment of mixed IDSs.
- Explore how featureless correlation and filtering apply to alerts mixed with other events. Relevant events include those that can be observed in a corporate IT infrastructure, including non-security-oriented ones. This could for instance be DNS traffic from clients, file and service access, authentication logs, and errors of any kind.
- Explore how pre-registration can be implemented and evaluate it with a fixed data set, in order to obtain controlled, repeatable performance results in a controlled setting. Given our understanding of the pre-registration setting and our experience with detection of malicious domains, it appears relevant and promising to work towards an implementation.

- Explore how pre-registration can be applied in practice at registries or registrars. Deployment in practical settings where results are processed further by humans as part of business processes can offer the option to retrain and refine the method. This can provide an understanding of how well methods can perform in practice but will be harder to reproduce as the observed system includes feedback from e.g. human analysts and cybercriminals.

- Explore how the Certificate Transparency (CT) logs of issued certificates can be used to identify malicious domains as it provides a reasonable representation of domains accessible on the web. Such global perspective has, to the best of our knowledge, not been available before the emergence of CT. One specific issue to explore is the timings and delays in CT compared to domain names abuse, which could prove a challenge.

- Explore how query-only blacklists can be analysed using the feed of domain names from CT logs. Such a feed of domains is expected to provide much greater coverage of domains on the web than to e.g. domains caught spam traps, as seen in prior work.

Overall, there are many possible directions for future work on improving Internet security for corporations, and this only includes selected new directions stemming from the two topics which have been studied in this dissertation.

This concludes the dissertation.

References

[1] Statistics Denmark, “Latest use of internet - per cent of the population (16-74 years) by type, latest use and time,” Survey results, 2018, https://www.statistikbanken.dk/BEBRIT02.

[2] “Lov om offentlig digital post,” 2012, https://www.retsinformation.dk/Forms/R0710.aspx?id=142234, Danish law.

[3] G. F. Knudsen and P. S. Rasmussen, “It-anvendelse i virksomheder 2018 – hver anden virksomhed bruger avanceret teknologi,” Survey report, 2018, https://www.dst.dk/nytpdf/27484.

[4] G. F. Knudsen, “It-anvendelse i virksomheder 2017 – virksomhedernes digitalisering,” Survey report, 2017, https://www.dst.dk/da/Statistik/Publikationer/VisPub?cid=20742#.

[5] V. Cerf and R. Kahn, “A protocol for packet network intercommunication,” *IEEE Transactions on communications*, vol. 22, no. 5, pp. 637–648, 1974, https://doi.org/10.1109/TCOM.1974.1092259.

[6] A. K. Sood, R. Bansal, and R. J. Enbody, “Cybercrime: Dissecting the state of underground enterprise,” *IEEE internet computing*, vol. 17, no. 1, pp. 60–68, 2013, https://doi.org/10.1109/MIC.2012.61.

[7] J. Lewis, “Economic impact of cybercrime - no slowing down,” Online, 2017, https://www.mcafee.com/enterprise/en-us/assets/reports/restricted/rp-economic-impact-cybercrime.pdf, Accessed Aug 31, 2018.

[8] “European cybercrime centre - ec3: About,” Online, Europol, https://www.europol.europa.eu/about-europol/european-cybercrime-centre-ec3, Accessed Oct 30, 2018.

[9] “Target breach began with contractor’s electronic billing link,” Online, Feb 06 2014, https://search.proquest.com/docview/2082102164?accountid=8144, Accessed Oct 15, 2018.

[10] M. J. Schwartz, "Target ignored data breach alarms," Online, Dark Reading, 2014, https://www.darkreading.com/attacks-and-breaches/target-ignored-data-breach-alarms/d/d-id/1127712, Accessed Oct 30, 2018.

[11] E. Basu, "Target ceo fired - can you be fired if your company is hacked?" Online, Jun 15 2014, https://www.forbes.com/sites/ericbasu/2014/06/15/target-ceo-fired-can-you-be-fired-if-your-company-is-hacked/#1fbf70bc7c9f, Accessed Sep 15, 2018.

[12] BBC, "Cyber-attack: Europol says it was unprecedented in scale," Online, May 13 2017, https://www.bbc.com/news/world-europe-39907965, Accessed Sep 15, 2018.

[13] L. Mathews, "Notpetya ransomware attack cost shipping giant maersk over $200 million," Online, Aug 16 2017, https://www.forbes.com/sites/leemathews/2017/08/16/notpetya-ransomware-attack-cost-shipping-giant-maersk-over-200-million/#141a06814f9a, Accessed Sep 15, 2018.

