Network Measurements and Security

Untangling the Web of Domain Names, Certificates, and Mobile Apps

Hageman, Kaspar David

DOI (link to publication from Publisher):
10.54337/aau461887228

Publication date:
2021

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

NETWORK MEASUREMENTS AND SECURITY

UNTANGLING THE WEB OF DOMAIN NAMES, CERTIFICATES, AND MOBILE APPS

BY
KASPAR DAVID HAGEMAN

DISSERTATION SUBMITTED 2021

AALBORG UNIVERSITY
DENMARK

NETWORK MEASUREMENTS AND SECURITY

Untangling the Web of Domain Names, Certificates, and Mobile Apps

Ph.D. Dissertation

Kaspar David Hageman

Dissertation submitted October, 2021

Dissertation submitted:	October, 2021
PhD supervisor:	Prof. Jens Myrup Pedersen Department of Electronic Systems Aalborg University
PhD Co-supervisor:	Assoc. Prof. René Rydhof Hansen Department of Computer Science Aalborg University
PhD committee:	Associate Professor Sokol Kosta (chairman) Aalborg University Associate Professor Maciej Korczynski Université Grenoble Alpes Professor Michał Choraś University of Science and Technology (UTP) in Bydgoszcz
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Electronic Systems

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-985-1

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2022

Abstract

The Internet plays a crucial role in many facets of our everyday lives and has attracted the attention of malicious actors. This dissertation explores the central and crucial role that domain names and digital certificates play in our online interactions, and how this role can be used, or is lacking, for securing the Internet against these actors. It also provides a significant contribution towards generating a stronger interest in the field of information security in students, and thereby to a better education of the information security industry.

Firstly, it investigates how the relationship between domain names and digital certificates can be used for the detection of phishing attacks at an early stage. By producing a framework for collecting domains and certificates at scale – and allowing for convenient post-analysis of this data – it provides a foundation for further domain name-based research. Differences between certificates issued for phishing domains and other domains are identified, showing (among others) that these certificates have both a relationship to phishing domains and benign domains. This 'ambiguous' relationship between certificates and domains is then further explored by training a machine learning model for identifying the likelihood of certificates being used in phishing attacks. This dissertation also provides insight into network traffic that, from a domain name perspective, is invisible, and concludes that domain name (and certificate) based security is unable to block a significant fraction of network traffic.

Secondly, the dissertation addresses the specific role of digital certificates in Android, and the implications this role has on the trustworthiness of published apps. Whereas certificates usually provide a guarantee about the identity of the certificate owner in other systems, this is not the case for Android. As a result, end users, regulators and security researchers cannot rely on this certificate information for any meaningful identification of the developer behind an app.

Lastly, this thesis contributes with a novel virtualized training platform for security education, *Haaukins*, that fulfills a need that was previously not

met: the ability to quickly set up short penetration testing training sessions for inexperienced students. The major design considerations of this platform have been documented along the way and are included in this dissertation.

Resumé

Internettet spiller en afgørende rolle i mange aspekter af vores hverdag og har tiltrukket opmærksomhed fra ondsindede aktører. Denne afhandling undersøger den centrale og afgørende rolle, som domænenavne og digitale certifikater spiller i vores online-interaktioner, og hvordan denne rolle kan bruges, eller er utilstrækkelig, til at sikre Internettet mod disse aktører. Afhandlingen yder også et betydeligt bidrag til at skabe en større interesse for informationssikkerhed hos studerende og derved til et højere uddannelsesniveau i informationssikkerhedsindustrien.

For det første undersøger afhandlingen, hvordan forholdet mellem domænenavne og digitale certifikater kan bruges til at opdage phishing-angreb på et tidligt tidspunkt. Ved at producere et framework til indsamling af domæner og certifikater i stor skala — og give mulighed for praktisk efteranalyse af disse data — dannes grundlaget for yderligere domænenavnsbaseret forskning. Herefter identificeres forskelle mellem certifikater udstedt til phishing-domæner og certifikater udstedt til andre domæner, hvilket blandt andet viser, at disse certifikater både har en relation til phishing-domæner og godartede domæner. Dette 'tvetydige forhold mellem certifikater og domæner udforskes derefter yderligere ved at træne en maskinlæringsmodel til at identificere sandsynligheden for, at certifikater bruges i phishing-angreb. Resultaterne fra afhandlingen giver også indsigt i netværkstrafik, der fra et domænenavnsperspektiv er usynligt, og konkluderer, at domænenavns-/certifikatbaseret sikkerhed *ikke* er i stand til at blokere ondsindet netværkstrafik i betydeligt omfang.

For det andet omhandler afhandlingen den rolle, som digitale certifikater spiller i Android, og hvilke konsekvenser disse har for pålideligheden af publicerede apps. Mens certifikater normalt giver en garanti for identiteten af certifikatindehaveren i andre systemer, er dette ikke tilfældet for Android. Afhandlingen viser yderligere, at tilsynsmyndigheder og sikkerhedsforskere som følge heraf ikke kan bruge disse certifikatoplysninger til meningsfyldt at

identificere udviklederen af en app.

Endelig har dette ph.d. projekt, bidraget til en ny virtualiseret træningsplatform til IT-sikkerhedsuddannelse, kaldet *Haaukins*, der opfylder et behov, der tidligere ikke var opfyldt: muligheden for hurtigt at oprette korte penetrationstesttræningssessioner for studerende uden tidligere sikkerheds-viden/-erfaring. De overordnede designovervejelser for denne platform er blevet dokumenteret undervejs og indgår i denne behandling.

Curriculum Vitae

Kaspar Hageman obtained his BSc in Computer Science in 2013 and his MSc in Telematics in 2015 at the University of Twente, the Netherlands. In his master thesis, he investigated the impact of elliptic curve cryptographic algorithms as a replacement for RSA on the performance of DNSSEC-enabled resolvers. Afterwards, he spent a year and a half as a software engineer at Nedap Healthcare.

Since 2017, he has been working as a Ph.D. student at the Department of Electronic Systems at Aalborg University. He has been part of the 'SecDNS' project during this time, and he has contributed to eight papers, of which three have currently been published. In addition, he contributed to *Haaukins*, a popular security education platform in Denmark. His Ph.D. project included a three-month visit at IMDEA Networks in Madrid, Spain.

His research interests include network security, machine learning, and protocol standards.

Preface

This Ph.D. dissertation is the result of a four-year effort and is organized as a collection of papers. The dissertation is composed of three parts, with Part I providing the motivation for, the background of, and the context of the main contributions of the dissertation. Part II consists of the papers written during the duration of the Ph.D. project, and Part III reflects on this work in a discussion and conclusion.

The dissertation is composed of seven papers, four of which I am the main author, and the remaining three are co-authored by me. The contribution to each paper is outlined in co-author statements, distributed to the PhD committee prior to the defense. The papers, as they appear in this dissertation, are unchanged from their published and submitted form beside the layout and minor spelling corrections.

List of publications The following papers comprise the research contributions made during the Ph.D. project. At the time of the dissertation submission, three of these papers have been published, one is accepted but unpublished (*), two are currently in review (†), one is in preparation for submission (‡), and one is not included in this dissertation (§).

1. T.K. Panum, K. Hageman, J.M. Pedersen, R.H. Hansen, "Haaukins: A Highly Accessible and Automated Virtualization Platform for Security Education", in *"The 19th IEEE International Conference on Advanced Learning Technologies (ICALT)"*, 2019

2. T.K. Panum, K. Hageman, R.H. Hansen, J.M. Pedersen, "Towards Adversarial Phishing Detection", in *"The 13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)"*, 2020§

3. G.M. Mennecozzi, K. Hageman, T.K. Panum, A. Türkmen, RV. Mahmoud, J.M. Pedersen, "Bridging the Gap: Adapting a Security Education

Platform to a New Audience", in *"The IEEE Global Engineering Education Conference (EDUCON 2021)"*, 2021

4. K. Hageman, E. Kidmose, R.H. Hansen, J.M. Pedersen, "Can a TLS Certificate Be Phishy?", in *"The 18th International Conference on Security and Cryptography (SECRYPT 2021)"*, 2021
5. K. Hageman, R.H. Hansen, J.M. Pedersen, "Gollector: Measuring Domain Name Dark Matter from Different Vantage Points", in *"The 26th Nordic Conference on Secure IT Systems (NordSec 2021)"*, 2021*
6. S. Homayoun, K.Hageman, S. Afzal-Houshmand, C.D. Jensen, J.M. Pedersen, "Detecting Ambiguous Phishing Certificates using Machine Learning", in *"The 36th International Conference on Information Networking (ICOIN 2022)"*, 2021†
7. K. Hageman, E. Kidmose, R.H. Hansen, J.M. Pedersen, "Understanding the Challenges of Blocking Unnamed Traffic", in *"IEEE/IFIP Network Operations and Management Symposium (NOMS 2022)"*, 2021†
8. K. Hageman, Á. Feal, J. Gamba, A. Girish, J. Bleier, M. Lindorfer, J. Tapiador, N. Vallina-Rodriguez, "Kaonashi: On Developer Attribution Challenges in Android App Marketplaces", 2021‡ [1]

[1] This paper was previously submitted and rejected at the Internet Measurement Conference (IMC 2021), and is now in preparation for resubmission to another conference.

Acknowledgments

No decent Ph.D. dissertation is ever written in a complete vacuum, and in my case, this is no different. Many individuals directly helped me out with my work, engaged in interesting discussions, or gave me the motivation to continue the process, and without whom I would definitely not have finished this dissertation.

Most importantly, I must express my immense thanks to my two supervisors, Jens and René. It is hard to believe how Jens and I started in 2017 on a project related to DNS, and how it evolved into a dissertation that is so much more than that. I am not sure that you, René, understand how much our discussions over the years have not only helped improve the quality of the research and papers but also raised the confidence I ended up having in my own work. In the moments where I felt like giving up, you helped me get up and steer my research in an ever-so-slightly different direction that made me feel like I was coming up with significant contributions to the research field. This Ph.D. dissertation would not have been possible without the funding from the Innovation Fund Denmark and the Danish Industry Foundation, many thanks to them as well.

I cannot write this part without thanking the fellow Ph.D. students, postdocs, and professors at our security research group: Egon, Vlad, Gian, Martin, Tatiana, and Rasmus. Thank you for the collaborations that we set up, the research projects, and – last but not least – the many lunches that we shared together, I am going to miss them. A specific mention goes out to Thomas, with who I not only shared nearly my entire Ph.D. project, but also all my frustrations, celebrations, and everything in between. I have good memories of the first few Haaukins demos we gave that barely worked, and the afternoons we spent just chatting about research and IT in general, distracting ourselves from the next deadline. Without any doubt, I would have left Denmark empty-handed if it wasn't for us sharing an office for more than three years. Furthermore, I am grateful for Finn's assistance in the data collection from the AAU network up to the last moments.

And of course, there is a second research group I would like to thank, IMDEA Networks that was so kind to host me for three months. Thank you, Narseo, for the opportunity to work together with you, and Julien and Álvaro for making me feel at home in Madrid. It's a shame the COVID situation kept me from visiting again! We did some great work together, and it has definitely been a rich addition to my Ph.D.

In 2017, I made the decision to spend the next three years in a country I knew no one, and I am grateful for the friends and family I still have back in the Netherlands. A massive thanks goes out to David, Bas, Joche, and Robin for keeping me sane, especially during the COVID lockdown period. The support from my family – my mom, my dad, Jolijn and Tijmen – made me feel like we were not in different countries at all, thank you!

I have been told in several instances that a Ph.D. is supposed to be the best period of your life. During some periods of this Ph.D. project, this was definitely not the case, and I would say it was in fact the opposite of that. However, in the situations it *was* the case, there was generally always one person there, Lisha. Thank you so much for sticking around when I was in a bad mood and grumpy, and for celebrating the good things together; I am looking forwards to both in the future!

Kaspar Hageman, October 2021

Contents

Part I

Introduction

Chapter 1

Motivation

The security community has engaged with malicious actors in a decade-long arms race to defend Internet users against numerous types of attacks. Defense mechanisms that are being put in place may become obsolete if malicious actors find their way around them, thus raising the need for novel defense and mitigation strategies. Furthermore, new developments on the Internet change the landscape continuously, through the adaptation of new technical proposals or new trends that change how users interact with the Internet. The implication of this ongoing arms race and changing landscape is that both Danish [1] and international [2] end users and businesses remain vulnerable for digital attacks.

Malicious activities on the Internet are diverse in their methodology, type of victims, and goals. The purpose for examples can range from financial gain [3], to political espionage [4] or digital warfare [5], and their methodologies can range from obtaining sensitive data to bringing down a particular service [6]. One class of attacks relies on malicious software or malware, to be installed on a victim's computer system. There are various ways for this software to infect a computer system, such as worms (i.e., software that propagates itself across a network) or drive-by-downloads (i.e., the victim is unaware that the software is downloaded and installed [7]) Another class of attacks operates fully remotely, such as phishing (i.e., the victim is deceived into disclosing sensitive information [8]) or denial of service attacks (i.e., the operation of a legitimate service is disrupted). The remainder of this chapter highlights several impactful problem domains with Internet abuse.

Phishing This attack is characterized by the persuasion of victims into performing actions through impersonation and deception [8]. Victims are typically contacted via automated means, such as email or text messages. The general purpose is to convince the victim to disclose sensitive information – such as login credentials or credit card information – by impersonating a legitimate entity. It is common for the attacker to refer the victim to a website, that looks indistinguishable from a legitimate one, intended to increase the perceived trustworthiness of the site.

Since its conception in the 90s, phishing attacks have received scrutiny from the security community, and a variety of mitigation techniques have been proposed, including browser-based protection system [9] and phishing email detection mechanisms [10, 11]. This Anti-Phishing Working Group (AWPG) reports hundreds of thousands of such websites monthly [12], showing that the problem is still rampant. According to an estimation, the cost of conducting a phishing attack targeting a hundred thousand victims is less than $30 [13]. This, in combination with access to phishing kits – off-the-shelf software that can be used to easily set up a phishing attack with limited technical knowledge – makes the barrier to entry low [14]. Several studies have shown the lack of awareness in users [15, 16], illustrating that phishing requires both a technical and human solution.

Entity attribution Phishing attacks can be successful due to the lack of control over the entities that are abused in phishing attacks. Registering domain names similar to other legitimate brands, obtaining digital certificates for encrypting traffic to these domains, and deploying websites that are indistinguishable from other websites are not only possible but trivial to execute. There are technical solutions that prevent attackers from spoofing certain entities, such as digital certificates that ensure web browsers are connecting to the correct web server, and the public key infrastructures behind these certificates that prevent attackers to obtain false certificates. However, these solutions apply to entities that are owned by a legitimate organization, and not to entities owned by the malicious actors that look similar to the legitimate ones. Another ecosystem that suffers from similar problems is the mobile operating system Android, which has seen large-scale exploitation of its users through the distribution of cloned apps. In both cases, users would benefit from better attribution, or the identification of the owner behind an entity.

Security education The fight against online crime requires a well-educated workforce that has a technical understanding of the technology that they are protecting. This does not only require an understanding of the types of attacks that are currently known, but also a deep understanding of the systems under attack so that new attacks can be quickly recognized and understood, and novel attacks can be anticipated. Within Denmark, there are only few opportunities for students to pursue a higher education in information security [17, 18]. Security is not always a core priority within organizations and is often treated as an afterthought. According to [1], Danish employees mention a lack of security training in their company, predominantly (1) a lack of awareness of available training, (2) a lack of awareness that there are opportunities for training, and (3) a lack of awareness that they need training. As such, there is much to be gained in Danish education and training facilities for producing a better-educated workforce.

1.1 Problem statement

A common element in phishing attacks and entity attribution is the involvement of both domain names and digital certificates. These two entities are central components of not only phishing attacks, but network traffic in general. As such, we concern ourselves with the following task:

> "To understand and leverage the role of domain names and
> digital certificates in Internet security"

This dissertation addresses several of the challenges that are laid out: the role of domains names in malicious activities, the importance of digital certificates in phishing attacks, the lack of attribution in the Android ecosystem, and the improvement of security education. The contribution to security education here is a secondary one.

Chapter 2

Background

This chapter describes the relevant background that is required to understand the individual papers and provides a context in which the research described in these papers was conducted.

2.1 Traffic measurements

To identify, prevent and mitigate malicious software, security researchers commonly look for evidence that something malicious is occurring. One approach is to monitor state changes of an individual computer system through anti-virus software, finding software that interacts in suspicious ways with the system. Alternatively, network traffic can be *passively* monitored with the same purpose of identifying suspicious behavior. The latter has the significant advantage of covering a much larger set of end user devices. Lastly, security researchers can *actively* probe remote computer systems for suspicious content that they may serve. This also includes consulting third-party applications, such as search engines, or third-party APIs. This dissertation is primarily concerned with the latter two, passive and active traffic measurements.

Figure 2.1 shows a typical communication pattern between an end user and systems on the Internet – in this case a user visiting a website. Firstly, the end user's system translates the website address to an IP address, using the Domain Name System (DNS) protocol (Section 2.1.1). The system then establishes both a Transmission Control Protocol (TCP) connection and a Transport Layer Security (TLS) connection for encrypting the traffic (Section 2.1.3) through a handshake process, which is followed by the actual data exchange. The data

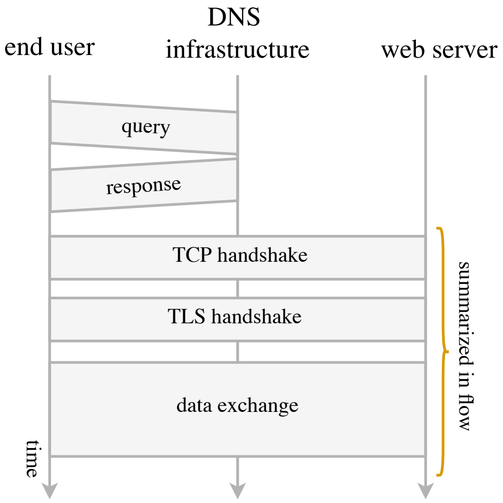


Figure 2.1: Typical overview of the traffic generated by an end user connecting to a website running HTTPS

that is exchanged typically consists of various Hypertext Transfer Protocol (HTTP) requests and responses. In each of these steps, any party that operates in the path between the end user or the other servers can intervene and interfere with this pattern. Examples include DNS servers that respond with a false IP address [19], or a browser that blocks a user from visiting a malicious website. The steps in the figure prior to the data exchange may be absent. Some types of traffic that do not rely on DNS traffic include peer-to-peer protocols [20, Chapter 5] or worms propagating on a local network [21–23]. The User Datagram Protocol (UDP), the major alternative transport layer protocol to TCP, operates without a handshake, and web servers can serve their data without TLS in case the traffic is not to be encrypted.

2.1.1 DNS

The Domain Name System (DNS) provides the translation between (among others) the domain name space and the IP name space [24]. Domain names are a way to name resources across networks and protocols, and the DNS facilitates a distributed method to translate them to IP addresses, through distributing the mappings between domain names and IP addresses (i.e., DNS resource records) across millions of authoritative name servers. The full domain name space is divided into individual zones, or parts of the domain name space that is operated by a single organization. Systems can query name servers in a query-response fashion, obtaining either a set of valid resource records, or a referral to another name servers that is authoritative for the requested data. Commonly, larger organizations, Internet Service Providers (ISPs) and other service providers operate *recursive* name servers, who conduct DNS queries on behalf of their clients, acting as a proxy and caching layer between end devices and authoritative name servers. The DNS was designed without security or privacy considerations, and as such intercepting and monitoring DNS traffic is trivial and provides strong insight into the intention of an application. As illustrated in Figure 2.1, DNS is commonly the first initiated step in the communication between a end user's device and a server. As a result, DNS measurements have been a massive asset for security researchers, demonstrated by the heavy use in it against botnet command and control communications [25, 26], spamming [27] or malicious domains in general [28, 29].

Throughout the years, new proposals introduced encryption or message signing to add security guarantees to the protocol. DNS over HTTPS (DoH) is one of the more impactful ones, as its quick adoption by DNS service providers and vendors [30–33] has led to a shift from unencrypted DNS traffic towards local resolvers to encrypted DNS traffic to a small set of key DNS providers. Other proposals include DNS over TLS (DoT) [34], DNSCrypt [35] and DNSSEC [36]. The implication of the adoption of these systems is that DNS traffic becomes harder to passively observe, which hits both malicious (e.g., malicious Internet service providers) and benign observers (e.g., security researchers).

2.1.2 Flow monitoring

DNS is a highly valuable source of information, but it represents only a small fraction of the total traffic that is generated in a typical network, both in packet volume and in byte size. The sheer volume of network traffic makes monitoring all of it highly costly, if done without a method to reduce the data to analyze. NetFlow was devised as a method by Cisco to compress characteristics of the network traffic in a reduced form and formed the basis for the IETF's IPFIX standard [37]. A flow is defined as a "sets of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties" [38]. These common properties typically consist of a source and destination IP address, the source and destination port number, and the layer 4 protocol (e.g., UDP or TCP). These properties are enhanced with measured properties of the flow, such as packet and byte counters. The data exchange in Figure 2.1 between the end user and web server would be exported as a single flow. In flow monitoring, a *flow exporter* is typically responsible for collecting raw packets, aggregating them into flows, and exporting a (potentially sampled) set of flows. By both summarizing raw packets into a small number of fields and aggressively sampling the flows (for instance exporting each individual flow with a 1% probability), network traffic can be reduced to only a fraction of its original size. A flow exporter maintains an internal cache of flows and must decide when to clear an entry from the cache and export the entry. Typically, entries expire due to timeouts (e.g., no packet belonging to a flow has been monitored for a period of time), resource constraints (e.g., a cache can only contain a maximum number of active flows), or natural expirations (i.e., a TCP packet that terminates a connection) [39]. Although flow monitoring loses a large amount of detailed information that is contained in the payload and headers of packets, it does capture the general traffic characteristics. These characteristics have proven to be highly useful for combating attacks such as worm propagation [21–23, 40–42].

2.1.3 Digital certificates

The data that is exchanged between browsers and web servers often contain sensitive information, such as user names and passwords from the client's side, or personal information embedded in documents from the web server's side. To prevent malicious actors from intercepting the traffic and seeing the content of these files, the data is encrypted during transportation. The protocol scheme that facilitates this mechanism is Transport Layer Security. TLS encrypts and decrypts traffic using a symmetric key that is being established under an asymmetric key exchange. The server provides a *digital certificate* to the client, in the X.509 certificate standard format [43]. This certificate represents a binding between a public key (whose corresponding private key is owned by the web server) and a set of domain names for which the certificate is valid. The embedded domain names, known as Subject Alternative Names (SANs) can exist in a wildcard format – denoted by a star (*) – that indicates that the SAN can match *any* value for the label with the star. When presented with a certificate, a client must verify that the host name of the server, that they attempt to connect to, is one of the embedded domains in the certificate.

Furthermore, the certificate must have been cryptographically signed by trusted third-party organizations, who verified the identity behind the certificate owner. By constructing a chain of trust (with each shackle in the chain of certificates signing the next certificate in the chain), a client can verify that the certificate originates from a trusted third-party. Devices, operating systems, and browsers come pre-installed with their own list of *root certificates*, or the first certificate in the certificate chain [44]. The trusted third parties are referred to as Certificate Authorities (CAs) who own these root certificates and issue certificates for customers that can prove ownership of the SANs that are to be covered by the newly-signed certificate. Each certificate has a time window (defined by a `Not Before` and `Not After` field) outside which the certificate should always be rejected by a client. Further, the X.509 standard allows a set of extensions to be included, which are used for mechanisms such as informing the client whether the certificate can be used as a root certificate or not, or the location of revocation lists (i.e., a list of certificate fingerprints that should be rejected by any client). Although heavily used in the HTTP ecosystem, these digital certificates are used across different systems as well, such as in email communication [45, 46], and encryption of Virtual Private Network (VPN) traffic [47].

2.2 Mobile apps and Android

The mobile operating system market is dominated by two players, with Google's Android and Apple's iOS having 72.7% and 26.4% market share respectively [48]. Mobile devices come with the functionality (in terms of hardware and software) for collecting highly sensitive information about an individual [49, 50], making security and privacy a highly important factor. In contrast to the desktop computer market, the distribution of software for mobile devices is scattered across a set of market places, from which end users can download new applications, or *apps*. The two dominant mobile operating systems, Apple's iOS and Google's Android, both provide a default market from which to download apps. The iOS market is more tightly controlled, with the distribution of apps being centralized in their own market, and with stronger requirements put on the developer prior to releasing an app [51]. Whereas in Android, the requirements for releasing an app are more relaxed; the registration process for new developers is relatively simple [52] and apps can be distributed from multiple markets.

A large factor for the lack of control in the Android ecosystem compared to iOS is the fact that iOS apps must be signed by a certificate issued by Apple [53], whereas Android apps can be signed by a self-signed certificate [54]. Android applications are distributed as *.apk* packages, and a certificate signs the content in the package; the certificate merely provides integrity (i.e., the guarantee that the package has not been tampered with). While certificates are used within

TLS as a method of authenticating the server, certificates as being used in Android do not provide such a guarantee at all. Instead, they are used in several mechanisms with Android devices, such as enabling the release of updates for apps signed by the same certificate or sharing data between apps signed by the same certificate. From the perspective of an Android device, a newly installed app can be considered author-less, as the only proof-of-identity in the package (i.e., the certificate) is completely fabricated and unverified. As a result of these design choices, the Android ecosystem has seen numerous cases of malicious (cloned) apps being published and its detection remains a relevant field of research [55].

2.3 Capture the flag

Capture the flags (CTFs) are a form of competitive events in which teams compete in exploiting vulnerable computer systems [56]. By finding so-called *flags*, or random strings of text deliberately placed there by the organizers of the event, participants score points, striving to retrieve the most points of all teams. The CTF community grew from hacker communities in the 90s, with DEFCON's prestigious competition being the longest-running annual CTF [57]. These competitions started as an opportunity for different hacker groups to demonstrate their abilities and remain highly popular to this date [58]. There are different styles of CTFs, most notably *Jeopardy* and *Attack and Defense*. In the former, teams have limited time to exploit a set of vulnerabilities, and each exploit results in a number of points, generally according to the difficulty of the exploit. In contrast, *Attack and Defense* CTFs require both teams to both exploit the machines operated by their rival teams, while simultaneously patching their own systems to prevent other teams from doing the same. Capture the flag events are typically conducted in an isolated environment, with participants exploiting systems that are not providing legitimate services to other users, to ensure that exploiting vulnerabilities will not harm non-participants. Education programs have started to embrace CTFs as an education tool, due to the practical experience it gives to students and their competitive element. These academic CTFs serve as a tool for students to practice their penetration testing skills in a safe environment, but also to instill an interest in younger students for the field of information security [59]. CTFs are being used as an education tool ranging from high school students [59] to teaching university courses [60] and the US military [61].

Chapter 3

Contributions

The contributions of this dissertation can be loosely divided across three distinct areas: domain name and certificates, Android attribution, and security education. Within the area, there is a strong cohesion between the individual papers, and for each of these areas, this chapter highlights the main contributions that were made in the individual papers.

3.1 Domain names and certificates

Domain names, and the protocols closely related to them, have played a significant role in the security of the Internet. This part of the dissertation consists of four papers that contribute to this field. Rather than considering domain names isolated entities that exist in a vacuum, the papers consider the relationship between domain and digital certificates specifically, since the encryption that these certificates provide has been playing a larger role in the security of Internet users recently.

Paper A – ***Gollector: Measuring Domain Name Dark Matter from Different Vantage Points*** This paper addresses the impact of the distributed nature of the Internet on domain name measurements. By measuring the domain name space from various measuring points (or *vantage points*), it is possible to compare the relative differences between the vantage points. We present a novel tool, *Gollector*, that collects and stores domain names from four vantage points in a structured manner. The paper introduces the notion of domain name *dark matter*, or the fraction of the domain name space that is unobserved from the perspective of a particular vantage point. By leveraging the differences between the dark matter of four different vantage points, we showcase the utility of *Gollector*.

The tooling and insights from this work provide a foundation for the data collection of the remaining papers in this section. Most importantly, the potential role of Certificate Transparency in domain registrations became apparent, thereby facilitating early detection of malicious domains.

Paper B – *Can a TLS Certificate be Phishy?* The field of phishing has focused on classifying entities such as URLs, websites, or emails as benign or malicious. The increasing usage of encryption (HTTPS) and visibility of this ecosystem (CT logs) makes digital certificates a potentially relevant target for classification. This paper performs a first look into this potential, by analyzing differences between certificates issued for phishing domains, and benign domains. We find that phishers resort to a relatively small set of – generally cheap or free – certificate authorities. Moreover, we identify that a significant portion of certificates issued for phishing domains cover not only a phishing domain but other unlabeled domains as well, suggesting that the reputation of a domain can be propagated to other domains covered by that same certificate. It also highlights that certificates can be somewhat ambiguous in nature, being both phishy and benign at the same time.

Paper C – *Detecting Ambiguous Phishing Certificates using Machine Learning* Following the results from the previous paper, a machine learning classifier was implemented that aimed to classify certificates as malicious or benign. As opposed to prior work by other research groups, we saw this as a multi-class classification problem, including these 'conflicted' certificates as a class next to benign and phishing certificates. Such certificates have deliberately been excluded by previous work, and as such it remains unknown how such systems would act on these conflicted certificates. We inferred the labels of certificates by producing an aggregated label from the labels of the domains that the certificates cover. Using a relatively simple classification model, this paper showed that this model is capable of classifying certificates with high accuracy.

Paper D – *Understanding the Challenges of Blocking Unnamed Network Traffic* The previous papers in this contribution area are concerned with domain name based measurements and appling this to different security contexts (i.e., domain name dark matter or phishing certificate classification). These results could ultimately result in systems that prevent Internet users to interact with domain names that are deemed malicious. However, there is a fraction of network traffic that does not rely on DNS resolutions for obtaining IP addresses, but relies on other methods such as using hard-coded IP addresses. This so-called *unnamed traffic* can bypass any domain name based protection system. Malicious software, such as worms, have been shown to propagate via unnamed traffic and as such there is a need to understand unnamed traffic better to block potential threats in networks. This paper proposes a novel method for extracting named and unnamed flows from raw network traffic and describes several challenges in identifying unnamed traffic. Using this method, we collected the unnamed traffic from the VPN server of our university network for a week and found that the majority of services, clients, and destination IP addresses communicate using unnamed traffic. Our results suggest that unnamed traffic is ubiquitous and cannot be simply blocked without significantly affecting the benign operations of a network.

3.2 Android attribution

Whereas certificates provide authenticity between browser and web servers on the Internet, they perform a somewhat different role in the Android ecosystem. There exists no public key infrastructure in which a third-party certificate authority performs domain ownership verifications prior to issuing a certificate. Here, they act solely as a means to exchange cryptographic public keys between Android devices, market stores, and developers, and do not provide any guarantees about the owner of the key and certificate.

Paper E – *Kaonashi: On Developer Attribution Challenges in Android App Marketplaces* Authorship attribution is an important aspect in the mobile security and privacy field, as devices collect highly sensitive information that should not fall in the wrong hands. In addition, regulators must have access to accurate contact information of developers and researchers must have a solid ground truth about the owner of apps. Android specifically, has had a rampant problem with malicious apps that abuse the lack of attribution, due to a lack of a public key infrastructure, and the low barrier of entry for new developers. In this paper, we investigate the extent to which the Android ecosystem makes attribution difficult and imprecise, and as a consequence, what the implications are on the research field as a whole.

3.3 Security education

The security group at Aalborg University has been developing the *Haaukins* platform since 2018. The project focuses on creating parallel virtual lab environments in which participants can practice penetration testing exercises for educational purposes. The design and development process has been documented in two papers, that are a part of this dissertation.

Paper F – *Haaukins: A Highly Accessible and Automated Virtualization Platform for Security Education* In this paper, *Haaukins* is introduced as a platform that creates virtual security labs automatically, intended for a high school setting. The paper presents the requirements that shaped the development process of the platform, and the implementation details that led to the fulfillment of these requirements. A major contribution of Haaukins is its ease-of-access, as participants access their labs through a remote desktop in their browser, which significantly reduces the set-up time for newcomers in the field compared to alternative methods.

Paper G – *Bridging the Gap: Adapting a Security Education Platform to a New Audience* After the publication of the previous paper, the development of *Haaukins* continued in a somewhat different direction. The most significant change was a shift in the audience, which tended to be older and more experienced. As a result, the requirements that shaped the development process changed over time and are addressed in this paper. This paper describes the new, changed design goals of the platform, and how these goals were met.

The combined results of these seven papers led us to understand the role of domain names and certificates in Internet security, and TLS an Android attribution in specific. Furthermore, we leverage the relationship between the two entities to classify certificates. These contributions conclude the first part of this dissertation, and part II comprises the seven papers described in this chapter.

Part II

Papers

Chapter 4

Gollector: Measuring Domain Name Dark Matter from Different Vantage Points

Paper A

Kaspar Hageman, René Rydhof Hansen, Jens Myrup Pedersen

The paper is accepted by
The 26th Nordic Conference on Secure IT Systems (NordSec 2021)

The layout of this work has been revised.

Abstract

This paper proposes Gollector, a novel tool for measuring the domain name space from different vantage points. Whereas such measurements have typically been conducted from a single (or few) vantage point, our proposed solution combines multiple measurements in a single system. Gollector allows us to express the relative difference in the covered domain name space, and the temporal characteristics, as domain name dark matter. *We leverage a three-week trace from four vantage points, by applying the tool to three security-related use cases: early domain registration detection, data leakage in a split-horizon situation, and a proposed method for subdomain enumeration. We release the Gollector source code to the research community to support future research in this field.*

4.1 Introduction

The Domain Name System (DNS) has historically played, and continues to play, a vital role in many different areas of network security research, including examining Internet censorship [A1], spam detection [A2, A3], and identifying botnet communication [A4]. The highly distributed nature of DNS, with information scattered across millions of domain name servers, means there exists neither a method to reliably observe all interactions with the DNS, nor a method to reconstruct the full domain name space. Consequently, researchers (implicitly or explicitly) choose one or more *vantage points*, e.g., network locations or network datasets, from which to conduct DNS measurements.

The (DNS) vantage point has a major impact on the DNS-related data that can be collected and processed. This is the case both for *passive* measurements, e.g., traffic monitoring of DNS resolvers that is highly dependent on physical location, but also for *active* measurements, e.g., due to geographical split horizons [A5] or censorship [A1]. We will refer to the part(s) of the domain name space that cannot be observed from a given vantage point as *DNS dark matter* (with respect to that viewpoint).

Different (partial) solutions have been deployed by the research community, including simply increasing the number of vantage points covered as well as including more different types of vantage points. However, to the best of our knowledge, there have been no studies focusing specifically on the impact of choosing different vantage points, and solutions discussed in the literature have been mostly ad-hoc. In this paper, we introduce the *Gollector* tool and framework as a step towards a more systematic and structured treatment of vantage points specifically, and DNS data collection and analysis in general.

In particular, we intend *Gollector* to become the "one-stop-shop" for DNS collection and analysis. Therefore, we open-source the tool for the research community to use, and the source code can be found at:

`https://github.com/aau-network-security/gollector`

We start the paper by providing the relevant background (Section 4.2) and place our work in the context of prior research (Section 4.3). In the remainder of the paper, we present the main contributions of the paper:

- We provide an overview of the existing vantage points, describing their conceptual advantages and drawbacks (Section 4.4).
- We present a novel tool and framework, *Gollector*, which combines data collected from different vantage points, allowing for the comparison of each vantage point (Section 4.5).
- We apply the tool to a sample dataset (Section 4.6) and leverage the data (in Section 4.7) for three previously unexplored use cases related to *DNS dark matter* and show that combining vantage points has a positive impact on DNS studies: *(i)* early domain detection, *(ii)* data leakage with split horizons, and *(iii)* subdomain enumeration.

4.2 Background

Systems on the Internet are commonly addressed within two namespaces: the IP address name space and the domain name space. IP addresses are used by routing devices to forward network traffic to the correct host, whereas domain names are more user-friendly and are therefore used by humans to address hosts. The DNS has been facilitating the translation between these two namespaces since the 1980s [A6]. The domain name space forms a tree structure, where each node in the tree is a label, and a domain name is a composition of all labels in a path of the tree from the root to a leaf. The nodes in the first layer are referred to as top-level domains (TLDs) and for the right-most label in a domain name (e.g., `.com` is the TLD for the domain `www.example.com`). A domain – sometimes referred to as a Fully Qualified Domain Name (FQDN) – can further be decomposed into an apex part and a subdomain part; the apex domain is the part that is registered at a domain registry, with the subdomain being the remaining part of the domain name. The higher levels of the domain name space are highly-regulated and reserved, and newly created domains (i.e., the apex domains) can be created under so-called public suffixes only [A7]. A part of the domain name space that is managed by a particular organization is referred to as a zone. For instance, registries - the operators of the TLDs – are authoritative for the zone which comprises all domains under that TLD, and maintains a zone file in which all mappings between the domain and IP name space are stored, or DNS records. The content of these zone files is served by *authoritative name servers* to DNS clients, answering queries with the appropriate DNS records if the name server is authoritative for the queried data, or with a reference to another name server if not. Clients commonly rely on intermediate DNS servers, referred to as resolving name servers or *resolvers*, to resolve DNS queries on their behalf while caching data locally to increase the performance of the DNS ecosystem as a whole.

The functionality of the DNS forms a fundamental basis for the workings of many other protocols on the Internet, including the Hypertext Transfer Protocol (HTTP) and Transport Layer Security (TLS). HTTP and TLS form the secure communication channel used by web browsers and web servers to communicate. Documents are exchanged by requesting specific locations identified by (among others) the domain name of the server that hosts the resource. TLS, providing the encryption layer to this exchange, relies on digital certificates exchanged between the browser and server which are used by clients to verify the authenticity of the web server [A8]. These digital certificates embed a set of domain names in them, which denote the domains for which the certificate is valid. A certificate is signed by one of hundreds of trusted third parties, the Certificate Authority (CA), who is tasked to verify the identity of an owner behind a certificate request before issuing the certificate. Due to two major incidents in which a CA erroneously issued a certificate [A9, A10], the Certificate Transparency (CT) was developed to audit the issuance behavior of each CA [A11]. In this framework, CAs submit newly issued certificates to publicly available, append-only logs, and as such, any third party can monitor these logs.

4.3 Related work

Given the ubiquitous nature of domain names on the Internet, the vast majority of applications interact with them in one way or another, including applications with malicious intent. For example, phishing websites are commonly hosted on typosquatted domains (i.e., a catch-all term for domain names that are similar-looking to benign domains) [A12] and bots within a botnet rely on the DNS as a communication channel to dynamically identify the location of their bot masters [A13]. As such, the security community has relied on data collected through domain name measurements to better understand and mitigate such types of malicious behavior. Relying on network traces, including DNS and TLS, to build extensive domain name-related datasets has been an ongoing process for many years. Passive DNS was proposed in 2004 as a method to support DNS data recovery, by collecting DNS queries in the wild, thereby being able to replicate the state of zone files at a particular point in time [A14]. The ISC implemented a version of these ideas in 2012 [A15], which resulted in the commercialization of the framework in 2013 under the company Farsight Security [A16]. The ENTRADA project focuses on collecting passive DNS traffic from the perspective of authoritative name servers instead [A17]. Alternatively, researchers relied on active measurements for creating longitudinal datasets. The OpenIntel project collects a fixed set of DNS records for all apex domains within a set of TLD zones daily [A18]. In their paper, the authors have collected data from three general TLD zone files comprising 50% of the apex domain name space, but have since then expanded to more general TLDs and sixteen country-code TLDs[1] [A19]. Hohlfeld [A20] expands upon this approach, by both collecting more than only DNS records (e.g., TLS support and particular TCP settings) and by relying on more domain name sources besides zone files (i.e., passive DNS and domains extracted from CT logs). Similarly, Project Sonar scans the IPv4 address space for (among others) TLS certificates, reverse DNS misconfigurations, and various TCP and UDP services [A21].

Reconstructing the full domain name space is a complex task due to the distributed nature of the DNS. For many TLDs, access to zone files is controlled through the Centralized Zone Data Service (CZDS) [A22], whereas access to other TLDs is more restricted, available ad-hoc involving non-disclosure agreements [A20], making a full replication of the apex domain space difficult. Several techniques have been used to circumvent this, such as "zone-walking" for DNSSEC-enabled zones [A23, A24] or abusing misconfigurations of DNS name servers, that allowed for a full zone file disclosure through zone transfer request [A25]. Alternatively, these zones can be partially reconstructed by relying on other data sources, including certificate transparency [A26] and the aforementioned passive DNS [A14, A15]. Even though passive DNS collects FQDNs (in addition to apex domains), mapping out the full FQDN domain name space is even more complex than the apex domain name space. The penetration testing community has relied on domain enumeration as one method of "reconnaissance", or information gathering about a particular target. As a result, a number of tools exist that support subdomain enumeration or DNS-based reconnaissance [A27–A31]. Typically, these subdomains are identified

[1] as of July 2021

by scraping third-party sources that have collected this information prior (e.g., search engines) [A32] or by generating candidate FQDNs [A27, A28, A31].

Gollector stands apart because it is intended to collect passive domain name-related data from more vantage points than any of these research or commercial initiatives. Furthermore, the tool is unique in the sense that it specifically emphasizes the *differences* between vantage points, allowing us to evaluate the relative dark matter between each vantage point. Lastly, in contrast to tools from the penetration testing community, *Gollector* collects traffic from a global perspective, rather than focusing on a small set of individual domain names.