[14] E. Nakashima, "Russian military was behind 'notpetya' cyberattack in ukraine, cia concludes," Online, Jan 12 2018, https://www.washingtonpost.com/world/national-security/russian-military-was-behind-notpetya-cyberattack-in-ukraine-cia-concludes/2018/01/12/048d8506-f7ca-11e7-b34a-b85626af34ef_story.html?noredirect=on&utm_term=.d925adcb213c, Accessed Sep 15, 2018.

[15] E. Messmer, "Eurograbber online banking scam netted $47 million," Online, Dec 05 2012, https://www.networkworld.com/article/2161854/malware-cybercrime/-eurograbber--online-banking-scam-netted--47-million.html, Accessed Nov 8, 2018.

[16] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013, http://dx.doi.org/10.1016/j.comnet.2012.07.021.

[17] J. M. Bauer, M. J. G. van Eeten, T. Chattopadhyay, and Y. Wu, "Itu study on the financial aspects of network security: Malware and spam," International Telecommunication Union, Tech. Rep., 2008.

[18] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 635–647, https://doi.org/10.1145/1653662.1653738.

[19] M. Roesch, "Snort: Lightweight intrusion detection for networks." in *LISA*, 1999, pp. 229–238.

[20] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23, pp. 2435–2463, 1999, https://doi.org/10.1016/S1389-1286(99)00112-7.

[21] R. Vaarandi and K. Podiņš, "Network ids alert classification with frequent itemset mining and data clustering," in *Network and Service Management (CNSM), 2010 International Conference on*. IEEE, 2010, pp. 451–456, https://doi.org/10.1109/CNSM.2010.5691262.

[22] O. Dain and R. K. Cunningham, "Fusing a heterogeneous alert stream into scenarios," in *Proceedings of the 2001 ACM workshop on Data Mining for Security Applications*, vol. 13. Citeseer, 2001, pp. 1–13.

[23] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *Recent advances in intrusion detection*. Springer, 2001, pp. 54–68, https://doi.org/10.1007/3-540-45474-8_4.

[24] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 202–215.

[25] B. Zhu and A. A. Ghorbani, "Alert correlation for extracting attack strategies," *International Journal of Network Security*, vol. 3, no. 3, pp. 244–258, 2006.

[26] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Proceedings of 16th USENIX Security Symposium*. USENIX Association, 2007, p. 12.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997, https://doi.org/10.1162/neco.1997.9.8.1735.

[28] G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection." in *USENIX Security Symposium*, vol. 5, no. 2, 2008, pp. 139–154.

[29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, vol. 96, no. 34, 1996, pp. 226–231.

[30] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014, https://doi.org/10.1016/j.cose.2014.05.011.

[31] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure." in *EMNLP-CoNLL*, vol. 7, 2007, pp. 410–420.

[32] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically generating models for botnet detection," in *European symposium on research in computer security*. Springer, 2009, pp. 232–249, https://doi.org/10.1007/978-3-642-04444-1_15.

[33] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, 2008.

[34] J. Finkle, "Exclusive: Microsoft and symantec disrupt cyber crime ring," Online, Feb 2013, http://www.reuters.com/article/us-cybercrime-raid-idUSBRE91515K20130206, Accessed Nov 8, 2018.

[35] J. Demarest, "Statement before the senate judiciary committee, subcommittee on crime and terrorism: Taking down botnets," Online, Jul 2014, https://www.fbi.gov/news/testimony/taking-down-botnets, Accessed Jan 25, 2018.

[36] P. OKane, S. Sezer, and K. McLaughlin, "Obfuscation: the hidden malware," *IEEE Security & Privacy*, vol. 9, no. 5, pp. 41–47, 2011, https://doi.org/10.1109/MSP.2011.98.

[37] V. Julien, "Suricata ids," Online, 2015, http://suricata-ids.org/, Accessed Nov 8, 2018.

[38] D. M. Chess and S. R. White, "Undetectable computer viruses," in *Virus Bulletin*, 2000, pp. 107–115.

[39] G. C. Tjhai, S. M. Furnell, M. Papadaki, and N. L. Clarke, "A preliminary two-stage alarm correlation and filtering system using som neural network and k-means algorithm," *Computers & Security*, vol. 29, no. 6, pp. 712 – 723, 2010, https://doi.org/10.1016/j.cose.2010.02.001.