4.4 Vantage points

When passively collecting traffic in the DNS (i.e., capturing traffic generated by a client population, rather than generating own DNS traffic) the measurement vantage point is a determining factor for what fraction of the total DNS traffic one can observe, as illustrated by Figure 4.1. Similarly, TLS traffic (prior to version 1.3) can be passively collected to observe the certificate (and other parameters) exchanged during a TLS handshake, which suffers from the same vantage point limitation as passive DNS collection. Alternatively, domain name related information can be extracted from other data sources that are not related to passive traffic. Besides active measurement – the process of actively probing servers for their responses to acquire information – sources are available that provide insight into the management and operation of a domain name. TLD zone files act as the ground-truth for all domains registered directly under a TLD, and can be used to infer registrations and domain expirations. The CT framework is an alternative source of TLS certificates, as it provides researchers access to publicly available, append-only logs of newly issued certificates by CAs. The logs guarantee that new certificates are published within a certain time frame – the Maximum Merge Delay (MMD) – such that the logs remain up to date with the latest issued certificates.

Even though these vantage points have their inherent differences, there is a commonality between them: the part of the domain name space they observe and the timing of those observations. Certain domain names or even full TLDs may be observed from one vantage point but would never be observable from another. As such, a part of the domain name space can be considered dark matter for the latter. Out of our previously-discussed vantage points, we identify four vantage points that significantly differ from each other[2]: passive

[2] The difference between data collected from a routing device from a network operator and a DNS resolver may be insignificant if both vantage points are owned by the same party, in the case

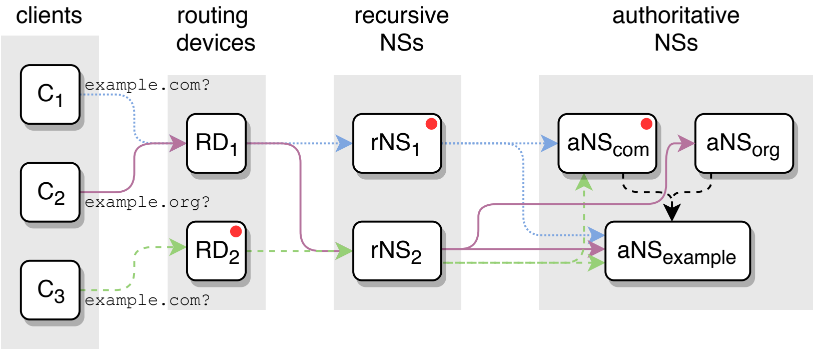


Figure 4.1: The different vantage points (denoted by ●) from which to conduct passive DNS measurements. Each arrow represents a DNS query send between two devices, where the colors indicate from which client the request originated. The various vantage points have different observations based on the querying behavior of the three clients: RD$_1$ observes $\{C_1, C_2\}$, rNS$_2$ observes $\{C_1, C_3\}$ and aNS$_{\text{com}}$ observes $\{C_2, C_3\}$.

DNS from a resolver, passive DNS from an authoritative name server, CT logs, and zone files. We can compare these vantage points according to the following properties. The WIDTH indicates how many TLDs the vantage point is capable of capturing domains across. The DEPTH shows what part of the FQDN a vantage point is capable of collecting domain names from. The TIME GRANULARITY represents the precision at which particular events are registered. A related dimension is the MAXIMUM TIME DELAY, which denotes how long it takes, worst case, for an event to be registered by a particular vantage point. Lastly, for the vantage points that passively collect data from a number of clients, the POPULATION COVERAGE illustrates the size of the overall population that is being covered by the vantage point.

Table 4.1 shows an overview of the four vantage points and their dimensions. Both a DNS resolver and CT logs are capable of observing across a variety of TLDs, although it depends on the DNS client population and certificate issuers, respectively, which TLDs are actually observed. The domains covered by the zone file of a TLD are registered at an apex domain level, and thereby this vantage point does not cover FQDNs as opposed to the others. Furthermore, the zone files provided by the CZDS are updated daily [A33], and therefore have a one-day granularity, whereas the other vantage points

of an ISP.

Table 4.1: Summary of vantage points

	Dimension				
Vantage points	Width	Depth	Time granularity	Maximum time delay	Population coverage
CT logs	All TLDs	FQDN	Precise	MMD	–
Resolver	All TLDs	FQDN	Precise	–	Local
Authoritative NS	One TLD	FQDN	Precise	–	Global
TLD Zone file	One TLD	Apex	Daily	One day	–

have a highly precise (i.e., sub-second) granularity. Every CT log operator defines a MMD, or maximum merge delay, which denotes the amount of time the operator will take as a maximum to publish newly issued certificates to the log, which tends to be 24 hours. The one-day granularity of zone files indicates that it can take up to a day for a newly registered domain to appear in the zone file. Lastly, for the two passive DNS vantage points, there is a difference in the DNS population coverage; an authoritative name server receives global traffic for domains within its zone, whereas a resolver serves only a local, smaller population.

4.5 Design

The current state-of-the-art tooling lacks the possibility of conducting analyses between vantage points at a full domain name space scale. Based on this, we derived a set of design goals that shaped the design and implementation of *Gollector*. We first present these design goals, followed by an overview of the architecture of *Gollector* describing how each of the individual goals is met.

Firstly, the main purpose of *Gollector* is to allow for the data collection of DNS and domain name-related information from different vantage points (G_1). The design of the tool must allow the collection from new vantage points to be added at a future point in time (G_2). Given the large number of existing domain names, and the volume of DNS and TLS data that is generated on the Internet, the tool should handle data collection at a large scale and remain highly performant (G_3). The data structures in which the data is stored must allow for post-collection analysis (G_4). The collection of DNS traces may contain sensitive information, which third-party data sources may be hesitant to share with researchers and may only be willing to do so in an anonymized form. However, the anonymization of data may make post-collection analysis more difficult, and less detailed, and as such we would like to preserve the relationship between unanonymized data and anonymized data in *Gollector* (G_5).

4.5.1 Architecture

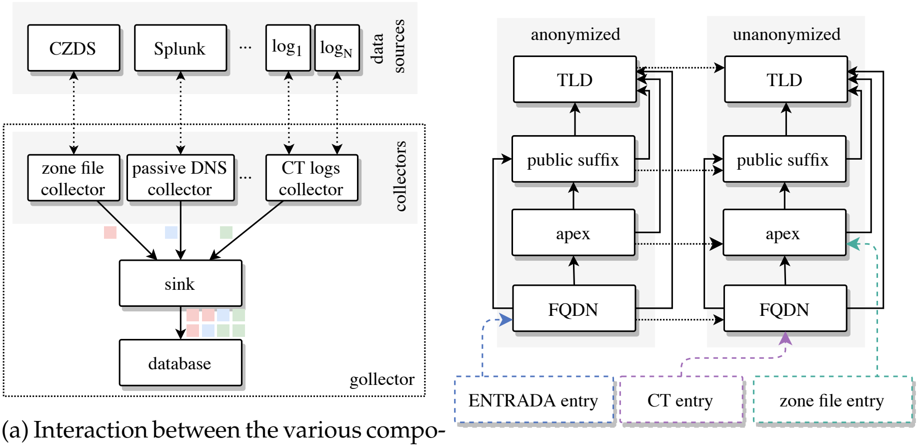
To meet G_1, the architecture of *Gollector* consists of modular components: (1) a set of data collectors, (2) a data sink, and (3) a database for persistent data storage (see Figure 4.2a). Each individual collector is a small component dedicated to collecting domain name-related data from one vantage point and sending the resulting data to the sink. So far, we implemented four collectors (see 4.5.2). The collectors and the sink communicate securely using gRPC, allowing the collectors and sink to operate on different machines and thereby collectors to operate in different network environments (i.e., collect data from different vantage points). The decoupling of collectors from the other components of *Gollector* allows new collector modules to be developed in the future, thereby meeting G_2.

The sink is designed to accept messages from the various collectors, extract database models from the messages and insert these models in the underlying database. As of now, we use PostgreSQL as the underlying database, as our database models naturally fit in a relational database model, and for future implementations, we can switch to a database intended for big-data analytics. The sink inserts new models in the database in batches rather than individual queries, resulting in a high-performance insertion rate (meeting G_3).

Domain names are stored in the database as a collection of database models (see Figure 4.2b how the models relate to each other). Each collected FQDN is segmented in its parts, according to the domain name hierarchy (i.e., top-level domain, its public suffix, the apex domain, and the FQDN). Each segment is inserted as its separate row in its own table and has a foreign key to all parts higher in the hierarchy. The data in this database is enriched by adding more tables with pointers (i.e., foreign keys) to these domain-related tables; a timestamped certificate may point to an FQDN, whereas a timestamped zone file entry may point to an apex domain instead. This makes it easy to answer questions such as *"How many unique apex domains are observed under each TLD?"* or *"For a given apex domain, how many certificates have we seen?"*, and thereby fulfills G_4.

Lastly, the segmented storage of domain names also applies to anonymized domain names. Instead of storing the domain name directly, we store an anonymized version of each segment by hashing[3] the segment after appending a salt to the segment. The database maintains a mirrored set of tables for

[3] using SHA256



(a) Interaction between the various components

(b) The database models.

Figure 4.2: Design of *Gollector*

anonymized segments, including the foreign keys from segments lower in the hierarchy to upper segments. To analyze data collected in both their anonymized and unanonymized form, we link the two sets of tables by adding a reference from the anonymized table to the unanonymized table (meeting G5). This link will only exist if a particular segment has been seen in both an anonymized dataset and an un-anonymized dataset, and thus will not apply to all observed domain names. As such, this method only provides anonymity for segments that have only been seen in their anonymized form, and only until an unanonymized form is collected.

4.5.2 Collectors

Each collector registers *events* related to a domain name. These events range from individual DNS queries to domain registrations. Depending on the collector, a collector may generate only one or a few events per domain or may generate many events over the course of a measurement. The current implementation of *Gollector* consists of the collectors described below.

Zone file collection This collector can fetch zone files from the CZDS [A22] and zone files over HTTP from servers that provide access. The former is an API provided by ICANN that allows for a standardized way to access zone files of over a thousand gTLDs including `.com` and `.net`, whereas the latter is used to fetch the Danish `.dk` TLD. In both cases, the authentication is handled by the collector and is configured through a configuration file during startup. The collector automatically requests access for zone files daily when the granted access expires, ensuring that data collection continues during long measurements. All available zone files are then collected daily, and any changes between zone files of two consecutive days are tracked. Domain names that appear and disappear in the most recent zone file are considered new registrations and expirations/removals respectively. Furthermore, we collect all domain names observed on the first day but do not consider these to be registrations or expirations. *Gollector* stores the zone files both raw on disk as well as in a processed form in the underlying database, so it allows researchers to work with the raw zone file if needed.

Passive resolver DNS *Gollector* itself does not perform any passive DNS measurements itself, but rather relies on previously collected datasets instead. The supported format for parsing passive DNS data is in Splunk Stream [A34] export data, which consists of a condensed form of individual DNS request-response pairs in a JSON format. From the logs, *Gollector* extracts the queried domain name and timestamp of the resolution, omitting any DNS-specific information, such as query types or resolved IP addresses.

CT logs Each CT log provides an HTTP API that can be used to fetch CT log entries. Such entries contain a full certificate chain of the newly-signed certificate, including the timestamp it was added to the log. *Gollector* traverses each log (as recognized by Google[4]), fetching all entries per log. This collector parses each entry, extracts the embedded domain names from the newly-signed certificate, and stores them with a timestamp when the certificate was submitted to the CT logs. Furthermore, the certificates are stored in their raw format in the underlying database, allowing for further, more in-depth, investigation when necessary.

ENTRADA This collector interfaces with the dataset generated by ENTRADA, used to collect DNS resolutions at an authoritative name server level. This dataset comprises DNS-specific attributes of each resolution, such as the query type, the specific resolved IP address(es), and the IP addresses from which the query originates. We summarize this information by collecting some basic statistics related to an individual domain name that has been queried; the first

[4] `https://github.com/google/certificate-transparency-community-site/blob/master/docs/google/known-logs.md`

Table 4.2: Overview of the collected data, denoting the unique number of events, FQDNs, apexes, public suffixes and TLDs observed per vantage point.

	Events	FQDNs	Apexes	Public suffixes	TLDs
Zone files	8,371,731	–	7,920,217	572	607
CT logs	114,182,670†	89,989,143	27,807,193	4,222	1,087
Passive DNS	200,146,260	6,046,480	1,213,405	1,125	580
ENTRADA	161,497,905	161,497,905	124,318,163	328	272

†We identify a CT event as a uniquely observed certificate.

time and the last time the domain was queried. As such, the information in *Gollector* is far smaller in size than the source dataset, at the cost of losing details.

4.6 Dataset

We applied *Gollector* to four types of data sources, for which we implemented the aforementioned four collectors (Section 4.5.2). We collected the data over a time period of three weeks[5]. For our experiments, we collected the passive DNS traffic from our Danish university network (with 10s of thousands of users) and the ENTRADA data from the Danish `.dk` TLD. We collected our certificate data from all recognized CT logs and attempted to retrieve all TLDs available from CZDS. Figure 4.2 illustrates the unique number of events, FQDNs, apexes, public suffixes, and TLDs observed per vantage point. Note that for the zone files, we only collected domains registrations and expirations, rather than all entries in the zone files. Since domains are registered at an apex domain level at a DNS registry, the collected zone files do not contain any FQDNs. Whereas our ENTRADA collector found the most unique FQDNs (161.5M), these FQDNs tend to be centralized under a relatively small set of TLDs (272) compared to the other vantage points. Conversely, our passive DNS collector identified the smallest number of FQDNs (6.0M), which is unsurprising given the relatively small number and homogeneity of clients the university network serves (i.e., primarily Danish students and academic staff). The CT log data spans most TLDs (1,087), which comprises 72.6% of all recognized TLDs [A35].

We hypothesize that the ENTRADA and the passive DNS traffic are highly

[5] Between February 1st, 2021 and February 21st, 2021

Table 4.3: The top ten TLDs per vantage point in terms of unique number of apex domains identified under the TLDs.

CT		Zones		Passive		ENTRADA	
com	44.3%	com	59.2%	mynet	23.9%	dk	100%
tk	4.5%	icu	10.7%	my_net	22.0%	arpa	0.0%
de	4.4%	net	4.4%	home	19.1%	com	0.0%
net	3.9%	xyz	3.9%	lan	16.5%	org	0.0%
org	2.9%	org	3.1%	com	4.9%	net	0.0%
uk	2.3%	wang	2.4%	dk	2.0%	se	0.0%
ru	1.6%	page	2.2%	localdomain	1.2%		
nl	1.6%	site	1.8%	net	1.1%		
br	1.5%	bar	1.0%	dlinkrouter	0.7%		
it	1.4%	club	0.8%	org	0.6%		

biased towards Danish traffic. To test this hypothesis, we analyze the distribution of unique apexes found per TLD for each of the vantage points. Table 4.3 shows for each vantage points, the percentage of apexes identified by that vantage point under a particular TLD, showing the top 10 TLDs per vantage point. The results show our hypothesis to be confirmed for ENTRADA, with nearly all apexes falling under the `.dk` TLD, whereas this is not the case for the passive DNS traffic. The passive DNS traffic contains a large number of apexes under reserved TLDs for internal use (i.e., `mynet`, `my_net`, `home`, `lan` [A36]), and the `.com` TLD is more popular than the Danish TLD. The CT and zone file datasets are more in line with the general size of the TLDs; `.com` and `.net` are the largest TLDs.

4.7 Use cases

We demonstrate the utility of *Gollector* by diving deeper into three use cases. Firstly, we evaluate the impact of the time differences of the four vantage points by analyzing how effective they are in recognizing newly registered domains. Additionally, we leverage the relative dark matter differences between passive DNS measurements from a resolver and an authoritative name server perspective to investigate the split-horizon setup of our local university network. These two use cases showcase the benefits of multiple vantage points. Lastly, we leverage the full set of FQDNs for a domain name generation algorithm, as an alternative to brute-force subdomain enumeration techniques employed by penetration testing tools.

4.7.1 Early detection of domain names

Various malicious actors rely on domain registrations for their operations, such as botnet operators (for domain fluxing) and phishers (for typo-squatting and hosting phishing sites in general). Prior work has demonstrated that the involved domains tend to be abused within a few days after their registration, after which they have already served their (malicious) purpose [A37]. From a defense perspective, identifying such domains in the early part of their lifecycle is therefore of critical importance. A domain registration can be detected at different points of time depending on the vantage point. Zone files are a logical choice, as they originate from the party that registers new domains (i.e., registries) but have as a limitation that they are created with a one-day granularity[6]. We investigate if other vantage points provide a more accurate – and thereby earlier – time of registration, especially focusing on CT logs, as they cover all TLDs rather than just one (which is the case for ENTRADA and to a lesser extent the passive DNS from university network).

We identified a registration of 4,438,966 domains over the course of 20 days[7] for an average of 221,948 domains per day. For each of these registrations, we identify if the other three vantage points (i.e., CT logs, passive DNS, and ENTRADA) observed the domains as well. Firstly, we identify if these vantage points observed the domain registrations *at all* in the full timeline. This serves as an indication to what extent domain registrations remain undetected and the coverage of the domain name space the vantage points have compared to zone files. We follow this up by identifying which of these domains were detected *before* the zone files registered these domains. For those domains, the one-day granularity of zone files is surpassed by the granularity of the other vantage point. Lastly, we identify the domains that were detected *within seven days* after the zone file identified the domain as registered. Wullink *et al.* [A37] showed that phishing domains tend to be most active within the first seven days of their registration (based on the DNS traffic the domain receives). Therefore, identifying such domain registrations within seven days is highly beneficial for mitigation these attacks.

Table 4.4 shows the results, both in absolute numbers and the percentage of the total number of zone file registrations. Since the ENTRADA dataset operates at a single TLD's name server, we differentiate between the full dataset and the dataset for the `.dk` domains only. Across all TLDs, the CT logs have relatively high coverage, with almost one in four domain registrations being

[6] Registries will have access to more accurate registration data than just the zone files, so this is a limitation for researchers who only have access to the zone files

[7] Since a registration is detected by computing the difference of the zone files of two subsequent days, we are missing the registrations on the first day of our measurement.

Table 4.4: Detection of newly registered domain names for non-zone files vantage points. The results for both the full set of TLDs and the `.dk` zone only are shown.

			All TLDs			
		Absolute			Percentual	
	CT	Passive	ENTRADA	CT	Passive	ENTRADA
Overall	971,318	533	46,628	23.6%	0.01%	1.1%
Before	568,436	216	25,713	13.8%	0.01%	0.62%
Within 7 days	325,277	169	4688	7.9%	0.00%	0.11%
			`.dk` only			
Overall	16,476	63	46,495	34.9%	0.13%	98.5%
Before	0	0	25,673	0.00%	0.00%	54.4%
Within 7 days	639	3	4,601	1.35%	0.01%	9.74%

detected across the whole dataset. The passive DNS and ENTRADA datasets have a much smaller coverage, with only 0.01% and 1.1% of domain registrations being detected. Notably, CT logs provide earlier detection of domains compared to zone files for 13.8% of domains. When looking at the `.dk` domain only, all vantage points detect more registrations than the full dataset, with ENTRADA detecting nearly all registrations. Furthermore, of the domain registrations detected by the passive DNS dataset, almost all of them were `.dk` domains (with only 133 non-Danish registrations detected). In none of the cases, the percentage of identified registrations was significantly improved by including the first seven days after registration. For identifying new domain registrations, zone files are still primarily the best vantage points, but this can be supported by CT logs and ENTRADA for individual TLDs.

4.7.2 Split horizon and data leakage

Large organizations commonly operate a *split-horizon* DNS infrastructure, where DNS resolutions receive different responses depending on the location of the requester. Use cases include load balancing or protecting sensitive information that should only be accessible from within a corporate network [A38]. Furthermore, the exposure of the existence of a particular hostname can already provide insight into an organization's inner workings and should potentially be protected against. We leverage our passive DNS data and ENTRADA dataset – both relatively biased towards the `dk` TLD – to investigate potential data leakage in our dataset. We identify which domain names are likely to be only used internally and what data is leaked outside the network through DNS queries. The split-horizon setup should result in particular domain names only being queried within the university network.

As a first step, we identify what apex domain names are likely to be owned by the university network. We assume that internal apex domains are heavily used for various services within the network, thereby having many different FQDNs in use. As such, we collect the unique number of FQDNs observed under each apex domain in our dataset. Table 4.5 shows this count for the 10 most prevalent apex domains and also shows the percentage of total FQDNs observed in the passive DNS dataset it encompasses. FQDNs under `aau.dk` are seen most often (more than 63% of observed FQDNs fall under this apex), suggesting that this domain is used for internal systems within the network. Indeed, this domain is owned by the university, whereas the other domains in the table are related to background services such as advertisement/analytics (e.g., `googlesyndication.com`, `cedexis-radar.net`), or network management (e.g., `bbsyd.net`, `emnet.dk`), and are not associated with the university.

From our ENTRADA dataset, we found 18,499 FQDNs under the `aau.dk` apex domain, a much lower number than the 3,8M seen in the passive DNS dataset. Not all of these domain names are necessarily sensitive information, as some of the domains used by the university are likely used to host public websites. Therefore, we turn to the domains that have both been seen by the passive DNS and the ENTRADA dataset: this set of domains comprises 2,813 FQDNs, or 15.2% of the `aau.dk` FQDNs seen in the ENTRADA dataset. Since we have no ground truth of what is a sensitive domain and what is not, we compare this list of domains to the most common subdomains instead [A39]. We found 435 (or 15.5%) of these domains is in the public list, leaving more than 2,300 subdomains potentially leaked. As a result of the anonymization practice of *Gollector*, we are unable to further investigate these potentially leaked domains, as these domains are anonymized and the unanonymized version can (deliberately) not be retrieved.

4.7.3 Subdomain enumeration

Part of the reconnaissance phase in penetration testing is subdomain enumeration, or the process of identifying all subdomains under a given apex domain. Strategies include scraping third parties or generating (i.e., brute-forcing) candidate FQDNs [A27, A28, A31]. *Gollector* supports the former, as its database model allows to easily query all FQDNs observed under a particular apex domain. We present a method to support the latter as well. As opposed to the existing brute-forcing techniques, we infer a relationship between sets of subdomains, based on the co-occurrence of these subdomains under a shared

Table 4.5: The 10 apex domains with the most observed unique FQDNs in the passive DNS dataset collected from the university network.

Apex domain	Unique FQDN count	%
`aau.dk`	3,829,837	63%
`googlesyndication.com`	344,058	6%
`technicolor.net`	61,151	1.01%
`cedexis-radar.net`	44,771	0.74%
`sophosxl.net`	39,297	0.65%
`bbsyd.net`	36,758	0.61%
`office.com`	30,215	0.50%
`emnet.dk`	23,540	0.39%
`obelnet.dk`	22,909	0.38%
`webspeed.dk`	21,569	0.36%

set of apex domains. As a result, our proposed method generates accurate candidate FQDNs and identifies relationships between subdomains that otherwise would not be found. This inference is motivated by a particular use case in which subdomains are likely to co-occur under the same apex domain; `cPanel` defines a set of *Service Subdomains*, or subdomains exposed by cPanel to provide interfaces to external components [A40]. Therefore, the existence of a `cPanel` subdomain may indicate that the other *Service Subdomains* are also "in use" under the particular apex domain, even if a DNS dataset has not identified its existence. Our proposed method consists of the following steps: (1) we convert our dataset of subdomains and apex domains in a graph, (2) we compute a clique cover of this graph, (3) we prune these cliques according to the weights in each clique to filter out nodes that are not relevant to the clique, and (4) we generate a set of candidate FQDNs, based on the pruned cliques.

As a first step, we split up our set of FQDNs into their subdomain and apex parts, and subsequently create a graph in which the subdomains are modeled as nodes. Edges between two subdomains express the measure of overlap of the sets of apex domains under which both subdomains have been seen. The edge weight is computed as the Jaccard index [A41] of the set of apex domains under which the first subdomain has been seen and the set of apex domains under which the second subdomain has been. We prune the edges that have a weight of zero (i.e., between subdomains that are never seen under the same apex domain), and remove any nodes that are without edges (i.e., subdomains never seen under the same apex domain as another subdomain).

We split up the nodes in our graph into a *clique cover*. Cliques are induced subgraphs such that each node is adjacent to all other nodes in the subgraph. This implies that every subdomain within a clique has been observed under the same apex domain with all other subdomains at least once. By assigning each node to a clique we reach a clique cover, which we achieve by relying on the algorithm defined in Appendix 4.A[8].

A clique cover ensures every subdomain falls in a clique, but this does not guarantee there is a strong connection between the nodes within the clique. Therefore, we prune each clique to remove nodes that are not considered relevant. We scale the edge weights in each clique such that the highest weight equals 1, and then prune the nodes whose maximum edge weight falls under a given threshold. In our experiments, we used a threshold value of 0.6, which we found through thorough experimentation.

For each clique, we can now generate a set of candidate FQDNs. We maintain a set for each subdomain, denoting the apex domains under which the subdomain has been observed, based on all FQDNs in our dataset. For each clique, we define the set of apex domains that *any* of the subdomains in the clique has been observed under. The Cartesian product of the apex domains and subdomains then forms the tuples of apex domains and subdomains representing the candidate FQDNs. The FQDNs already seen in data are left out from this set, forming the final set of candidate FQDNs.

We applied this methodology to a dataset of 2 million randomly sampled FQDNs from our dataset, In total, we identified 8,410 cliques comprising 22,519 subdomains. Appendix 4.B shows several examples of subdomains that form a clique. These subdomains were previously seen under a set of 1,021,175 apex domains. Given our cliques, we generated 2,349,911 FQDN candidates, resulting in an average of only 2.3 FQDNs per apex domain. Out of these candidates, we could successfully resolve 1,396,129 FQDNs or 59% of candidates. Additionally, we also manually investigated some of the cliques to understand what the nature of these cliques is. This manual investigation was far from exhaustive, but we found cliques related to the software that runs on these domains (such as the `cPanel` example that drove this research) and cliques pointing to a specific organization. An example of the former clique type is cliques for subdomains used by Magento, a highly-popular open-source eCommerce platform [A42]. We identified 470 cliques related to this platform with subdomains containing the keyword `magento`, often having `shop` or `store` as another keyword being embedded in one of the subdomains. The latter type includes a clique formed by subdomains under the apex domains `fbcdn.net`

[8] There are potentially many clique covers, and our purpose is not to achieve a minimal clique cover

and `whatsapp.net`, containing 118 subdomains, indicating the relationship between Whatsapp and Facebook.

Our proposed method can be integrated into existing penetration testing tools as an alternative to wordlist-based domain generators. On top of that, our cliques can be used to identify shared domain name ownership, and to assist security researchers in identifying domains hosting the same services.

4.8 Conclusions

In this paper, we introduced *Gollector* as a novel platform for collecting domain name and DNS-related information. Through a thorough overview of the DNS and TLS ecosystem, we present a set of vantage points from which this information can be retrieved. Through three uses cases, we leverage the differences between these vantage points. Firstly, we show that that CT logs and passive DNS traffic collected at an authoritative name server can serve as a source for early domain registration detection. Zone files are outperformed by the CT logs in 13.8% of domains under all TLDs, and by the passive authoritative traffic in 54.4% of domains under the `.dk` TLD. Secondly, we compare passive DNS measurements from a university network with authoritative name server measurements to shed light on potential data leakage of subdomains under the main domain name in use by the university. Lastly, we present a method to generate potentially existing FQDNs, which infers these FQDNs based on the association of subdomains and apex domains.

Acknowledgments

This research was carried out under the SecDNS project, funded by Innovation Fund Denmark. We would like to express our gratitude to Finn Büttner and Erwin Lansing for their assistance in collecting our passive DNS datasets.

References

[A1] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson, "Global measurement of DNS manipulation," in *26th USENIX Security Symposium (USENIX Security 17)*, (Vancouver, BC), pp. 307–323, USENIX Association, Aug. 2017.

[A2] O. van der Toorn, R. van Rijswijk-Deij, B. Geesink, and A. Sperotto, "Melting the snow: Using active DNS measurements to detect snowshoe spam domains," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2018)*, pp. 1–9, 2018.

[A3] 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*, CCS '16, (New York, NY, USA), pp. 1568–1579, Association for Computing Machinery, 2016.

[A4] K. Alieyan, A. Almomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Computing and Applications*, vol. 28, no. 7, pp. 1541–1558, 2017.

[A5] "Using GeoIP with BIND 9." `https://kb.isc.org/docs/aa-01149`, 2020. Accessed: 2021-07-10.

[A6] P. Mockapetris, "Domain names - implementation and specification," STD 13, RFC Editor, November 1987. `http://www.rfc-editor.org/rfc/rfc1035.txt`.

[A7] "Public suffix list." `https://publicsuffix.org/`. Accessed: 2021-07-10.

[A8] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," RFC 8446, RFC Editor, August 2018. `http://www.rfc-editor.org/rfc/rfc8446.txt`.

[A9] J. Prins, "DigiNotar certificate authority breach operation black tulip." `https://media.threatpost.com/wp-content/uploads/sites/103/2011/09/07061400/rapport-fox-it-operation-black-tulip-v1-0.pdf`, 2011. Accessed: 2021-07-23.

[A10] "Comodo SSL affiliate the recent RA compromise." `https://blog.comodo.com/other/the-recent-ra-compromise/`. Accessed: 2021-07-23.

[A11] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," RFC 6962, RFC Editor, June 2013. `http://www.rfc-editor.org/rfc/rfc6962.txt`.

[A12] J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C. Kanich, "The long "taile" of typosquatting domain names," in *23rd USENIX Security Symposium (USENIX Security 14)*, (San Diego, CA), pp. 191–206, USENIX Association, Aug. 2014.

[A13] M. Singh, M. Singh, and S. Kaur, "Issues and challenges in DNS based botnet detection: A survey," *Computers & Security*, vol. 86, pp. 28–52, 2019.

[A14] F. Weimer, "Passive DNS replication," in *FIRST Conference on Computer Security Incident*, 2005.

[A15] R. Edmonds, "ISC passive DNS architecture." `https://mirror.yongbok.net/isc/kb-files/passive-dns-architecture.pdf`, 2012.

[A16] A. Behjat, "ISC spins off its security business unit." `https://www.isc.org/blogs/isc-spins-off-its-security-business-unit/`, 2013.

[A17] M. Wullink, G. C. M. Moura, M. Müller, and C. Hesselman, "Entrada: A high-performance network traffic data streaming warehouse," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2016)*, pp. 913–918, 2016.

[A18] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, and A. Pras, "A high-performance, scalable infrastructure for large-scale active DNS measurements," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1877–1888, 2016.

[A19] "Openintel - current coverage." `https://openintel.nl/coverage/`, 2020. Accessed: 2021-07-10.

[A20] O. Hohlfeld, "Operating a DNS-based active Internet observatory," in *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, SIGCOMM '18, (New York, NY, USA), pp. 60–62, Association for Computing Machinery, 2018.

[A21] "Project Sonar." `https://opendata.rapid7.com/about/`, 2021. Accessed: 2021-07-10.

[A22] "Centralized zone data service." `https://czds.icann.org/`, 2021. Accessed 30-08-2021.

[A23] J. Schlyter, "DNS security (DNSSEC) NextSECure (NSEC) RDATA format," RFC 3845, RFC Editor, August 2004. `http://www.rfc-editor.org/rfc/rfc3845.txt`.

[A24] N. L. M. van Adrichem, N. Blenn, A. R. Lúa, X. Wang, M. Wasif, F. Fatturrahman, and F. A. Kuipers, "A measurement study of DNSSEC misconfigurations," *Security Informatics*, vol. 4, oct 2015.

[A25] E. Borges, "Wrong Bind configuration exposes the complete list of russian TLDs to the Internet." https://securitytrails.com/blog/russian-tlds, March 2018. Accessed 30-08-2021.

[A26] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman, "Towards a complete view of the certificate ecosystem," in *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, (New York, NY, USA), pp. 543–549, Association for Computing Machinery, 2016.

[A27] "OWASP/Amass." https://github.com/OWASP/Amass. Accessed: 2021-07-10.

[A28] "Sublist3r." https://github.com/aboul3la/Sublist3r. Accessed: 2021-07-10.

[A29] "Subfinder." https://github.com/projectdiscovery/subfinder. Accessed: 2021-07-10.

[A30] "DNSdumpster." https://dnsdumpster.com/. Accessed: 2021-07-10.

[A31] "DSNRecon." https://github.com/darkoperator/dnsrecon. Accessed: 2021-07-10.

[A32] Bharath, "A penetration testers guide to subdomain enumeration." https://blog.appsecco.com/a-penetration-testers-guide-to-sub-domain-enumeration-7d842d5570f6, 2018. Accessed: 2021-07-24.

[A33] "About zone file access." https://www.icann.org/resources/pages/zfa-2013-06-28-en, 2021. Accessed 30-08-2021.

[A34] "About Splunk stream." https://docs.splunk.com/Documentation/StreamApp/7.3.0/DeployStreamApp/AboutSplunkStream, 2020. Accessed: 2021-07-10.

[A35] "List of top-level domains." https://www.icann.org/resources/pages/tlds-2012-02-25-en, 2021. Accessed 30-08-2021.

[A36] D. Eastlake and A. Panitz, "Reserved top level DNS names," BCP 32, RFC Editor, June 1999. http://www.rfc-editor.org/rfc/rfc2606.txt.

[A37] M. Wullink, M. Muller, M. Davids, G. C. M. Moura, and C. Hesselman, "ENTRADA: enabling DNS big data applications," in *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pp. 1–11, 2016.

[A38] R. Aitchison, *DNS Techniques*, pp. 163–207. Berkeley, CA: Apress, 2011.

[A39] "The most popular subdomains on the Internet." `https://bitquark.co.uk/blog/2016/02/29/the_most_popular_subdomains_on_the_internet`, 2016. Accessed: 2021-07-27.

[A40] "Service subdomains explanation." `https://documentation.cpanel.net/display/CKB/Service+Subdomains+Explanation`, 2021. Accessed 30-08-2021.

[A41] P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901.

[A42] "Magento." `https://magento.com/`. Accessed: 2021-07-27.

Appendices

4.A Clique cover algorithm

Algorithm 1 denotes the algorithm used to compute a clique cover for graph G. The intuition behind the algorithm is that two nodes – connected through an edge with the largest weight – have the largest priority to form a clique. The algorithm iterates over all edges in the graph and assigns a clique to each node in the graph based on the interactions that are observed through the edges. Depending on whether the source and destination nodes of the edge are already in a clique, the algorithm creates new cliques, adds nodes to existing cliques, or merges cliques. The output of the algorithm is a hashmap of the clique assigned to each node in the graph. The implementation of the algorithm includes several optimizations to reduce the edges to evaluate.

4.B Examples of cliques

Table 4.B.1 contains several examples of cliques. The table shows a general description of what the subdomains may be intended for, the number of subdomains in the clique, the number of apexes associated with these subdomains, and the list of subdomains comprised by the clique.

```
Function cliqueCover (G);
Input   : graph G of subdomain nodes
Output: set of subdomain lists
edges = edgeListFrom(G);
edges = sortByWeight(edges);
cliques = {};
for edge in edges do
    src, dst = nodes in edge;
    cliqueSrc = cliques[src];
    cliqueDst = cliques[dst];
    if src not in clique and dst not in clique then
        /* both are without clique, create a new one                */
        c = newClique(src, dst);
        cliques[src] = c;
        cliques[dst] = c;
    else if src in same clique as dst then
        /* src and dst are already in the same clique               */
    else if src not in clique and dst in clique then
        /* try to add dst to cliqueSrc                              */
        if cliqueSrc.formsCliqueWith(dst) then
            cliques[src].add(dst);
        end
    else if src in clique and dst not in clique then
        /* try to add src to cliqueDst                              */
        if cliqueDst.formsCliqueWith(src) then
            cliques[dst].add(src);
        end
    else if src and dst in different cliques then
        /* try to merge the two cliques                             */
        if cliqueSrc.formsCliqueWith(cliqueDst) then
            c = mergeCliques(cliqueSrc, cliqueDst);
            cliques[src] = c;
            cliques[dst] = c;
        end
end
return cliques;
```

Algorithm 1: Clique cover algorithm

Table 4.B.1: Examples of cliques

Description	Subdomain count	Apex count	Subdomains
High-entropy subdomains	237	2	`adfqjkxr`, `aeovrpvk`, `anhpfctcxzcp`, `asqzcggxiy`, `bdzvxofezaejku`, . . .
Email servers	5	34,249	`imap`, `xwa`, `xas`, `pop`, `smtp`
Western language-related subdomains	7	26,730	`en`, `es`, `fr`, `pt`, `it`, `ru`, `de`
More language-related subdomains	6	3,764	`ko`, `zh`, `cs`, `nl`, `ar`, `ja`
Content deliver network	9	5,197	`cdn-1`, `cdn-3`, `cdn-2`, `cdn-5`, `cdn-7`, . . .

Chapter 5

Can a TLS Certificate Be Phishy?

Paper B

Kaspar Hageman, Egon Kidmose, René Rydhof Hansen, Jens Myrup Pedersen

The paper was published in the
Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)

Abstract

This paper investigates the potential of using digital certificates for the detection of phishing domains. This is motivated by phishing domains that have started to abuse the (erroneous) trust of the public in browser padlock symbols, and by the large-scale adoption of the Certificate Transparency (CT) framework. This publicly accessible evidence trail of Transport Layer Security (TLS) certificates has made the TLS landscape more transparent than ever. By comparing samples of phishing, popular benign, and non-popular benign domains, we provide insight into the TLS certificates issuance behavior for phishing domains, focusing on the selection of the certificate authority, the validation level of the certificates, and the phenomenon of certificate sharing *among phishing domains. Our results show that phishing domains gravitate to a relatively small selection of certificate authorities, and disproportionally to* cPanel, *and tend to rely on certificates with a low, and cheap, validation level. Additionally, we demonstrate that the vast majority of certificates issued for phishing domains cover more than only phishing domains. These results suggest that a more pro-active role of CAs and putting more emphasis on certificate revocation can have a crucial impact in the defense against phishing attacks.*

5.1 Introduction

Decades after its inception in the '90s, phishing remains a significant problem. This scalable form of criminal activity can be characterized by the use of deception in which impersonation is used to obtain information from a target [B1]. The Anti-Phishing Working Group (APWG) still reports the discovery of tens of thousands of phishing sites monthly [B2]. This indicates that phishing is far from a solved problem, and that there remains a need for novel and improved detection, prevention, and mitigation methods.

A significant effort, from both an academic and commercial perspective, has been made towards the detection and identification of phishing entities, such as URLs, emails, websites, and domains. Existing approaches are in many cases based on identifying similarities between suspicious entities and known legitimate ones, as criminals conducting phishing attacks (referred to as phishers) often attempt to deceive victims into believing they are interacting with a legitimate system. However, entities that so far ha not received a similar degree of scrutiny are digital certificates. Such certificates are used in establishing a secure communication channel between (among others) web browsers and web servers. It raises the research question on whether TLS certificates can be labeled as 'phishy' or benign in the same manner URLs and domains have historically been given these labels, ultimately preventing Internet users from interacting with websites serving these certificates. The recent large-scale adoption of two technologies has resulted in a trail of certificates, which potentially can be used for an alternative detection method of phishing attacks:

- HTTPS, i.e.,, HTTP over Transport Layer Security, by phishing websites, for encrypting traffic between the browser and web server
- The submission of newly-issued TLS certificates to Certificate Transparency (CT) logs by certificate authorities

Certificate Transparency has been developed for third parties to monitor the logs for fraudulently issued certificates, resulting in TLS certificates being inserted in these logs in near real-time and additionally on a global scale. Furthermore, the issuance of certificates is assumed to occur early in the lifecycle of a domain, and thereby also of the phishing attack. Prior research has identified a general short domain lifetime and disposable nature of phishing domains, which we hypothesize to be reflected in the CT log artifacts we can observe:

1. The selected *certificate authority* that phishing domains resort to for issuing their certificates are expected to be cheap and certificate issuance is expected to be automated. In addition, an analysis of CA selection may reveal certain patterns related to the phishing hosting infrastructure of phishers.

2. The *validation level* of certificates is a proxy for the monetary cost that phishers invest into increasing the perceived legitimacy of phishing websites.

3. Phishing attacks may be part of a larger campaign, and the preparation of those attacks may be coordinated. It is hypothesized that this is reflected in certificates covering more than one phishing domain, which in practice would simplify the hosting infrastructure of the phishing attack.

In this work, we test these hypotheses by analyzing a large collection of TLS certificates, hinting towards the 'phishyness' of the certificates. The results serve as a preliminary motivation for pursuing a CT-based phishing mitigation system. It is namely important that phishing domains and non-phishing domains handle their infrastructure significantly different from a TLS perspective, in order to rely on them for such as mitigation system.

Prior to testing these hypotheses, we used our collected data to show that for the majority of phishing domains, a certificate is issued before the domain gets blacklisted, which emphasizes the relevancy of a CT log-based protection system. Our main findings are as follows:

- Phishers resort to a relatively small set of CAs for their certificate issuance, and the issuer of certificates reveals information regarding the infrastructure on which services for domains are served, as illustrated by a significant number of cPanel servers for phishing domains.

- Phishers seldom resort to the more expensive EV certificates, and rarely to OV certificates, although these results apply to non-popular domains as well.

- Certificates rarely cover *only* phishing certificates, but a large fraction of certificates issued for phishing domains cover other domains as well.

The remainder of the paper is structured as follows. We present the context of the paper in the background and related work in Sections 5.2 and 5.3. This is followed by a description of the methodology and the results in Sections 5.4 and 5.5. The overall impact of the results is discussed in Section 5.6.

5.2 Background

Transport Layer Security (TLS) — like its predecessor SSL — is the underlying protocol suite for encrypting communication on the Internet. The secure communication it provides is facilitated by a public key infrastructure, in which the identity of an entity (such as a domain name) is bound to a cryptographic public key. The proof of such a binding is stored and distributed in the form of an X.509 certificate, and consists of the identities, the public key and a cryptographic signature that allows a web browser to verify the identity of the web server it is initiating a TLS connection with. The process of issuing certificates is handled by one of the hundreds of certificate authorities (CAs), which are inherently-trusted, third-party organizations. The X.509 certificate standard supports the binding with multiple identities [B3], and these identities are referred to as Subject Alternative Names (SANs). This allows an organization to request a single certificate for multiple domains (or other entity types such as IP addresses), thereby reducing the number of certificates required to secure web traffic towards their infrastructure.

An applicant applies for a certificate at a CA with information about the to-be-created certificate, such as the SANs to cover, the validation level of the certificate, and the period for which the certificate is valid. The CAs are tasked to verify that the requester of a certificate does in fact own all entities that the certificate is about to cover. This verification can be done via a variety of methods depending on the CA, including email verification, an HTTP endpoint, or more thorough background checks. After an applicant proves ownership of all SANs, the CA issues a certificate, signed by their own root (or intermediate) certificate.

Each certificate has a validation level associated with it, indicating the depth - and therefore also the cost - of the verification process the CA and requester went through to get the certificate issued. Domain validated (DV) certificates require the least validation, followed by organization validated (OV) certificates and lastly extended validated (EV) certificates. The latter validation type is reserved for corporations and recognized entities and requires an in-depth background check. The benefit of these more expensive OV and EV certificates used to be a different indicator in the browser, although most browsers are moving towards removing these differences[1], as different indicators have proven to be ineffective [B4].

In addition to the validation level and SANs to be covered in the domain, an applicant must declare the duration of the *validity period* of the certificate. As a security mechanism, certificates expire after a time span after which a certificate cannot successfully be verified and TLS connections must be rejected. Typically, this period lies between a few months and couple of years, and since 2020, the *CA/Browser Baseline Requirements* restricts the validity period to a maximum of 13 months [B5]. This period prevents certificates from being abused for long periods of time, in case the private key of the cryptographic key pair of the certificate owner is compromised.

5.2.1 Certificate Transparency

Fraudulent issuance of a certificate — regardless of its malicious intent — can have a disastrous impact, as demonstrated by two incidents in 2011. Attackers were able to obtain certificates for domains including `google.com` and `microsoft.com`, allowing them to perform large-scale man-in-the-middle attacks [B6]. The reputation damage eventually resulted in the bankruptcy of one of the CAs. As a direct response to these incidents, the Certificate Transparency project was initiated with the goal of monitoring and auditing the certificate issuance of CAs. The project encourages CAs to submit every issued certificate to publicly accessible, append-only CT 'logs'. These CT logs can be run by third-party organizations and, due to their public availability, al-

[1] https://blog.chromium.org/2018/05/evolving-chromes-security-indicators.html

low anyone to monitor for fraudulent issuance of certificates for their domains. As of February 2021, Chrome's CT policy recognizes more than a hundred logs being operated by 21 different organizations, and they contain over 12 billion issued certificates[2].

After submitting a certificate to a CT log, a CA obtains a Signed Certificate Timestamp (SCT), which acts as a proof from the log operator that the certificate is, or will soon be, appended to the log. This SCT is embedded in the certificate, which can be used during the TLS handshake between a browser, and a web server to verify the inclusion of the certificate in the logs. As of April 2018, Chrome has started to require any certificate to comply with its CT policy, effectively requiring all certificates to be logged in at least two CT logs[3]. A similar policy was introduced by Apple in October 2018[4]. Due to the large combined browser market share of Google and Apple (an estimated 82.3% as of February 2021[5]), this has resulted in the large-scale adoption of the CT framework. Combined with a generally increased adoption of HTTPS (an estimated 84% for phishing URLs [B2]), this makes the CT logs a promising data source for phishing detection.

5.3 Related work

Scheitle *et al.* [B7] recognized the potential of CT logs for phishing detection based on a preliminary experiment, focusing primarily on typosquatting domains, the practice of registering domains looking similar to other domains in order to create confusion. More recently, Fasllija *et al.* [B8] explored this idea further by implementing and evaluating a classifier based on the certificates contained in the CT logs. Similarly, Sakurai *et al.* [B9] built domain name templates and match newly issued certificates against these templates to identify new phishing domains. Commercial initiatives such as CertSpotter[6], PhishFinder[7], and Facebook[8] started to provide protection services that alert domain owners whenever a certificate for their domains is submitted to a CT log. These initiatives primarily rely on the lexical properties of the domains and do not explore the TLS-specific information contained in the logs.

Alternative early detection systems for domain abuse have been proposed

[2] `https://www.certificate-transparency.org/known-logs`

[3] `https://github.com/chromium/ct-policy/blob/master/ct_policy.md`

[4] `https://support.apple.com/en-us/HT205280`

[5] `https://www.w3counter.com/globalstats.php`

[6] `https://sslmate.com/certspotter/`

[7] `https://phishfinder.io/`

[8] `https://developers.facebook.com/tools/ct`

in prior research. The works of Hao *et al.* [B10, B11] resulted in PREDATOR, a proactive detection system of spamming domains. The Dutch registry, SIDN, uses nDEWS [B12] as an early detection system for various types of domain abuse, operating at the level of a top-level domain. Lever *et al.* [B13] detect domain ownership changes for identifying malicious re-registrations. These systems primarily rely on the DNS for their data, and could potentially be combined with CT log data analysis for better performances.

The CT logs have been used as a source data set for other application areas besides phishing domain detection. Manousis *et al.* [B14] analyse the impact of *Let's Encrypt* on the TLS ecosystem, finding early evidence for the adoption of typosquatters and malware hosters. Similarly, Aersten *et al.* [B15] identify that in the first year of *Let's Encrypt*'s introduction, primarily low-cost domains started to resort to *Let's Encrypt* as their CA. VanderSloot *et al.* [B16] investigated the coverage of the entire TLS ecosystem from various viewpoints (in terms of observed certificates). They identified that CT logs at the time already captured over 90% of all certificates, which since then presumably has only increased further.

Prior research has relied on alternative certificate-related datasets. These datasets were often collected by either actively probing TLS servers (i.e., active measurements) or monitoring network traffic for TLS handshakes (i.e., passive measurements). Active measurements allow researchers to not only capture the certificate, but also parameters exchanged during the TLS handshake [B17]. Drawbacks of active measurements include its reliance a list of known domains[9], partial coverage caused by unavailability of servers and the difficulty of performing continuous measurements. In passive measurements, such as the works by Razaghpanah *et al.* [B19], have no control of which certificate are observed. The degree of coverage of the TLS ecosystem is highly dependent on the quality and quantity of the observation point(s), as a small and homogenous client population is unlikely to query a significant portion of TLS-enabled servers. In contrast to CT logs, neither active nor passive measurements are guaranteed to provide certificates in a timely fashion (i.e.,, immediately after the certificate has been issued). Note that different measurement types can be used in conjunction [B20], compensating each other's drawbacks and thereby resulting in a higher quality dataset.

[9] A TLS handshake involves the *Server Name Indication* extension of TLS [B18], i.e.,, a field that contains the domain name that the client intends to connect to.

5.4 Methodology

To obtain a manageable dataset of certificates (in terms of the cost of analyzing these certificates), we collect certificates issued for a sample of three domain types. An exploratory analysis of all certificates available from the CT logs is considered infeasible, as they contain over twelve billion certificates. Rather than considering Fully Qualified Domain Names (FQDNs), we are primarily concerned with analyzing *root domains*, the part of the FQDN under which the domain is registered by its owners[10]. Certificates issued for domains under this root domain are generally requested by the domain owner since the domain owner is tasked to demonstrate ownership of the domain to the CA, although newer verification methods (such as an HTTP-based method) challenge this assumption. The aforementioned three domain types used for analysis are defined as follows:

- *Phishing domains* (D_{phish}) are those domains that have historically been used in phishing attacks.
- *Popular benign domains* (D_{pop}) represent the set of domains used for highly popular services that receive the majority of traffic on the Internet.
- *Non-popular benign domains* (D_{nonpop}) are domains that receive a low amount of traffic, yet have not been seen as part of phishing attacks.

For each domain type, we sampled 10,000 domains, resulting in a total number of 30,000 domains for which certificates were collected. Figure 1 shows the process to retrieve the sample of domains and their associated certificates. The *domain sampling* step consists of converting the source domains into a set of sampled, labeled domains. In the *certificate retrieval* step, all certificates for these sampled domains are retrieved, which in turn are analyzed further.

Domain sampling The Tranco list [B21] combines three other domain lists to provide a robust list of the top one-million popular domains, in terms of the popularity of those domains[11]. We assume that the domains on the top of this list are inherently benign (i.e.,, they were not registered for malicious purposes), under the assumption that a maliciously registered domain will never become popular enough to receive high amounts of traffic. The top of the list therefore represents popular, benign domains.

[10] For instance, the root domain for the FQDN `www.example.co.uk` is `example.co.uk`

[11] The definition of 'popularity' differs slightly between the three source lists, but generally represents the amount of traffic the domain receives.

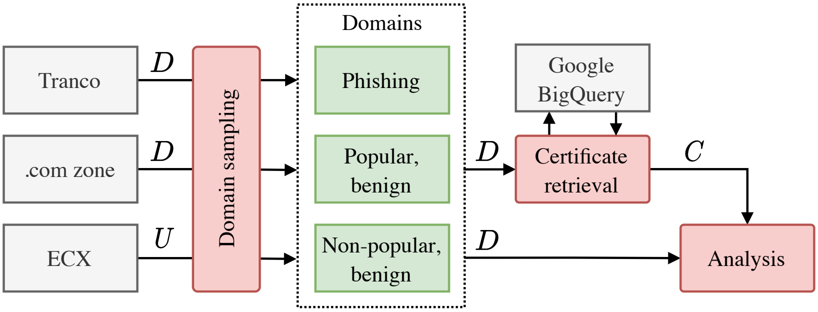


Figure 1: The process of retrieving the three samples of domains and their associated certificates. The grey boxes represent the source data sets, the green boxes represent three domain types and the red boxes represent the actions performed by the authors. D = set of domains, U = set of URLs, C = set of certificates.

The APWG's eCrime eXchange (eCX) platform[12] contains millions of known phishing URLs submitted by contributing organizations, whenever the submitted URL is found to host phishing content. The root domains extracted from these URLs form the base of the phishing domain sample, denoted as D_{ecx}. Some popular domains are often abused for hosting phishing content, such as Facebook or Google Docs. This however does not imply those domains are registered for malicious purposes; the content on these sites is merely user-generated and therefore contains a mix of malicious and benign content. The set of *phishing domains* is defined as $D_{phish} = D_{ecx} \setminus D_{pop}$, or the root domains extracted from eCX URLs, excluding any domain that is in the top 1M Tranco domains.

Lastly, the non-popular, benign domain type is defined as $D_{nonpop} = D_{com} \setminus (D_{ecx} \cup D_{pop})$, where D_{com} is the set of all domains in the `.com` zone (comprising almost half of all domains globally). None of these domains receive a large amount of traffic and have never been seen in a phishing attack according to the eCX platform.

Certificate retrieval We collect any certificate submitted to the CT log ecosystem that covers (at least) one of the domains in the domain sets. This collection is done through a Google BigQuery dataset provided by the Censys search engine [B22]. The criterion for a certificate to cover a domain name is that

[12] `https://apwg.org/ecx/`

the SANs list of the certificate contains (1) the root domain itself, (2) a subdomain of the root domain, or (3) a wildcard domain under the root domain. In addition to the raw certificates, we also collect several extra fields, including the fingerprint of the certificate, and the validity of the certificate according to different root stores.

5.5 Results

All four datasets (i.e.,, the Tranco list, the .com zone file, the eCX URLs, and certificates) were collected in the beginning of 2020. We used the Tranco list from February 20th, 2020[13]. The list of eCX URLs comprises over 8.6 million URLs (belonging to 1.02 million unique root domains), and contains URLs discovered up to February 27th, 2020. The certificates from Censys were collected on March 12th, 2020. An overview of the resulting dataset is shown in Table 1. The vast majority of the 79.1 million collected certificates were issued for popular domains. This is partially explained by the fact that only a small fraction of popular domains had no certificate issued for it, compared to 4,235 phishing domains, and 6,283 of the non-popular domains. In our further analysis, we discard certificates that are *untrusted* by browsers, as those certificates cause browsers to present users a warning page, instead of the actual content. The purpose of issuing certificates in the first place is to make pages seem legitimate, which is defeated when a warning page is shown. A certificate can be untrusted for various reasons, such as being self-signed or being signed by a root certificate that a particular browser or operating system does not trust. This filtering of untrusted certificates primarily affects the popular domains, with nearly 60% of all certificates issued for those domains being untrusted. The table suggests that phishing domains have adopted HTTPS to a higher degree than non-popular domains, with nearly twice the number of certificates issued for them.

5.5.1 Temporal analysis

As previously stated, one advantage of CT log analysis over passive and active measurements is the presumed early submission and publication of certificates to the CT logs. Using CT logs as a data source for early detection is only viable if certificates can consistently be monitored prior to the blacklisting of the domains they cover, and as such we test the following hypothesis:

Hypothesis 1. *Certificate issuance occurs prior to blacklisting of phishing domains*

[13] Available at `https://tranco-list.eu/list/XWNN`.

Table 1: Overview of the collected dataset.

	# domains	# domains without cert	# certificates	% certificates trusted
Phishing	10.0k	4.2k	184.9k	95.9
Popular	10.0k	175	78.9M	40.4
Non-pop.	10.0k	6.3k	95.6k	99.0
Total	30.0k	10.7k	79.1M	

We analyze the time difference between (1) the blacklisting of a phishing domain and (2) the issuance of its certificate(s). This process is made more difficult because a domain can both have been blacklisted multiple times, and have multiple certificates issued for it. The former challenge is tackled by selecting the first timestamp of blacklisting seen for that specific domain, whereas the latter is tackled by introducing the notion of the *closest certificate*. For each timestamp of first blacklisting, the most recent certificate issuance *before* blacklisting is considered. If no certificate before blacklisting exists, the oldest certificate *after* blacklisting is considered. The first motivation for this definition is that the most recent certificate before blacklisting is most likely to be issued by the same owner of the domain at the time of blacklisting, as older certificates may have been issued by a previous owner of the domain. Secondly, certificates issued after blacklisting are meaningless for early protection against phishing attacks abusing that domain, hence the low priority of including those certificates. This process is illustrated in Figure 2.

Certificates issued before the first blacklisting date are not necessarily requested by the same domain owner that caused the domain to be blacklisted. Given the decade-spanning time window of the CT logs, it is possible to observe a certificate issued for a domain ten years before it got blacklisted. Since the identification of domain ownership is inherently a difficult task, we instead estimated a lower and upper bound for the number of phishing domains for which we can identify a relevant certificate, issued before the domain got blacklisted.

Upper bound For all phishing domains in the dataset, the time difference between the issuance of the closest certificate (t_C) and the earliest URL blacklisting timestamp (t_B) is computed (denoted as $\Delta(t_C, t_B)$). A negative time difference implies that the closest certificate was issued before the URL was blacklisted. For 75.66% of all phishing domains, the earliest certificate timestamp occurs

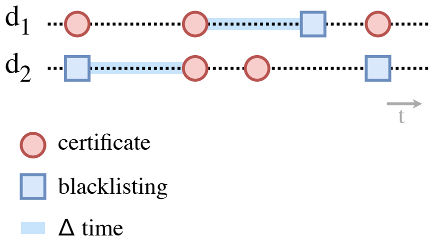


Figure 2: The computation of the time difference between domain blacklisting and the closest certificate issuance. For domain d_1, the most recent certificate *before* the first blacklisting timestamp is taken (resulting in a negative time difference), whereas for domain d_2 the earliest certificate issuance timestamp *after* the first blacklisting timestamp is taken (resulting in a positive time difference).

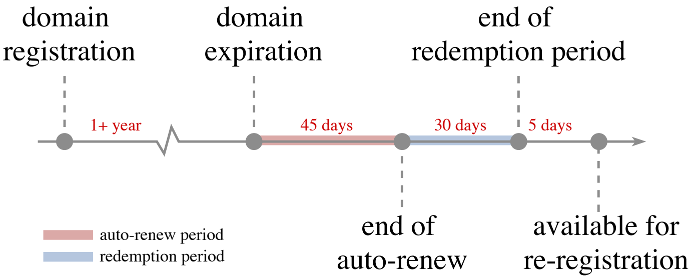


Figure 3: Timeline of the re-registration process of a domain name.

before the earliest blacklisting timestamp, which serves as the upper bound of the percentage of domains a CT log-based phishing domain detection system can protect against.

Lower bound In order to estimate a lower bound, the registration process of a domain is taken into account. Since domains can change their ownership during their lifecycle, it is vital to only consider certificates issued by the same owner that owned the domain during the time of blacklisting. A domain registered under the `.com` top-level domain (TLD) goes through a process after a domain expires, during which the control of the domain is taken back

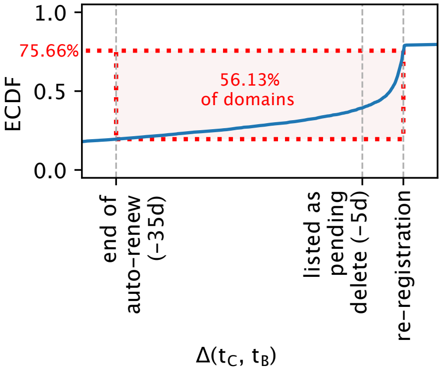


Figure 4: The ECDF of the time difference between the issuance of the closest certificate and the earliest URL blacklisting timestamp, for each phishing domain. The figure is zoomed in on the period around the auto-renew and registration period of a domain.

by the registrar, and after which a domain can potentially change owner[14]. An auto-renew period of 45 days is followed by a 30-day redemption period, after which the domain is listed as 'pending delete' for 5 days. After this period, the domain is up for re-registration [B23] (see Figure 3). In the auto-renew period, domain owners have the opportunity to renew or sell their domain, and as such a domain cannot change ownership in the 35-day period between the end of the auto-renew period, and the earliest potential to re-register a domain. Given a domain for which a URL was blacklisted at an arbitrary timestamp $t = 0$, all certificates issued in the period $[-35 \text{ days}, 0]$ are requested by the same owner that owned the domain at the time of blacklisting. Figure 4 shows a zoomed-in portion of the Empirical Cumulative Distribution Function (ECDF) of $\Delta(t_C, t_B)$ of all phishing domains, and shows that 56.13% of domains had their closest certificate issued in this period for which there exists the previously described certainty about the ownership of the domain. As such, this percentage is considered the lower bound. The figure also illustrates the aforementioned upper bound.

These results indicate that for between 56.13% and 75.66% of phishing domains, a certificate issuance can be observed before a URL for that domain is being blacklisted. Even though this implies that perfect coverage of all domains does not seem feasible, it is important to note that CT logs cover

[14] Note that other TLDs may have different processes regarding the expiration of a domain, but given the large market share of `.com`, their expiration process drives our methodology

domains across all TLDs and their coverage is arguably only getting higher due to the ever-increasing adoption of HTTPS. Therefore, CT log-based detection of phishing domains could potentially cover a larger set of domains than active or passive measurement-based methods.

5.5.2 Issuers

Each certificate in circulation has been signed by a CA, which has deliberately been approached by the requester of the certificate (through the submission of a certificate signing request). As such, the choice of CA of a certificate reflects the behavior of the domain owner. CAs have individual differences, such as pricing models, countries in which they operate, the process of verifying the identity of the requester, etc. As these differences may be important considerations for phishers, we test the following hypothesis:

Hypothesis 2. *The CA selection of phishing domains provides insight in the monetary considerations, and the operations of phishers*

In reality, it is not trivial to identify the underlying CA that signed a certificate, as certificates are generally signed by intermediate certificates[15], and a single CA may operate many intermediate certificates. In addition, *cross-signing* of certificates is not uncommon, in which the root certificate of a CA signs the intermediate certificate of another CA, resulting in a chain of trust rooted in more than one CA. We therefore analyze the issuer of certificates rather than identifying the underlying CA, as the issuer selection may provide a similar insight into phishing as the CA selection would.

In order to analyze the selection of CA, it is important to handle any potential biases that can be induced. More specifically, it is imperative that a single domain should not dominate other domains due to the number of certificates issued for it (as demonstrated earlier in Table 1), or a specific issuer to dominate others due to the short validity period of their certificate (thereby requiring customers to frequently re-issue certificates).

The set of certificates issued for domain d_i issued by issuer a_j is formalized as $C_{issuer}(d_i, a_j)$. The duration of an individual certificate c's validity period is denoted as $v(c)$. The set of all existing issuers is $\mathscr{A}$. Using these formulations, we define *domain issuer preference* $\mathscr{P}_{issuer}(d_i, a_j)$, which represents how much

[15] These intermediate certificates are in turn signed by a root certificate or another intermediate certificate

Table 2: The domain type issuer preference (as percentage) for the ten most preferred issuers and the three domain categories.

a_j	$\mathscr{P}_{issuer}(D, a_j)$ (in %) D_{phish}	 D_{pop}	 D_{nonpop}
Let's Encrypt Authority X3	*46.19	*15.70	*61.46
cPanel, Inc. Certification Authority	*33.53	1.23	*10.69
COMODO ECC Domain Validation Secure Server CA 2	*7.94	5.23	*4.51
DigiCert SHA2 Secure Server CA	0.06	*6.88	0.15
Amazon	0.14	*5.91	0.73
Go Daddy Secure Certificate Authority - G2	1.91	4.50	4.21
CloudFlare Inc ECC CA-2	3.65	1.33	4.43
GlobalSign CloudSSL CA - SHA256 - G3	0.32	4.23	0.06
DigiCert SHA2 High Assurance Server CA	0.01	4.09	0.02
Sectigo RSA Domain Validation Secure Server CA	1.38	1.55	4.03

*among the three most preferred authorities within the domain category

a particular issuer is preferred by a domain:

$$\mathscr{P}_{issuer}(d_i, a_j) = \frac{\sum\limits_{c \in C_{issuer}(d_i, a_j)} v(c)}{\sum_{a \in \mathscr{A}} \sum_{c \in C_{issuer}(d_i, a)} v(c)} \tag{5.1}$$

This measure is a value between one and zero, and $\sum_{a \in \mathscr{A}} \mathscr{P}_{issuer}(d_i, a) = 1$. This definition is used to express the *domain type issuer preference* for domain type k towards an issuer a_j:

$$\mathscr{P}_{issuer}(D_k, a_j) = \frac{\sum_{d \in D_k} \mathscr{P}_{issuer}(d, a_j)}{\sum_{a \in \mathscr{A}} \sum_{d \in D_k} \mathscr{P}_{issuer}(d, a)} \tag{5.2}$$

Where D_k is the set of all domains for domain type k. Similar to Eq. 5.1, the following holds: $\sum_{a \in A} \mathscr{P}_{issuer}(D_k, a) = 1$. Eq. 5.1 ensures that certificates are weighted proportionally to their validity period, and that each domain within a domain type is equally weighted disregarding the number of certificates issued for the domain, whereas Eq. 5.2 ensures a proper comparison between domain types.

Using Eq. 5.2, we compute the domain type preferences for all combinations of issuers and domain types, of which the results are illustrated in Table 2. The top 10 most preferred issuers are shown in the table. In total, 853 issuer certificates were used to sign the set of certificates, of which 846 were used

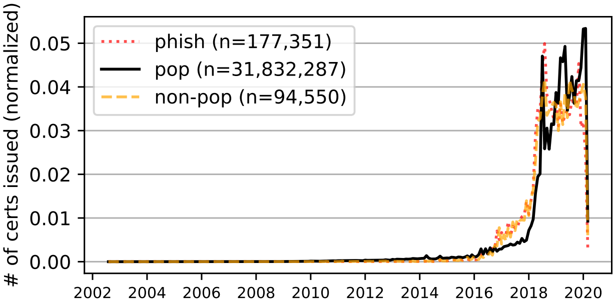


Figure 5: The distribution of the month of issuance of certificates for the three domain types.

for popular domains, 132 for phishing domains and only 125 for non-popular domains. Generally, the preference for phishing domains and non-popular domains are fairly similar, with the popular domains having a vastly different preference profile.

Although *Let's Encrypt* is the most preferred issuer for popular domains (with 15.7%), it is not nearly as popular as the other two domain types (46.19% and 61.46% for phishing and non-popular domains respectively). One explanation could be that *Let's Encrypt* is a relatively new CA, and popular domains tend to be long-lived, resulting in a stronger preference for CAs that have been operating longer, or for now-defunct CAs. Figure 5 shows the distribution of the month of issuance of certificates for the three domain types. The figure shows that the vast majority of certificates are issued after 2018 and that there is no major difference between the domain types, suggesting that the age of *Let's Encrypt* does not play a large role in the issuer preference.

The issuer preference provides information regarding the infrastructure used to serve services using popular domains, of which the high preference for the *Amazon* issuer by popular domains (5.91%, or the third-most preferred issuer) is an example. Amazon issues public certificates for customers of their AWS cloud platform, which specifically focuses on “securing public websites with significant traffic requirements"[16], indicating that part of the infrastructure of popular domains with such certificates operates on AWS. Similarly, we find that phishing domains have a strong preference for the *cPanel* issuer (33.53%) compared to popular (1.23%) and non-popular (10.69%) benign domain types. cPanel is a popular web hosting platform with the option for

[16] `https://docs.aws.amazon.com/acm/latest/userguide/acm-overview.html`

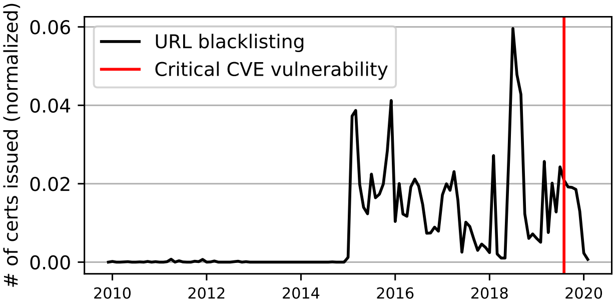


Figure 6: Distribution of the month of URLs blacklisted for phishing domains with cPanel certificates (black). The month of the discovery of several critical vulnerabilities is displayed in red.

automatic issuance and deployment of TLS certificates. We hypothesize that either (1) phishers rely on cPanel for their hosting infrastructure on a large scale, or (2) that cPanel accounts are compromised at a large scale and repurposed for conducting phishing attacks. Under the assumption that the second hypothesis can only occur on a large scale in case of the discovery of a critical vulnerability, we would expect to see a peak of submissions of phishing URLs on the eCX for domains running cPanel software, as cPanel hosts are compromised around the time of the vulnerability disclosure. Several critical vulnerabilities were reported for cPanel in September 2019[17], but according to Figure 6 this did not coincide with unusual numbers of related URLs being reported. We have found no evidence of peaks of URL submissions for phishing domains running on cPanel software, suggesting that the first hypothesis holds true, but this requires further research to confirm.

5.5.3 Validation level

In addition to the CA selection, the selection of a validation level provides insight in the monetary investment in conducting phishing attacks, which leads to the following hypothesis:

Hypothesis 3. *Phishers resort to certificates with higher validation levels to increase the perceived legitimacy of phishing websites.*

[17] We used a CVE record with a score of 8 or higher as the definition of a critical vulnerability, obtained from `https://www.cvedetails.com/vulnerability-list/vendor_id-1766/product_id-3023/Cpanel-Cpanel.html`

Similar to the issuer analysis, the analysis of validation level differences requires a normalization step taking into account the number of certificates and the validity duration of certificates. The set of all certificates issued for domain d_i with a validation level l_j is denoted as $C_{val}(d_i, l_j)$. The set of possible validation levels (i.e.,, DV, OV, EV, and unknown) is denoted as $\mathscr{L}$. The *domain validation level preference* $\mathscr{P}_{val}(d_i, l_j)$ represents how much a particular validation level is preferred by a specific domain:

$$\mathscr{P}_{val}(d_i, l_j) = \frac{\sum\limits_{c \in C_{val}(d_i, l_j)} v(c)}{\sum_{l \in \mathscr{L}} \sum_{c \in C_{val}(d_i, l)} v(c)} \tag{5.3}$$

Again, $v(c)$ denotes the duration of the validity period of certificate c, and this measure is a value between one and zero, and $\sum_{l \in \mathscr{L}} \mathscr{P}_{val}(d_i, l) = 1$. Then, the *domain type validation level preference* for each of the domain categories D_k (i.e.,, D_{phish}, D_{pop} and D_{nonpop} for phishing, popular and non-popular sampled domains respectively) for a specific validation level l_j is defined as follows:

$$\mathscr{P}_{val}(D_k, l_j) = \frac{\sum_{d \in D_k} \mathscr{P}_{val}(d, l_j)}{\sum_{l \in \mathscr{L}} \sum_{d \in D_k} \mathscr{P}_{val}(d, l)} \tag{5.4}$$

We rely on the reported validation levels from the Censys dataset[18] to compute the preferences. Table 3 shows the results of the computation of these values, for all domain type and validation level combinations. Unsurprisingly, popular domains have a stronger preference for OV and EV certificates, as popular domains have a larger incentive and budget for protecting the legitimacy of their brand. Phishing and non-popular benign domains have a highly similar preference.

Notably, we identified 133 EV certificates issued for 16 unique phishing domains. Further inspection shows that only two of those 16 domains are marked by VirusTotal as malicious, questioning whether the other 14 are really phishing domains in the first place. As such, we conclude that EV certificates are virtually unused by phishers in our dataset, and this suggests that EV certificates are in fact a meaningful method for marking websites as trustworthy, even though browsers stopped presenting such indicators to users.

[18] We verified the correctness of these values by computing a validation level based on the existence of specific Object Identifiers (OIDs) in the X.509 certificate extensions and matching those against the Censys values. Our method in general led to more inconclusive results.

Table 3: Values of $P_{val}(D, l)$ for all domain categories and validation levels (as percentage).

$\downarrow D \rightarrow l$	DV	OV	EV	Unknown
D_{phish}	92.94	6.65	0.11	0.31
D_{pop}	49.41	39.17	4.03	7.38
D_{nonpop}	91.99	7.28	0.14	0.59

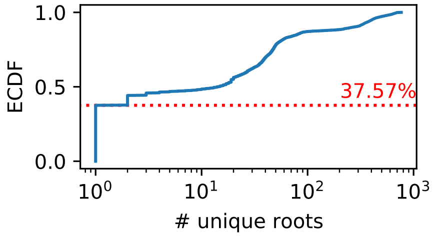


Figure 7: ECDF of the number unique roots of certificates covering at least a single phishing domain ($n = 975, 426$). The red line represents the fraction of covering only a single unique root domain

5.5.4 Certificate sharing

Based on the following hypothesis, there is an expectation that phishing domains may be deployed on a shared infrastructure, which would be reflected by *shared certificates*, or certificates that cover more than a single phishing domain.

Hypothesis 4. *Individual phishing attacks may be part of a larger campaign, and the deployment of those attacks is coordinated.*

In order to test this hypothesis, we collect all certificates in our dataset that cover at least a single phishing domain. Note that we consider all phishing domains, not only the 10,000 domains in the initial sample. In total, 975k certificates were identified. For each certificate, we extract the unique root domains, and the ECDF of the count is shown in Figure 7. 37.57% of certificates issued for phishing domains cover only a single unique root, indicating that the remaining 62.43% (or 608,934 certificates) is potential evidence that certificate sharing is rampant in phishing attacks.

Surprisingly, only 0.90% (or 5,452) of these certificates cover *only* phishing roots, meaning that the remaining certificates cover not only phishing roots, but also domains of unknown nature. The implication of this is that there is a small portion of certificates that can be considered unambiguously 'phishy', whereas the vast majority of certificates cannot. The existence of these 'ambiguous' certificates could be interpreted in several ways, including:

1. Certificates are requested by a third party to which domain control is delegated by the actual owner, and this third party requests certificates that cover benign and phishing domains.

2. There exists a vast number of domains that have not yet been discovered to be phishing domains, which means that in reality these ambiguous certificates are in fact 'phishy'.

3. Not all domains owned by a particular domain owner are used for phishing attacks. This includes for example benign domain owners whose websites are hacked and repurposed for phishing attacks, or phishers who manage domains for non-phishing activities, and domains with user-generated content of mixed nature.

We investigate the first point through a case study: Cloudflare. Cloudflare's Content Delivery Network (CDN) enables customers to encrypt traffic to their websites by proxying traffic through Cloudflare servers. This requires them to change the DNS name server of their domain to Cloudflare, effectively delegating the control of their DNS configuration to Cloudflare, which is used by the ACME protocol to verify the ownership of domains. Rather than requesting a single certificate for each domain, a single certificate is requested for a set of domains, all from different owners. In addition, these certificates contain a Cloudflare-specific domain (usually the first domain name in the list of SANs), matching the structure `sni{6 digits}.cloudflaressl.com`. Naturally, none of these certificates can unambiguously be assumed to be phishy, as they cover domains from many different owners. This set of Cloudflare certificates alone already comprises 91.369 certificates, or 9.37% of the full set of certificates issued for phishing domains. We identified other similar certificate structures to Cloudflare certificates, that include SANs such as `statuspage.io` and `incapsula.com`, leading us to believe that those domains are used for similar purposes as the `sni{6 digits}.cloudflaressl.com` domains. Further research is needed to fully differentiate 'phishy' certificates from these CDN-type certificates.

5.6 Discussion

The fact that between 56.13% and 75.66% of phishing domains observe a relevant certificate being issued before a URL is blacklisted is a promising result and a strong motivation for pursuing the development of CT log-based abuse prevention systems (accepting Hypthesis 1). Given the constant growth of phishing attacks being conducted over HTTPS (over 84% of identified phishing attacks in the last quarter of 2020 according to the APWG [B2], compared to only 10% in the first quarter of 2017), CT log's early coverage of phishing domains is only expected to grow in the future. Simultaneously, the introduction of more traffic encryption methods, such as ECN[19] could render passive measurements less effective, emphasizing the relevance of CT logs even more.

We have demonstrated that phishing domains and non-popular benign domains have similar preference profiles for the issuer selection of their certificates (except *cPanel*), but a vastly different profile compared to popular domains. Our results provide some insight in the operations of phishing domains (e.g.,, preference for *cPanel*), thereby we accept Hypothesis 2. Further research is required to draw stronger conclusions regarding the domains relying on *cPanel*, but these results emphasize the importance of the position of CAs in the fight against phishing. CAs have the opportunity to interfere with the TLS ecosystem through the revocation of certificates and in fact some CAs state in their policies that malicious activity is grounds for certificate revocation[20]. Although the effectiveness of certificate revocation has historically been limited [B24], our results are an argument for better handling of certificate revocation.

Our results for the validation level of certificates show that phishers rarely resort to EV certificates (thereby rejecting Hypothesis 3). Given the relatively high cost of OV and EV certificates ($27.44 and $72.18 per year at COMODO respectively for example[21]), this suggests phishers are not willing to significant amounts of money, or are actually rejected during the vetting process. Even though browser indicators have been demonstrated to be ineffective from the perspective of users [B4], higher validation levels could be an effective signal for identifying certificates that are not used in phishing attacks. EV certificates could act as a white-listing method for identifying 'benign' certificates.

Lastly, we found few unambiguous cases of shared certificates between phishing domains, with 5,452 certificates *only* covering phishing domains.

[19] https://tools.ietf.org/html/draft-ietf-tls-esni-10#page-6

[20] Example of Sectigo: https://sectigo.com/uploads/files/Sectigo-CPS-v5_2_2.pdf

[21] https://comodosslstore.com/resources/dv-vs-ov-vs-ev-ssl-which-certificates-are-good-for-site-security/

The vast majority of certificates that cover at least a single phishing domain also covers domains whose nature is unknown, which makes it difficult to extract meaningful information from the certificates about the decisions of the phishers. A substantial part of this challenge can be explained by the practice of domain owners delegating control to CDNs, which aggregate many domains from many owners, and obtain certificates covering many domains of various types, including phishing. We found that 9.37% of the phishing domains had been mixed with other domains by Cloudflare, and indications that other CDNs apply similar practices. Consequently, CDNs have a role and a responsibility in the fight against phishing, when they offer the service of managing certificates, and in general when applying practices that allows phishers to mix with benign domain owners. We cannot reject nor accept Hypothesis 4, which remains inconclusive.

This work provides preliminary characteristics of phishing certificates, suggesting that a there is a certain degree of 'phishyness' that can be assigned to a certificate. Even a simple heuristics-based warning system can potentially be useful for identifying candidate phishing domains, by for example identifying *cPanel* certificates covering several domains, including a domain that previously already was blacklisted by the eCx.

Limitations Our methodology differentiates between three domain types, and considers phishing domains a homogenous group of domains registered for phishing purposes. Even though we accounted for domain ownership changes in the collection of certificates for phishing domains, we do not address the potential of domain compromise. Maroofi *et al.*. [B25] manually collected a dataset of phishing URLs and identified 58% to be maliciously registered and 42% to be compromised. This implies that there is a likelihood that our analysis of phishing domains leads to conclusions for compromised domains, rather than maliciously registered domains. Identifying whether a domain is compromised in itself is already a challenging task, which is significantly more difficult with historical data, where the website may not be online anymore.

It is possible that the domain type samples contain false positives of domains that in reality are placed in the wrong class. One cause of this could be the (lack of) vetting of URLs in the eCX platform, which could result in URLs submitted to the platform that are in reality not phishing URLs. In addition, it is unlikely that the eCX URLs are fully complete, covering every phishing URL in existence, as not all phishing attacks are detected and reported. Additionally, the eCX relies on member contributions, and these are unlikely to be fully complete. As a result, there are likely to be domains in the non-popular domain set that are in reality phishing domains. Unfortunately, this is a core limitation of the CT framework for phishing prevention, as TLS certificates are issued on a domain basis instead of on a URL basis.

5.7 Conclusions

This paper addresses the potential of using the phishyness of digital certificates as a method to identify phishing domains early in their lifecycle. By comparing the certificates issued for three distinct domain sets, we identify relevant patterns in the differences across these domain sets. Firstly, our temporal analysis shows that for 56.13% to 75.66% of phishing domains, a certificate is issued before the domain is being blacklisted, indicating the scale at which CT-based mitigation can protect against phishing. Furthermore, our results show that phishing domains resort to a relatively small group of issuers, particularly gravitating to *cPanel*, which emphasizes that stronger adherence to certificate revocation lists produced by these issuers can be highly valuable. We have also shown that phishers are unlikely to resort to (expensive) EV certificates, which could suggest that domains that *do* employ them could serve as a whitelist for non-phishing domains. Lastly, we found that certificates are unlikely to be unambiguously phishy or benign, given the set of phishing domains they encompass. Although we identified only a few certificates that *only* cover phishing domains, the majority of certificates issued for phishing domains cover multiple phishing domains, which is a hopeful takeaway. These results led us to provide several suggestions for changes to the TLS ecosystem.

Our work opens several pathways for future work. Firstly, given the preliminary nature of our results, we encourage the research community to integrate our results in novel or existing domain classification methods. An alternative promising direction is the disambiguation of certificates, which - if successful - could lead to a very effective way to propagate the 'phishyness' of a certificate to all the domains it covers. Additionally, the potential impact of CAs (*cPanel* in particular) and certificate revocation could be explored in more detail.

ACKNOWLEDGEMENTS

This research is carried out under the SecDNS project, funded by Innovation Fund Denmark. We thank Censys.io for sharing their CT log data with us, and we are thankful for the APWG for granting us access to their eCX platform.

References

[B1] E. E. Lastdrager, "Achieving a consensual definition of phishing based on a systematic review of the literature," *Crime Science*, vol. 3, Sept. 2014.

[B2] Anti-Phishing Working Group, "Phishing Activity Trends Report - 4th Quarter 2020." `https://docs.apwg.org/reports/apwg_trends_report_q4_2020.pdf`, February 2021. Accessed: 2021-02-12.

[B3] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC 5280, RFC Editor, May 2008. `http://www.rfc-editor.org/rfc/rfc5280.txt`.

[B4] C. Thompson, M. Shelton, E. Stark, M. Walker, E. Schechter, and A. P. Felt, "The web's identity crisis: understanding the effectiveness of website identity indicators," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1715–1732, 2019.

[B5] CA/Browser Forum, "Baseline requirements for the issuance and management of publicly-trusted certificates (version 1.7.3)," tech. rep., CA/B Forum, October 2020. `https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.7.3.pdf`.

[B6] J. Prins, "DigiNotar certificate authority breach operation black tulip." `https://media.threatpost.com/wp-content/uploads/sites/103/2011/09/07061400/rapport-fox-it-operation-black-tulip-v1-0.pdf`, 2011. Accessed: 2021-07-23.

[B7] Q. Scheitle, O. Gasser, T. Nolte, J. Amann, L. Brent, G. Carle, R. Holz, T. C. Schmidt, and M. Wählisch, "The rise of Certificate Transparency and its implications on the Internet ecosystem," in *Proceedings of the Internet Measurement Conference 2018*, IMC '18, (New York, NY, USA), p. 343349, Association for Computing Machinery, 2018.

[B8] E. Fasllija, H. F. Enişer, and B. Prünster, "Phish-hook: Detecting phishing certificates using Certificate Transparency logs," in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 320–334, Springer International Publishing, 2019.