[40] P. Ning and D. S. Reeves, "Correlating alerts using prerequisites of intrusions: Towards reducing false alerts and uncovering high level attack strategies," DTIC Document, Tech. Rep., 2005.

[41] R. Shittu, A. Healing, R. Ghanea-Hercock, R. Bloomfield, and R. Muttukrishnan, "Outmet: A new metric for prioritising intrusion alerts using correlation and outlier analysis," in *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*. IEEE, 2014, pp. 322–330.

[42] A. Severyn and A. Moschitti, "Automatic feature engineering for answer selection and extraction," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 458–467.

[43] M. R. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. J. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang, "Brainwash: A data system for feature engineering," in *Conference on Innovative Data Systems Research (CIDR)*, 2013.

[44] U. Khurana, D. Turaga, H. Samulowitz, and S. Parthasrathy, "Cognito: Automated feature engineering for supervised learning," in *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 1304–1307.

[45] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA, 2007, p. 35.

[46] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager *et al.*, "Building watson: An overview of the deepqa project," *AI magazine*, vol. 31, no. 3, pp. 59–79, 2010, https://doi.org/10.1609/aimag.v31i3.2303.

[47] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2001, pp. 85–103.

[48] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, 2002, pp. 245–254.

[49] G. P. Spathoulas and S. K. Katsikas, "Enhancing ids performance through comprehensive alert post-processing," *Computers & Security*, vol. 37, pp. 176–196, 2013, https://doi.org/10.1016/j.cose.2013.03.005.

[50] R. Shittu, A. Healing, R. Ghanea-Hercock, R. Bloomfield, and M. Rajarajan, "Intrusion alert prioritisation and attack detection using post-correlation analysis," *Computers & Security*, vol. 50, pp. 1–15, 2015, https://doi.org/10.1016/j.cose.2014.12.003.

[51] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li, "Ai^2: Training a big data machine to defend," in *Big Data Security on Cloud*

(BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on. IEEE, 2016, pp. 49–54, https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.79.

[52] The Shmoo Group, “Defcon 8 ctf data set,” Online, 2000, http://web.archive.org/web/20080623064229/http://cctf.shmoo.com:80/data/cctf-defcon8/, Accessed Nov 8, 2018.

[53] Cyber Security and Information Sciences Group, MIT Lincoln Laboratory, “2000 darpa intrusion detection evaluation data set,” Online, 2000, https://www.ll.mit.edu/ideval/data/2000data.html, Accessed Nov 8, 2018.

[54] ——, “1999 darpa intrusion detection evaluation data set,” Online, 1999, https://www.ll.mit.edu/ideval/data/1999data.html, Accessed Nov 8, 2018.

[55] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[56] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in *The International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116.

[57] E. Kidmose, M. Stevanovic, and J. M. Pedersen, “Correlating intrusion detection alerts on bot malware infections using neural network,” in *International Conference on Cyber Security and Protection of Digital Services (Cyber Security 2016)*, C-MRIC. IEEE, 2016, pp. 195–211.

[58] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[59] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural network design*. PWS publishing company Boston, 1996, vol. 20.

[60] W. S. Sarle, “Neural network faq,” Online, 1997, http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html, Accessed Nov 3, 2016.

[61] J. Heaton, *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.

[62] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, “Detecting malware domains at the upper dns hierarchy.” in *USENIX security symposium*, vol. 11, 2011, pp. 1–16.

[63] G. C. M. Moura, M. Müller, M. Davids, M. Wullink, and C. Hesselman, “Domain names abuse and tlds: from monetization towards mitigation,” in *3rd IEEE/IFIP Workshop on Security for Emerging Distributed Network Technologies (DISSECT 2017), co-located with IFIP/IEEE International Symposium on Integrated Network Management (IM 2017)*, May 2017, https://doi.org/10.23919/INM.2017.7987441.

[64] M. Felegyhazi, C. Kreibich, and V. Paxson, “On the potential of proactive domain blacklisting,” in *3rd USENIX workshop on large-scale exploits and emergent threats (LEET ’10)*, vol. 10, 2010, pp. 6–6.

[65] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck, “Understanding the domain registration behavior of spammers,” in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 63–76, https://doi.org/10.1145/2504730.2504753.

[66] S. Bortzmeyer, “DNS Query Name Minimisation to Improve Privacy,” RFC 7816, Mar. 2016, https://doi.org/10.17487/RFC7816.