[B9] Y. Sakurai, T. Watanabe, T. Okuda, M. Akiyama, and T. Mori, "Discovering HTTPSified phishing websites using the TLS certificates footprints," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 522–531, IEEE, Sept. 2020.

[B10] 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*, IMC '13, (New York, NY, USA), p. 6376, Association for Computing Machinery, 2013.

[B11] 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*, CCS '16, (New York, NY, USA), pp. 1568–1579, Association for Computing Machinery, 2016.

[B12] G. C. M. Moura, M. Muller, M. Wullink, and C. Hesselman, "nDEWS: A new domains early warning system for TLDs," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2016)*, pp. 1061–1066, IEEE, Apr. 2016.

[B13] 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 *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706, IEEE, May 2016.

[B14] A. Manousis, R. Ragsdale, B. Draffin, A. Agrawal, and V. Sekar, "Shedding light on the adoption of Let's Encrypt," 2016.

[B15] M. Aertsen, M. Korczyński, G. C. M. Moura, S. Tajalizadehkhoob, and J. van den Berg, "No domain left behind: is Let's Encrypt democratizing encryption?," in *Proceedings of the Applied Networking Research Workshop*, ACM, July 2017.

[B16] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman, "Towards a complete view of the certificate ecosystem," in *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, (New York, NY, USA), pp. 543–549, Association for Computing Machinery, 2016.

[B17] L. Zhang, D. Choffnes, D. Levin, T. Dumitraş, A. Mislove, A. Schulman, and C. Wilson, "Analysis of SSL certificate reissues and revocations in the wake of Heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, (New York, NY, USA), p. 489502, Association for Computing Machinery, 2014.

[B18] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright, "Transport layer security (TLS) extensions," RFC 3546, RFC Editor, June 2003. Accessed: 2021-02-12.

[B19] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS usage in Android apps," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pp. 350–362, ACM, Nov. 2017.

[B20] R. Holz, J. Amann, A. Razaghpanah, and N. Vallina-Rodriguez, "The era of TLS 1.3: Measuring deployment and use with active and passive methods," *CoRR*, vol. abs/1907.12762, 2019.

[B21] V. L. Pochat, T. V. Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings 2019 Network and Distributed System Security Symposium*, Internet Society, 2019.

[B22] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 542–553, ACM, Oct. 2015.

[B23] 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 *26th USENIX Security Symposium (USENIX Security 17)*, (Vancouver, BC), pp. 865–880, USENIX Association, Aug. 2017.

[B24] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's PKI," in *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, (New York, NY, USA), p. 183196, Association for Computing Machinery, 2015.

[B25] S. Maroofi, M. Korczyski, C. Hesselman, B. Ampeau, and A. Duda, "COMAR: Classification of compromised versus maliciously registered domains," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 607–623, 2020.

Chapter 6

Detecting Ambiguous Phishing Certificates using Machine Learning

Paper C

Sajad Homayoun, Kaspar Hageman, Sam Afzal-Houshmand,
Christian Damsgaard Jensen, Jens Myrup Pedersen

The paper was submitted to the
The 36th International Conference on Information Networking (ICOIN 2022)

The layout of this work has been revised.

6.1 Introduction

Phishing is still one of the most common attacks which has a plethora of reported cases on a daily basis [C1]. In these attacks, victims are persuaded in disclosing sensitive information by an attacker that impersonates a legitimate organization or person. Modern and more advanced attackers have begun to use phishing websites offering legitimate digital certificates as a way to persuade the users into diverging their information willingly, by making the websites look more legitimate through visible padlock icons in browsers. Although phishing websites with legitimate certificates can be more effective than their unencrypted counterparts, they have the drawback from the attacker's perspective that they have are visible in publicly available certificate transparency (CT) logs [C2]. The inclusion of certificates in these logs is a requirement to be a trusted website in most modern mainstream browsers [C3]. Monitoring CT logs for newly-issued certificates may enable the detection of malicious phishing websites using legitimate certificates in the earlier stages of the attack. In fact, prior research has proposed automated solutions for detecting digital certificates being involved in phishing attacks [C4–C7]. Most of this research has tackled this as a binary classification problem, with certificates being considered either malicious or benign. Identifying a high-quality ground truth is challenging, because the label of a certificate tends to be based on the label of the domain name or website that serves a given certificate. Not only is the nature of the domain not always clear-cut benign or malicious [C4,C8], but there may also be conflicting domain labels for a given certificate [C9].

In this paper, we propose a novel machine learning technique to address the issues with this task in the state of the art. In contrast to prior work, our solution considers the classification of certificates to be a multi-class classification problem and is capable of finding certificates that fit neither the traditional benign or malicious labels, but are rather considered ambiguous or 'conflicted'. More specifically, the contributions in this paper are summarized as follows:

- we combine both features described in prior work and newly-introduced features,
- we present a novel labeling mechanism that takes into account the individual labels of the domains that are being covered by a certificate,
- we study how our features can predict a phishyness score for each certificates,
- we validate our methods using a Time-based cross validation scheme

and show that our classifier and regression models achieve a high performance.

The remainder of this paper is divided as follows. Section 6.2 gives brief background on digital certificates and Section 6.3, we review some related works. Section 6.4 describes our data collection pipeline, which extracts certificate related and domain related features. In Section 6.5 presents our classification scenario, and Section 6.6 explained our regression scenario. In Section 6.7, we evaluate our scenarios using a time-based approach. We discuss the achievements and limitations in Section 6.8. Finally, Section 6.9 concludes the paper.

6.2 Background

The Transport Layer Security (TLS) protocol suite provides encryption functionality that other networking protocols such as HTTP and IMAP. Besides encrypting the communication between a client and server, TLS guarantees authenticity by operating under a public key infrastructure (PKI). When establishing a connection, a server provides a digital certificate (as defined in the X.509 format [C10]), which consists of a public key (whose private key is only known to the server operator), a set of identities (usually one or more domain names) and a signature from a trusted third-party. These third parties, or Certificate Authorities (CA), only sign new certificates after successfully validating that the requestor of the certificate can demonstrate ownership of all identities to be included in the certificate. Clients can verify the authenticity of a certificate by validating that the signature was created using a private key that the client inherently trusts, as part of the certificate root store installed on their system. Originally, a certificate contained a single *subject* field, which indicated for what domain name the certificate was valid. An extension was later introduced to support multiple domain names to be included, named the `Subject Alternative Name` extension, and the domain names embedded in a certificate are therefore commonly referred to as SANs.

In 2011, two CAs issued certificates for high-profile domain names for malicious actors, which allowed these actors to perform large-scale impersonation attacks. As a response, the Certificate Transparency (CT) framework [C2] was developed with the intention of monitoring the certificate issuance behavior of the CAs. In this framework, CAs are encouraged to submit newly-issued certificates to CT logs, publicly-available repositories of certificates. Browser vendors started to require certificates to be included in these logs, and as a results the CT logs capture nearly all certificates that are issued worldwide.

6.3 Related Work

Artificial intelligence based approaches for classifying malicious certificates have been studied during the past few years. Inspired by classifiers for URL detection, there have been works on classifiers that utilize the information that can be gained from the certificates itself which is encompassed in the work presented in this paper. Mohammad *et al.* provide a dataset that illustrates how using the certificate issuer from HTTPS information can be used to high avail which has bled over to multitudes of other works [C11]. It ought to be noted that certificate information often needs additional information extraction of viable features due to the limited information offering as compared to URL databases[C12].

Across works in the area of detecting phishing websites based on certificates, there is a recurring theme of feature engineering. Mishari *et al.* proposed a selection of certificate features from phishing websites to be used for training a random forest, decision tree and nearest neighbor to detect phishing website which denoted an accuracy above 85% [C13]. A more holistic real-time version with a similar approach and results was proposed by Dong *et al.* where the framework of feature extractor, classifier and decision process was included with phishing information used to train the same models in addition to naive bayes tree, logistic regression, decision table and k-nearest neighbor [C14]. A deep neural network version was also proposed for classification by Dong *et al.* which denoted an accuracy above 95% [C15]. Relying on deep learning, Torroledo *et al.* use a long short-term memory (LSTM) based model for detection [C7]. Recently, Drichel *et al.* propose a pipeline where they could easily test a lot of these classifiers using CT log data which may help in classifier selection process [C4].

The security research community tends to rely on information from domain names to provide a label for certificates that cover the given domain names. Typically, a list of popular domains serves as a ground truth for benign domains, and lists of phishing domains or URLs (e.g., PhishTank [C16]) as benign domains. Certificates that are served by these two respective domain groups can as a result be labeled as benign and phishing as well. Hageman *et al.* showed that labeling certificates in such as fashion may result in mislabeling [C9]. Content Delivery Networks (CDNs) such as CloudFlare and Incapsula that provide HTTPs based protection services issue certificates for sets of domains originating from different owners. In case this set of domains include both malicious and benign domains, labeling these certificates is a challenge.

Even though a significant effort was made by the security community towards the detection of certificates involved in malicious activity, to the best of our knowledge no one has recognized it as an ambiguous problem that should not be tackled as a binary classification problem.

6.4 Data Collection

To train and validate our approach, we rely on a vast number of labeled certificates. First, we describe how we extract a relevant label and feature space from a certificate, and then explain how we obtained a large dataset of certificates.

6.4.1 Feature extraction and labeling

Figure 1 shows the feature extraction and label extraction process. For each unlabeled certificate, and a set of labeled domains, it produces a feature space and a label for the certificates. This set of labeled domains is prepared in advance and contains both benign (i.e., domains that are – with high confidence – have not been part of a phishing attack) and phishing domains (i.e., domains that have been observed as part of a phishing attack). The resulting feature space is a combination of features extracted from the certificate itself, and aggregated features extracted from the list of SANs that are covered by the certificate. For the feature extraction components, we rely on a combination of features that have been used in prior research [C4–C6] or are derived from insights from other work [C9] resulting in 107 features. Due to page limit, we have uploaded a list of our features at our page on the Internet[1]. The first 58 features are extracted from the X.509 certificates themselves and include features such as the signing parameters (e.g., key size, signature algorithm), extensions (i.e., the presence and content of certain X.509 extensions) and the composition of the subject field. The remaining 49 features are extracted from the lexical properties of SANs covered by the certificate, such as the presence of particular keywords or features related to the characters composition and diversity in the string. Each certificate covers a variable number of SANs, and the feature space of these individual SANs are condensed in a fixed-length feature space. We rely on simple statistical functions (i.e., min, max, mean, median) to summarize, and express the diversity of, the numerical domain features and compute a ratio for condensing binary domain features. The dataset contains a significant number of duplicate samples, which we filter out. It is not uncommon for certificates to be renewed after they expire, covering

[1] https://phish-certs.github.io/

the same SANs and being signed by the same CA with the same parameters, resulting in all our extracted features to remain identical[2].

The label of a certificate is inferred from the collection of the labels from the SANs. Depending on which model is being trained, the label is one of three classes (benign, phishy or conflicted) or a continuous "phishyness" score. In the first case, a certificate is benign or phishy when the list of SANs is only composed of benign or phishing domains respectively (and may include unlabeled domains as well). A certificate covering both at least one benign and phishy domain is considered conflicted, as it is not trivial to claim the maliciousness of the certificate. In the latter case, we use a function of all SANs covered by the certificate for computing a phishyness core. This score (s_i) is expressed as a ratio of the number of phishing domains (p_i) and phishing domains, benign domains (b_i) and unlabeled domains (u_i):

$$s_i = \frac{p_i}{p_i + b_i + u_i} \tag{6.1}$$

Note that the label extraction is only done during the training process, and not during the operations of the framework.

6.4.2 Dataset

To train and validate our machine learning models, we collected a vast collection of certificates. This requires a set of certificates for which a ground-truth is known. We can rely on known phishing and benign domains and infer the ground-truth of certificates issued for those domains. Under the assumption that a highly-popular domain is inherently abused for phishing attacks (e.g., `youtube.com`), we rely on the top one million most popular domains from the Tranco list [C17] as a ground truth of benign domains. For establishing a ground truth of phishing entities, we collect phishing URLs from the eCrime Exchange (ECX) platform [C18], a platform in which various anti-phishing organizations share newly identified phishing URLs with one another. The root domains from those URLs (e.g., `example.org` from `https://www.example.co.uk?help`) form the basis of our set of phishing domains. There is an overlap between this set of domains and the benign domains, since some popular domains host some user-generated phishing content, such as Google Forms and Facebook. As such, we remove any of the top 1 million domains from the ECX domains to form our ground truth of phishing domains. Furthermore, we take into account that the nature of a phishing

[2] The only distinction between these certificates are the timestamps when they become active and expire

Table 1: Collected dataset after removing duplicates

Samples Type	Number of Samples
Benign Certificates	213,353
Phishing Certificates	46,256
Conflicted Certificates	15,578

domain can change over time (e.g., a domain may have been registered for benign purposes for years, after which it was registered by phisher and used for hosting phishing content). It is common for phishing domains to only be abused for several days [C19]. In the label aggregation phase of Figure 1, a domain is only considered "phishy" in the context of a particular certificate, if the validity period overlaps with the identification of a phishing URL associated with the domain in the ECX platform.

Similar to [C4], we rely on certificates from the CT logs as a basis for our ground truth. From both of our domain sets, we sampled 10,000 domains each, and collected all certificates that cover any of these 20,000 resulting domain names [C9]. As a result, our dataset contains not only the certificates that are currently deployed on web servers – a common method for related work to retrieve their certificate data from [C6] –, but also historical data and certificates used for non-HTTPS related services, such as mail servers. The certificates were collected from the Censys search engine [C20], which provides extra information to these certificates, most notably the validation status of three root stores (Microsoft, Mozilla's NSS and Apple). We discard all certificates that, according to Censys, do not have any valid certificate chain to any of the three root stores.

6.5 Scenario 1: Distinguishing between phishing and benign certificates

Figure 2 depicts the underpinnings of a multi-class classification training and testing phase. The training phase comprises two steps: (1) feature engineering and preprocessing and (2) training the multi-class classifier. The former receives benign (B), phishing (P), and conflicted (C) certificates. It prepares data format for the training algorithm and filters features with low variance or constant values. This step is also responsible for normalizing the values of different features. The latter outputs a trained classifier to separate between phishing, benign, and conflicted certificates. The training step is not limited to

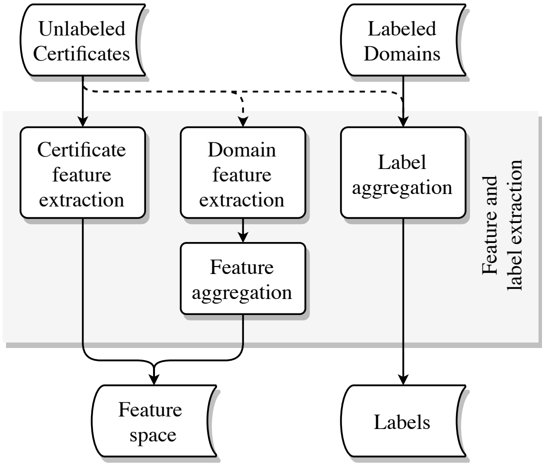


Figure 1: The feature engineering and preprocessing pipeline that turns unlabeled certificates and labeled domains in a labeled feature space. The dashed lines between the certificate data set and the processing modules represent the set of SANs covered by the certificate.

certain multi-class classification algorithms. In the test phase, for classifying new certificates, we first have a data collection and preprocessing step to extract certificate related and domain related features for feeding to the multi-class classifier. Then the classifier outputs a label for the new certificate.

6.5.1 Experimental results

To show the performance of our extracted features in separating phishing, benign and conflicted certificates, we selected five well-known classification algorithms namely *stochastic gradient descent (SGD)*, *k-nearest neighbors (kNN)*, *random forest (RF)*, *decision tree (DT)* and *support vector machines (SVM)*. We used simple classification algorithms to show the robustness of our algorithms in classifying phishing certificates. To avoid the complexity of parameter tuning for each algorithm, we used default parameters given by *sklearn* Python library. As kNN needs k as a required parameter, we set k as the square root of the number of training samples ($k = \sqrt{N}$). We use F1 Score as our comparison metric because it expresses the precision and recall in a single metric [C21]. Table 2 compares our trained classifiers with different metrics for 5-fold cross validation. Our evaluation shows that our proposed features for detecting

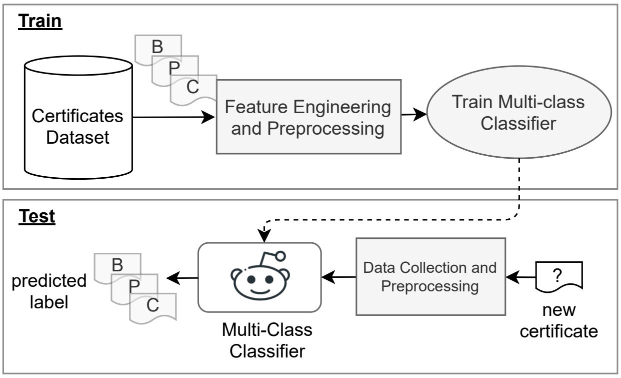

Train
Certificates Dataset
B
P
C
Feature Engineering and Preprocessing
Train Multi-class Classifier
Test
predicted label
B
P
C
Multi-Class Classifier
Data Collection and Preprocessing
?
new certificate

Figure 2: Multi-class classification to distinguish between Phishing, Benign and Conflicted certificates

Table 2: F1 Score of different classifiers under K-Fold cross validation

Fold	SGD	KNN	DT	RF	SVM
1	0.78	0.74	0.96	0.99	0.84
2	0.76	0.74	0.97	0.99	0.84
3	0.75	0.74	0.97	0.99	0.83
4	0.79	0.74	0.97	0.98	0.84
5	0.76	0.74	0.97	0.99	0.84
avg.	0.77	0.74	0.97	0.99	0.84

phishing certificates can help to build high quality classifiers. The RF classifiers generated higher performance in comparison to other classifiers.

6.6 Scenario 2: Predicting phishyness scores

Figure 3 explains training and testing phases of a regression model to predict a phishyness score for each certificate. Similar to *Scenario 1*, this scenario has one step for preprocessing and one step for training the regression model. In the preprocessing step we calculate a phishyness score for each certificate based on equation (6.1). This step is also responsible for normalizing the values of different features as a data preparation task. The output would be a phishyness score for each certificate which can be between 0 and 1. In the test phase,

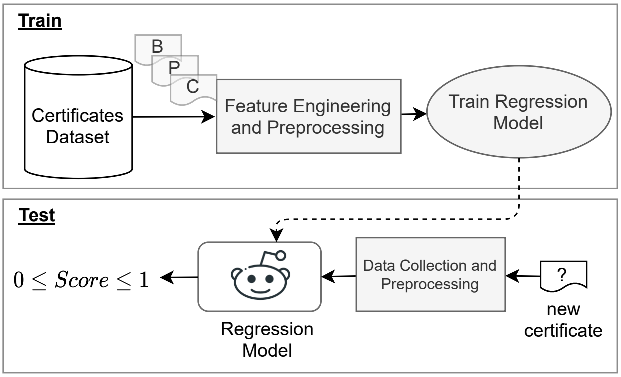


Figure 3: Regression model to predict maliciousness score between 0 and 1 for each certificate

for predicting a phishyness score for new certificates, the data collection and preprocessing step extracts all required features (certificate related and domain related features), and then the trained regression model outputs a score.

6.6.1 Experimental results

To show the performance of our extracted features, we applied different regression algorithms on our dataset. To avoid parameter tuning of different algorithms, we applied each algorithm using default parameter set from *sklearn* [C22]. Table 3 compare the results achieved by different algorithms such as *lasso regression*, *ridge regression*, *ElasticNet*, *random forest regressor (RFR)*, *decision tree regressor (DTR)* and *bagging regressor (BR)*. We use the *Root Mean Square Error (RMSE)* in Equation (6.2) as it is commonly used in regression analysis to verify experimental results. RMSE is the standard deviation of the residuals, which is errors between real value (y) and predicted value ($\hat{y}$) for all samples (n).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)} \tag{6.2}$$

Table 3: RMSE of different regression models under 5-fold cross validation

Fold	Lasso	Ridge	ElasticNet	RFR	DTR	BR
1	0.37	0.2	0.37	0.04	0.06	0.05
2	0.37	0.2	0.37	0.04	0.06	0.05
3	0.37	0.2	0.37	0.04	0.06	0.04
4	0.37	0.2	0.37	0.04	0.06	0.04
5	0.37	0.2	0.37	0.04	0.06	0.05
avg.	0.37	0.20	0.37	0.04	0.06	0.05

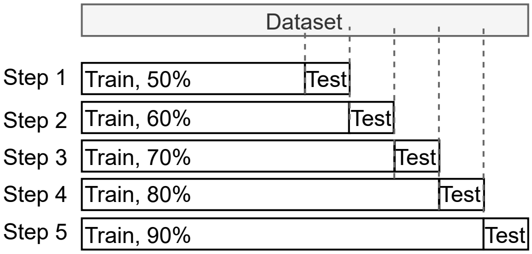


Figure 4: Time-based cross validation.

6.7 Validation

The CT framework was created in 2011 and as a result, the CT logs consist of certificates spanning almost a decade. In cross validation, both the training and validation sets included mixed samples from different periods in time. As such, this validation does not evaluate how well the model generalizes to changes in the TLS ecosystem over time. An example of a major shift was the introduction of *Let's Encrypt*, the first certificate authority that issued certificates for free fully automated, which suddenly enabled small websites to serve their content over HTTPS.

We perform a time-based cross validation to evaluate the generalization of our models over time. In this evaluation, we take the first 50% of our certificates, as defined by their validity date, and produce the performance metrics over the next 10% of certificates. We repeat this process by taking the first 60, 70, 80 and 90% of certificates and test on the next 10% (see Figure 4).

Tables 4 show the F1 scores of time-based validation for the classifiers. Comparing Tables 2 & 4 proves that our classifiers should be retrained on need

Table 4: F1 Score of different classifiers under time-based cross validation

Fold	SGD	KNN	DT	RF	SVM
1	0.81	0.60	0.79	0.85	0.66
2	0.75	0.59	0.84	0.85	0.62
3	0.72	0.59	0.77	0.84	0.65
4	0.72	0.58	0.79	0.81	0.61
5	0.74	0.59	0.81	0.83	0.65
avg.	0.75	0.59	0.80	0.84	0.63

Table 5: RMSE of different regression models under time-based cross validation

Fold	Lasso	Ridge	ElasticNet	RFR	DTR	BR
1	0.37	0.21	0.37	0.05	0.07	0.05
2	0.34	0.18	0.34	0.04	0.06	0.05
3	0.31	0.17	0.31	0.04	0.06	0.04
4	0.34	0.20	0.34	0.05	0.08	0.06
5	0.30	0.20	0.30	0.05	0.08	0.06
avg.	0.33	0.19	0.33	0.05	0.07	0.05

data as there is a decrease in the F1 Score for classifying recent certificates. Our trained random forest classifier could achieve an F1 score of 0.84, while it could get to 0.99 in our 5-fold cross validation (2). The kNN model gave the worst results in both 5-fold and time-based cross validations, which shows that our feature space and dataset need more complex classification to separate phishing, benign and conflicted samples.

Table 5 describes RMSE of each studied regression model. Our time-based evaluation shows that the RFR and BR as ensemble-based algorithms have achieved RMSE of 0.05, which is best among our regression algorithms. The lasso and ElasticNet regression algorithms achieved the worst RMSE, which is 0.33. We are interested in a more in-depth look into the errors that RFR as our best model produces. As such, we calculate the error level (i.e., the absolute difference between the predicted value and the real value) for each sample. We plot these figure in a cumulative distribution function in Figure 5. The figure illustrates for instance that 93.5% of samples has an error level of smaller than 0.01, which we believe gives an acceptable approximation of the actual value.

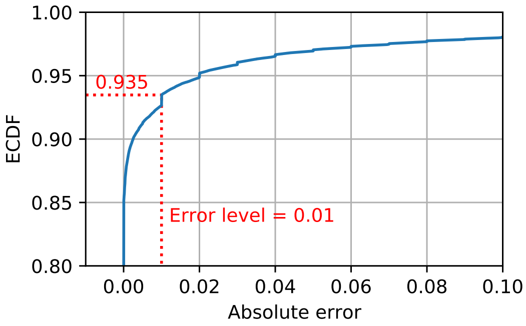


Figure 5: ECDF of the error level of real and the predicted errors for all test samples for time-based cross validation (n=123,835).

6.8 Discussion

Our results have shown that our proposed approach can classify with a high performance and generalizes well over time on historical data. We deliberately selected more classical and simpler classifier and regression models over more novel models, such as deep neural networks, to show that the selected features are powerful and avoid parameter settings.

Adverserial robustness A major challenge for employing machine learning models is an adversarial environment, and we should consider the robustness of the model against behavioral changes of phishers trying to evade our model. Evasion here is the ability of an attacker to modify the feature space of a certificate to circumvent a 'phishing' label to be produced by the classifier. A number of features are domain name independent and are controlled by the CA rather than the phisher (who merely requests the issuance of the certificate), and can therefore only be influenced by the attacker by requesting their certificate from a different CA, which may have monetary consequences. The domain-related features in the feature space are derived from *all* SANs covered by the certificate. These features are changed by a phisher by requesting certificates for different sets or, or even individual, domain names, which has once again a monetary impact and may also impact the hosting infrastructure that phishers resort to. As future work, we consider evaluating the performance of our proposed approach in the absence of domain-related features or CA related features to emulate eliminating features due to the evasion strategies of phishers.

Ground truth challenges As described in Section 6.3, our work is not the first attempt at certificate classification for various security purposes. However, to the best of our knowledge we are the first to acknowledge that there can be conflicts in the label for a certificate. As a result, it is difficult to compare our results with prior work in a fair representative manner.

By monitoring CT logs, we are merely observing snapshots of a domain, and miss the changes of the domain afterwards. Phishers are known to compromise (i.e., hack) existing benign websites, re-purposing them for malicious purposes. As such, we may be labeling certificates as phishing or conflicted due to a domain being reported as phishing domains months after the certificate was issued, even though the certificates at time of issuance should have been labeled differently.

We devised a method to express the maliciousness of a certificate based on the composition of labels of the domains covered by the certificate. Even though this method provides a continuous scale to put certificates on, one should be careful with relying on the results for any automated decision making. Blocking traffic to web servers serving these certificates may block benign traffic and disproportionately hit organizations that provide security services. We believe that the results can instead be highly valuable as a warning signal for security researchers or regulators to follow up on manually. The CT logs are already being used for similar purposes [C23].

6.9 Conclusion

In this paper, we proposed a novel method to automatically classify digital certificates as benign, as used in phishing attacks, or assign it an conflicted label. Our work is motivated by prior work on phishing certificate classification and on the existence of ambiguous certificates that are associated with both benign and phishing domains. We consider both a (1) multi-class classification problem, where certificates can be benign, phishy or conflicted, and (2) a regression problem where certificates have a phishyness score. By training different machine learning models in both scenarios, we show a highly performant system. Furthermore, we evaluate the resulting classifiers and regressors using time-based cross validation to show that our approach generalizes decently over time.

6.10 Acknowledgment

This work was supported by Innovation Fund Denmark (IFD) under the SecDNS project.

References

[C1] FBI Internet Crime Complaint Center, "Internet crime report 2020," 2020. `https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf`.

[C2] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," RFC 6962, RFC Editor, June 2013. `http://www.rfc-editor.org/rfc/rfc6962.txt`.

[C3] B. Laurie, "Certificate transparency," *ACM Queue*, vol. 57, p. 4046, Sept. 2014.

[C4] A. Drichel, V. Drury, J. von Brandt, and U. Meyer, "Finding phish in a haystack: A pipeline for phishing classification on certificate transparency logs," in *The 16th International Conference on Availability, Reliability and Security*, pp. 1–12, 2021.

[C5] E. Fasllija, H. F. Enişer, and B. Prünster, "Phish-hook: Detecting phishing certificates using Certificate Transparency logs," in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 320–334, Springer International Publishing, 2019.

[C6] J. Li, Z. Zhang, and C. Guo, "Machine learning-based malicious X.509 certificates detection," *Applied Sciences*, vol. 11, no. 5, p. 2164, 2021.

[C7] I. Torroledo, L. D. Camacho, and A. C. Bahnsen, "Hunting malicious TLS certificates with deep neural networks," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, AISec '18, (New York, NY, USA), p. 6473, Association for Computing Machinery, 2018.

[C8] S. Maroofi, M. Korczyski, C. Hesselman, B. Ampeau, and A. Duda, "COMAR: Classification of compromised versus maliciously registered domains," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 607–623, 2020.

[C9] K. Hageman, E. Kidmose, R. R. Hansen, and J. M. Pedersen, "Can a TLS certificate be phishy?," in *18th International Conference on Security and Cryptography, SECRYPT 2021*, pp. 38–49, SCITEPRESS Digital Library, 2021.

[C10] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC 5280, RFC Editor, May 2008. `http://www.rfc-editor.org/rfc/rfc5280.txt`.

[C11] R. Mohammad, F. Thabtah, and T. Mccluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, pp. 443–458, 08 2013.

[C12] U. Meyer and V. Drury, "Certified phishing: Taking a look at public key certificates of phishing websites," pp. 211–223, Aug. 2019.

[C13] M. Mishari, E. De Cristofaro, K. Eldefrawy, and G. Tsudik, "Harvesting SSL certificate data to mitigate web-fraud," *CoRR*, vol. abs/0909.3688, 01 2009.

[C14] Z. Dong, A. Kapadia, J. Blythe, and L. J. Camp, "Beyond the lock icon: real-time detection of phishing websites using public key certificates," in *2015 APWG Symposium on Electronic Crime Research (eCrime)*, pp. 1–12, 2015.

[C15] Z. Dong, K. Kane, and L. Camp, "Detection of rogue certificates from trusted certificate authorities using deep neural networks," *ACM Transactions on Privacy and Security (TOPS)*, vol. 19, p. 5, 09 2016.

[C16] Cisco Talos Intelligence Group, "Phishtank," n.d. `https://phishtank.org/`.

[C17] V. L. Pochat, T. V. Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings 2019 Network and Distributed System Security Symposium*, Internet Society, 2019.

[C18] Anti-Phishing Working Group, "The APWG eCrime Exchange (eCX)." `https://apwg.org/ecx/`, n.d. Accessed: 30-09-2021.

[C19] M. Wullink, M. Muller, M. Davids, G. C. M. Moura, and C. Hesselman, "ENTRADA: enabling DNS big data applications," in *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pp. 1–11, 2016.

[C20] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 542–553, ACM, Oct. 2015.

[C21] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K.-K. R. Choo, and D. E. Newton, "Drthis: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Generation Computer Systems*, vol. 90, pp. 94 – 104, 2019.

[C22] "scikit-learn: machine learning in python scikit-learn 1.0 documentation." `https://scikit-learn.org/stable/`. (Accessed on 10/15/2021).

[C23] Facebook, "Certificate transparency monitoring," n.d. `https://developers.facebook.com/tools/ct/search/`.

Chapter 7

Understanding the Challenges of Blocking Unnamed Traffic

Paper D

Kaspar Hageman, Egon Kidmose, René Rydhof Hansen, Jens Myrup Pedersen

The paper is under review at the
IEEE/IFIP Network Operations and Management Symposium (NOMS 2022)

The layout of this work has been revised.

Abstract

Network traffic that is not preceded by any Domain Name System (DNS) resolutions is referred to as unnamed traffic. Any DNS-based security system is ineffective against malicious content distributed through this traffic. In this paper, we introduce a novel method for identifying unnamed traffic based on the correlation of flows and DNS responses extracted from raw network traces. We describe two challenges that affect the validity of our method, and how to handle them. By applying our method to a one-week trace of network traffic, we illustrate that unnamed traffic is ubiquitous in a university network across nearly all client systems, destination IP addresses, and destination services. We conclude by presenting several open problems that prevent us from blocking unnamed traffic for security reasons.

7.1 Introduction

For network administrators of large-scale networks with a multitude of clients, it is of natural interest to ensure the safety of the clients and prevent criminal activities from taking place. Automated security mechanisms focus on eliminating access to certain hosts that are considered malicious. A popular class of these methods employs the analysis of Domain Name System (DNS) traffic for identifying hosts and quantifying the intent of traffic towards them. These methods are naturally challenged by *unnamed traffic*, the traffic that is not relying on DNS to resolve domain names to IP addresses, which completely circumvents these DNS-based security methods. Not only malicious applications [D1–D6] but also benign applications [D7, D8] are known to rely on unnamed traffic, making network operators unable to simply block out all unnamed traffic. This dilemma enables the unnamed traffic to exist within networks, while potentially being a crucial component of malicious activities.

In this work, we take a first step towards systematically classifying network traffic to block unnamed traffic, with the intentions of preventing malicious software to communicate with services outside a network. This process has several unique challenges that we would like the research community to be aware of. More specifically, we contribute with the following:

- We describe a novel process of collecting, processing, and correlating raw network traffic into anonymized named and unnamed flows.

- We analyze the impact of DNS encryption methods and caching of DNS records on the validity of this process.

- We apply our method to traffic from a medium-sized university network to illustrate the characteristics of existing unnamed traffic.

7.2 Background

The DNS provides a method for translating domain names to IP addresses [D9]. Client systems resolve domains to IP addresses by querying sets of distributed *name servers*, or NSs, for *resource records*, or RRs. Each RR comes with a Time-to-live (TTL) value that specifies how long the records should be cached. Plain DNS resolutions are transmitted over a network unencrypted, allowing network operators to monitor the traffic and make restriction policies for certain domains. Several privacy extensions to DNS have been proposed over the years, most notably DNS-over-HTTPS (DoH) [D10] and DNS-over-TLS (DoT) [D11], which both rely on Transport Layer Security (TLS) for establishing an encrypted channel to communicate over. These extensions threaten the ability of network operators to monitor the queries if clients do not employ the network operator's DoH or DoT-enabled DNS server.

Flow monitoring was introduced as a scalable alternative to raw traffic monitoring. Instead of capturing information about individual packets, the packets are aggregated into *flows*. A flow is identified as "a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties" [D12]. The set of common properties is referred to as a *flow key*, and is often a 5-tuple[1]. In addition to this key, each flow contains aggregated data fields, such as a beginning and end timestamp, byte counts, and packet counts.

7.3 Related work

Earlier works on worm spreading mitigation recognized that a lack of DNS traffic was suspicious and blocking such traffic prevents the worms from fully spreading across a network [D1–D6]. These projects have performed controlled experiments, in which an isolated network is infected with a worm, but they do not consider the impact of blocking unnamed traffic on benign traffic in their results. Janbeglou *et al.* produced several papers related to unnamed traffic from a benign traffic standpoint. In [D7], they conduct a passive analysis

[1] Source and destination IP addresses, source and destination port numbers and the IP protocol number

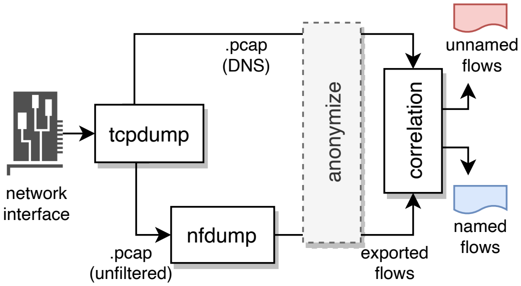


Figure 1: The different components to extract the data

on several datasets to understand its content, followed up by [D8, Chapter 5] in which they find that several popular benign peer-to-peer applications rely on unnamed traffic to function. Both of these directions give insight into a specific *known* type of traffic (i.e., worm spreading or popular applications). This paper complements these results by analyzing the challenges of accurately identifying all unnamed traffic in a typical network.

7.4 Unnamed traffic identification

We propose a pipeline of components to process raw network traffic (as captured from a network interface) and produce named and unnamed flows (Figure 1). This process relies on a monitoring device to passively collect network traffic passing through a network. Using `tcpdump`, raw traffic is captured and written to `.pcap` files. These `.pcap` files are used to extract both flows and DNS responses, using `nfdump` and `tcpdump` with a filter respectively. More specifically, we extract any IP addresses from DNS responses for A/AAAA queries, i.e., queries for resolving domain names to IP addresses. Both the resulting flows and DNS responses are anonymized and afterwards we identify the preceding DNS resolutions for each flow. Flows without a preceding DNS resolution are reported as unnamed.

Anonymization The anonymization process must preserve the ability for us to correlate flows and DNS resource records, and as such there must be a one-to-one mapping between an unanonymized and anonymized IP address. For IP addresses within the monitored network, we employ Crypto-PAN [D13] to anonymize CIDR host identifiers and preserve the network prefixes. As a result, we are still able to identify which hosts are located within the monitored network but do not know the original IP address (before anonymization). All MAC addresses are overwritten with a default value, removing any identifiable information contained in these fields. For the anonymization of `.pcap` files, we recompute the checksum of packets so that conventional network tools accept these traces.

Correlation For a flow to be preceded by a DNS RR, they must match on the following properties: (1) the source IP address of the flow must be equal to the requesting IP address of the RR, (2) the destination IP address of the flow must be equal to the resolved IP address of the RR, and (3) the start timestamp of the flow must occur after the observation of the RR and before its TTL expires.

7.4.1 Validation

We conduct a set of controlled experiments to ensure that our method correctly correlates flows and DNS RRs. For these experiments, we prepared a Docker container to execute a simple task and collected the resulting traffic in a raw `.pcap` file. We run the pipeline on this file and analyze the identified number of named and unnamed flows. We evaluated the following use cases: an idle Ubuntu container without any applications running, a single URL request, a basic Nmap scan, a Chrome or Firefox browser visit `facebook.com` or play a Youtube video. We hypothesize that all experiments, except for the one involving Nmap, generate no unnamed traffic.

The results are shown in Table 1, illustrating the number of observed DNS responses, total flows, named flows, and unnamed flows. For the reported flows, we exclude flows that correspond to DNS resolutions. Nmap only generates TCP packets with the `SYN` flag set, to evaluate if a particular port is open, hence the large number of observed unnamed flows. The results confirm our hypothesis, except for Firefox generating a single unnamed flow when playing a YouTube video. Although these experiments are limited in scope, they show that even benign traffic – in this case watching a YouTube video – can lead to unnamed traffic, breaking the assumption that network traffic is generally preceded by DNS.

7.5 Results

We collected a dataset from a medium-sized university network in Denmark, spanning one week from 11:00 AM, October 4th, 2021 until 09:30 AM, October 11th, 2021. The data was collected using our pipeline deployed at the

Table 1: Results of correlating traffic from several basic network activities, showing the number of DNS queries, and breakdown of the number of flows (*F* = Firefox, *C* = Chrome).

Activities	**DNS Responses**	**Flows (excl. DNS)** Total	Named	Unnamed
Idle	0	0	0	0
cURL	1	1	1	0
Nmap	0	49991	0	49991
Facebook (F)	53	44	44	0
YouTube (F)	71	79	78	1
Facebook (C)	2	10	10	0
YouTube (C)	7	22	22	0

VPN server of the university network, monitoring the traffic of any client that is connected to the university's VPN. We are primarily interested in blocking network traffic from leaving a network and therefore we only considered traffic between internal and external IP addresses, omitting any internal traffic. Moreover, the dataset comprises IPv4 traffic only, as the university's VPN server operates with IPv4 addresses only. Raw traffic was fed into our pipeline in two-minute intervals.

In the process of extracting named and unnamed traffic, we ran into several challenges that potentially could threaten the validity of the extraction process. We discuss how two of these caveats could impact our results, and how we handle these cases: TTL caching and the usage of alternative DNS resolvers.

TTL caching We must take into account that the clients whose traffic we monitor have already locally cached a significant number of DNS resource records prior to our measurement. Any flow matching those cached records would be (incorrectly) identified as unnamed in our measurements, as the DNS resolutions are missing in our dataset. The vast majority (99.16%) of DNS RRs in our dataset have TTL values lower than a day. By running our correlation on the full DNS dataset and discarding the first day of the flow measurement, we reduce the number of flows falsely marked as unnamed.

Alternative DNS resolvers It is also possible for a flow to be falsely labeled unnamed if the preceding DNS resolution was unobserved by the DNS monitoring setup, which happens when clients use an alternative DNS resolver.

Table 2: Estimated DNS query breakdown in type

Resolver types	# Queries	Percentage	# Nameservers
Local (plain)	11,498,808	92.61	2
Open (plain)	598,903	4.82	331
DoT	0	0.00	0
DoH	318,059	2.56	8

We evaluate how impactful the use of these other resolvers is for the validity of our results. We focus on encrypted DNS traffic (DoT and DoH), and open DNS resolvers using plain DNS. Since we are unable to access the content of encrypted DNS traffic, and do not store the resolver IP addresses in our DNS dataset, we must in all cases estimate the usage of the various resolver types based on flows. For plain DNS, we estimate every packet to port 53 to be an individual DNS resolution, and distinguish between the university and open DNS resolvers based on the destination IP addresses of the flows. For DoT and DoH, the query volume is estimated by observing traffic towards TCP port 853 [D11] for DoT and by observing traffic to known port 443 for IP addresses of known public DoH providers [D14] for DoH respectively. In addition, for IP addresses for (1) which we observed plain DNS resolutions towards and (2) which we successfully can resolve a DNS query towards, we also consider traffic towards port 443 as DoH traffic. The query count is estimated to be the number of packets contained in the flows. For DoT and DoH, these flows include the packets for establishing a TCP and TLS connection, and as such relying on the packets in the flow overestimates the number of actual DNS requests. The estimated number of queries is shown in Table 2. The vast majority of queries (97.44%) remain observable with the remaining queries being forwarded to eight DoH resolvers, and as such we do not address encrypted DNS resolutions in our data analysis.

7.5.1 Data overview

Table 3 shows an overview of the collected data, both its flow part and its DNS part. For the flows, we report the total number of identified instances (e.g., flow count, unique IP addresses), and the instances involved in unnamed and named flows. Similarly, the DNS portion of the table denotes the total resolutions and involved IP addresses, and the instances related to DNS lookups that are (and are not) succeeded by at least one flow.

We collected nearly 37 million individual flows, of which 18.93 million (or

Table 3: General overview of the data

	Flows		
	Total	Named	Unnamed
Flow count	36.83 M	18.93 M	17.90 M
Unique IP_{src}	975	956	975
Unique IP_{dst}	343,462	52,039	317,044
Unique services	69,419	6,297	66,392
Packets (sent)	1.64 B	0.98 B	0.66 B
Packets (rcvd.)	1.59 B	0.95 B	0.64 B
Bytes (sent)	464.09 B	284.42 B	179.67 B
Bytes (rcvd.)	2702.94 B	1738.26 B	964.68 B
	DNS		
	Total	Succeeded	Not succeeded
Resolution count	5.72 M	4.38 M	1.34 M
Unique IP_{req}	1,077	1,077	1,075
Unique IP_{res}	79,483	75,073	33,737
Unique query name	119,911	113,671	33,838

48.61%) were unnamed. The percentage of packets and bytes exchanged are similar, ranging between 35.69% for received bytes to 40.42% for sent packets. Interestingly, a significant portion of DNS resolutions was never followed up by a flow, indicating that these DNS resolutions have been performed unnecessarily.

Service and IP addresses breakdown We aim to understand how the usage of unnamed traffic is distributed across source IP addresses, destination IP addresses, and the services that are being accessed (as inferred from the destination port and protocol). For each IP address and service, we compute the percentage of traffic origination from, or destined towards, that entity, in terms of flows, sent packets, and received packets. The cumulative distributions of these percentages are shown in Figure 2, only including services and addresses involved in at least fifty flows.

Nearly 40% of all services with at least fifty flows are accessed purely through unnamed traffic, and only 2% being accessed purely named (top figure). For the common web browsing services – HTTP and HTTPS – we observe 17.6% and 20.6% of flows to be unnamed, whereas DNS is almost exclusively accessed unnamed (as expected).

The distribution of unnamed traffic usage across source IP addresses (middle figure) is highly concentrated between 20% and 80% of flows (solid black trace) being unnamed. Only 1.46% of these IP addresses has an unnamed traffic percentage outside this window. As a result, we cannot identify a small set of hosts that is responsible for most unnamed traffic. Instead, many source IP addresses contribute evenly to the amount of observed unnamed flows. The distribution of sent and received packets is even more skewed towards a higher unnamed percentage.

The distribution of the percentage of traffic for the destination IP addresses (bottom figure) shows that 13.55% and 9.19% of destination IP addresses are accessed exclusively through unnamed and named traffic respectively. For the IP addresses associated with `youtube.com` and `google.com`, 0.67% and 14.55% of traffic is unnamed respectively.

7.6 Discussion

Our results show that almost half of all flows in our data set are unnamed, Furthermore, we show that there is an almost universal existence of unnamed traffic towards services, originating from clients and towards destination IP addresses. From a network management perspective, this suggests that blocking unnamed traffic might affect the normal operations of a large number of services, and nearly the entire client population of a network. Although we described several caveats in this work, there are several notable remaining challenges.

Flow exporting interval The methodology that we took in this work relies on `.pcap` being generated at a two-minute interval. This approach likely splits a single flow into two separate flows in some cases, of which the later flow may be falsely marked as unnamed. By increasing the interval, this likelihood decreases but remains present. Alternatively, one could export flows in real-time using a router that supports exporting flows. However, these flow exporters typically export flows prematurely for performance reasons, which makes this method imperfect too. The relatively short measurement interval in our data set could partially explain why unnamed traffic is so prevalent across all source IP addresses.

Changing of IP addresses We collected traffic from a VPN server, which assigns IP addresses each time a client connects to the server. This IP address

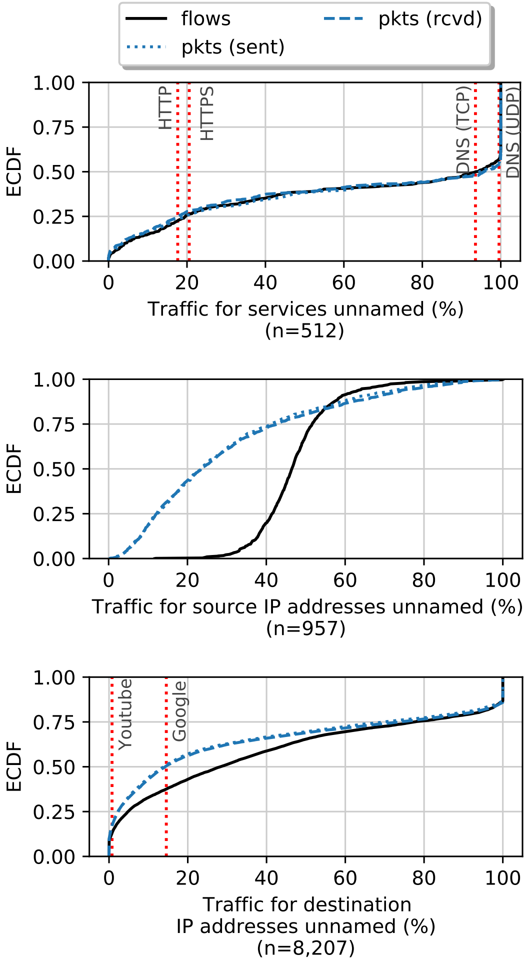


Figure 2: The percentage of flows, and packets towards or originating from services (top), source IP addresses (middle) and destination IP addresses (bottom) being unnamed.

assignment does not guarantee that a computer system obtains the same IP address every time it establishes a connection. As such, our methodology is incapable of matching DNS records resolved with an old IP address against a flow after a change of the IP address of a client. To overcome this problem, the allocation of IP addresses by the VPN server could be taken into account, allowing us to correlate flows and DNS records across different source IP addresses. However, this would require the integration of a VPN server and our solution.

Clients not adhering to TTL values The validity of the correlation of flows and DNS records is heavily dependent on the extent to which clients adhere to the TTL values that name servers return. These TTL values are a suggestion, and there is no enforcement that clients actually adhere to them. It is possible for a client to cache a record indefinitely – resulting in potentially many falsely identified unnamed flows – but the extent to which this is done in practice is unknown. We recommend a more in-depth study into the DNS caching behavior of various stub resolver implementations (i.e., the DNS client running locally on an end user's system).

7.7 Conclusion

In this paper, we proposed a method for extracting named and unnamed traffic from a raw network traffic trace, i.e., traffic that is or is not preceded by a DNS resolution respectively. By applying our proposed method to a one-week dataset, we illustrate three potential challenges, which we address. Firstly, we take into account that DNS records can be locally cached prior to monitoring network traffic and conclude that nearly all records have a TTL of shorter than a day. Secondly, over 97% of DNS resolutions are unencrypted and can be observed, with the remaining DNS resolutions being a cause of misidentifying flows. A deeper dive into the unnamed traffic in our dataset reveals that unnamed traffic is ubiquitous in our dataset, being generated by nearly all source IP addresses, towards a significant number of services and affecting the majority of destination IP addresses. Relying on the "unnamedness" of traffic to make decisions about blocking this traffic is at this stage infeasible. We recommend the research community to further investigate this traffic to better understand the intent behind unnamed traffic.

Acknowledgments

The authors would like to thank Finn Büttner for his continuous efforts towards data capturing at our university network. This research is carried out under the SecDNS project, funded by Innovation Fund Denmark.

References

[D1] G. R. Ganger, G. Economou, and S. M. Bielski, "Self-securing network interfaces: What, why and how," 2002. https://apps.dtic.mil/sti/citations/ADA457627.

[D2] D. Whyte, E. Kranakis, and P. C. Van Oorschot, "DNS-based detection of scanning worms in an enterprise network.," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2005)*, 2005.

[D3] K. Shahzad and S. Woodhead, "Towards automated distributed containment of zero-day network worms," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–7, IEEE, 2014.

[D4] K. Shahzad and S. Woodhead, "Empirical analysis of rate limiting + leap ahead (rl+la) countermeasure against witty worm," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 2055–2061, 2015.

[D5] M. Ahmad and S. Woodhead, "Containment of fast scanning computer network worms," in *International Conference on Internet and Distributed Computing Systems (IDCS)*, vol. 9258, pp. 235–247, Springer, 09 2015.

[D6] M. A. Ahmad, S. Woodhead, and D. Gan, "A countermeasure mechanism for fast scanning malware," in *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*, pp. 1–8, IEEE, 2016.

[D7] M. Janbeglou, H. Naderi, and N. Brownlee, "Effectiveness of DNS-based security approaches in large-scale networks," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 524–529, IEEE, 2014.

[D8] M. Janbeglou, *Understanding and Controlling Unnamed Internet Traffic*. PhD thesis, University of Auckland, 2017.

[D9] P. Mockapetris, "Domain names - implementation and specification," STD 13, RFC Editor, November 1987. `http://www.rfc-editor.org/rfc/rfc1035.txt`.

[D10] P. Hoffman and P. McManus, "DNS queries over HTTPS (DoH)," RFC 8484, RFC Editor, October 2018. `http://www.rfc-editor.org/rfc/rfc8484.txt`.

[D11] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over transport layer security (TLS)," RFC 7858, RFC Editor, May 2016. `https://www.rfc-editor.org/rfc/rfc7858.txt`.

[D12] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP flow information export (ipfix) protocol for the exchange of flow information," STD 77, RFC Editor, September 2013. `http://www.rfc-editor.org/rfc/rfc7011.txt`.

[D13] J. Xu, J. Fan, M. Ammar, and S. Moon, "Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, pp. 280–289, 2002.

[D14] DNS Privacy Project, "Public resolvers." `https://dnsprivacy.org/public_resolvers/`.

Chapter 8

Kaonashi: On Developer Attribution Challenges in Android App Marketplaces

Paper E

Kaspar Hageman, Álvaro Feal, Julien Gamba, Aniketh Girish, Jakob Bleier, Martina Lindorfer, Juan Tapiador, Narseo Vallina-Rodriguez

The paper is in preparation for submission

The layout of this work has been revised.

Abstract

Mobile applications (apps) have become one of the main gateways to access online content and services. The Android ecosystem is vast and open, and an important part of understanding it is being able to attribute apps to the company behind them. Users can download apps from different app stores, which have their own set of requirements, policies and enforcement in place for app publishing and market sanitization. This translates into different, potentially conflicting information available to attribute a given app to its original developer or publisher. Attribution in the Android ecosystem is complex, but also critical for stakeholders other than the markets themselves. Regulators, data protection agencies and researchers need accurate attribution for when they identify privacy violations or security vulnerabilities, or measuring the Android ecosystem and its markets.

In this paper, we perform a comprehensive analysis of the attribution challenges on the Android ecosystem. We identify the signals that can be (and have previously been) used for attribution in 9 different markets, including those related to the app itself and those related to each specific market. We show that these signals are often missing from markets, that they can change over time, and that they are not consistent either within or across markets. We explore the case of the Google Play Store in detail and show that previously used signals—notably the signing certificate—are not reliable for attribution as there is no clear relationship between them and the company behind an app. We conclude by providing a constructive discussion on the limitations of our findings, and the implications for the Android ecosystem and potential solutions.

8.1 Introduction

With an estimated 57% market share [E1], mobile devices have become the prevailing vehicle to access Internet services, exceeding in popularity traditional desktop devices. Online services are accessed from mobile devices not only via browsers, but predominantly through third-party applications (apps) that users can download from a number of app stores or marketplaces (markets hereafter). Unfortunately, the dynamism of the mobile developer ecosystem and the lack of control over app developers and their practices can be detrimental to markets and users alike, leading to issues such as the distribution of malware or app clones [E2, E3]. For instance, the popular *Barcode Scanner* Android app was purposely injected with malware in November 2020 after the app transferred ownership. The existing user base ignored this, and the malware remained undetected for almost a month [E4]. In early 2021, the sudden popularity of the iOS-only app *Clubhouse* led many Android users to mistakenly download an identically named but completely unrelated app from the Google Play Store [E5]. Similarly, attackers created apps posing as belonging to *Trezor*, a hardware cryptocurrency wallet that does not have any corresponding apps yet, and in turn stole the entirety of the users' accounts [E6,E7]. More generally, the ease of repackaging apps with minor changes has led to numerous app clones, which impersonate an already existing (and often highly popular) app but incur in harmful activities, such as injecting malware or removing payment options [E8–E10].

Attribution is the process of determining the entity (either an individual developer, organization, or company) that publishes an app in a market and is ultimately responsible for it.[1] Correct and reliable attribution is essential for security, accountability, and regulatory reasons. Attribution facilitates the detection of fraudulent copies published by malicious actors to get access to user data from mobile devices [E2, E11], and is instrumental to conduct measurement studies about the Android developer ecosystem, market dynamics, and trends [E12–E15]. Additionally, it is well known that mobile apps can collect highly sensitive and personal data by accessing phone sensors and restricted features that can compromise users' privacy and security [E16–E21]. Thus, correct attribution contributes to a better risk assessment when granting apps access to sensitive data.

Despite its importance, attribution remains an open research challenge in mobile measurement studies, security, and software engineering research. This is particularly problematic in the Android ecosystem. During the development and publication process, Android apps acquire certain features that can be helpful when attributing them to their original developer or publisher. Such features derive both from the app package (e.g., signing certificates) and its market profile (e.g., application developer name, email and website). We refer to these features as *attribution signals*. However, some of the fields available on the markets are self-declared by the developer, including the name of the app, contact information and links to their website. Additionally, the mandatory cryptographic app signature bundled with every Android app is self-signed by the developer. To complicate things further, each market has different publication policies, requirements, and procedures to publish an app, which translates into an inconsistent set of features for app attribution across markets [E22, E23]. The overall result is that, due to the lack of strong attribution mechanisms and strict publication policies, Android markets struggle to keep their stores sanitized, identify and remove apps from dubious developers, and ensure that the developer ecosystem is sufficiently transparent for

[1] We note that the actual author of the software might be different from the publisher, e.g.,for companies outsourcing their software development.

end-users [E2, E3, E24].

Attribution for Android is harder than it is for iOS, the web or other software platforms, like Windows. iOS apps must be signed by a certificate issued by Apple after verifying the identity of the developer [E25, E26].[2] The use of a Public Key Infrastructure (PKI) on the web provides authenticity between a web browser and a web server through the use of X.509 certificates issued by Certificate Authorities (CAs) and even DNS(SEC) records. Similarly, Windows software is signed using a public-key cryptography standard [E27]. Whenever a piece of software is installed on a device, a central authority verifies the signature and includes the publisher's name on the notification shown to users [E28].

None of these mechanisms exist in Android. In the absence of reliable attribution mechanisms, researchers have relied on multiple signals to conduct studies on the Android ecosystem. Most approaches use information contained in the app's certificate [E3, E11, E12, E22, E23, E29–E38] or information about the developer gathered from the market [E11, E13–E15, E31, E38–E42]. Other approaches have also relied on the app name [E11, E13, E38, E41], the package name [E22, E34, E39], and multiple code features [E22, E33, E43], including strings [E12] and resources [E34] found in the app's package file. However, it is unclear to which extent such signals are indeed a good indicator for identifying an app and its developer; and, if they are not, the negative effects this might have on the findings of such studies and our understanding of the Android ecosystem.

To further complicate matters, the notion of valid attribution in legal texts is not aligned with the signals used by users, researchers, and market operators. Current legislation—i.e., the General Data Protection Regulation (GDPR) in the EU [E44]—mandates that software must allow users to execute their digital rights [E45] (i.e., access, correct or delete their data) as well as assess the risks of software through the privacy policy. However, the company behind the creation, development, and even publication of an app in the market can be completely decoupled from the company processing (i.e., the data processor, in terms of the GDPR) the user's data. These potential differences between market data and legal texts add an extra level of complexity for users who just want to be able to understand who is the company that is offering a given app. Regulators and data protection agencies must also be able to attribute an app to a company in order to enforce the legislation in place. Similarly, researchers

[2] iOS developers must enroll in the developer program which requires the verification of their legal identity. Developers then receive a certificate issued by Apple that must be used to sign their apps in order to be published in the official store. Additionally, a developer account on the Apple store is more expensive than in the Google Play store: $99 per year vs. a one-time $25 fee, thus adding a higher entry barrier for many ill-intended actors.

face the same issue when disclosing any found security vulnerabilities or privacy violations back to the app's responsible party.

Contributions This work aims at gaining a better understanding of the procedures, difficulties, and potential pitfalls of attributing Android apps in absence of sound mechanisms using metadata available on markets or in the app itself. To do so, we seek answers to the following questions: *(i)* what signals can be used for attribution; *(ii)* how consistent and volatile are such signals both within a market and across them; and *(iii)* what is the role of different market policies on attribution and to which extent they are enforced? To address these questions, we first review the app signing procedure and the policies defined by different markets to identify app developers (§8.2). We then perform a longitudinal study of the problem domains and attribution approaches followed by researchers in the last decade (§8.3). Our findings translate into a set of attribution signals (§8.4) that we explore in detail in the remainder of the paper. For our measurement, we gather a large dataset containing 2.6M market entries and 1.3M apps from 9 relevant Android markets (§8.5). We then (§8.6) study issues that can negatively impact attribution, such as missing values, volatility, and inconsistencies within the attribution signals at the app, intra market and across market levels. Given its official nature and predominant position, we explore the Google Play Store in more detail (§8.7). We conclude the paper by discussing the implications of our findings and providing recommendations for market operators and researchers that can contribute to improving attribution in the Android ecosystem (§8.8). We make the source code of our crawler available (at the time of publication of this work) in an effort to make our work reproducible and to foster further research [E46].

8.2 Background

8.2.1 App signing process

Android apps are distributed as Android Package (APK) files through app markets where users can freely shop mobile software. In order to provide *integrity* (i.e., to prevent tampering of the content of the APK) and *authenticity* (i.e., to prove the identity of the author), each package must be cryptographically signed prior to being published on a market or being installed on a device. The signing certificate thus plays a vital role as an indication of authorship in the Android ecosystem, as stated in Google's policy: *"the certificate associates the APK or app bundle to you and your corresponding private key. This helps Android ensure that any future updates to your app are authentic and come from the original* *author*" [E47]. In fact, an app can only be updated if the update is signed with the same certificate as the installed app itself.

Signing certificates. Beside the *release certificate* used for publishing, developers can also make use of *debug certificates* during the development and testing process. These certificates are either generated by the Android Studio build environment with a limited lifetime [E48], or test keys used for debug builds of the Android Open Source Project (AOSP). In the latter case, the private key is publicly available as well, allowing anyone to update an app without breaking the signature and making them popular among malware authors [E2, E49]. Consequently, the Google Play Store does not accept apps signed with debug certificates [E48].

The signature scheme for Android packages has been revised multiple times, each version fixing deficiencies present in the previous ones. Version 2 (released with Android 7.0 in 2016) corrected a severe security vulnerability in the first version [E50], and version 3 (released with Android 9.0 in 2018) enabled the rotation of signing certificates [E51]. For the sake of backward compatibility, APKs may be signed using one or more different signature schemes.

Yet, the main issue preventing signing certificates to be useful for attribution and accountability is that they can be self-signed. This means that there is no guarantee that the information about the signing company is complete or even accurate. On the web, it is common to rely on information related to the SSL certificate of a web page to attribute a given domain to the company behind it [E17]. Windows software is signed using a public key infrastructure that allows the OS to verify who signed a given program [E28]. Android, however, lacks a central authority that confirms the validity and veracity of a given certificate. This disallows the use of this information for attribution.

Signing delegation. Since 2017, Google has been offering "Play App Signing" as a way to protect signing keys from being lost or compromised. By using it, developers can delegate the key management and signing process to the Google Play Store. The developer signs a candidate APK prior to uploading the file to the market using an *upload key*, resulting in a signature that is verified by the Play Store. In turn, the Play Store re-signs the APK with a *signing key* managed by Google and distributes the resulting APK to users [E48]. Google further introduced the concept of app bundles in 2018, which are essentially "split APKs" that are built and optimized by the Google Play Store and, consequently, require signature delegation. While the use of app bundles has been optional to date, starting in August 2021 Google will require all apps to be published as app bundles, i.e., all apps will be signed by Google instead of the developers [E52]. This further invalidates the actual (limited) function of

Table 1: Application publishing policies per market. Note that we could not find Baidu's policy.

	Google Play	**APKMonk**	**Tencent**	**Mi**	**Baidu**	**APKMirror**	**Huawei**	**Qihoo 360**	**F-Droid**
Publisher info	✓	✓	✓	✓	—	✓	✓	✓	
Anti-malware	✓				—		✓	✓	
Anti-clones	✓			✓	—	✓	✓	✓	
False identity	✓			✓	—	✓	✓	✓	
Reference to policy	[E56]	[E57]	[E58]	[E59]	—	[E60]	[E61]	[E62]	[E63]

the signing certificate as an indicator of authorship. However, Google might gain control over the process if this is complemented with strict developer identity verification processes.

Yet, signature delegation is a market-specific mechanism. Alternative markets might implement their own (less restricting) signing procedures. Huawei offers a similar app signing service [E53], while F-Droid, an open source market, provides a developer toolkit including the option to handle certificate management and APK signing [E54]. Other markets, such as APKMonk [E55], rely on pre-signed apps to be uploaded. We discuss market policies next.

8.2.2 Market policies and metadata

After signing an app, developers can publish their apps on one (or several) of the many available markets. The package name of the APK acts as a unique app identifier on a specific Android device, and as such only one app under a specific package name can be installed at the same time. Presumably, markets do not allow for two different published apps to share their package name. Across most app stores, besides uploading the APK (or the app bundle), each published app is accompanied by some basic information about the functionality of the app (e.g., its description and category) and the developer (e.g., name and contact information). Individual markets may support different types of this *metadata* and, as markets operate independently, the metadata may not be consistent for apps published on multiple markets. Furthermore, this metadata is often introduced by the developers themselves and, therefore, accurate or not depending on developers following best practices and the verification policies implemented on each market.

Table 1 summarizes the publishing policies and terms of service (ToS) of the nine markets studied in this work. We manually analyzed these ToS to determine whether they explicitly prohibit specific behaviors. We note that these documents often use ambiguous language, which could result in different understandings to different readers. Nonetheless, this shows that different markets do have different policies.

The Google Play Store is the official and dominant Android market [E64], to the extent that Android's security settings prohibit installing apps from other "unknown sources" by default. All apps in the store (and optionally from other sources as well) are verified with Google Play Protect for violations of Google's Unwanted Software Policy [E65]. The Developer Program Policies [E66] further restrict not only the content of apps, but also how they are marketed and advertised to users: the Play Store prohibits deceptive behavior [E67], misrepresentation [E68], and impersonation [E69]. These restrictions also apply to the market metadata, such as the developer name, title and screenshots. Specifically, this data should not be *"non-descriptive, irrelevant, excessive or inappropriate"* [E70]. Google claims to identify violations of these policies, but it is unclear how they perform this task. Prior work has shown instances in which developers break such policies, reporting empty or inaccurate privacy policies [E24]. While only anecdotal, we were able to publish one app that did not comply with these policies, pointing towards the fact that the Google Play Store is not able to automatically detect violations of these policies at scale (see Appendix 8.F for an extended explanation of our experiments).

The policies of alternative markets range from nearly non-existing and vague (e.g., APKMonk refers to their ToS, which cannot be found on their website [E55]) to policy items that are stricter than in the Play Store. Different policies have a direct impact not only on the number and type of attribution signals that are available, but also on their accuracy and validity as attribution signals. For instance, not all markets have a notion of app developer (i.e., a name that identifies the organization behind the development of an app). Of the nine markets we study, only Tencent and Mi (from Xiaomi) require identification for registration, although Tencent only requires it for Chinese citizens, residents, or companies. Mi requires a picture of developers holding a valid ID card or passport to create an account [E71].

8.3 Attribution in prior research

Since the early days of mobile markets, the research community has studied different aspects of the developer ecosystem and its dynamics, such as developer popularity and diversity, the content offered by published apps, and the prevalence of malicious actors [E14, E15, E31, E40, E72]. To study these aspects, researchers have relied on various attribution signals, such as market metadata and signing certificates. However, no systematic research has questioned the validity and pitfalls of existing attribution techniques. This section explores the most common attribution signals used in prior research. Our aim is both to identify them and to understand the reasons provided to consider them suitable attribution signals.

Ecosystem analysis. Petsas *et al.* [E14] performed an early market-level analysis on four alternative markets. Although they did rely on attribution signals to link developers across markets, they used market metadata to calculate statistics per developer. Several research efforts developed scalable techniques for crawling the Google Play Store [E31, E36] to retain historical data of both market information and published APKs for research purposes. Androzoo [E36] collects APKs from multiple markets but does not collect market metadata so any research effort relying on their dataset has to perform attribution based solely on signing certificates. Viennot *et al.* [E31] proposed a more comprehensive crawling technique that harvests both market- and APK-level information. They used both of them to evaluate a similarity measure based on resource names and assets.

Another line of research tracked apps across markets to understand the overlap of their catalogs [E13, E31, E73–E75]. Wang *et al.* [E23, E40] used market-level metadata to track app developers across Chinese app markets and the Google Play Store. However, this metadata is not consistent across markets (see Section §8.8), which leads many researchers to rely on the signing certificate for cross-platform analysis. For instance, Wang *et al.* [E15] rely on the certificates along with market signals. They note that larger entities, such as Samsung, have multiple developer names and certificates without characterizing the scale of the problem. Recent efforts focused on studying the Android supply chain and firmware-level customizations faced these attribution challenges. Pre-installed apps are generally not publicly available in any market and therefore lack a developer profile. Consequently, researchers had to rely on certificates to define authorship [E30, E37, E76].

Security and privacy issues. Security research requires identifying the author behind a malware sample, a privacy-intrusive app, or a potentially harmful app (PHAs). To that end, the malware analysis community [E22, E29, E32, E43, E77] has typically approached attribution through app-level information, such as the certificate, package name, or app' bytecode. Unfortunately, techniques that rely on (byte)code to infer authorship are computationally expensive and do not scale for market-level analysis. An empirical analysis by Barrera *et al.* [E29] on the security aspects of Android app installation and the limitations of sandboxing based on the Linux user ID relies on certificates and app information as an indicator of authorship. They showed that malicious actors often rely on the possibility to create new certificates to prevent linkability. Similarly, Oltrogge *et al.* [E34] shows that app building frameworks, which automate app development and distribution, invalidate the assumption that two apps with the same certificate belong to the same company.

A long-standing problem in mobile app markets is detecting app clones, i.e., apps that pretend to be a different app. Identifying which app is the original one is a well-known problem that arises from the unreliability of certificate information [E3, E31, E35, E41, E78, E79], and attribution signals are critical for solving this problem. The clone detection approach by Crussell *et al.* [E3] relies on the assumption that two apps only belong to the same developer if they are signed using the same certificate. Viennot *et al.* [E31] relied on developer and certificate information to report rebranded and cloned apps. Li *et al.* [E35] define two apps as clones if they share 80% of their code but are signed with different certificates.

Recent research efforts have characterized app installation practices from multiple perspectives [E11, E42, E76]. Kotzias *et al.* [E11] explore the distribution of potentially unwanted apps through Pay-Per-Install (PPI) schemes. Farooqi *et al.* [E42] identify developers making use of incentivized installs to manipulate app store metadata. These studies relied both on certificate information along with developer market level information to identify underlying publishers and developers making use of these app distribution schemes.

Attribution is also critical for the correct functioning of security apps. Aonzo *et al.* [E39] show that to identify an app–website relationship, several password managers rely on the developer website information provided in the Google Play Store. More recently, Sebastian *et al.* [E38] rely on chains of shared metadata between apps to identify developer account polymorphism, not using any code similarity methods. The authors propose a method that uses on Indicators of Compromise (IOCs) to find sets of developers that are related (i.e., using signals from the app itself, such as certificate information, and signals from the market). They evaluated their approach on a set of rogueware and adware operations and discovered at least one previously unknown developer account for 94% of them.

Finally, attribution is important for responsible disclosure. In a recent study on GDPR violations, Nguyen *et al.* [E80] use the developer email addresses listed on the Google Play Store to notify developers and reported that they were only able to reach the owner of 15.7% of apps.

Alternative attribution approaches. The shortcomings in the Android attribution ecosystem have led the software engineering community to propose complementary attribution signals and methods. Kalgutkar *et al.* [E12] use strings found in the APK for attribution to build a classifier and rely on the signing certificate to establish a ground truth on apps by the same author. They acknowledge that this approach can have both false negatives and false positives, which they attempt to reduce by filtering out apps signed by publicly available certificates or by debug certificates. Xu *et al.* [E41] and their framework AppAuth, and Gonzalez *et al.* [E33] both attribute apps to authors based on code-style features. Specifically, Gonzalez *et al.* [E33] tackle the problem of attribution by generating a dataset of known developer profiles and attribute apps to profiles based on the development style to identify malware authors. They further illustrate the similarity of signals between a pair of cloned apps, indicating that app clones leverage the lack of vetting of these signals.

Key takeaways. Our literature survey (see Table 8.A.1 in Appendix 8.A for a summary) shows that developer metadata from the market and the apps' signing certificate are predominately used for attribution. While the first can be helpful to link apps within the same market, it does not indicate the author of an app, merely its publisher. Additionally, the developer signal is inconsistent across markets and, in some cases (e.g., pre-installed apps), not available at all. Signing certificates, however, are tied to an app but not necessarily to the author of the app: they only represent the signing entity. At the same time, an entity such as a large company or malicious actor can use multiple certificates for different apps. These signals are easily scalable to market-level analysis, in contrast to other more computationally complex such as code-level similarity. Even in those cases, clone detection often relies on market data or certificates for evaluating their technique and as ground truth. Although many studies acknowledge the limitations of this approach, there is a lack of understanding of the value of these signals.

8.4 Attribution signals

This section examines the potential and accuracy of various types of market metadata and app features as attribution signals. We cover signals extracted from an exhaustive analysis of Android's official documentation, [E81]—including app store guidelines and policies [E56–E60, E62, E63]—, and signals used by

Table 2: Signals collected from market metadata.

	# Markets	Google Play	APKMonk	Tencent	Mi	Baidu	APKMirror	Huawei	Qihoo 360	F-Droid
Package name	5	✓	✓	✓			✓			✓
App name	9	✓	✓	✓	✓	✓	✓	✓	✓	✓
Developer name	7	✓	✓	✓	✓		✓		✓	
Developer website	1	✓								
Privacy policy URL	1	✓								
Developer email	1	✓								

previous research efforts detailed in Section 8.3. This results in seven attribution signals: the signing certificate, the developer name, the developer website, the developer email, the privacy policy URL, the app name, and the package name. With the exception of the signing certificate, which is mandatory for any Android app to be installed, we report the public availability of these signals across the nine markets we studied in Table 2.

Package name. The package name serves as the unique (string) identifier of an app at the operating system level: Android does not allow installing two apps with the same package name on the same device. Google's documentation recommends following the Java package naming convention, and to "*use Internet domain ownership as the basis for package names (in reverse), to avoid conflicts with other developers*" [E82]. This stresses the theoretical importance of the package name as an attribution signal. Some markets use the package name as a unique and visible identifier (usually visible in the URL for the app's market entry), whereas others use an internal identifier instead (more commonly in Chinese markets). Since we did not observe any package name published under different internal identifiers, we posit that these markets also enforce internally the uniqueness of package names across their catalogs of apps.

App name. The app name is how most users would search for an app in any market. We note that this signal does not necessarily help attributing an app to the company publishing it (e.g., Instagram is not directly mapped to Facebook). Nevertheless, we include it in this study in order to assess its validity and consistency. The app name is not only available in an app's metadata on the market, but also listed in the APK's manifest file. The latter is visible once the app is installed on a device. While we expect that both names are the same, we empirically verify this assumption in Section 8.6.3 and we validate the uniqueness of the app name across all the apps published within every market.

Developer information. Developers publishing apps on markets are identified by a *developer name* and (depending on the market) contact information, such as an *email address* or a *website*, as well as a *privacy policy URL*. The potential value of developer details as attribution signals depends on whether they are used to attribute two apps *on the same market* to the author (intra-market), or whether they are attributed *across markets* (inter-market). In contrast to the app and developer names, we collect the other three signals only for the Google Play Store.

Signing certificate. Android apps must be signed with a self-signed certificate, which in principle can act as an attribution signal (§8.2.1). Since the private key associated with the certificate is supposedly kept secret by its owner, we consider that two apps signed by the same certificate belong to the same organization. Google Play recommends the signing certificate to be valid for over 25 years, implying this is sufficient for the life cycle of the app [E48]. Furthermore, Google does not provide certificate revocation methods for the Play Store, instead recommending to either rotate keys [E83] or to replace the signing key altogether [E84] in case of a private key compromise. As a result, a certificate used to sign an APK remains valid for the full life cycle of an Android app. Each X.509 certificate contains a *subject* field, which indicates the *owner* of the certificate, and an *issuer*, which serves as an indication of the entity that provided the certificate to the owner. However, in the case of self-signed certificates the subject and issuer field are the same, and developers can generate and sign their apps with a self-signed certificate with arbitrary information in both fields. In addition, for apps that delegate their signing process to Google Play (§8.2.1), all certificates share the same subject field.

Discarded signals. Other techniques previously used in the research literature, such as code diffing, market metadata, the app icon, app screenshots, or the app description, could be considered as potential (visual, in some cases) attribution signals. However, they present significant scalability and interpretability challenges that impede their application across markets and app versions [E12, E35, E85–E88].

8.5 Dataset

In order to build a comprehensive and cross-market dataset of app signals, we implement a crawler based on Scrapy [E89] which downloads APKs as well as

Table 3: Dataset overview in terms of total number of market entries and APKs, as well as unique per package names (PkgN). The last column shows the percentage of unique package names for which we collected both the APK and market metadata.

Market	Market Entries		APKs		Overlap
	Total	PkgN	SHA-256	PkgN	
Google Play	1,293,738	873,059	776,395	723,128	83%
APKMonk	682,112	456,698	345,535	287,421	63%
Tencent	209,950	131,550	143,271	129,277	98%
Mi	25,350	19,396	15,766	13,894	72%
APKMirror	13,273	896	9,784	885	99%
Baidu	12,532	4,440	9,819	4,440	100%
Huawei	9,705	5,203	7,212	5,203	100%
Qihoo 360	5,943	4,365	4,748	4,365	100%
F-Droid	5,221	1,537	3,044	1,462	95%

their corresponding app metadata from nine markets with varying features: the Google Play Store, which is the largest market in terms of users and number of apps; three alternative markets: APKMonk [E90], APKMirror [E91], which is an aggregator of apps published in other stores, and F-Droid [E92], which deliberately position themselves as an alternative to the Play Store for open-source apps; and five major app markets with a large market share in China and Asia: Mi [E93], Baidu [E94], Tencent [E95], Huawei [E96] and Qihoo 360 [E97].

Dataset collection. For each market, we automatically crawl app profiles and harvest the market signals described in Section 8.4 from the page's HTML, when available. We observe that the availability of these signals is highly inconsistent in markets like F-Droid, and we choose not to collect them in this case. This crawling process resulted in several errors, such as gathering truncated developer names from Huawei, as the market only lists a fixed number of characters. We refer to the full set of signals extracted from the app profile in the market as a *market entry*. We kickstarted each market crawl with a seed of known package names and identified further profile pages by following links found in the downloaded profile pages. We deployed the crawler on separate occasions between Dec. 2019 and Aug. 2020 in an attempt to discover newly published apps. This approach also allowed us to collect multiple market entries for a subset of apps.

Table 3 provides a general overview of our dataset. The two market entry columns show the total number of market entries collected from the market (left column) and the number of unique package names for which we collected a market entry (right column). Our app discovery strategy yields a different number of apps for each market, as we lack a complete and comprehensive list of apps per market to exhaustively crawl each of them. The APK column shows the total number of unique APK files as identified by their SHA-256 hash (left column) and the number of unique package names for which we downloaded APK files (right column). The discrepancy between the package name counts in each category is caused by the crawling process, as in some cases we could not download APK files in conjunction with market signals (due to server errors, timeouts, or rate limiting). The overlap column denotes the percentage of unique package names for which we have both the APK and market metadata simultaneously at least once. For the remainder of the paper, we exclude all market entries for which we failed to collect the accompanying APK. In addition, for each unique package name on a market, we use only the most recently collected market entry for evaluation purposes (§8.6.1, §8.6.3, §8.6.4, §8.6.5 and §8.7). The exception to this approach is §8.6.2, in which we perform a longitudinal analysis of the signals and, therefore, consider all market entries collected for a given package name for each market.

8.6 Attribution signal analysis

Each app market has its own policies for publishing apps (§8.2) with different degrees of enforcement (§8.6.1). All attribution signals are nonetheless self-declared, so it is possible that developers (intentionally or by mistake) publish apps under false information (see Appendix 8.F). This section explores this issue by measuring the presence, volatility, consistency, and validity of attribution signals for identifying developer accounts, both within a particular market and across markets.

8.6.1 Missing attribution signals

Table 4 shows the percentage of unique package names for which there are missing signals, across markets. This preliminary analysis is critical to assess the enforcement of market-specific publication policies and the availability and validity of the different attribution signals.

Market metadata. Nearly all apps collected across the markets are published under a developer name, although there are isolated cases on Google Play (6) and APKMonk (160) without one. Of the cases on the Play Store, most apps have been unlisted either by the developer or by the platform operator, except for one app [E98] that remains active as of this writing. Unfortunately,

Table 4: Percentage of unique market entries with missing signals on the different markets.

	Signal	Google Play	APKMonk	Tencent	Mi	APKMirror	Baidu	Huawei	Qihoo 360	F-Droid
Market	Developer name	<0.1%	<0.1%	0%	0%	0%	—	—	0%	—
	Developer website	35%	—	—	—	—	—	—	—	—
	Privacy policy URL	18%	—	—	—	—	—	—	—	—
	Developer email	<0.1%	—	—	—	—	—	—	—	—
Certificate	`commonName`	7%	10%	9%	7%	11%	7%	8%	9%	<0.1%
	`organization`	18%	25%	20%	23%	10%	21%	21%	18%	<0.1%
	`organizationalUnit`	28%	37%	30%	23%	25%	23%	25%	24%	<1%
	`locality`	25%	35%	28%	24%	17%	21%	24%	24%	<1%
	`state`	30%	40%	32%	28%	21%	25%	27%	27%	<1%
	`country`	22%	30%	27%	27%	16%	24%	25%	23%	<1%

we lack information to identify the reason behind the removal. In the case of APKMonk, manual inspection shows that missing metadata happens due to a limitation in our crawler, as APKMonk relies on CloudFlare's email obfuscation [E99] and developer names are recognized as email addresses whenever they contain the "@" character. More significant, however, is the extent to which developer websites (35%) and privacy policy URLs (18%) are missing across markets. Without this information, it becomes challenging for end-users, regulators and researchers to identify the legal or liable entity behind an app.

Signing certificates. When looking at the signing certificates' subject, we find that relative distinguished name (RDN) components are missing for a significant fraction of unique market entries, ranging from 7% for the common names on (among others) Google Play to 40% on APKMonk. The lack of oversight on the self-signed certificates results in the inconsistent availability of these RDNs. A clear exception is F-Droid, whose published apps tend to be signed by its signing framework, which embeds all RDNs in the certificate.

8.6.2 Attribution signal volatility

The Android ecosystem is highly dynamic, with new apps being published and apps being updated regularly [E19, E100]. One effect of this dynamism is that signals are also prone to change, resulting in updates of the app's profile. Signal changes can be relatively small, such as fixing errors in the metadata, or more impactful, such as apps being transferred from one entity to another.

Table 5: Signal volatility, i.e., percentage of package names per market for which we observed a change in signals over time. The package name coverage reports the percentage of the total unique packages within the market for which we have several versions (collected between Dec. 2019 and Aug. 2020).

		Google Play	**APKMonk**	**Tencent**	**Mi**	**APKMirror**	**Baidu**	**Huawei**	**Qihoo 360**	**F-Droid**
	Package name coverage	34%	28%	43%	39%	39%	38%	76%	25%	84%
Market	App name	2.54%	0.09%	0.32%	1.16%	5.87%	1.82%	1.19%	1.00%	0.30%
	Developer name	2.09%	0.02%	0.68%	0.68%	0.29%	—	—	0.45%	—
	Developer website	1.63%	—	—	—	—	—	—	—	—
	Privacy policy URL	1.57%	—	—	—	—	—	—	—	—
	Developer email	1.09%	—	—	—	—	—	—	—	—
Cert.	Added	0%	0%	0%	0%	0%	0.06%	0.03%	0%	0%
	Removed	0%	0%	0%	0.02%	0%	0%	0%	0%	0%
	Fully replaced	0.01%	0.02%	0.11%	0%	0.29%	0.47%	0.03%	0%	0.2%

This subsection investigates the degree of *volatility* (or changes of value) of signals over time and their impact on attribution efforts.

To measure volatility, we consider the subset of package names in our dataset for which we collect multiple market entries. This varies across markets, ranging from 25.2% for Qihoo 360 to 83.7% for F-Droid. For this subset, we measure the percentage of package names whose signal values changed during the course of our measurements, across collected market entries for that particular package name. The results are shown in Table 5. All markets seem to allow app names and developer names of published app to change. The app name volatility ranges from 0.09% of package names on APKMonk to 5.87% of package names on APKMirror. The volatility of the developer name ranges from 0.02% on APKMonk to 2.09% on Google Play. For the signals captured on Google Play only (i.e., the developer website and email address, and the privacy policy URL) we see a similar volatility percentage (1.63%, 1.57% and 1.09% respectively).