[67] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, “Building a dynamic reputation system for dns.” in *USENIX security symposium*, 2010, pp. 273–290.

[68] H. Choi and H. Lee, “Identifying botnets by capturing group activities in dns traffic,” *Computer Networks*, vol. 56, no. 1, pp. 20–33, 2012, https://doi.org/10.1016/j.comnet.2011.07.018.

[69] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, “Exposure: a passive dns analysis service to detect and report malicious domains,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, p. 14, 2014, https://doi.org/10.1145/2584679.

[70] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, “Phoenix: Dga-based botnet tracking and intelligence,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014, pp. 192–211, https://doi.org/10.1007/978-3-319-08509-8_11.

[71] P. Luo, R. Torres, Z.-L. Zhang, S. Saha, S.-J. Lee, A. Nucci, and M. Mellia, “Leveraging client-side dns failure patterns to identify malicious behaviors,” in *Communications and Network Security (CNS), 2015 IEEE Conference on*. IEEE, 2015, pp. 406–414, https://doi.org/10.1109/CNS.2015.7346852.

[72] R. Sharifnya and M. Abadi, “Dfbotkiller: Domain-flux botnet detection based on the history of group activities and failures in dns traffic,” *Digital Investigation*, vol. 12, pp. 15–26, 2015, https://doi.org/10.1016/j.diin.2014.11.001.

[73] S. Hao, N. Feamster, and R. Pandrangi, “Monitoring the initial dns behavior of malicious domains,” in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 269–278, https://doi.org/10.1145/2068816.2068842.

[74] R. Perdisci, I. Corona, and G. Giacinto, “Early detection of malicious flux networks via large-scale passive dns traffic analysis,” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 5, pp. 714–726, 2012, https://doi.org/10.1109/TDSC.2012.35.

[75] S. Hao, A. Kantchelian, B. Miller, V. Paxson, and N. Feamster, “Predator: proactive recognition and elimination of domain abuse at time-of-registration,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1568–1579, https://doi.org/10.1145/2976749.2978317.

[76] T. Vissers, J. Spooren, P. Agten, D. Jumpertz, P. Janssen, M. Van Wesemael, F. Piessens, W. Joosen, and L. Desmet, “Exploring the ecosystem of malicious domain registrations in the. eu tld,” in *20th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2017)*, 2017, pp. 472–493, https://doi.org/10.1007/978-3-319-66332-6_21.

[77] M. Stevanovic, J. M. Pedersen, A. D’Alconzo, S. Ruehrup, and A. Berger, “On the ground truth problem of malicious dns traffic analysis,” *computers & security*, vol. 55, pp. 142–158, 2015.

[78] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, “An empirical analysis of phishing blacklists,” in *Sixth conference on email and anti-spam (CEAS)*, 2009.

[79] G. C. Moura, M. Müller, M. Wullink, and C. Hesselman, “ndews: A new domains early warning system for tlds,” in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 1061–1066.

[80] Palo Alto Networks, Inc., “Minemeld threat intelligence sharing,” Online, https://github.com/PaloAltoNetworks/minemeld/wiki, Accessed Nov 9, 2018.

[81] CSIS Security Group A/S, Online, https://www.csis.dk/prevent-secure-dns/, Accessed Nov 8, 2018.

[82] W. Wang and K. Shirley, "Breaking bad: Detecting malicious domains using word segmentation," in *In Proceedings of the 9th Workshop on Web 2.0 Security and Privacy (W2SP)*, 2015.

[83] P. Zhang, T. Liu, Y. Zhang, J. Ya, J. Shi, and Y. Wang, "Domain watcher: Detecting malicious domains based on local and global textual features," *Procedia Computer Science*, vol. 108, pp. 2408–2412, 2017, https://doi.org/10.1016/j.procs.2017.05.204.

[84] M. Stevanovic, J. M. Pedersen, A. D'Alconzo, S. Ruehrup, and A. Berger, "On the ground truth problem of malicious dns traffic analysis," *computers & security*, vol. 55, pp. 142–158, 2015.

[85] S. S. Hansen, T. M. T. Larsen, M. Stevanovic, and J. M. Pedersen, "An approach for detection and family classification of malware based on behavioral analysis," in *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2016.