We acknowledge that it is likely that some of the cases in which a signal changes over time might make no difference for the sake of attribution. We take a deeper look at the signals that have a domain name embedded in them: the developer email address, the developer website, and the privacy policy URL. Signals available from online domains (i.e., SSL certificates and WHOIS information) are stronger than the self-reported signals in Android, and can thus give us an indication how significant our found volatility percentages are. If a signal is volatile across market entries for a given package name, but the embedded domain name is not, the signal can be considered to be non-volatile. The percentages drop to 0.60%, 1.33% and 1.15% for the developer email, the developer website and privacy policy URL respectively. Even though these adjusted volatility percentages are lower than when taking only Android attribution signals into account, the fact remains that signal changes occur on all markets.

Lastly, we turn to the volatility of signing certificates. The certificate percentages in Table 5 represent the fraction of package names that added a new certificate to the APK, removed a certificate or completely changed certificates across their market entries. Even though the percentages for the latter are relatively low, they can contain apps re-published either by their original or by a completely different developer. The self-signed nature of the Android ecosystem disallows us, unfortunately, to differentiate between these two cases due to the lack of ground truth.

8.6.3 Signal consistency within apps

While we collect different signals for each app from the market metadata and the APK file (see §8.4), one signal type that is present in both is the app name. For every market entry of a given app, we compare the market metadata with the name in the manifest file and observe that only in 46% of the cases the name is exactly the same. While we see that these inconsistencies are less common in some of the markets—namely Huawei, Baidu, F-Droid and Mi, in which nearly 90% of apps names are consistent between the market metadata and the manifest file—none of them achieve 100% consistency, which hints towards the lack of any automatic enforcement mechanisms.

Such signal inconsistencies impede attribution at the app-name level and make external analysis of these apps harder as one has to decide whether to rely on market metadata or the app's manifest (e.g., when researchers make their results available to the public). The possibility to create an inconsistency between the market and the installed app can be (and has already been) abused by malware and phishing attacks [E101]. To better quantify the extent of this issue, we measure, for those apps in which there is not an exact match, how similar both names are. To do so, we rely on the cosine similarity metric [E102]. We find that around half of the apps have a similarity below 60%. In some cases these differences are a result of language differences (e.g., we find apps for which the manifest name is in Russian and the market entry is in English), while in other cases one name is a substring of the other (e.g., *"racing lap timer & stopwatch"*—*"laptimer"*, *"localwifinlpbackend"*—*"wifi location service"*). Finally, we find some cases in which both names appear to be completely unrelated (e.g., *"marshmallow adventure"*—*"flappy candy"* or *"filebox"*—*"myfaves"*).

8.6.4 Signal consistency within markets

In this subsection, we assess signal consistency within a given market from a developer perspective. It is assumed that all apps are signed by the same certificate, as Android relies on this information for many of its developer-based features (i.e., apps with the same certificate can share their functionality). However, we observe that it is common across all markets for apps to be signed with different certificates. We count the number of certificates used to sign every app in our dataset (by unique hash) and find that while the vast majority of apps are signed with only one certificate, we find examples of apps signed with two certificates. For our analysis, if a given app is signed by several certificates, we group all of them together (for the sake of completeness).

We first group the entries by the SHA-256 fingerprint of the signing certificate and count the number of apps published under more than one developer name. We rely on certificate information as this field should only be available to the developer (i.e., we assume that the developer is the one who generates the certificate used for singing an app). To do so, we normalize all strings to lowercase and use exact string matching. Note that we exclude Baidu, Huawei and F-Droid from this study, as the developer name is not available on these markets (§8.4). We find that it is relatively common for apps to be published under more than one developer name in all markets. The percentage of market entries whose APKs share a signing certificate but are published under different developer names varies from 11% for APKMirror to 23% for Qihoo 360. On Google Play this issue concerns 15% of market entries. While some of these market entries are likely to be apps from the same developer published under different developer names, other cases are due to app building frameworks, such as Andromo [E103] or AppyPie [E104]. These companies offer solutions to app publishers to off-load the process of creating their app, either by providing a framework for automatic app creation or by developing the app for the publisher [E34]. The final app is signed directly by the app building framework, which results in many unrelated apps sharing the same signing certificate. We study such app building frameworks in depth in Section 8.7.2.

We now conduct the opposite measurement (i.e., we group market entries by their developer name, and count how many unique certificates are signing these entries). We find that it is common practice for developers to use more than one certificate: the percentage of market entries published by a developer that uses more than one certificate varies from 30% for APKMirror up to 63% for Google Play. In fact, we find that the majority (80%) of Google Play developers with multiple certificates use a different certificate per market entry.

Using the same method, we group the market entries by their app name, and count the number of apps published under more than one developer name. Again, we ignore entries from Baidu and F-Droid. We find that the percentage of entries that share the same app name but are published by different developers is lower overall, but still significant: it varies from 0.3% for APKMirror to 8% for the Google Play Store. A notable example on Google Play is the app name *Messages*, which has been published under eight different package names, with Google's own version being the most popular with more than a billion downloads. These numbers are similar when counting the number of apps published under more than one unique package name: the percentage of market entries varies from 0.9% for Mi to 8% for the Google Play Store.

These results complicate accurate attribution as one cannot fully trust either the developer name or the app name. Furthermore, attackers might take advantage of this lack of enforcement by publishing clones under the name of the original app. Other signals such as the developer website or email address might help fix this issue; however, such signals are only available on Google Play. We investigate the consistency of these signals in depth in Section 8.7.

8.6.5 Signal consistency across markets

The package name of an app is a unique identifier at the Android OS level. This principle is followed on individual markets, with only one app to be advertised under a particular package name. However, since a market operator can only enforce this policy on their own market, the association of the unique package name with a particular app might not prevail from market to market. As a result, self-declared signals across markets can be even more unreliable than they are within a single market. For instance, two different entities might publish each an app, both sharing the same package name, under the same developer name but on two different markets. The signals would suggest the same underlying developer, even though this is not the case.

We investigate how inconsistent signals across markets are in the current ecosystem. First, we evaluate the number of apps (by package name) that are published on more than one market. In total, 1,040,868 package names have been published across the various markets. Of those, 106,574 (or 10.24%) have been published on multiple markets (see Appendix 8.B for a breakdown across pairs of markets). Figure 1 shows the percentage of apps, per market pairs, that are signed by the same certificate on both markets. The figure shows the impact of F-Droid's (and to a lesser extent Google's) signing delegation, resulting in nearly no package names being published on other markets with the same certificate. Even though both markets have a similar signing framework, the overlap of package names published on Google Play with other markets is vastly higher. Of the apps co-published on both Google Play and another market, a relatively small percentage (ranging from 0% for F-Droid and Huawei to 24.4% for Tencent) of package names are signed by a Google certificate. Those are examples of apps that are likely to have first been published on the Play Store, after which their signed APKs were published on the other market (as recommended by Google [E84]). As such, Google Play's signing policy does not only affect their own market but has also started to affect attribution of apps across markets.

Similarly, we compute the percentage of package names published under the same app name between market pairs. Among the Chinese markets, the overlap of app names is relatively high: 81.06% of package names published on a pair of Chinese markets are published under the same app name. This percentage for western markets is even higher (93.56%), but for those pack-

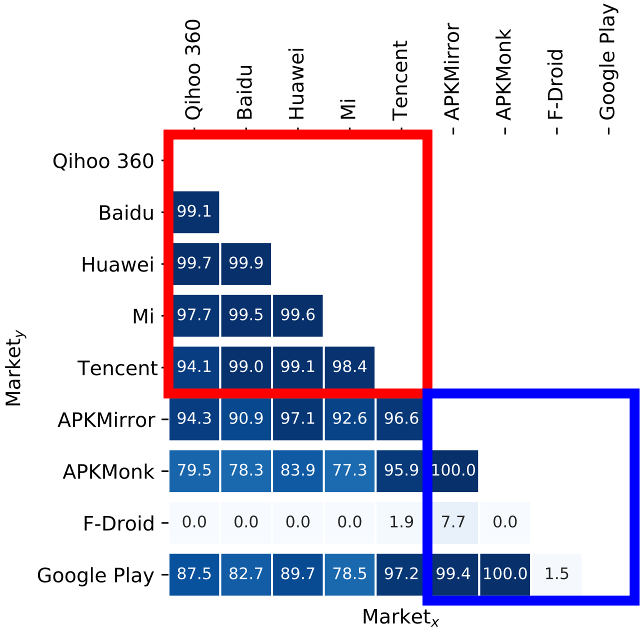


Figure 1: Percentage of unique package names published on both market$_x$ and market$_y$ that have been signed using the same certificate.

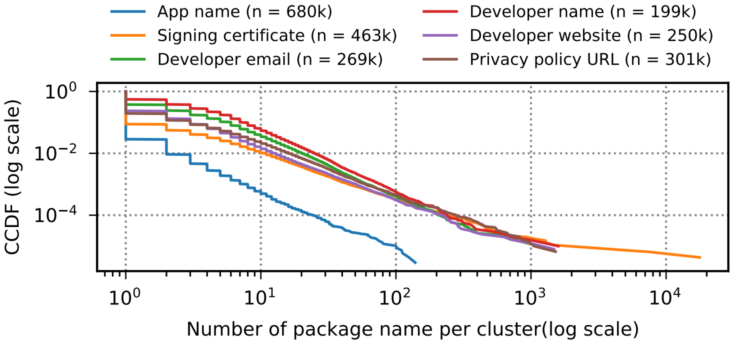


Figure 2: Complementary CDF of the number of package names per cluster, broken down by signal.

ages names published across a western and Chinese market, the percentage is only 53.47%. Similarly, we compute these percentages for the developer name. Across Chinese markets package names are published under the same developer name in 66.82% of the cases, and 92.13% among western markets. Across Chinese and western markets this percentage is 65.98%. The general poor overlap between western and Chinese markets can be explained by the language and alphabet difference, with apps on Chinese markets largely being advertised with Chinese characters, and primarily advertised on western markets with the Latin alphabet, and in rarer cases other languages such as Russian.

These results highlight the overall weakness of the signals available for attribution across markets. Even those signals that are relied upon when studying individual market dynamics, and used by the research community for attribution (§8.3), are inconsistent intra- and inter-market and cannot be trusted for cross-market attribution (e.g., the certificate).

8.7 Case study: Google Play Store

The results in previous section confirm that available attribution signals are unsound. This may be a side effect of lax enforcement of market policies (or the lack of thereof), as suggested by empirical evidence (see Appendix 8.F). Here we focus on measuring the accurate attribution using signals only from the Google Play Store, as (i) it is the official and most prevalent Android marketplace [E64], (ii) it is the most rich in signals from this market, and (iii)

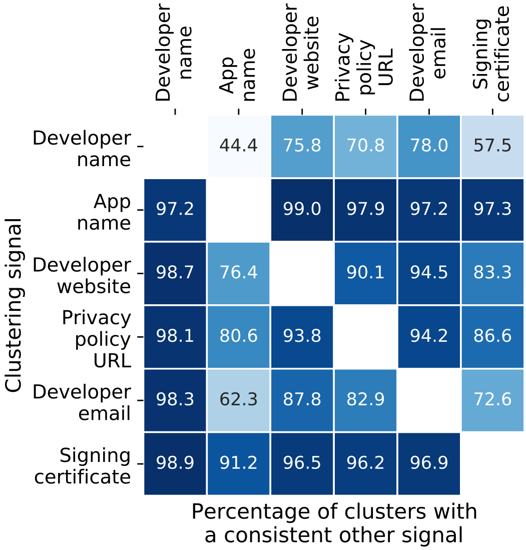
Google Play has a significant number of publication policies (§8.2.2).

8.7.1 Signal-based clustering

We have seen that individual signals can be inconsistent, even within a single market (§8.6.4). In order to explore the different perspectives that the choice of signals can provide, we cluster the Google Play apps in our dataset following the rule that apps in the same cluster have the same signal value. We cluster Google Play apps for each of the 6 signals: developer name, app name, developer website, privacy policy URL, developer email, and signing certificate (as identified by its SHA-256 fingerprint). Figure 2 shows the complementary cumulative distribution function (CCDF) of the number of unique package names in each cluster (i.e.,, the cluster size) in a log-log scale, broken down by signal. For each signal we also report the final number of clusters we obtain. The developer name signal divides the dataset in the least amount of clusters (199k), and the app name in the most number of clusters (680k). We argue that app names are not a signal that uniquely identifies an app (given the curve in the CCDF), thus confirming the results presented in the previous section. Even though the other signals tend to cluster more package names together, they are more relevant as as signal for author attribution. Furthermore, the developer name can be an appropriate signal to group together apps that have been published by the same company (as this signal is unique within Google Play).

We explore how many other unique signal values these clusters contain as a proxy to assess the consistency of other attribution signals in conjunction with those present in a given cluster. For instance, we find two apps published under the developer name *Rad3 Limited* but with two different email addresses (*mark@smalesfarm.co.nz* and *contact@wakachangi.com*), showing an inconsistency between the developer name and the developer email address. While these inconsistencies can be legitimate, they create problems for attribution, as they disconnect apps that are indeed related to the same company.

For each of the clusters created by the 6 available attribution signals, we investigate the consistency with the remaining 5 signals. Figure 3 shows the results as a heatmap for each pair of clustering signal (y-axis) and targeted signal (x-axis). The value in each cell shows the percentage of clusters containing only a unique value of another signal; e.g., in 44.4% of the clusters created by the developer name signal, there is more than a single app name, implying that the remaining 55.6% of developer names publish more than one app on the Google Play Store. Figure 3 shows that there is a relatively low correlation between the developer name and other developer-related signals:



	Developer name	App name	Developer website	Privacy policy URL	Developer email	Signing certificate
Developer name		44.4	75.8	70.8	78.0	57.5
App name	97.2		99.0	97.9	97.2	97.3
Developer website	98.7	76.4		90.1	94.5	83.3
Privacy policy URL	98.1	80.6	93.8		94.2	86.6
Developer email	98.3	62.3	87.8	82.9		72.6
Signing certificate	98.9	91.2	96.5	96.2	96.9	

Figure 3: Heatmap of the signal consistency between clustering signal and target signals.

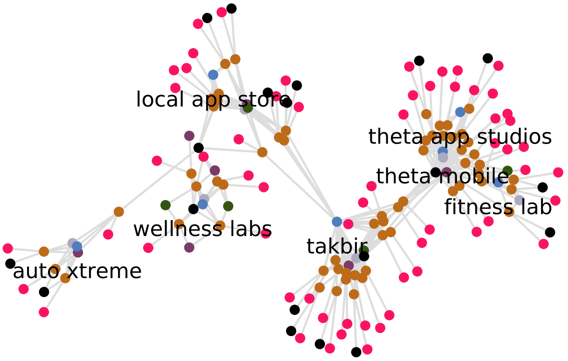


Figure 4: A cluster from the Google Play attribution graph annotated with developer names (● = market entry, ● = developer name, ● = developer website, ● = developer email, ● = privacy policy URL, ● = app name, ● = certificate).

24.2% of developer names publish apps with more than one developer website and 22.0% with multiple developer email addresses. Furthermore, only 57.5% of developer names publish apps with a single signing certificate. A salient result is that relying only on the certificate to identify apps developed by the same entity may be short-sighted. Conversely, there are signals that are re-used across multiple developer names. Although this represents a small fraction of the respective signal (e.g., 1.1% of certificates are used across developer names), it shows that some of the self-reported contact information is currently inconsistent across developer names on Google Play.

8.7.2 Multiple signal attribution graphs

We have seen that individual signals tend to correlate poorly with other signals on Google Play. To explore this issue in more depth, we create *attribution clusters* on multiple signals, rather than just one. To do so, we model unique market entries and signals as nodes in a graph, with edges between them representing the measured relationship between the market entry and its signals. This large attribution graph consists of connected components, or clusters, that we can consider to be associated with a particular entity. Figure 4 shows an example cluster. Additional examples are provided in Appendix 8.C.

The resulting graph contains 150,630 clusters for 718,741 market entries. Figure 5 shows a CCDF of the size of these clusters in terms of package names

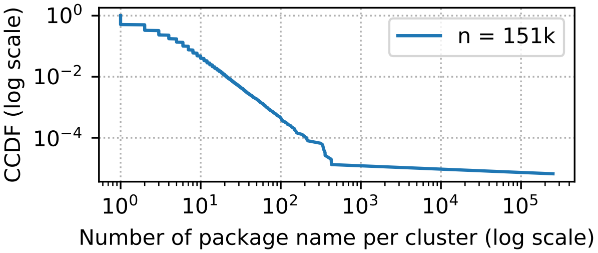


Figure 5: # of apps per cluster based on all signals.

contained in each cluster. We find that 73,703 (or 10.25%) of market entries are isolated in their own cluster, and that the largest cluster comprises 250,937 (or 34.91% of all) market entries. We find 5,501 (or 3.7%) clusters to be fully consistent (i.e., all apps in these clusters are published under the same signals, except for the app name)), indicating that the majority of these clusters contain ambiguous information. We count the number of certificates (by their unique hash) used to sign every app in our dataset. We find that while the vast majority of apps are signed with only one certificate, there are 31 examples of apps signed with two certificates. In these cases, the app signed by two certificates can bridge two, otherwise non-connected apps, with each other. The size of this large cluster suggests that a naive attribution graph creation method can lead to substantial over-attribution. To identify the signal causing this over-attribution, we compute the *betweenness centrality* of each signal, a centrality measure expressing the fraction of the shortest paths in the graph going through a particular node [E105]. The intuition behind this approach is that the most central node must be key for connecting vertices in the graph that would otherwise not be connected, and therefore must be a likely cause for over-attribution. When looking at the centrality value of nodes, we see that the vast majority have a centrality close to zero, with a few outliers with a relatively high centrality. Hence, we assume these nodes are important causes for the over-attribution in this largest cluster.

Highly central nodes. By investigating highly-central signals we find patterns that make attribution challenging. App names are not unique on the Play Store, thereby making it difficult to differentiate between these apps. Examples include *BMI Calculator* (published 79 times by 78 different developers), *Music Player* (published 100 times by 98 developers) and *Flashlight* (published 139 times by 136 developers). We also observe that the highest centrality corresponds to a set of certificates which are highly prevalent on Google Play. Relying on the subject information, we manually analyze the top 10 companies (by centrality of the certificate) and observe that all are related to frameworks or companies that build apps for others [E34]. Namely, the signal with the highest centrality is a certificate associated with Andromo, which is used to sign 17,721 apps. Andromo is an app development framework to build apps based on pre-existing components [E103]. It builds APKs that are all signed by the same certificate and are ready to submit for publication on the market. Andromo does not mention this signing practice in their terms of service [E106], and in fact recommends against enrolling for signing delegation to the Play Store [E107]. Apps built with Andromo have been published under 1,602 different developer names, which—other than the signing certificate—have little in common with each other. In this particular case, the package name also indicates the use of this development framework, as 91.2% of Andromo apps follow the package name scheme `{com,net}.andromo.dev<dev_id>.app<app_id>`. Another example of a highly central certificate is the one used by the Qbiki Network, an organization related to the Seattle Clouds company that used to develop apps for different OSes [E108] (the service seems to be discontinued in 2021). Their certificate was used across 7,579 apps published by 961 developer names. Examples of still operating app building companies with highly central certificates are QuickApp Ninja [E109], AppyPie [E104] and AppMachine [E110], having signed 512, 938, and 216 apps respectively. Appendix 8.D provides a full overview of the number of signals used across the apps published by these app builders.

As a result of this practice, customers are fully reliant on the app builder to provide updates for the app, as the initial signing certificate is required to do so. In addition, as we discuss in §8.8, certificates have use cases beyond enforcing update integrity. As a result, the app builders have full control over the declared and used permissions and can potentially automatically grant permissions across apps from different developers without user awareness. Furthermore, from an attribution standpoint, these certificates have become meaningless in identifying the responsible developer behind an app. When looking at the privacy policy as a centrality signal, we see a similar pattern to the app builder certificates: 389 apps developed using AppsGeyser (another app building framework [E111]) share the same privacy policy (hosted on the AppsGeyser website), otherwise not sharing a common signal among apps published by different developers.

Large organizations. The attribution graphs allow us to analyze the signal usage behavior of prominent tech companies. We select a curated list of developer names based on the number of popular apps (i.e., more than 1B downloads) that they published. Given the large number of downloads, we assume these developer profiles are the legitimate ones as opposed to developers trying to imitate the organizations. For these identified developer names, we collect the number of unique signals used across apps published under the same name (i.e.,account on Google Play). We also measure how many other developer names use the same certificate as the main developer account to sign their apps. Most of these organizations use multiple certificates to sign their apps (with the exception of TikTok, WhatsApp and Instagram). An extreme case is that of HP, which has a separate certificate per published app. For three organizations (Google, Samsung and HP), we see that one or more alternative developer name has used one of the certificates to publish an app. Conversely, both WhatsApp and Instagram are subsidiary companies of Facebook Inc., but do not share any cryptographic link with their parent company. A similar relationship holds for Microsoft and Skype. The full results are shown in Table 8.E.1 in Appendix 8.E. Overall, these results paint a diverse picture of publication and development strategies that can impede automatic attribution, even for software released by well-known companies.

8.8 Discussion

Our results show that attribution in the Android ecosystem is a complex and still unsolved problem. While previous research has mostly relied on certificates for attribution, the fact that they can be self-signed implies that the information provided by them can be forged, rendering platform-level attribution unfeasible. In fact, although all markets have a notion of developer, we have shown that this developer is not always the same as the company behind the offered service. The proliferation of app development frameworks and companies that build apps for others calls for a re-definition of roles as well as a need to adjust the way in which researchers approach Android attribution and build their ground truth for different studies.

Implications. Imprecise app attribution has negative consequences in a number of research areas and applications. It affects measurements and other quantitative or qualitative analyses on the app ecosystem for which attribution is a relevant variable, such as understanding market dynamics or identifying deceptive actors and practices. Another area severely impacted is accountability, as the inability to map a given app to the company behind it prevents different stakeholders to act correctly whenever they identify an issue with the app, resulting in a lack of regulatory enforcement. Similarly, attribution issues translate into a lack of transparency for users as the absence of clear signals about which company has published a given app prevents them from making an informed decision about installing it. Furthermore, inconsistencies across markets can lead to users installing an app by mistake, only because it presents an ambiguous signal (e.g., the app name). We have shown not only that, but also that the Play Store vetting process is not fit to detect apps in which developer information is false, leading to potential app clones being published.

Signatures as proof of ownership. The unreliability of the signing certificate has profound consequences on Android security, as it has long been used as a proof of authorship. Prior to API level 29 (Android 10, released in 2019), two apps signed with the same certificate were capable of running under the same Linux user ID, easily share data and even run in the same process [E112]. The Android permission system still automatically grants permissions with a `signature` protection level, if the requesting and declaring apps are signed by the same certificate. Apps signed with the same certificate as the OS can even be granted system permissions [E37]. The use of the signing certificate as a proxy for the ownership of an app extends beyond the operating system itself. With *Digital Asset Links* [E113], developers can declare associations between websites and apps, and even share credentials [E114]. In this case, apps are identified based on their package name and the SHA-256 fingerprint of their signing certificate.

Recommendations. We believe that only the platform itself can solve the predicament of Android attribution. To provide sound attribution mechanisms, the Android ecosystem should move away from self-signing certificates. Other prominent software ecosystems already rely on more sound signature mechanisms, such as Apple issuing all valid certificates for its app store [E25] and Windows relying on a public key infrastructure [E28]. While not perfect [E115], we argue that these approaches could limit the number of certificates that have incomplete or invalid information. We have shown that apps in different markets tend to be signed by the same certificate, meaning that this approach could also help make signals across markets more reliable. If a given app has already been published on another market using a different certificate, the market could either reject the app or show a warning to developers and users that this app might have been tampered with. An alternative option would be to rely on third-party verification for attribution, but we argue that this would simply be a shift of responsibility from the market to third-party entities.

Finally, we recommend that researchers become aware of the limitations of the ecosystem before making any judgments or gathering ground truth, as the use of signing as an attribution signal is not as robust as it seems. We found counterexamples to both the assumption "one developer uses one certificate" as well as "one certificate belongs to a single developer." We argue that the limits of market metadata, and also certificates, should be discussed in the context of their use as attribution signals, because neither is an infallible source of attribution.

8.9 Conclusions

We have shown that the validity, availability, and fidelity of attribution signals vary both within and across markets due to the diversity (and lack of) policies and enforcement mechanisms (including no enforcement at all). We crawled different markets multiple times for close to a year and empirically demonstrated that attribution signals such as developer websites and privacy policies are missing (or can be tampered) for a large portion of entries. Developer's deceptiveness, malpractice, and lack of maintenance, along with the lack of automatic policy enforcement by market operators lead to incorrect and outdated information being shown on market profiles. We have shown that the same developer can use multiple certificates, and that the same certificate can even be used by multiple developers (e.g., by using app development frameworks). New signing practices, such as Google signing apps themselves, affect attribution even across other markets. In summary, our empirical findings suggest that none of the currently available signals are sufficient for in-market or intra-market attribution of apps.

References

[E1] "Desktop vs Mobile vs Tablet Market Share Worldwide." `https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet`, 2021. "Accessed March 2021".

[E2] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis–1,000,000 apps later: A view on current Android malware behaviors," in *2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, IEEE, 2014.

[E3] J. Crussell, C. Gibler, and H. Chen, "Attack of the clones: Detecting cloned applications on Android markets," in *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, pp. 37–54, 2012.

[E4] "Who is to blame for the malicious barcode scanner that got on the Google Play store?." `https://blog.malwarebytes.com/android/2021/`

02/who-is-to-blame-for-the-malicious-barcode-scanner-that-got-on-the-google-play-store/, 2021. "Accessed March 2021".

[E5] "Android users looking for Elon Musk on Clubhouse caused an identically named app to pull itself." https://9to5google.com/2021/02/01/clubhouse-android-app-elon-musk/, 2021. "Accessed April 2021".

[E6] "Fake Trezor app steals more than $1 million worth of crypto coins." https://blog.malwarebytes.com/social-engineering/2021/04/fake-trezor-app-steals-more-that-1-million-worth-of-crypto-coins/, 2021.

[E7] "He believed Apple's App Store was safe. Then a fake app stole his life savings in bitcoin." https://www.washingtonpost.com/technology/2021/03/30/trezor-scam-bitcoin-1-million/, 2021.

[E8] B. Kim, K. Lim, S.-J. Cho, and M. Park, "Romadroid: A robust and efficient technique for detecting ndroid app clones using a tree structure and components of each app's manifest file," *IEEE Access*, pp. 72182–72196, 2019.

[E9] H. Wang, Y. Guo, Z. Ma, and X. Chen, "Wukong: A scalable and accurate two-phase approach to Android app clone detection," in *Proceedings of the International Symposium on Software Testing and Analysis*, 2015.

[E10] K. Chen, P. Liu, and Y. Zhang, "Achieving accuracy and scalability simultaneously in detecting application clones on Android markets," in *Proceedings of the International Conference on Software Engineering*, 2014.

[E11] L. B. Platon Kotzias, Juan Caballero, "How did that get in my phone? unwanted app distribution on Android device," in *IEEE Symposium on Security and Privacy (SP)*, 2020.

[E12] V. Kalgutkar, N. Stakhanova, P. Cook, and A. Matyukhina, "Android authorship attribution through string analysis," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pp. 1–10, 2018.

[E13] M. Ali, M. E. Joorabchi, and A. Mesbah, "Same app, different app stores: A comparative study," in *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*, 2017.

[E14] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, and T. Karagiannis, "Rise of the planet of the apps: A systematic study of the mobile app ecosystem," in *Proceedings of the Internet Measurement Conference (IMC)*, 2013.

[E15] H. Wang, Z. Liu, Y. Guo, X. Chen, M. Zhang, G. Xu, and J. Hong, "An explorative study of the mobile app ecosystem from app developers' perspective," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2017.

[E16] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis," in *Proceedings of the USENIX Security Symposium*, (Washington, D.C.), pp. 785–800, USENIX Association, Aug. 2015.

[E17] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[E18] V. Avdiienko, K. Kuznetsov, A. Gorla, A. Zeller, S. Arzt, S. Rasthofer, and E. Bodden, "Mining apps for abnormal usage of sensitive data," in *Proceedings of the International Conference on Software Engineering*, 2015.

[E19] J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez, "Bug fixes, improvements,... and privacy leaks: A longitudinal study of PII leaks across Android app versions," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[E20] Y. Hu, H. Wang, R. He, L. Li, G. Tyson, I. Castro, Y. Guo, L. Wu, and G. Xu, "Mobile app squatting," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2020.

[E21] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2011.

[E22] M. Lindorfer, S. Volanis, A. Sisto, M. Neugschwandtner, E. Athanasopoulos, F. Maggi, C. Platzer, S. Zanero, and S. Ioannidis, "AndRadar: fast discovery of Android applications in alternative markets," in *Proceedings of the International conference on detection of intrusions and malware, and vulnerability assessment (DIMVA)*, 2014.

[E23] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, "Beyond Google Play: A large-scale comparative study of chinese Android app markets," in *Proceedings of the Internet Measurement Conference (IMC)*, 2018.

[E24] E. Okoyomon, N. Samarin, P. Wijesekera, A. Elazari Bar On, N. Vallina-Rodriguez, I. Reyes, Á. Feal, and S. Egelman, "On the ridiculousness of notice and consent: Contradictions in app privacy policies," *Workshop on Technology and Consumer Protection (ConPro)*, 2019.

[E25] "What You Need To Enroll." `https://developer.apple.com/programs/enroll/`. Acccessed: 16-09-2021.

[E26] "Identity Verification." `https://developer.apple.com/support/identity-verification/`, 2021.

[E27] B. Kaliski, "Pkcs# 7: Cryptographic message syntax version 1.5," 1998.

[E28] D. Kim, B. J. Kwon, and T. Dumitraş, "Certified malware: Measuring breaches of trust in the windows code-signing PKI," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[E29] D. Barrera, J. Clark, D. McCarney, and P. C. van Oorschot, "Understanding and improving app installation security mechanisms through empirical analysis of Android," in *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2012.

[E30] L. Wu, M. Grace, Y. Zhou, C. Wu, and X. Jiang, "The impact of vendor customizations on Android security," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2013.

[E31] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of Google Play," in *The 2014 ACM international conference on Measurement and modeling of computer systems*, 2014.

[E32] K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, W. Zou, and P. Liu, "Finding unknown malice in 10 seconds: Mass vetting for new threats at the Google-Play scale," in *Proceedings of the USENIX Security Symposium*, 2015.

[E33] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Authorship attribution of Android apps," in *Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2018.

[E34] M. Oltrogge, E. Derr, C. Stransky, Y. Acar, S. Fahl, C. Rossow, G. Pellegrino, S. Bugiel, and M. Backes, "The rise of the citizen developer: Assessing the security impact of online app generators," in *IEEE Symposium on Security and Privacy (SP)*, 2018.

[E35] L. Li, T. F. Bissyandé, and J. Klein, "Rebooting research on detecting repackaged Android apps: Literature review and benchmark," *IEEE Transactions on Software Engineering*, 2019.

[E36] P. Liu, L. Li, Y. Zhao, X. Sun, and J. Grundy, "AndroZooOpen: Collecting large-scale open source Android apps for the research community," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*, 2020.

[E37] J. Gamba, M. Rashed, A. Razaghpanah, J. Tapiador, and N. Vallina-Rodriguez, "An analysis of pre-installed Android software," in *IEEE Symposium on Security and Privacy (SP)*, 2020.

[E38] S. Sebastian and J. Caballero, "Towards attribution in mobile markets: Identifying developer account polymorphism," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[E39] S. Aonzo, A. Merlo, G. Tavella, and Y. Fratantonio, "Phishing attacks on modern Android," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[E40] H. Wang, H. Li, and Y. Guo, "Understanding the evolution of mobile app ecosystems: A longitudinal measurement study of Google Play," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2019.

[E41] G. Xu, C. Zhang, B. Sun, X. Yang, Y. Guo, C. Li, and H. Wang, "AppAuth: Authorship attribution for Android app clones," *IEEE Access*, 2019.

[E42] S. Farooqi, A. Feal, T. Lauinger, D. McCoy, Z. Shafiq, and N. Vallina-Rodriguez, "Understanding incentivized mobile app installs on Google Play store," in *Proceedings of the Internet Measurement Conference (IMC)*, 2020.

[E43] N. Marastoni, A. Continella, D. Quarta, S. Zanero, and M. D. Preda, "Groupdroid: Automatically grouping mobile malware by extracting code similarities," in *Proceedings of the 7th Software Security, Protection, and Reverse Engineering / Software Security and Protection Workshop*, 2017.

[E44] "2018 reform of eu data protection rules." https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf.

[E45] "Chapter 3 – rights of the data subject." https://gdpr-info.eu/chapter-3/, 2018. "Accessed May 2021".

[E46] "PyStoreCrawler." https://github.com/kdhageman/pystorecrawler, 2021.

[E47] "App Signing Considerations." https://developer.android.com/studio/publish/app-signing#considerations, 2021.

[E48] "Sign your app." https://developer.android.com/studio/publish/app-signing, 2021.

[E49] H. Wang, H. Liu, X. Xiao, G. Meng, and Y. Guo, "Characterizing Android App Signing Issues," in *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019.

[E50] "New Android vulnerability allows attackers to modify apps without affecting their signatures." https://www.guardsquare.com/en/blog/new-android-vulnerability-allows-attackers-modify-apps-without-affecting-their-signatures, 2017.

[E51] "APK Signing V3 Scheme." https://source.android.com/security/apksigning/v3, 2021.

[E52] "Recent Android App Bundle improvements and timeline for new apps on Google Play." https://android-developers.googleblog.com/2020/08/recent-android-app-bundle-improvements.html, 2020. "Accessed May 2021".

[E53] "App Signing-Service Introduction." https://developer.huawei.com/consumer/en/doc/development/AppGallery-connect-Guides/agc-appsigning-introduction-0000001051379577.

[E54] "F-Droid Signing Process." https://f-droid.org/en/docs/Signing_Process/, 2021.

[E55] "Submit Android apps." https://www.apkmonk.com/submit-app/, 2021.

[E56] "Google Play Policy Center." https://support.google.com/googleplay/android-developer#topic=3450769, 2021.

[E57] “Apkmonk - market policy.” https://www.apkmonk.com/submit-app/, 2021. Accessed May 2021 archived at http://archive.today/2GVPl.

[E58] “Tencent - market policy.” https://open.qq.com/eng/reg, 2021. Accessed May 2021 archived at http://archive.today/Nbusz.

[E59] “Mi - market policy.” https://global.developer.mi.com/home, 2021. Accessed May 2021 archived at http://archive.today/TBHNs.

[E60] “Apkmirror - market policy.” https://www.apkmirror.com/faq/, 2021. Accessed May 2021 archived at http://archive.today/mXflW.

[E61] “Huawei market policy.” https://developer.huawei.com/consumer/en/doc/30202, 2021. Accessed May 2021, archived at http://archive.today/WCYu0.

[E62] “Qihooo 360 - market policy.” http://dev.360.cn/Wiki/0-2.html, 2018. Accessed May 2021, archived at http://archive.today/ZJb3Z.

[E63] “F-droid market policy.” https://gitlab.com/fdroid/fdroiddata/blob/master/CONTRIBUTING.md, 2021. Accessed May 2021, archived at http://archive.today/PBi1p.

[E64] “A look at the Android market (aka Google Play) on its 10th anniversary.” https://techcrunch.com/2018/10/22/a-look-at-the-android-market-aka-google-play-on-its-10th-anniversary/, 2018. "Accessed May 2021".

[E65] “Help protect against harmful apps with Google Play protect.” https://support.google.com/googleplay/answer/2812853. "Accessed May 2021".

[E66] “Google Play Policy Center.” https://support.google.com/googleplay/android-developer/topic/9858052, 2021. Archived at http://archive.today/gBMYU.

[E67] “Google Play Policy Center.” https://support.google.com/googleplay/android-developer/answer/9888077, 2021.

[E68] “Google Play Policy Center.” https://support.google.com/googleplay/android-developer/answer/9888689, 2021.

[E69] “Google Play Policy Center.” https://support.google.com/googleplay/android-developer/answer/9888374, 2021.

[E70] "Google Play Policy Center." https://support.google.com/googleplay/android-developer/answer/9898842, 2021.

[E71] "Mi Developer." https://global.developer.mi.com/document?doc=accountRegistration.becomeADeveloper, 2021.

[E72] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, "Measuring pay-per-install: The commoditization of malware distribution," in *Proceedings of the USENIX Security Symposium*, 2011.

[E73] N. Zhong and F. Michahelles, "Google Play is not a long tail market: An empirical analysis of app adoption on the Google Play app market," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013.

[E74] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," *Telemat. Inf.*, 2011.

[E75] N. d'Heureuse, F. Huici, M. Arumaithurai, M. Ahmed, K. Papagiannaki, and S. Niccolini, "What's app? A wide-scale measurement study of smart phone markets," *Proceedings of the SIGMOBILE Mobile Computing and Communications Review*, 2012.

[E76] E. Blázquez, S. Pastrana, Á. Feal, J. Gamba, P. Kotizias, N. Vallina-Rodriguez, and J. Tapiador, "Trouble over-the-air: An analysis of FOTA apps in the Android ecosystem," in *IEEE Symposium on Security and Privacy (SP)*, 2021.

[E77] V. Rastogi, Y. Chen, and W. Enck, "Appsplayground: Automatic security analysis of smartphone applications," in *Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2013.

[E78] J. Crussell, C. Gibler, and H. Chen, "Andarwin: Scalable detection of Android application clones based on semantics," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2007–2019, 2014.

[E79] C. Soh, H. B. K. Tan, Y. L. Arnatovich, and L. Wang, "Detecting clones in Android applications through analyzing user interfaces," in *2015 IEEE 23rd international conference on program comprehension*, pp. 163–173, IEEE, 2015.

[E80] T. T. Nguyen, M. Backes, N. Marnau, and B. Stock, "Share First, Ask Later (or Never?) - Studying Violations of GDPR's Explicit Consent in Android Apps," in *Proceedings of the USENIX Security Symposium*, 2021.

[E81] "Documentation for app developers." https://developer.android.com/docs/, 2021.

[E82] "Android developers — manifest element." https://developer.android.com/guide/topics/manifest/manifest-element.html#package, 2021.

[E83] "APK signature scheme with key rotation." https://developer.android.com/about/versions/pie/android-9.0#apk-key-rotation, 2021.

[E84] "Use Play App Signing." https://support.google.com/googleplay/android-developer/answer/9842756?visit_id=637574359279602919-373176865, 2021.

[E85] M. Sun, M. Li, and J. C. S. Lui, "DroidEagle: Seamless detection of visually similar Android apps," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '15, (New York, NY, USA), pp. 1–12, Association for Computing Machinery, June 2015.

[E86] J. Rajasegaran, N. Karunanayake, A. Gunathillake, S. Seneviratne, and G. Jourjon, "A Multi-modal Neural Embeddings Approach for Detecting Mobile Counterfeit Apps," in *The World Wide Web Conference on - WWW '19*, (San Francisco, CA, USA), pp. 3165–3171, ACM Press, 2019.

[E87] L. Simko, L. Zettlemoyer, and T. Kohno, "Recognizing and imitating programmer style: Adversaries in program authorship attribution," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, pp. 127–144, Jan. 2018.

[E88] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android HIV: A study of repackaging malware for evading machine-learning detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 987–1001, 2019.

[E89] "Scrapy | A Fast and Powerful Scraping and Web Crawling Framework." https://scrapy.org/, 2021.

[E90] "APKMonk." https://www.apkmonk.com/, 2021.

[E91] "APKMirror." https://www.apkmirror.com/, 2021.

[E92] "F-Droid." https://f-droid.org/, 2021.

[E93] "Xiaomi Mi App Store." http://app.mi.com/, 2021.

[E94] "Baidu App Store." https://shouji.baidu.com/, 2021.

[E95] "Tencent App Store." https://android.myapp.com/, 2021.

[E96] "Huawei App Store." https://appgallery1.huawei.com/#/Featured, 2021.

[E97] "Qihoo 360 App Store." http://zhushou.360.cn/, 2021.

[E98] "Hope mankind foundation." https://play.google.com/store/apps/details?id=com.hmf, 2021. "Accessed May 2021".

[E99] "What is email address obfuscation?." https://support.cloudflare.com/hc/en-us/articles/200170016-What-is-Email-Address-Obfuscation-, 2021.

[E100] P. Calciati, K. Kuznetsov, X. Bai, and A. Gorla, "What did really change with the new release of the app?," in *Proceedings of the 15th International Conference on Mining Software Repositories*, pp. 142–152, 2018.

[E101] Z. Chen, "Thousands of HiddenAds Trojan Apps Masquerade as Google Play Apps." https://www.mcafee.com/blogs/other-blogs/mcafee-labs/multi-tricks-hiddenads-malware/, 2020.

[E102] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Asian conference on computer vision*, pp. 709–720, Springer, 2010.

[E103] "Andromo - Mobile App builder for Android. No coding." https://www.andromo.com/, 2021.

[E104] "Appypie - homepage." https://www.appypie.com/, 2021. "Accessed May 2021".

[E105] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[E106] "Terms and Conditions - Andromo." https://www.andromo.com/en/terms/, 2021.

[E107] "How to put your app in Google Play | Andromo Support." https://support.andromo.com/i47-how-to-put-your-app-in-google-play, 2020.

[E108] "Seattle Cloud." https://web.archive.org/web/20210123083513/http://seattleclouds.com/, 2021.

[E109] "Quickapp Ninja: Create Android games without any coding!." https://quickappninja.com/, 2021.

[E110] "Appmachine." https://www.appmachine.com/, 2021.

[E111] "AppsGeyser: Free App Creator & App Maker. Create Android Apps No Code." https://appsgeyser.com/, 2021.

[E112] "Shared User Id." https://developer.android.com/guide/topics/manifest/manifest-element#uid, 2021.

[E113] "Verify Android App Links." https://developer.android.com/training/app-links/verify-site-associations.

[E114] "Enable automatic sign-in across apps and websites." https://developers.google.com/identity/smartlock-passwords/android/associate-apps-and-sites.

[E115] P. Kotzias, S. Matic, R. Rivera, and J. Caballero, "Certified pup: abuse in authenticode code signing," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[E116] "Bizzness apps - homepage." https://www.biznessapps.com/, 2021. "Accessed May 2021".

[E117] "Mobincube the best APP BUILDER DIY for Android iPhone/iPAD." https://mobincube.com/, 2021.

[E118] "Swiftic: iPhone & Android App Maker - Create Your Own App in Minutes!." https://www.swiftic.com/, 2021.

[E119] https://developers.google.com/android/play-protect/cloud-based-protections, 2021.

[E120] "Google Play Policy Center — Impersonation." https://support.google.com/googleplay/android-developer/topic/9969539, 2021.

[E121] "Average number of new Android app releases via Google Play per month from March 2019 to February 2021." https://www.statista.com/statistics/1020956/android-app-releases-worldwide/, 2021. "Accessed May 2021".

[E122] “Number of available applications in the Google Play store from December 2009 to December 2020.” https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2020. "Accessed May 2021".

Appendices

8.A Attribution research survey

Table 8.A.1 shows a summary of our survey on Android attribution strategies in prior research discussed in 8.7.2.

8.B Cross-market analysis

As a supplement to the discussion on signal consistency across markets in Section 8.6.5, Figure 8.B.1 illustrates the overlap between apps published across markets based on the package name. Figures 8.B.2 and 8.B.3 further illustrate the consistency of the app name and developer name signals across markets.

8.C Example attribution graphs

As a supplement to the attribution graphs discussed as part of our our case study on Google Play in Section 8.7.2, Figure 8.C.1 shows four additional examples of attribution graphs of different sizes.

8.D Details on app builders

As a supplement to our discussion in Section 8.7.2, Table 8.D.1 illustrates the number of unique signals for the most popular app builders used by apps on the Google Play Store, in terms of the centrality of the certificate used to sign apps.

Table 8.A.1: Overview of selected related work and the signals used for attributing apps: C = Certificates, D = Developer as presented in market(s), O = Other signals.

Ref	Year	Research Goal	C	D	O
[E3]	2012	Clone detection	✓		
[E29]	2012	Inter-market malware study	✓		
[E14]	2013	Inter-market study		✓	
[E30]	2013	Pre-installed apps study	✓		
[E77]	2013	Malware study			
[E31]	2014	Intra-market study, clone detection	✓	✓	
[E22]	2014	Market measurement, clone detection	✓		✓‡§
[E32]	2015	Inter-market malware study	✓		
[E13]	2017	Inter-market study		✓	✓*
[E15]	2017	Intra-market study		✓	
[E43]	2017	Malware, clone detection			✓§
[E33]	2018	Clone detection	✓		
[E12]	2018	Clone detection	✓		
[E34]	2018	Online app generator study	✓		✓†‡¶
[E39]	2018	Phishing study		✓	✓‡
[E23]	2018	Inter-market study	✓		
[E35]	2019	Clone detection	✓		
[E40]	2019	Intra-market study		✓	
[E41]	2019	Code stylometry		✓	✓*
[E36]	2020	Dataset creation	✓		
[E37]	2020	Pre-installed apps study	✓		✓†
[E42]	2020	Incentivized install study		✓	
[E11]	2020	Malware/pre-installed apps study	✓	✓	✓†
[E38]	2020	Dataset creation, malware study	✓	✓	✓†

Additional signals: *app name, ‡package name, †certificate fields (e.g., Issuer or Organization), §code properties (e.g., method signatures or the control flow graph), ¶resource files in the app's package.

8.E Details on large organizations

As a supplement to our discussion in Section 8.7.2, Table 8.E.1 shows details on the signal usage by high-profile organizations active on the Google Play

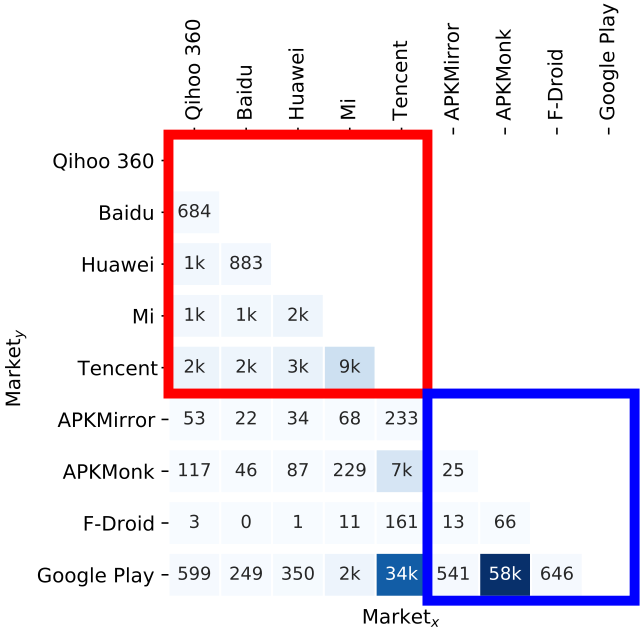


Figure 8.B.1: Total number of unique package names published on both market_x and market_y.

Store.

8.F Enforcement of publication policies on Google Play

We investigated the enforcement of the publication policies, particularly those related to attribution signals, through different experiments on the Google Play Store. Our choice of Google Play is motivated by its global prevalence and official nature, and because it has the most comprehensive set of publication policies among the markets we surveyed (§ 8.2.2). Each app submitted to Google Play goes through an automated vetting process that flags it for manual inspection if it is found to contain suspicious behavior [E119]. As impersonation and tampering is explicitly prohibited (§8.2.2), apps aiming to violate policies can be expected to be rejected by the market. The primary goal of our experiment is to assess the robustness of this vetting process to detect and prevent the publication of apps that attempt to impersonate or tamper

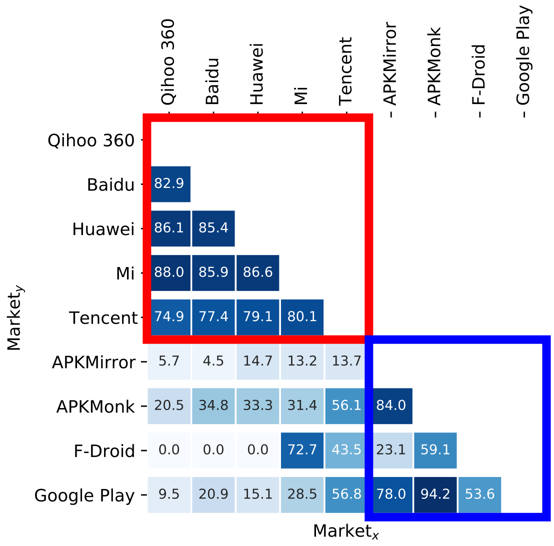


Figure 8.B.2: Percentage of unique package names published on both market$_x$ and market$_y$ under the same app name.

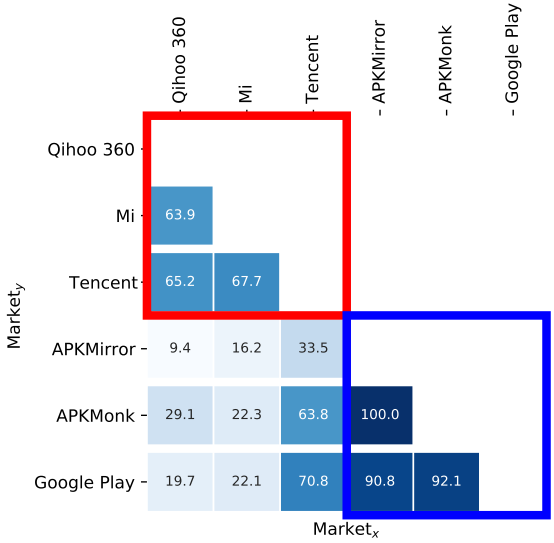
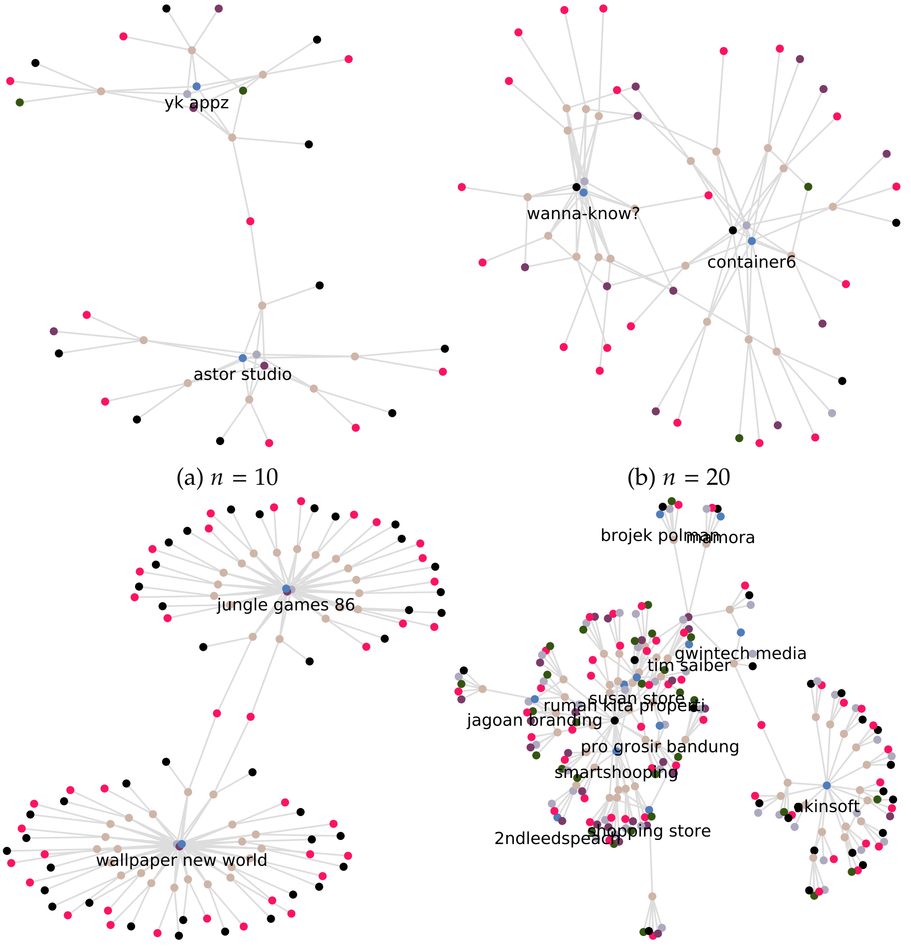


Figure 8.B.3: Percentage of unique package names published on both market$_x$ and market$_y$ under the same developer name.



(a) n = 10

(b) n = 20

(c) n = 40

(d) n = 50

Figure 8.C.1: Attribution graphs on Google Play for clusters of different package name counts (as denoted by n) annotated with developer names (● = market entry, ● = developer name, ● = developer website, ● = developer email, ● = privacy policy URL, ● = app name, ● = certificate).

Table 8.D.1: Unique signal usage by app signed by the most popular app building frameworks.

App building framework	**# package name**	**# developer names**	**# developer emails**	**# developer websites**
Andromo [E103]	17,721	1,602	1,908	1,030
Qbiki/Seattle Cloud [E108]	7,579	961	1,142	713
Bizness Apps [E116]	3,052	587	1,231	1,431
Mobincube [E117]	1,283	274	379	328
AppyPie [E104]	938	561	638	500
Como/Swiftic [E118]	528	319	382	445
QuickApp Ninja [E109]	512	174	185	99
DoubleDutch	356	210	121	96
AppMachine [E110]	216	174	168	148

with other apps or developer profiles. In doing so, we also shed light on the fidelity of the attribution signals available in the market.

In each experiment, we compiled and attempted to publish a repackaged app that includes partially fake developer information. To avoid breaking any intellectual property regulation, we chose two open source apps with licenses that allow their redistribution, namely *Open Sudoku*,[3] an app with a moderate amount of downloads (10,000+), and the *Signal Private Messenger*,[4] a popular privacy-focused messenger app, with more than 50 million downloads as of May 2021. Specifically, we evaluated whether the Google Play Store: *(i)* conducts any source code or image similarity analysis between new and existing apps; *(ii)* evaluates the similarities between market signals; and *(iii)* carries out a more exhaustive vetting if the affected app has a high profile (i.e., a larger number of downloads). To prevent apps from being rejected based on the developer reputation, we released each app under a new developer account. We discuss the ethical implications of this experiment in Appendix 8.G.

Code and image similarity verification. We posit that performing any code similarity for every new app submission might be too computationally expen-

[3] Open Sudoku in the Play Store: `https://play.google.com/store/apps/details?id=org.moire.opensudoku`, Developer website: `https://opensudoku.moire.org/`

[4] Signal in the Play Store: `https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms`, Developer website: `https://signal.org/`

Table 8.E.1: Unique signal usage by large organizations. The last column lists the developer names signing apps with the same certificates as the organization.

Developer name	# app names	# developer emails	# developer websites	# signing certificate	Other developer names
Google LLC	114	38	89	68	The Infatuation Inc.
Facebook Inc.	16	6	13	7	—
Microsoft Corporation	30	26	23	12	—
Samsung Electronics Co., Ltd.	49	18	24	22	Sophos Limited; Communitake Technologies Ltd.; Barco Limited (Awind); Nsl Utils; Teamviewer; Sidi; Ubridge; Rsupport Co., Ltd.; Koino; Maas360; Bomgar Corporation; Logmein, Inc.
Twitter Inc.	2	2	2	2	—
Snap Inc.	1	1	1	1	—
Tiktok Inc.	2	1	1	1	—
Netflix, inc.	5	5	3	4	—
WhatsApp Inc.	3	3	2	1	—
HP Inc.	21	14	20	21	Hewlett Packard Enterprise Company; Hp Inc; Printeron Inc
Instagram	5	2	2	1	—
King	16	15	14	6	—
Skype	1	1	1	1	—

sive given the scale of the market, even excluding different app versions. To test if the Google Play Store performs such an analysis, we released a clone of the Open Sudoku app, with all the market signals completely modified. This app was accepted for publication, which suggests that the Play Store vetting process does not perform any source code comparison. We left the app online for around three months before removing it ourselves.

Market signals similarity verification. The Play Store's Developer Program Policy specifically prohibits impersonation [E120] and publishing an app with highly similar market signals should therefore be prohibited. We released another clone of Open Sudoku, this time with signals that were equal to the original apps where possible, or highly similar where not possible (i.e., the package name and the developer name). We used `org.moire.attribution.opensudoku` instead of `org.moire.opensudoku` as its package name, and *Oscar García Amor* as the developer name instead of the original *Óscar García Amor*. The Play Store accepted the app, which shows that impersonation is still possible despite the official policy (we left the app on the market for three months before removing it ourselves).

Protection of high-profile apps. We hypothesize that the Play Store prioritizes the protection of popular or high-profile apps. To test this hypothesis, we release a clone of Signal Private Messenger, with all the attribution signals resembling the original app as closely as possible. The legitimate app has more than 50 million downloads as of May 2021. The Play Store rejected our first clone of the app for violating the impersonation policy, specifically referring to the title of the app (and in a subsequent submission, the icon of the app). Compared to the previous experiment, these rejections suggest that the Play Store is indeed prioritizing selected, high-profile apps. At this point, the app seemed to have become part of a manual vetting process, as a consecutive rejection based on violating in-app payment policies was accompanied with a set of screenshots to a PayPal donation link, which under our assumption would unlikely to be done by an automatic process.

Takeaways. Our results indicate that the mechanisms currently in place to enforce publication policies in the Google Play Store are imperfect. As a consequence, market and app signals that are self-reported by developers might be unreliable, especially for non-popular apps, and therefore lead to conflicting attribution results.

8.G Ethical considerations

As part of this study, we conducted an experiment in which we released legitimate clones of two open source apps on the Google Play Store. We took several steps to ensure our experiments would cause no harm to users nor to the platform itself. We received approval from our institutional ethical review board before conducting this experiment.

Both apps that we released display a notification to the users, clearly stating that these apps are part of a research project and they are not the original app, rather a clone. Additionally, this notification redirected them to the original apps and advised them to uninstall our clone of the app. Besides, we do not collect any information about the users that have install our apps. Finally, we removed the apps from the Play Store as soon as the experiment was over to prevent any confusion. Our experiment might have affected the platform, though we considered it to be negligible. Since all apps go through an automatic vetting process when published and with an estimation of around 100k apps [E121] published monthly, the overhead of two additional apps is minimal. Furthermore, there are over 3 million apps [E122] publicly available on the Google Play Store, consequently two apps will not affect any potential measurement or research conducted on the whole spectrum of the Google Play Store in any meaningful way. We estimate the benefits of understanding the vetting process of the Google Play Store to be far greater than the potential overhead of publicly releasing two apps into the Google market.

Chapter 9

Haaukins: A Highly Accessible and Automated Virtualization Platform for Security Education

Paper F

Thomas Kobber Panum, Kaspar Hageman, Jens Myrup Pedersen, René Rydhof Hansen

The paper was published in the

Proceedings of the 19th IEEE International Conference on Advanced Learning Technologies (ICALT)

Abstract

Education of IT security can include a tedious and frustrating experience for novice students and organizers. We have sought out to create an education platform that improves upon this experience, through automation, and individualized learning labs. These learning labs hosts are isolated clusters of virtual computer instances, representing real and insecure computer networks. The platform, named Haaukins, improves upon typical accessibility issues of students and cumbersome configuration management for organizers. In order make the platform accessible for other organizations, it has been open sourced.

9.1 Introduction

On a global scale, there is an increasing demand for qualified personnel for information security related positions. The (ISC)^2 estimates the shortage of security professionals to be 2.93 million persons [F1]. The Danish government has recognized this gap and is increasing its funding of cyber security education and research in response [F2]. In parallel, Aalborg University (Denmark) has started to address this issue by adding cyber security to the curriculum of relevant engineering educations.

This effort is accompanied by the university's ambassador program, which engages university students to reach out and teach high school students about their respective field of education. For cyber security, this involves running introductory workshops with subjects such as vulnerability scanning and exploitation. The sessions are typically short, between one and two hours of duration, and are initiated by a short lecture followed by exercises where participants interact with a computer system, referred to as a *lab*. As an organizer, managing these labs is time-consuming and error-prone, especially as participants often need assistance, e.g., to gain access to the lab. Our collective experience from hosting sessions with existing solutions led to a series of requirements for an education platform.

In order to have a solution that conforms to these requirements, we have developed a new virtualization platform, Haaukins, which grew from the collaboration project 'Danish Cyber Security Clusters'. The platform differs from existing work by automating the tedious management of the labs, while also providing an individualized experience to improve learning and making labs accessible with no configuration. It is designed to support the common exercise format, Jeopardy capture-the-flag (CTF) [F3], in which participants must gain access to certain secret information contained within a given computer system.

9.2 Related Work

Although there exist numerous platforms for deploying labs of connected, virtual instances, none of these fulfill the requirements of our specific use case in its entirety. The popularity of CTFs as a competition format has resulted in a range of commercial, closed platforms to support running such events [F4, F5]. The fact that these platforms are not open makes them unsuitable for an educational use, since the educator is completely reliant on the owner in both the access to the platform and the material hosted on them.

A range of existing platforms places various teams in a single, shared virtual network. The motivations for doing so range from a performance consideration [F6] to the desire to host attack and defense CTFs (ADCTFs) [F7, F8]. In ADCTFs, participants not only attempt to hack other machines, but actively have to protect their own against others. Given the limited prior knowledge of our target audience, it is infeasible for us to host ADCTFs.

PicoCTF is organized yearly and — similarly to Haaukins — aims to create an interest in information security among high school students [F9]. The platform is developed as an interactive game, and thereby does not reflect a realistic scenario.

In summary, the related work is either closed, not suitable for our CTF format or does not represent the real world in a realistic fashion.

9.3 Design Goals

The design of Haaukins has primarily been driven by four design goals: automation, transparency, accessibility, and realism. In the following subsections, we describe each design goal in some detail.

Fully Automated (DG$_{FA}$) In the process of preparing a CTF, numerous components need to be instantiated, configured and connected correctly. Manual configuration of labs is time-consuming process and errors can have severe consequences, e.g., a virtual instance being completely unreachable from one wrongly assigned IP address. Haaukins reduces the preparation time for events through full automation of the configuration of its labs.

Transparent (DG$_{T}$) Discovering and identifying a security vulnerability requires investigation and exploration. This element of exploration must be

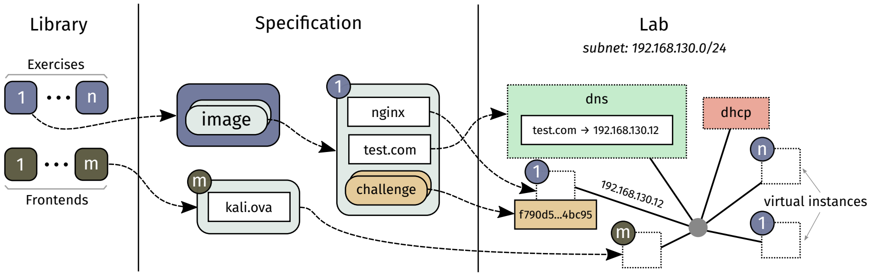


Figure 1: Levels of abstraction of a lab. Based on a selection of exercises and frontends (*left*), Haaukins uses their specification (*middle*) to create an individualized lab of instances per team (*right*).

contained in Haaukins, without causing participants to end up in dead ends that can hinder their learning [F10, F11]. If the design of the exercises is the cause of these dead ends, it has be identifiable. Ideally a platform allows for gaining insight into this, by providing a method for observing participants' behavior and make them available for further analysis.

Highly Accessible (DG$_{HA}$) In the setting of a one- or two hour lecture, time is a valuable asset that should not be wasted on irrelevant aspects. Given the short time span, the overhead of accessing a virtual lab can take up a significant fraction of the allocated time. Beginners might find this non-trivial and it may be a hurdle for progressing. We strive to minimize the overhead, and ideally want participants to access their virtual lab in a matter of minutes, independently from their physical location and the operating system they use.

Realistic (DG$_{R}$) For skills learned through a simulated environment to be valuable, they must be transferable to the real-world setting. Haaukins strives to do so by ensuring that the designed labs are realistic replicas of real vulnerable computers and their networks. The labs are interacted with using a professional toolkit, that continuously evolves to remain relevant for trends in security vulnerabilities. Experiences gained from the labs should be indistinguishable from real-world settings, and exercise developers should not be restricted by the limitations of the platform.

]

9.4 Overview

A key feature of Haaukins is the option for multiple organizers to host simultaneous sessions (referred to as EVENTS) with one or more exercises for participants. For an event, a group of participants registers as a TEAM which is assigned an environment upon registration. This environment is referred to as a LAB and is represented by a network of virtual INSTANCES. Teams are tasked to discover unique identifiers (or FLAGS), that function as proof for solving a CHALLENGE, which can be checked through a web application, and upon being valid, are counted as positive scores for the team.

A new event starts with any composition of EXERCISES and FRONTENDS (see Figure 1). A frontend is an instance that a team gets to control via a graphical user interface. An exercise is composed of any number of IMAGES, which are templates from which an instance is created. The specification of an image consists of: a virtual disk image, any number of records, and any number of challenges. The virtual disk image can either be a Docker image or an Open Virtual Appliance (OVA) package, e.g. *nginx* or *kali.ova* in Figure 1. The records are DNS records, which map domain names to IP addresses per lab, and are necessary for the communication among instances.

The instantiation of a lab consists of automatic creation and configuration of instances based on the collective specification of exercises and frontends. In addition to the associated instances, every lab also contains a set of core services to support connectivity and service discovery for instances, these services are DHCP and DNS respectively. The instantiation process also involves inserting unique flags in instances and randomizing IP address ranges, thereby individualizing each lab and its challenges.

9.5 Design

Haaukins consists of a client and a server component, hkn and hknd respectively, enabling multiple organizers to interact with the same instance of hknd independently of each other.

The daemon process, hknd, controls the life cycle of internal data structures and the orchestration of all components; it further serves as an application wide reverse HTTP proxy acting as a single point of entry for all the web traffic that comes in from the participants of the platform, and redirects the traffic to the correct virtual instances. On an event-level, there are two third-party components being managed: CTFd [F12] and Guacamole [F13]. CTFd is a web application responsible for the graphical user interface for the participants, which allows them to access their respective event through a web browser. Through this interface, the teams can view their respective exercises, fill in the results for their respective challenges, and are directed to their respective frontends which are accessible through Guacamole. Guacamole is a web application which allows for streaming remote desktops to a web browser and is more thoroughly covered in Section 9.5.

The client, hkn, provides a command-line interface (CLI) that allows organizers to interact with hknd, e.g., to create events, listing exercises or resetting instances for certain teams.

Automated Orchestration Haaukins uses Docker containers and Oracle VirtualBox (VirtualBox) virtual machines for deploying multiple isolated services within labs. These technologies are both used to allocate and isolate the resources (e.g., CPU and RAM) of a physical machine into virtual instances, but their ability to do so differs in terms of computation overhead and level of isolation.

All instances in a lab are connected to the same virtual network, to ensure that all instances can communicate among each other. Concretely, Docker Macvlan is used for the networks, resulting in a LAN network topology that allows for promiscuous network monitoring and gives participants the ability to observe the entire network traffic. External network access, such as the Internet, has been disabled to prevent participants from abusing the instances.

Lab Individualization In highly competitive CTFs, the sought-after flags are identical across participating teams, since the competitive nature is an incentive for keeping found flags private. From our experience this incentive does not transfer to an educational setting, as students are more inclined to share solutions, causing cheating that negatively influences DG_T through false progress. To combat this issue, Haaukins individualizes each lab through two techniques: dynamic flags and dynamic subnets. Creating dynamic flags is the process of creating unique flags on a per team basis in dynamic exercises, and thereby prevent the sharing of flags. Implementing dynamic subnets involves hosting labs on networks with randomized private IP ranges, which is a significant mutation in the setting of network analysis exercises. This prevents students from sharing information about IP addresses of instances, with only a few exceptions of core services, i.e., DNS and DHCPD.

Accessible Through a Web Browser Based on our experience, novice participants often encounter problems with the use of a (desktop-)client for remote access, such as VPN or RDP. These problems led to the choice of using Apache

Guacamole within Haaukins, which is a web application that allows for accessing remote desktop protocols, e.g., RDP, through a modern web browser. Guacamole uses a backend daemon for translating standardized protocols to WebSocket traffic that is interpretable by a JavaScript client in the user's browser. Within Haaukins, the built-in capabilities of VirtualBox is utilized for creating RDP access to frontends. This access is then translated by Guacamole in order to be accessed from the participant's browser, requiring no installation process and being accessible from any physical location.

Monitoring Participants The design of effective exercises is complicated, as the exercises need to be open-ended in order to adhere to DG_R, while ensuring the openness does not cause participants to become stuck. In Haaukins, participants are initially tasked with identifying vulnerabilities, before actively exploiting them. If the participants become stuck at this stage, it will hinder their ability to learn from the exercises. To determine when and how this behavior occurs, Haaukins has the ability to monitor and log participant interaction with the platform. Only if consent is granted, the WebSocket traffic of Guacamole is captured using the reverse proxy of `hknd` and transformed to a stream of key presses. These streams can then be further analyzed in order to determine participant behavior, e.g., programming activities and terminal usage, that can potentially influence changes to the teaching material.

9.6 Conclusions

We present a novel education platform, Haaukins, that differentiates itself from existing CTF platforms by having improved accessibility, full automation, observability of participant behavior and high degree of realism. The platform presents itself as a web application that provides highly accessible lab environments for participants, accessible within minutes without prior experience. It completely automates the creation, configuration, teardown of all its components. Each lab is personalized through unique mutations of flags and IP addresses, which discourages cheating among participants. Since the labs in Haaukins are designed to be a realistic representation of a realistic network, learnings from the platform translate directly to real-world scenarios. In order to support other education institutions in conducting short CTF workshops, the platform is available as an open source project on GitHub[1] with a GNU GPLv3 license.

[1] https://github.com/aau-network-security/haaukins

References

[F1] (ISC)^2, "Cybersecurity Workforce Study." https://www.isc2.org/research/workforce-study.

[F2] Danish Ministry of Finance, "Danish Cyber and Information Security Strategy 2018-2021." https://en.digst.dk/policy-and-strategy/danish-cyber-and-information-security-strategy/.

[F3] T. Chothia and C. Novakovic, "An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education," in *2015 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, 2015.

[F4] Hacking-Lab, "Hacking-lab." https://www.hacking-lab.com/.

[F5] Hack The Box Ltd, "Hackthebox.eu." https://www.hackthebox.eu/. Accessed: 24-09-2021.

[F6] L. McDaniel, E. Talvi, and B. Hay, "Capture the Flag as Cyber Security Introduction," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pp. 5479–5486, IEEE, 2016.

[F7] G. Vigna, K. Borgolte, J. Corbetta, A. Doupe, Y. Fratantonio, L. Invernizzi, D. Kirat, and Y. Shoshitaishvili, "Ten Years of iCTF: The Good, The Bad, and The Ugly," *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, pp. 1–7, 2014.

[F8] J. Mirkovic and P. A. H. Peterson, "Class Capture-the-Flag Exercises," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.

[F9] P. Chapman, J. Burket, and D. Brumley, "PicoCTF: A game-based computer security competition for high school students," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, (San Diego, CA), USENIX Association, Aug. 2014.

[F10] W.-C. Feng, "A Scaffolded, Metamorphic CTF for Reverse Engineering," *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, 2015.

[F11] K. Chung and J. Cohen, "Learning Obstacles in the Capture The Flag Model," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.

[F12] CTFd Maintainers, “Ctfd.” `https://github.com/CTFd/CTFd`.

[F13] The Apache Software Foundation, “Apache guacamole.” `https://guacamole.apache.org`.

Chapter 10

Bridging the Gap: Adapting a Security Education Platform to a New Audience

Paper G

Gian Marco Mennecozzi, Kaspar Hageman, Thomas Kobber Panum, Ahmet Türkmen, Rasmi-Vlad Mahmoud, Jens Myrup Pedersen

The paper was published in the
Proceedings of the IEEE Global Engineering Education Conference (EDUCON 2021)

Abstract

The current supply of a highly specialized cyber security professionals cannot meet the demands for societies seeking digitization. To close the skill gap, there is a need for introducing students in higher education to cyber security, and to combine theoretical knowledge with practical skills. This paper presents how the cyber security training platform Haaukins, initially developed to increase interest and knowledge of cyber security among high school students, was further developed to support the need for training in higher education. Based on the differences between the existing and new target audiences, a set of design principles were derived which shaped the technical adjustments required to provide a suitable platform - mainly related to dynamic tooling, centralized access to exercises, and scalability of the platform to support courses running over longer periods of time. The implementation of these adjustments has led to a series of teaching sessions in various institutions of higher education, demonstrating the viability for Haaukins for the new target audience.

10.1 Introduction

The demand for cyber security professionals has been increasing during the past years and is expected to increase even more in the future [G1]. To meet this increased demand, various higher education institutions globally have started to develop curricula specifically to educate engineers in cyber security, while others have integrated cyber security into their existing teaching curricula [G2, G3]. Practical exercises are an essential part in such curricula, and therefore several educational teaching platforms have been designed over the past years [G4–G7] to help out the learning process of students and encourage them to pursue a career in information security.

These platforms provide an environment that can be used for teaching and training purposes in cyber security courses by simulating real vulnerable systems and supporting complex cyber scenarios, as well as training users in monitoring and defending cyber infrastructure against malicious activities. This allows students to execute attacks on these systems without harming actual systems, and to obtain both an offensive and defensive perspective of security practices.

Most of these platforms are designed to support a common exercise format, Jeopardy Capture-The-Flag (CTF) [G8] which nowadays is a well known format to demonstrate the user skills in solving cyber security tasks and problems. In this format, the participants are tasked to find *flags* (i.e. a hidden string of text) by successfully exploiting vulnerable computer systems, being awarded points for each flag they discover. CTF competition brings several advantages when applied in teaching environments [G6]. For example, these types of competitions allow students to legally hack systems in a safe environment, by identifying vulnerabilities and trying to compromise them, while also allowing them to learn to defend against attacks [G9]. CTFs have also been shown to be effective in keeping the students engaged by their hands-on nature and through the entertaining experience [G10]. When involved in a CTF, being able to work as a group in a team is important for students to achieve their goals, leading the students to improve their communication skills, and to share, compare and broaden their knowledge [G11]. Moreover, challenge-based learning stimulates the development of problem solving skills leading the students to be involved in finding better solutions [G12].

Although these existing educational teaching platforms contribute to the learning process, none of them automate the process of creating custom vulnerable environments for teaching classes for high school students. This led Aalborg University, Denmark, to develop the first version of Haaukins [G13], an educational tool used for conducting short training events at high schools. These events, usually running between two hours up to a few days, were intended to engage students without prior information security knowledge, in a new topic, and to generate interest in a future career into the security field.

Since its launch, Haaukins has proven to be successful in high schools and attracted attention from other Danish institutions within higher education. In order to increase the usage of the platform, Aalborg University together with other educational institutions has started to adapt Haaukins from both a technical and an education perspective to accommodate IT and engineering students within higher education. The main contributions of this paper are as follows:

- Define a formalization of the differences in teaching environment, across high school and higher levels of education, for cyber security education.
- Present a set of design goals, based on the formalization, that the Haaukins platform is required to adopt to address these differences.
- Develop a solution for these design goals, that have been integrated into the existing open source platform.

The rest of this paper is organized as follows. A background of the initial design of Haaukins is presented in Section 10.2, followed by a comparison with the new target audience in Section 10.3. In order to adapt the platform to the new target audience, a list of design principles is presented in Section 10.4, which is followed by the fulfilment of those principles in Section 10.5. The platform deployment is presented in Section 10.6, followed by the conclusion in Section 10.7.

10.2 Background

Besides existing educational platforms there are a variety of commercial platforms where it is possible to practice cyber security in a CTF format, including 'Hack the Box' [G14] and the more recently developed 'Try Hack Me' [G15]. Whereas 'Hack the Box' provides challenging scenarios fitting for a more experienced target audience, 'Try Hack Me' provides several learning paths suitable for introducing beginners to the basics of security. However, such commercial platforms generally have no option to expand upon the training material and scenarios provided (such as adding more vulnerable machines), which is a severe restriction when teaching courses according to specific learning goals and objectives. An educational institution must have the opportunity to tailor the teaching material to their curriculum, and as such cannot rely on closed platforms.

'PicoCTF' [G16] is a platform developed by Carnegie Mellon University that achieved success during the past years. In order to encourage cyber security interest among high school students, it provided an interactive game and a terminal user interface used to interact with the exercises. Similar to Haaukins, it is open source and has it has been created for high school students. However, in contrast to Haaukins, it constraints itself to exercise types which students can do on any computer systems, and without relying on professional tooling.

Setting up an environment composed of computer systems, vulnerable hosts, and connections between them is time-consuming and errors can be hard to handle. Haaukins aims to facilitate the learning process by helping teachers to automate the tedious setup and management of those environments, by making them accessible with no prior, complex configuration. This allows students to have their own virtual and isolated environment to practice cyber security skills, while having the convenience of accessing it from their own devices simply through any web browser. High school students who, due to their limited amount of experience with cyber security and a limited understanding of underlying computer science topics in general, need an automated and highly accessible way to access those environments. Moreover, in order to support other educational institutions in conducting short CTF workshops, Haaukins has been made available as an open-source project on GitHub[1] with a GNU GPLv3 license.

[1] https://github.com/aau-network-security/haaukins

Haaukins is implemented as a Jeopardy-style CTF platform, in which the teacher can create an event and students access their own individual environments, referred to as *lab*s, created within the scope of that event. Each virtual *lab*, accessible through a student's laptop, consists of a computer network that has multiple computer systems, which are under control of the student or student group. Within this *lab*, students have to work towards solving a set of *exercises*, where each of them is related to a specific concept within information security. Most of the exercises are designed with specific learning objectives in mind. Exercises are solved by exploiting vulnerabilities of computer systems in the virtual labs; vulnerabilities that are intentionally embedded in the computer systems by the exercise developer. In order to exploit these vulnerabilities, the students must understand the underlying problem with the software, which makes these exercises a valuable teaching tool. Each event can be composed of any set of exercises that the teacher wishes to use in that session. Figure 1 illustrates the interaction between teacher, students and Haaukins itself. A website, which is automatically generated per event, provides both teacher and students with information about the event and its exercises.

The high school target audience required four main design goals to be fulfilled, that shaped the development of Haaukins in the very early stage of its existence. Those design goals are described in [G13] and are summarized as follow:

Fully Automated Haaukins was designed to automate the lab configuration process completely, and to do so in a relatively short time, making the preparation of an event painless.

The automation process will start and connect the required instances and components in order to have an environment available for the learning aspects. This process eliminates the need for manual configuration and provide error handling of labs for the teachers.

Transparent As students are more inclined to share solutions than in a competition setup, potentially leading to cheating that negatively influences their learning experience, a unique way of creating labs has been provided in Haaukins. For each lab, the flags to be found are unique, reducing the possibility of sharing flags. In addition, the platform monitors all actions that students make within it, allowing for the analysis of this afterwards.

Highly Accessible Given the short nature of the training events at high schools, it was considered imperative that students spend as little effort as possible on establishing access to their labs. In this case, Haaukins allows participants to access their virtual lab (1) quickly (i.e. in a matter of minutes), (2) without prior knowledge of accessing a virtual platform and (3) independently of their physical location and the operating system they use (i.e. access from anywhere via their Internet browser).

Realistic In order to teach students skills that are usable outside of the classroom, the platform was required to reflect a real world setting. The existing pool of exercises was thus designed to replicate real-life situations, and the platform had to support these types of exercises. In Haaukins all the exercises created are from real-life situations and interaction's realism is kept in order to ensure that students are gaining useful skills. Additionally, the taught techniques and available tool kit are identical to those used by security professionals.

10.3 Target audience

During the last year, the usage of Haaukins increased because of interest from new educational institutions. From the feedback received after every event, we have realized that the platform was able to satisfy the design goals allowing teachers to successfully conduct cyber security courses. The increase in usage has brought more visibility to the platform, thus leading to new expectations and new plans for its continuous development. Haaukins, could potentially be used in higher education as well, however due to the different target audience, it would need several improvements in order to make the new users comfortable with the platform.

This section seeks to clarify details of the new target audience, and their distinct traits, when compared to the traditional target audience for Haaukins. As previously stated, the audience intending to use the adjusted platform is *students in higher education*. To further clarify the expected skill set of this group, we define it as students who have attended at least one semester in computer science, computer engineering or comparable educations at least at undergraduate level. Concretely, students within this group should have a novice level of understanding of fundamental computer science topics, such as: (1) networking, (2) operating systems, and (3) programming. This thereby puts this group of students, in terms of competences, ahead of the original target audience of Haaukins (i.e. high school students).

Consequently, the established experience of the new target audience is reflected in their ability to interact with and understand computer systems.

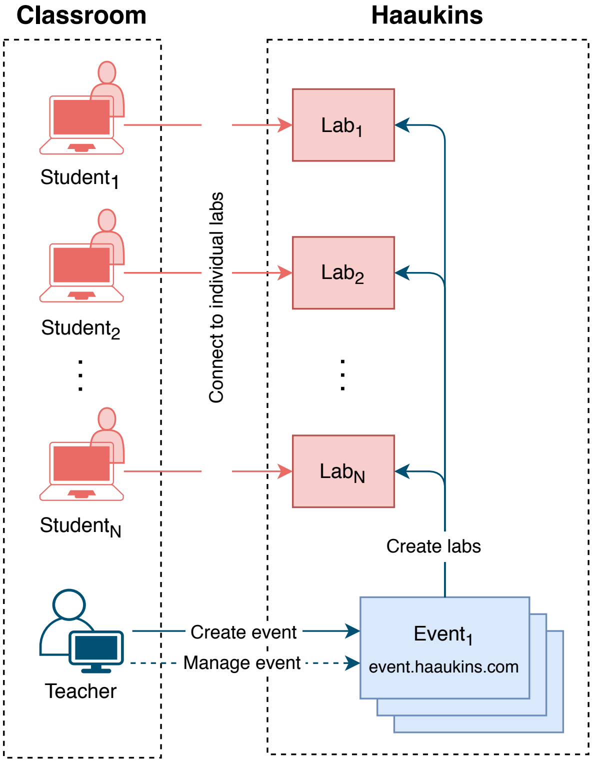


Figure 1: The interaction between a teacher, a classroom full of students and the Haaukins components in the first version of the platform

Trait 1. Higher education students are expected to have preliminary knowledge of various tools, selected to their preference, to diagnose and interact with computer systems and computer networks.

This is an important trait that differs from the original high school setting, where students are expected to have no or very limited technical understanding. This additional knowledge is also reflected in the type of teaching material to be used for the new target audience.