[86] R.-S. Pirscoveanu, S. S. Hansen, T. M. T. Larsen, M. Stevanovic, J. M. Pedersen, and A. Czech, "Analysis of malware behavior: Type classification using machine learning," in *International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. IEEE, 2015.

[87] M. McGuire, "Into the web of profit," Online, Bromium, Inc., 2018, https://learn.bromium.com/rprt-web-of-profit.html, Accessed Sep 18, 2018.

[88] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamalytics: An empirical analysis of spam marketing conversion," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 3–14.

[89] A. Abbasi and H. Chen, "A comparison of tools for detecting fake websites," *Computer*, no. 10, pp. 78–86, 2009, https://doi.org/10.1109/MC.2009.306.

[90] T. Vissers, W. Joosen, and N. Nikiforakis, "Parking sensors: Analyzing and detecting parked domains," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society, 2015, pp. 53–53, https://doi.org/10.14722/ndss.2015.23053.

[91] Z. Gyöngyi and H. Garcia-Molina, "Spam: It's not just for inboxes anymore," *IEEE Computer*, vol. 38, no. 10, pp. 28–34, 2005, https://doi.org/10.1109/MC.2005.352.

[92] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis, "Domain-z: 28 registrations later measuring the exploitation of residual trust in domains," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 691–706.

[93] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks." in *NDSS*, 2008.

[94] J. Nazario and T. Holz, "As the net churns: Fast-flux botnet observations," in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*. IEEE, 2008, pp. 24–31.

[95] P. Porras, H. Saidi, and V. Yegneswaran, "An analysis of conficker's logic and rendezvous points," Computer Science Laboratory, SRI International, Tech. Rep, Tech. Rep., 2009.

[96] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu *et al.*, "Click trajectories: End-to-end analysis of the spam value chain," in *2011 IEEE symposium on security and privacy*. IEEE, 2011, pp. 431–446, https://doi.org/10.1109/SP.2011.24.

[97] E. Kidmose, K. Gausel, S. Brandbyge, and J. M. Pedersen, "Assessing usefulness of blacklists without the ground truth," in *10th International Conference on Image Processing and Communications*, 2018.

[98] T. Moore and B. Edelman, "Measuring the perpetrators and funders of typosquatting," in *International Conference on Financial Cryptography and Data Security*. Springer, 2010, pp. 175–191, https://doi.org/10.1007/978-3-642-14577-3_15.

[99] T. Lauinger, A. Chaabane, A. S. Buyukkayhan, K. Onarlioglu, and W. Robertson, "Game of registrars: An empirical analysis of post-expiration domain name takeovers," in *Proceedings of the USENIX Security Symposium*, 2017.

[100] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 2016, pp. 2786–2792.

[101] M. Wander, "Measurement survey of server-side dnssec adoption," in *Network Traffic Measurement and Analysis Conference (TMA), 2017*. IEEE, 2017, pp. 1–9.

[102] D. Atkins and R. Austein, "Threat Analysis of the Domain Name System (DNS)," RFC 3833, 2004, https://doi.org/10.17487/RFC3833.

[103] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, "Measuring https adoption on the web," in *26th USENIX Security Symposium*, 2017, pp. 1323–1338.

[104] P. E. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, Oct. 2018, https://doi.org/10.17487/RFC8484.

[105] P. McManus, "Improving dns privacy in firefox," Online, 2018, online: https://blog.nightly.mozilla.org/2018/06/01/improving-dns-privacy-in-firefox/, Accessed Nov 2, 2018.

[106] Google Developers API, "Dns-over-https," Online, 2018, https://developers.google.com/speed/public-dns/docs/dns-over-https, Accessed Nov 8, 2018.

[107] J. Hodges, C. Jackson, and A. Barth, "HTTP Strict Transport Security (HSTS)," RFC 6797, Nov. 2012, https://doi.org/10.17487/RFC6797.

[108] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962, Jun. 2013, https://doi.org/10.17487/RFC6962.

[109] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gomez, N. Pitropakis, N. Nikiforakis, and M. Antonakakis, "Hiding in plain sight: a longitudinal study of combosquatting abuse," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 569–586, https://doi.org/10.1145/3133956.3134002.

[110] D. Huang, B. Niemczura, and A. Xu, "Detecting phishing domains using certificate transparency," Online, 2018, https://www.facebook.com/notes/protect-the-graph/detecting-phishing-domains-using-certificate-transparency/2037453483161459/, Accessed Nov 4, 2018.

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-356-3

AALBORG UNIVERSITY PRESS