Trait 2. With a deeper technical knowledge, students of higher education are able to work on more complex exercises that involve understanding and attacking highly-interconnected and complex multi-computer systems.

From a teacher perspective, designing such exercises can be challenging and time consuming, as ensuring the interconnection of a multitude of computers correctly is a complex task. These exercises typically encapsulate some intended-by-design vulnerabilities, to be found by the students, and the discovery of the given vulnerabilities serve as an objective of the exercise. Students are typically expected to interact with the computers of the systems, of the exercise, in an offensive and destructive manner. Therefore, exercise designers must avoid unexpected vulnerabilities in the systems to be attacked, since such unexpected vulnerabilities can potentially halt the completion of the intended exercise. From a course point of view, the duration itself is often longer than the high school events, and a class might run just once per week through a semester. As a consequence of this, reliability and availability must be provided even for such long-term events, so the users of the platform do not experience interruptions and resets of events, users or exercise progress.

Trait 3. Given the multitude of attack vectors that a computer system can have, with the typical length and size of a university course, keeping each students lab in a healthy state (ability to complete exercises) is challenging.

The identified and distinct traits of the new target audience capture the challenges sought out to be addressed by technical solutions that can be integrated into the existing Haaukins platform, thus expanding its application domain. It is important that the application domain is expanded rather than changed, and that Haaukins still fulfill the requirements for the original target audience. The improved platform thus also appeals to the groups in-between, i.e. students in higher education without the competences listed above.

10.4 Design principles

The three traits identified in Section 10.3, serve as the problems which have to be addressed in order for Haaukins to viable for the new teaching context. Prior to the technical design, we seek to put forward three design principles that will drive the changes to the existing platform. These design principles have to coexist with existing ones (covered in Section 10.2), thus their compatibility to the existing ones is discussed.

Haaukins provides rapid access to students' labs remotely (through a web browser) to a computer with pre-installed tools, serving as a fixed toolbox, used for accessing the lab environment and interacting with the exercises. This design was implemented in relation to the existing design principle of being highly accessible. However, the design of having a fixed toolbox (static tooling) could potentially violate Trait 1, as the desired set of tools might not match the provided set of tools. In order to address this problem, and in contrast to the existing static tooling currently provided, we present the following design principle:

Design Principle 1. *[Dynamic Tooling] Students should be able to use their desired set of tools within their lab of exercises.*

Having dynamic tooling could potentially be an undesired feature for the original teaching context (high schools), which includes more novice students that have fewer preliminary competences in the field. Thereby, it is important to ensure that this principle serves as an alternative, such that it can coexist with the current method of "static tooling". Knowing that the teaching context includes students with more preliminary knowledge (Trait 2), forces the exercises containing computer systems to become larger (groups of computers) and more inter-connected (network communication). These types of exercises are naturally costly to design due to their complexity, and the current architecture for Haaukins relies on teachers, on an individual level, to implementing these. Thereby, as a measure to reduce this cost across, we propose the following design principle that is based on sharing resources:

Design Principle 2. *[Centralization of Exercises] Complex exercises should be provided by a centralized source, such that teachers across various institutions can share implementations of exercises, thus enabling future teachers to benefit from existing work.*

Hosting and serving these exercises in individual labs for each student in a university class is a vastly different scale when compared to the high school setting (Trait 3). Moreover the platform has to support a higher volume of concurrent students coming from both the previous and new target audience. Scaling the platform to support a teaching context of a university class (more than 100 students) and an increased number of events running at the same time, is a two-fold challenge. Indeed, both the technical capabilities (efficiency, resources) and orchestration of labs (maintenance, restarts) are required to scale to this new volume of students. Moreover, teachers should have the possibility to monitor and manage student labs from a highly user friendly interface while the events are running.

Design Principle 3. *[Scalability] The platform should provide both computational efficiency (technical scaling) and orchestration features (teaching scaling) that ensure exercises within labs remain healthy and available while several events are running concurrently.*

10.5 Evolved Haaukins platform

The design principles defined in the previous section aim at improving Haaukins in order to make the new target audience comfortable in using the platform. The evolved platform should address the requirements of students in higher education without impacting the usability for high schools, which are still a core user group of Haaukins. This chapter presents a list of improvements implemented based on each of the principles defined in 10.4.

Dynamic tooling. In order to adhere to Design Principle 1, two alternatives were considered; either to integrate more tools in the lab or to provide an alternative access method to the lab. In the first approach, a more customizable lab environment can be provided to the users, e.g., by giving the teachers or students themselves the option to compose a toolkit from a curated list of tools and operating systems. This curated list would have to encompass all possible tools that students would like to use, which in practice would be difficult to accomplish. The second alternative instead relies on giving the students access to the lab through a different method than the browser-based method. Previously, students would remotely control a computer system prepared specifically for this, and these systems were identical across all labs in an event, leaving little room for customization. Instead, students could be given access to a *network connection* to the lab, thereby opening the option for students to connect their own computer systems to the labs, with their own custom tooling. In order to respect all previous and newly defined design principles, the platform must allow the user to still have a fully automated configuration process and a highly accessible way to the exercises.

From a technical point of view, the choice fell on the integration of a Virtual Private Network (VPN) [G17], and specifically Wireguard [G18], in which a secure connection to another network over the Internet is created, in our case to the *lab*. A VPN connection can be established from any operating system from any geographic location, merely requiring the students to configure their local Wireguard with a configuration file provided by Haaukins. Similar to the previous web-based access method, a VPN connection can generally be established from computer networks without requiring changes to the IT infrastructure. As a result, this solution does not violate the original highly-accessible design principle but contributes to Design Principle 1. Although configuring a VPN is considered a fairly simple setup step for an experienced student, this it not the case for the traditional target audience, and therefore Haaukins supports both (1) the web-based access method and (2) the VPN connection, and a teacher can make a per-event decision as to what access method suits the target audience best.

Centralization of exercises The first version of the platform has been developed to allow teachers to run events using either custom exercises or readily-available exercises provided by the platform itself through an exercise library. In the first case, teachers could create specific exercises for their courses and use them in their events in order to teach different topics not already provided by the platform. Although this feature brought more flexibility to the platform when referring to high schools, university teachers could not benefit much of it. This is largely due to the fact that different target audiences rely on different exercises that differ from each other in terms of both content and complexity (Trait 2). Exercises for higher education students are not only harder to solve compared to those for high school students, they are also more complex and take longer more time and efforts to create. These exercises, in fact, have to be created either focusing on a specific topic and go into details or have a broad approach where it covers several topics. In both cases, the teacher has to design and create several steps of difficulty in order to let students improve their skills and time spent on the exercise.

Creating an exercise for a course is time-consuming, especially for the more advanced exercises which are to be used within higher education, and therefore is not always an option. To support teachers, many exercises have been created and made available to the use in their events, and a clear workflow has been established for creating, testing, and including exercises for those who want to create their own. The main goal was to provide a centralized pool of exercises with different content and of different complexities where teachers can choose according to their courses (Design Principle 2). Each exercise is accompanied by a description, and a list of prerequisites and outcomes has been made in order to facilitate teachers in choosing a relevant composition of exercises for their courses. Finally, to facilitate an even better exercise selection phase, the exercises have been grouped based on different difficulty levels (e.g., the number of steps needed to solve it and the topic covered) and divided into different categories that cover different fields of cyber security (e.g., web exploitation, forensics, binary, reverse engineering and cryptography).

Scalability Haaukins must also support managing a higher number of events running at the same time. In order to provide a reliable and fault tolerance platform, it has to scale in two main directions (Design Principle 3) described as follows.

Teaching scaling: In order to maintain the labs healthy and available, a "reset functionality" has been created, available to both teachers and students in order to restart (i.e stop and start) *lab*s as well as individual exercises in case of crashes, which can happen if a student make mistakes when attempting to solve an exercise or when exercises are not properly developed with the destructive behaviour of the teaching context in mind (Trait 3). In such cases, one or more exercises in the *lab* are reverted to the initial state right after the lab was created. The students lose their progressions towards this specific exercise, but it allows them to experiment with potential destructive offensive techniques that will break the exercise. In fact, as exercises largely focus on breaking existing computer systems, those systems are brought to a high degree of stress. As such, platform allows a graceful recovery from errors, instead of burdening teachers with developing bulletproof exercises. This functionality has been made available on the event website for the students while for teachers it has been implemented in the *web client*.

In the previous version of Haaukins, the event creation and management controls were provided via a command-line interface (or *cli*), that had to be downloaded and installed on the computer of the teacher. This command-line program could be used to send some basic commands remotely to the physical server on which Haaukins was running, thereby managing events. Prior to be able to use this program, a teacher had to be granted access by another teacher as a security mechanism, which introduced another barrier for quickly setting up an event. Feedback from high school organizers showed that this approach was not user-friendly enough, and that it was too time-consuming to use.

In order to overcome this issue a different way to interact with the platform had to be provided, and a user interface *web client* connected to the platform was created. In detail, the *web client* is the web-application version of the *cli* which provides the same functionalities of the *cli* along with a number of new functionalities designed to improve scalability as well as the overall organization experience. With this solution it is no longer needed to download and install the *cli* on the teacher's computer machine. Teachers can access the *web client* upon request in order to create and manage their own events no matters where they are - simply by their web browser of choice. From an intuitive user interface, the teachers are able to choose the event configuration (e.g., event name, event capacity, exercises and VPN option) and check the status of the teams signed up in their events.

The *web client* is linked to the centralized exercises pool thus allowing teachers to insert new custom exercises and get all information about already existing and available exercises. This connection aims to facilitate the teacher in choosing the relevant exercises for his or her event.

Besides management of events and their respective *lab*s, Haaukins has the ability to monitor and log student's interactions with the platform, which enables the ability to identify if the participants become stuck while exploiting exercises. This functionality is only activated if consent is granted from individual students and is implemented by storing the stream of key presses to log files, that serve as the basis of the analysis of behavior.

Technical scaling: The new target audience will bring with it not just more events running simultaneously on the server, but also larger events due to the higher number of students for each course, thus leading to a higher computation load on the platform. A potential problem that might occur because of this higher demand, is that the platform might not have sufficient capacity to be able to manage all the events thus leading to the rejection of some of them. A main goal is therefore to provide both target audiences with a platform that is able to support all the requested events without affecting the performance of the platform or other events (Design Principle 3).

To meet this goal, two main approaches to make the platform more scalable have been evaluated, and both horizontal and vertical methods have been taken into consideration: the horizontal approach relies on replicating the platform on multiple servers, thus leading to a distributed version where events can run in different servers. The vertical approach instead consists of adding more resources (i.e. memory and hard disk) to the current server in order to make it more powerful. From our point of view both methods where suitable for the platform, the former being more expensive in terms of time due to the refactoring the code base of the platform but cheaper in terms of the monetary cost, while for the latter it is the other way around.

Also considering the possibility to make a platform cloud based, more effort has been made in making the platform available in a distributed way without affecting the usability for teachers and students. In this sense the platform has been split in microservices [G19] running on different servers, which also allows for an easier deployment to the cloud in the future [G20]. The vertical approach has been applied as well on the main server, where more resources have been installed.

From previous experience with high school events, it was found that labs in events that last longer than two weeks have a far lower resource utilization (i.e. the percentage of time that a lab is actively being used) than shorter events. In fact, some of the labs were not used for several full days before being used again afterwards, occupying resources of the host server and consequently potentially refraining other students from using the platform. As described in Trait 3, this scenario might occur more often, eventually denying requests for events due to the limited capacity of the server hosting the platform. Although the technical scaling improvements aim to provide the opportunity to everyone to use the platform, those long events might thus cause a problem. To overcome this issue, a 'sleep mode' feature has been developed, which automatically suspends *lab*s that have not been used for a while and resume them when the students log into the event again. This feature aims to save resources on the server - especially for the new usage - and thus boosts the scalability of the platform.

10.6 Deployment in higher education

Throughout the development of the evolved Haaukins platform, it has been used in various settings within higher education, and feedback has been collected as input to the development process. The usage includes courses within two universities and four university colleges, as well as larger events in the framework of higher education, such as summer schools and conferences. It was also used in university-facilitated events for IT professionals in companies including sectors such as finance, energy, IT and national authorities.

While different events and courses were organised differently, in general three steps were included: (1) In the preparation phase, the course or event was planned and set up. This includes choosing relevant exercises, determining whether VPN or web browser access should be used. In the beginning, this was in most cases done in close collaboration with the Haaukins developers, but as more teachers gained experience in using the platform and as the improvements described in this paper were developed - in particular the *web client* - this was increasingly done by the teachers independently. (2) In the next phase, the event/course was held. Unless any problems arose, this was usually done by the teachers. (3) Finally, in the evaluation phase, feedback was collected from the teachers and/or the students.

The feedback collected included the experience from both phase (1) and (2) together with general feedback and suggestions about the platform. This collection of feedback has served two purposes. One purpose was to use it for input to the overall platform design and development, where the resulting changes would be of a more fundamental character. Such changes would be incorporated to the overall development road map, which was discussed among partner institutions of higher education at regular meetings. Another purpose was to identify issues where smaller adjustments could improve the user experience. In many cases, these were straightforward to implement, e.g., better explanations of platform usage and exercises. By the time this paper was submitted, Haaukins has been used by more than 1.000 students from different target groups.

10.7 Conclusion and future work

In this work, we presented an evolved version of Haaukins, a cybersecurity training platform that facilitates the learning process by helping teachers in creating cybersecurity training scenarios in a secure, closed, and virtualized environment. Over the last years, the platform has been used in several high schools with consistently positive feedback and it was decided to improve the platform in a way that would support the usage in higher education.

The new target audience, compared to the previous one, has been identified as a more experienced student who is able to interact with computer systems and computer networks using various tools, and who is able to address more complex exercises. Due to those differences and the typical length and size of a university course, a list of design principles, which have to coexist with the existing ones, have been defined and afterwards shaped into technical improvements on the platform.

Examples of such improvements include that a VPN connection has been provided as an alternative way to connect the *lab*s, thus enabling the students to use their own tools. An exercises pool has been made available for teachers in order to let them benefit from already made exercises, thus avoiding the time invested in creating them. Finally, the platform has been made more scalable in order to handle a higher number of students and longer events running at the same time.

These collective changes made Haaukins a platform for both students of higher education and high school students, driven by an increased interest by schools in Denmark. The platform is currently being under further development to widen its appeal to even more target audiences and to be used on a larger scale, and additional research studies are also being undertaken: In order to obtain a better understanding of the challenges of the game based learning experience, we are planning to conduct user studies in the near future. Moreover, an investigation of which exercises should be developed to ensure a good progression in the learning path of different students will be carried out.

10.8 Acknowledgment

The authors would like to thank The Danish Industry Foundation for the continued support to the development of Haaukins.

References

[G1] S. Morgan, "Cybersecurity jobs report," 2017. https://www.herjavecgroup.com/wp-content/uploads/2018/07/HG-and-CV-The-Cybersecurity-Jobs-Report-2017.pdf.

[G2] E. C. 2018, "Resilience, deterrence and defence: Building strong cybersecurity in europe," 2018. https://ec.europa.eu/digital-single-market/en/news/resilience-deterrence-and-defence-building-strong-cybersecurity-europe.

[G3] A. University, "Cyber security, msc in engineering," 2020. https://www.en.aau.dk/education/master/cyber-security/.

[G4] P. Chapman, J. Burket, and D. Brumley, "PicoCTF: A game-based computer security competition for high school students," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, (San Diego, CA), USENIX Association, Aug. 2014.

[G5] E. Trickel, F. Disperati, E. Gustafson, F. Kalantari, M. Mabey, N. Tiwari, Y. Safaei, A. Doupé, and G. Vigna, "Shell we play a game? CTF-as-a-service for security education," in *2017 {USENIX} Workshop on Advances in Security Education ({ASE} 17)*, 2017.

[G6] A. Mansurov, "A CTF-based approach in information security education: an extracurricular activity in teaching students at altai state university, russia," *Modern Applied Science*, vol. 10, no. 11, p. 159, 2016.

[G7] L. Tobarra, A. P. Trapero, R. Pastor, A. Robles-Gómez, R. Hernández, A. Duque, and J. Cano, "Game-based learning approach to cybersecurity," in *2020 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1125–1132, IEEE, 2020.

[G8] T. Chothia and C. Novakovic, "An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education," in *2015 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, 2015.

[G9] S. Wi, J. Choi, and S. K. Cha, "Git-based {CTF}: A simple and effective approach to organizing in-course attack-and-defense security competition," in *2018 {USENIX} Workshop on Advances in Security Education ({ASE} 18)*, 2018.

[G10] M. Katsantonis, P. Fouliras, and I. Mavridis, "Conceptual analysis of cyber security education based on live competitions," in *2017 IEEE Global Engineering Education Conference (EDUCON)*, pp. 771–779, IEEE, 2017.

[G11] J. Werther, M. Zhivich, T. Leek, and N. Zeldovich, "Experiences in cyber security education: The MIT Lincoln laboratory capture-the-flag exercise.," in *CSET*, 2011.

[G12] R. S. Cheung, J. P. Cohen, H. Z. Lo, and F. Elia, "Challenge based learning in cybersecurity education," in *Proceedings of the International Conference on Security and Management (SAM)*, p. 1, The Steering Committee of The World Congress in Computer Science, Computer , 2011.

[G13] T. K. Panum, K. Hageman, J. M. Pedersen, and R. R. Hansen, "Haaukins: A highly accessible and automated virtualization platform for security education," in *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, vol. 2161, pp. 236–238, IEEE, 2019.

[G14] Hack The Box Ltd, "Hackthebox.eu." `https://www.hackthebox.eu/`. Accessed: 24-09-2021.

[G15] Try Hack Me - London, "Try Hack Me." `https://tryhackme.com/`.

[G16] Carnegie Mellon University, "picoCTF." `https://picoctf.org/`.

[G17] P. Ferguson and G. Huston, "What is a VPN?," 1998.

[G18] J. A. Donenfeld, "Wireguard: next generation kernel network tunnel," *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pp. 1–12, 2017.

[G19] N. Alshuqayran, N. Ali, and R. Evans, “A systematic mapping study in microservice architecture,” in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 44–51, IEEE, 2016.

[G20] V. Singh and S. K. Peddoju, “Container-based microservice architecture for cloud applications,” in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 847–852, 2017.

Part III

Epilogue

Chapter 11

Discussion

Throughout this dissertation, the individual results of the papers have been discussed. In this chapter, the results are discussed in the larger context of the dissertation and the security research field in general. More specific, it goes into details regarding shared limitations across the various papers (i.e., an ambiguous ground-truth and measurement bias) and the implications of future developments (i.e., encryption and transparency logs) on the Internet of this type of research.

11.1 The ambiguity of the ground-truth

Papers B and C investigated certificates related to phishing domains and worked towards distinguishing between these malicious certificates and benign certificates. Both papers acknowledge the challenges associated with identifying a valid ground-truth of certificates and the fact that we may have relied on an invalid and incomplete ground-truth in the papers. The data set that was the basis for both papers consisted of labeled domain names – phishing or benign – and the inferred labels for the certificates associated with these domains. In the former paper, the label of a certificate was inferred from a single domain name, whereas the latter paper relies on an aggregation of domain name labels. In both cases, a domain label was inferred from a blacklist of phishing URLs and a whitelist of benign domain names, as illustrated in Figure 1. The figure illustrates a 'many-to-one' mapping between certificates and apex domains, between apex domains and Fully Qualified Domain Names (FQDNs), and between FQDNs and URLS. In each of these steps, the labels of the many entities have been aggregated into a label of a single entity.

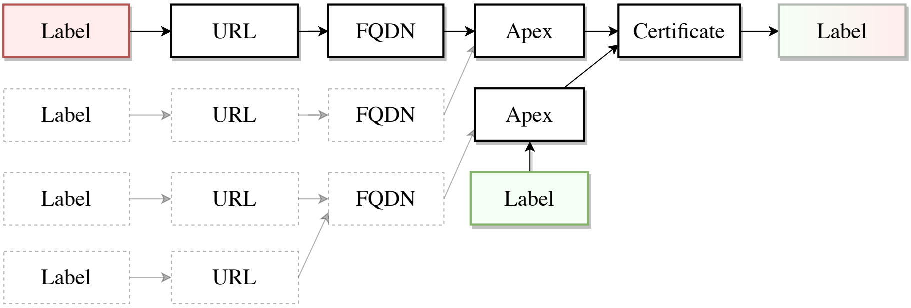


Figure 1: Inferring the label of a certificate based on the labels of URLs and apex domains

Paper C only addressed the 'many-to-one' mapping between certificates and apex domains, assigning a potential ambiguous label to certificates. Figure 1 shows that this might not be sufficient, as there exists a potential ambiguity in both FQDN and apex labels. Furthermore, the nature – and thereby the true label – of each entity can change over time, as demonstrated by domains that are repurposed for hosting malicious content after being compromised by malicious actors [62]. Even for labels that we assumed to be a ground-truth, there are counterexamples of entities related to this ground-truth that suggests another label. An example is the benign labeling of `facebook.com`, even though phishing attacks have been conducted on the site [63].

So how do we proceed with the limitations of labeling a certificate based on information we have about URLs and apex domains? Many commercially available phishing blacklists operate on a URL basis only, but other types of malicious activity, such as botnet communication, are heavily based around the domain registration process, and as such apex domain blacklists exist for this activity. Prior work has focused on identifying the differences between domains registered for phishing purposes and domains that were compromised and repurposed for phishing [62]. By focusing on the former class, we could focus on the domain registration process of phishing domains (on an apex level) alleviating the issues with URL-based labeling.

Android Attribution A different form of ambiguity was observed in Paper E. The results of this paper did not work towards app attribution directly; instead, it analyzed the effect of the Android ecosystem design on the ability of researchers and users to attribute apps to the underlying developer. This paper specifically focused on the responsible developer of the app, and not necessarily the developer that created the assets of the app. This distinction between a ‘publisher’ and ‘asset creator’ shows that authorship attribution can be considered an ambiguous problem; the publication of an app on an Android marketplace may have involved multiple different actors fulfilling different roles. Our results show clear examples of app builders where apps share the same *signer*, i.e., the actor that generates the digital signature but are published by different developer accounts, or conversely major app developers that rely on different signing keys to generate the signatures. Even though the identification of these roles fell outside the scope of this work, we believe that the research community should address this authorship ambiguity.

11.2 Data collection methods

The various papers in this dissertation relied on different data collection methods, ranging from active measurements through crawling app markets for their meta data and apps, to extracting DNS responses and flows from passively collected raw network traffic. Newly accepted, and newly proposed, Internet standards will affect the feasibility of these methods.

As touched upon in the individual papers, the encryption of different types of traffic will affect the ability to passively monitor traffic. Firstly, several DNS encryption proposals are receiving a steady adoption, such as DNS-over-HTTPS and DNS-over-TLS. Combined with TLS version 1.3 [64], in which certificates are encrypted as part of the handshake, and the proposed standard for encrypting more parts of the handshake [65], this will make it impossible for a passive observer to identify the destination domain name for traffic. Furthermore, it has become inexpensive for web sites to operate behind proxy servers, such as CloudFlare, which hides the IP address which hosts the actual content. Besides the general major adoption of network traffic encryption (up to 98% of pages loaded in Chrome are over HTTPS [66]), we see on an application-level that data become more encrypted. For instance, messaging apps such as Whatsapp and Signal employ end-to-end encryption which leads to the developers and operators of the underlying infrastructure being unable to decipher messages.

This does not necessarily mean that operators of corporate networks are doomed to manage networks based on completely opaque information. Firstly, deploying DoH or DoT-enabled resolvers provides network operators to monitor and enforce policies on DNS traffic. Furthermore, by enforcing user devices to deploy a self-signed root certificate, the network operators can proxy TLS traffic and inspect any TLS traffic passing through their firewalls. However, this applies to applications that adhere to these policies, which may be the case for major browsers, but not necessarily for malware. It would also break some existing techniques, such as certificate pinning and Certification Authority Authorization (CAA) DNS records. Also, in cases where the network operator does not control the user's devices – businesses with a bring-your-own-device policy for instance – enforcing the use of a TLS proxy is more difficult. Furthermore, the usage of this of deep packet inspection (DPI) raises privacy concerns of the users of such networks.

The massive adoption of the Certificate Transparency (CT) framework illustrates that standards can instead also lead to an increasingly transparent Internet. The major difference is that these aforementioned encryption standards preserve the privacy of end users, whereas the CT framework provides transparency for the services that are provided online by some of the crucial players: the Certificate Authorities. Perhaps we will see similar proposals for other major players, such as transparency for published apps on marketplaces such as the Google Playstore, or transparency of domain registrations from domain registries.

11.3 Measurement bias

Throughout this dissertation, we measured networks or otherwise collected data from different sources. In some of these cases, the coverage of this data collection resulted was limited: the university network's traffic (Papers A and D) covers a relatively small client population, traffic data was collected a small number of Android markets (Paper E), and DNS resolutions from ENTRADA cover only the Danish TLD (Paper A). As such, there is both a bias in these results towards a particular client population, top-level domain, and set of markets. Furthermore, data were collected only for a limited period.

The data that were collected for unnamed traffic represents a typical university network (albeit a fraction of this traffic due to the measurement of a VPN server only) and gives insight into how particular services are accessed (in terms of the 'namedness' of traffic). The results in this paper were aimed to illustrate the difficulties of relying on the identification of unnamed traffic for blocking malicious traffic, and for this goal, a relatively small network traffic trace fulfills the purpose. Collecting traffic for a longer time period, and from a different vantage point, would have been unlikely to lead us to draw a different conclusion; the 'unnamedness' of traffic to particular services and destination IP addresses are unlikely to differ across client populations (whether a person accesses Facebook from a Danish university network, or a corporate network in the US, the browser used in either situation will act the same way). As such, the measurement bias in Paper D is not a particular limitation from a service and destination IP address perspective. A different client population could perhaps reveal a vastly different perspective on the population's behavior. For instance, DNS over HTTPS is enabled by default for US-based Firefox users [67], resulting in a higher fraction of traffic being falsely labeled unnamed when measured from a US client population

Although data were collected from a large number of apps (including their .apk package and meta information) from various Android markets, the measurements were far from complete. Under ten markets were crawled – whereas there are dozens of markets in reality – and for those there were targeted, data were not exhaustively collected. However, Paper E successfully demonstrated some of the major issues with the Android ecosystem, and as such this limited data collection did not significantly impact our results. Throughout our measurements, it was noticed that several Android markets started to employ methods to make our crawling efforts more challenging, including enforcing stricter rate limits, anti-spamming web pages, and disabling the downloads of package files through a browser. Having a transparency log (similar to CT) for the Android ecosystem (as suggested in Section 11.2) would make the ecosystem more transparent for not only researchers but also regulators and users.

The DNS resolutions that we collected in Paper A, both from a local resolver and the authoritative name server, were heavily biased towards Danish traffic. By extending our measurements to a different, and larger, client population, and to authoritative name servers for general TLDs (such as .com and .net), different insights could have been provided. The .dk domain is relatively small and naturally tailored towards Danish users, whereas domain names under general TLDs – especially .com – are more international.

As a concluding remark to this chapter, the major limitation of this dissertation is the ground-truth on which Papers B and C are based (with respects to certificates and phishing URLs). Despite the limited data collection scope in this work, the bias in the data set is not considered to have significantly impacted the conclusions. Lastly, new standardization initiatives provide both less and more transparency, which both threatens and provides new opportunities for Internet traffic measurements.

Chapter 12

Conclusions

As stated earlier in Section 1.1, this dissertation was concerned with understanding and leveraging domain names and digital certificates in different security contexts. This dissertation has shown the clear relationship between domain names and certificates, and between certificates and Android apps. In the former, the ambiguous 'many-to-one' relationship between domain names and certificates was identified and a machine learning model for classifying the somewhat ambiguous certificates was produced. The grey 'phishy' nature of a digital certificate raises the question of how to interpret the results of this model and the results of prior research in the same research domain that did not acknowledge this greyness. The utility of this model is challenged by the existence of unnamed traffic, especially since new technologies are expected to amplify the volume of unnamed traffic in the future. For the latter, we show that certificates are a near meaningless signal for authorship attribution in Android. Hopefully, the results of this dissertation will lead to an actual change in the community, and lead to stronger enforcement of developer identity.

As a secondary contribution, *Haaukins* was released as an open-source platform, which at the time of writing of this dissertation has made quite a slash across Denmark: national events are being hosted using the platform, and the platform is still being funded for future development. The first iteration of Haaukins introduces the high-accessibility web-based user interfaced, which drew in inexperienced users but proved to be less popular among more experienced users. The introduction of a VPN solution met these demands and made the platform more suitable for "power users". This illustrates that the platform meets a previously unmet need and that our design goals and design decisions were the right ones.

12.1 Future work

This dissertation has given the research community both open source tools and insight for further research. The community is especially encouraged to pursue the following promising pathways:

- Adapting *Gollector* to, or collecting data from, more vantage points, thereby making it more suitable for the analysis of dark matter that were not addressed. Examples include Internet censorship and the evasion of DNS-based security mechanisms.

- Understanding the intent behind unnamed traffic better, potentially leading to DNS-aware firewalls that block unnamed traffic. This would either lead to dangerous unnamed traffic to be mitigated, or to funnel more DNS traffic to a DNS filter that blocks traffic to dangerous domain names, which could significantly improve the security of users inside a network.

- Establishing a better ground-truth for phishing domains, such that extracting a ground-truth for certificates is less prone to errors and ambiguity.

- Unraveling the different roles in the lifecycle of mobile apps to fully understand the different actors that produce signals of a published app. This would help researchers understand the confusing (lack of) overlap of signals between apps that were observed.

- The continuation of the development of *Haaukins* into in different directions to adapt the platform to a larger audience. The platform can be deployed to a cloud provider, making it easier to scale the platform to a larger user base. The nature of the teaching material the exploitation of a specific vulnerability does not allow the current exercises to be re-used by students. Developing a healthy ecosystem of developers for teaching material can be a promising next step, or alternatively introducing the automatic generation of dynamic challenges, such as [56], could be a potential source of new exercise material. This would retain users of the platform and keep them invested in the field of information security.

- Leveraging *Haaukins* for improving the approach to teaching information security. The platform is capable of monitoring the actions that participants of events take – ranging from the time when an exercise is solved to individual keypresses – and analyzing these actions into more detail can give us insight into what challenges are too complex or too simplistic.

- Incorporating *Haaukins* into non-security educations, such as general software or computer engineering programs. Combined with the previous point, the platform has the potential to make it more transparent to teachers how students approach certain learning tasks.

Generally speaking, the research community is asked to expand upon the software tools developed as part of this dissertation, to use this tooling to understand Internet phenomena even better, and to leverage this understanding for securing end users. This marks the end of this dissertation.

References

[1] O. Kulyk, J. Mauro, A. Dalela, S. Giallorenzo, B. H. Jakobsen, and E. Paja, "Assessment on the status of cybersecurity in Denmark," tech. rep., University of Southern Denmark, December 2020. `https://ascd.dk/results/report.pdf`.

[2] "State of software security: Volume 11," tech. rep., Veracode, 2020. `https://www.veracode.com/state-of-software-security-report`.

[3] "Ransomwhere." `https://ransomwhe.re/#report`. Accessed: 13-09-2021.

[4] F. Li, A. Lai, and D. Ddl, "Evidence of advanced persistent threat: A case study of malware for political espionage," in *2011 6th International Conference on Malicious and Unwanted Software*, pp. 102–109, IEEE, 2011.

[5] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[6] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[7] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda, "Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks," (Berlin, Heidelberg), pp. 88–106, Springer Berlin Heidelberg, 2009.

[8] E. E. Lastdrager, "Achieving a consensual definition of phishing based on a systematic review of the literature," *Crime Science*, vol. 3, Sept. 2014.

[9] "Dont be a phishing victim : Know your anti-phishing Chrome extension." `https://www.phishprotection.com/content/anti-phishing/anti-phishing-chrome-extension/`. Accessed: 27-09-2021.

[10] Y. Fang, C. Zhang, C. Huang, L. Liu, and Y. Yang, "Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism," *IEEE Access*, vol. 7, pp. 56329–56340, 2019.

[11] S. Smadi, N. Aslam, and L. Zhang, "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning," *Decision Support Systems*, vol. 107, pp. 88–102, 2018.

[12] Anti-Phishing Working Group, "Phishing Activity Trends Report - 1st Quarter 2021." `https://docs.apwg.org/reports/apwg_trends_report_q1_2021.pdf`, June 2021.

[13] Imperva, "Phishing made easy: Time to rethink your prevention strategy?." `https://www.imperva.com/docs/Imperva-HII-phishing-made-easy.pdf`, 2016.

[14] A. Oest, Y. Safei, A. Doupé, G.-J. Ahn, B. Wardman, and G. Warner, "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis," in *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pp. 1–12, IEEE, 2018.

[15] K. Jansson and R. Solms, "Phishing for phishing awareness," *Behaviour & Information Technology - Behaviour & IT*, vol. 32, pp. 1–10, 01 2011.

[16] D. D. Caputo, S. L. Pfleeger, J. D. Freeman, and M. E. Johnson, "Going spear phishing: Exploring embedded training and awareness," *IEEE Security Privacy*, vol. 12, no. 1, pp. 28–38, 2014.

[17] "Cyber security, kandidat." `https://www.aau.dk/uddannelser/kandidat/cyber-security/`. Accessed: 16-09-2021.

[18] "Specialisering i it-sikkerhed." `https://master-it-vest.dk/interesser/it-sikkerhed/`. Accessed: 16-09-2021.

[19] R. Badhwar, *The Advanced Malware Prevention Playbook*, pp. 165–172. Springer International Publishing, 2021.

[20] M. Janbeglou, *Understanding and Controlling Unnamed Internet Traffic*. PhD thesis, University of Auckland, 2017.

[21] M. Ahmad and S. Woodhead, "Containment of fast scanning computer network worms," in *International Conference on Internet and Distributed Computing Systems (IDCS)*, vol. 9258, pp. 235–247, Springer, 09 2015.

[22] M. A. Ahmad, S. Woodhead, and D. Gan, "A countermeasure mechanism for fast scanning malware," in *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*, pp. 1–8, IEEE, 2016.

[23] K. Shahzad and S. Woodhead, "Towards automated distributed containment of zero-day network worms," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–7, IEEE, 2014.

[24] P. Mockapetris, "Domain names - implementation and specification," STD 13, RFC Editor, November 1987. `http://www.rfc-editor.org/rfc/rfc1035.txt`.

[25] R. Villamarin-Salomon and J. C. Brustoloni, "Identifying botnets using anomaly detection techniques applied to DNS traffic," in *2008 5th IEEE Consumer Communications and Networking Conference*, pp. 476–481, 2008.

[26] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: A scalable botnet detection method for large-scale DNS traffic," *Computer Networks*, vol. 97, pp. 48–73, 2016.

[27] 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*, CCS '16, (New York, NY, USA), pp. 1568–1579, Association for Computing Machinery, 2016.

[28] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, (USA), p. 18, USENIX Association, 2010.

[29] M. Antonakakis, R. Perdisci, W. Lee, N. V. II, and D. Dagon, "Detecting malware domains at the upper DNS hierarchy," in *20th USENIX Security Symposium (USENIX Security 11)*, (San Francisco, CA), USENIX Association, Aug. 2011.

[30] C. Lu, B. Liu, Z. Li, S. Hao, H. Duan, M. Zhang, C. Leng, Y. Liu, Z. Zhang, and J. Wu, "An end-to-end, large-scale measurement of DNS-over-encryption: How far have we come?," in *Proceedings of the Internet Measurement Conference*, IMC '19, (New York, NY, USA), p. 2235, Association for Computing Machinery, 2019.

[31] K. Baheux, "A safer and more private browsing experience with Secure DNS." https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html, May 2020. Accessed: 27-09-2021.

[32] S. Deckelmann, "Firefox continues push to bring DNS over HTTPS by default for US users." https://blog.mozilla.org/en/products/firefox/firefox-continues-push-to-bring-dns-over-https-by-default-for-us-users/, February 2020. Accessed: 27-09-2021.

[33] Apple, "Enable encrypted DNS," 2020 Apple Worldwide Developers Conference, 2020.

[34] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over transport layer security (TLS)," RFC 7858, RFC Editor, May 2016. https://www.rfc-editor.org/rfc/rfc7858.txt.

[35] "DNSCrypt." https://dnscrypt.info/.

[36] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," RFC 4033, RFC Editor, March 2005. http://www.rfc-editor.org/rfc/rfc4033.txt.

[37] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP flow information export (IPFIX)," RFC 3917, RFC Editor, October 2004. https://www.rfc-editor.org/rfc/rfc3917.txt.

[38] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP flow information export (ipfix) protocol for the exchange of flow information," STD 77, RFC Editor, September 2013. http://www.rfc-editor.org/rfc/rfc7011.txt.

[39] R. Hofstede, P. eleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[40] K. Shahzad and S. Woodhead, "Towards automated distributed containment of zero-day network worms," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–7, 2014.

[41] K. Shahzad and S. Woodhead, "Empirical analysis of rate limiting + leap ahead (rl+la) countermeasure against witty worm," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Com-*

puting and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 2055–2061, 2015.

[42] D. Whyte, E. Kranakis, and P. C. Van Oorschot, "DNS-based detection of scanning worms in an enterprise network.," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2005)*, 2005.

[43] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC 5280, RFC Editor, May 2008. `http://www.rfc-editor.org/rfc/rfc5280.txt`.

[44] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson, "A tangled mass: The android root certificate stores," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, (New York, NY, USA), p. 141148, Association for Computing Machinery, 2014.

[45] V. Dukhovni, "Opportunistic security: Some protection most of the time," RFC 7435, RFC Editor, December 2014. `http://www.rfc-editor.org/rfc/rfc7435.txt`.

[46] P. Hoffman, "Smtp service extension for secure smtp over transport layer security," RFC 3207, RFC Editor, February 2002. `http://www.rfc-editor.org/rfc/rfc3207.txt`.

[47] C. Bonatti, S. Turner, and G. Lebovitz, "Requirements for an IPsec certificate management profile," RFC 4809, RFC Editor, February 2007. `http://www.rfc-editor.org/rfc/rfc4809.txt`.

[48] "Mobile operating system market share worldwide." `https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201912-202106`. Accessed: 14-09-2021.

[49] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[50] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis," in *Proceedings of the USENIX Security Symposium*, (Washington, D.C.), pp. 785–800, USENIX Association, Aug. 2015.

[51] "What You Need To Enroll." https://developer.apple.com/programs/enroll/. Acccessed: 16-09-2021.

[52] "Register for a Google Play Developer account." https://support.google.com/googleplay/android-developer/answer/6112435?hl. Accessed: 16-09-2021.

[53] "Certificates - support - Apple developer." https://developer.apple.com/support/certificates/. Accessed: 27-09-2021".

[54] "Sign your app | Android developers." https://developer.android.com/studio/publish/app-signing. Accessed: 27-09-2021".

[55] L. Li, T. F. Bissyandé, and J. Klein, "Rebooting research on detecting repackaged android apps: Literature review and benchmark," *IEEE Transactions on Software Engineering*, vol. 47, no. 4, pp. 676–693, 2021.

[56] Z. C. Schreuders, T. Shaw, M. Shan-A-Khuda, G. Ravichandran, J. Keighley, and M. Ordean, "Security scenario generator (SecGen): A framework for generating randomly vulnerable rich-scenario VMs for learning computer security and hosting CTF events," in *2017 USENIX Workshop on Advances in Security Education (ASE 17)*, (Vancouver, BC), USENIX Association, Aug. 2017.

[57] "DEFCON CTF archive." https://defcon.org/html/links/dc-ctf.html. Accessed: 28-09-2021.

[58] "CTFtime.org / All about CTF (Capture The Flag)." https://ctftime.org/. Accessed: 27-09-2021.

[59] P. Chapman, J. Burket, and D. Brumley, "PicoCTF: A game-based computer security competition for high school students," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, (San Diego, CA), USENIX Association, Aug. 2014.

[60] J. Vykopal, V. Švábenský, and E.-C. Chang, "Benefits and pitfalls of using capture the flag games in university courses," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, (New York, NY, USA), p. 752758, Association for Computing Machinery, 2020.

[61] R. L. Fanelli and T. J. O, Connor, "Experiences with practice-focused undergraduate security education," in *Proceedings of the 3rd International*

Conference on Cyber Security Experimentation and Test, CSET'10, (USA), p. 18, USENIX Association, 2010.

[62] S. Maroofi, M. Korczyski, C. Hesselman, B. Ampeau, and A. Duda, "COMAR: Classification of compromised versus maliciously registered domains," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 607–623, 2020.

[63] K. Parsons, "Why taking Facebook quizzes is a really bad idea," January 2020. https://www.cbc.ca/news/canada/nova-scotia/tech-columnist-warns-against-taking-social-media-quizzes-for-fun-1.5442282.

[64] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," RFC 8446, RFC Editor, August 2018. http://www.rfc-editor.org/rfc/rfc8446.txt.

[65] E. Rescorla, K. Oku, and N. Sullivan, "TLS encrypted Client Hello," tech. rep., RFC Editor, Aug. 2021. https://www.ietf.org/archive/id/draft-ietf-tls-esni-13.txt.

[66] "HTTPS encryption on the web." https://transparencyreport.google.com/https/overview. Accessed: 29-09-2021.

[67] S. Deckelmann, "Firefox continues push to bring DNS over HTTPS by default for US users," February 2020. https://blog.mozilla.org/en/products/firefox/firefox-continues-push-to-bring-dns-over-https-by-default-for-us-users/.

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-985-1

AALBORG UNIVERSITY PRESS