KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Large-scale analysis of attack techniques on Internet domain names

Thomas Vissers

Supervisors:
Prof. dr. ir. Wouter Joosen
Dr. ir. Lieven Desmet

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

August 2018

Large-scale analysis of attack techniques on Internet domain names

Thomas VISSERS

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

August 2018

Acknowledgments

This dissertation is the product of collaborative efforts, guidance, feedback and the support of many. I feel grateful for being surrounded by wonderful people that have, in many different ways, made this possible.

First and foremost, I want to thank my supervisor, Wouter Joosen. He introduced me to DistriNet, which ultimately led to this fantastic opportunity for me to work on topics I was passionate about and to grow as a security researcher. I greatly appreciate the strong professional mindset Wouter advocates throughout the research group. Over the years, it became more and more apparent to me what a unique, fruitful and positive environment I found myself in. Specifically, I want to thank Wouter for giving me guidance, autonomy and the tremendous amount of interesting opportunities during the PhD. This entire experience has, without a doubt, shaped the future of my professional life.

Secondly, I wish to express my gratitude to my co-supervisor, Lieven Desmet. I started working with Lieven during my second year at DistriNet, when I joined a research project under his management. This was, without a doubt, the most collaborative project I was part of during my PhD. I greatly enjoyed the team spirit, frequent brainstorming sessions, open project discussions and the technical solutions that Lieven facilitated. His door was always open. I am thankful for all his advice and support. Lieven's guidance quickly extended beyond the scope of the research project, evolving into the co-supervision of my PhD.

Next, I would also like to thank Nick Nikiforakis, who played such an important role throughout my entire PhD trajectory. When I first started at DistriNet, Nick was my daily mentor. He taught me the tricks of the trade, his experiences and his research strategies – many of which I still carry with me today. Nick closely guided me through my first research projects and we have been fruitfully collaborating ever since. Even after he took a position at Stony Brook University, we kept working together. Nick invited me there for a research visit at his PragSec lab, a wonderful opportunity that I greatly enjoyed. Apart from academic projects, Nick has always been there for professional advice and –not to forget– a good laugh and a good time.

My sincerest appreciation goes to the jury members, Herbert Bos, Nick Nikiforakis, Vincent Rijmen, Pierre Verbaeten, Frank Piessens and Danny Hughes, for taking the time to thoroughly evaluate my dissertation and to provide valuable feedback. I want to thank Omer Van der Biest for chairing the jury.

I am also thankful for the vital support of all the technical and administrative staff at the Computer Science department, with a special mention to Annick Vandijck and Katrien Janssens for their great efforts and care.

Next, I would like to explicitly mention my close colleague Tom Van Goethem, for our fruitful and enjoyable collaborations, fun conference trips and the many laughs we shared.

Following this, I would like to thank Wannes Meert, Jérôme Renaux and Alex Sarafianos from the DTAI group for letting me pick their brain on the data science-related challenges I faced throughout the years.

I would like to thank all my colleagues at the Computer Science department and DistriNet for the many nice chats, lunches, coffee breaks and inspiring discussions. The fun, open and friendly atmosphere we all bring to work is something I truly value. A warm heart goes out to my current and former office mates: Pieter, Job, Jan Tobias, Nick, Zubair, Vera, Jan, Tom, Tim, Victor, Stéphane, Sven and Stelios.

I would also like to mention my colleagues at Stony Brook University: Najmeh, Timothy, Meng and Olekssi. Thank you for your hospitality and the great time.

Last but not least, I want to express my gratitude to all my caring friends and loving family. In particular, I would like to thank my partner, my brother and my parents. I have been surrounded by their strong and positive support for as long as I can remember. For that, I am forever grateful. Thank you.

Thomas Vissers
August 9, 2018.

Abstract

The Domain Name System (DNS) is a fundamental element of today's Internet. Virtually all online connections depend on resolving a domain name to an IP address. Consequently, DNS is a high-profile target for attackers. Domain names are both exploited and abused to hijack traffic, as well as employed by cybercriminals to set up their infrastructure.

In this dissertation, we report on large-scale, data-driven analyses of new attacks and cybercriminal ecosystems concerning Internet Domain names. We start the thesis by presenting an updated view on recent developments in the DNS abuse landscape, incorporating a wide variety of attack vectors.

This work presents two ecosystem analyses that provide insights into how and why malicious registrations are made. The first study extensively scrutinizes 14 months of registration data of the .eu TLD to identify large-scale malicious campaigns. We explore the ecosystem and modus operandi of elaborate cybercriminal entities that recurrently register large amounts of domains for one-shot, malicious use. The second ecosystem analysis focuses on the domain parking industry. We provide an in-depth exploration of domain parking services. In particular, we examine this monetization strategy of deceptive and parasitic registrations and analyze the harmful consequences for users who unwillingly land on parked domains.

Furthermore, the thesis covers two newly discovered DNS vulnerabilities that enable attackers to hijack domain names through their nameserver: "nameserver typosquatting" and "nameserver bitsquatting". Specifically, we focus on the exploitation of nameserver configuration issues and hardware errors. These server-side security issues allow attackers to seize control over nameserver requests to fully hijack domains.

In addition, we present a large-scale analysis that resulted in a high-impact disclosure concerning DNS-based cloud security services. We extensively studied cloud-protected domains to assess the prevalence of "origin-exposure", i.e. the possibility of bypassing the cloud security and attacking the customer's server directly.

The research on DNS security, as outlined above, yields a broader reflection on the complex and dispersed attack surface of domain names. Based on the insights gathered from these analyses, we are able to formulate an agenda of key elements to be addressed to achieve a more secure DNS.

Beknopte samenvatting

Het Domain Name System (DNS) is een fundamenteel element van het internet van vandaag. Vrijwel alle online verbindingen zijn afhankelijk van de juiste resolutie van een domeinnaam tot een routeerbaar IP-adres. Daarom is DNS een belangrijk doelwit voor aanvallers. Domeinnamen worden uitgebuit en misbruikt om internetverkeer af te leiden, maar ook gebruikt door cybercriminelen om de infrastructuur voor uitgebreide malafide campagnes op te zetten.

In dit proefschrift rapporteren we over grootschalige, datagedreven analyses van nieuwe aanvallen en cybercriminele ecosystemen met betrekking tot internet domeinnamen. We beginnen de doctoraatsthesis met een recente weergave van ontwikkelingen in het landschap van DNS-misbruik, waarin een breed scala aan aanvalsvectoren is verwerkt.

In dit werk presenteren we twee ecosysteemanalyses die inzicht bieden in hoe en waarom malafide registraties worden gemaakt. De eerste analyse bestudeert 14 maanden aan registratiegegevens van het .eu-TLD om grootschalige malafide campagnes te identificeren. We verkennen het ecosysteem en de modus operandi van uitgebreide cybercriminele entiteiten die herhaaldelijk grote hoeveelheden domeinen registreren voor eenmalig, kwaadwillig gebruik. De tweede ecosysteemanalyse richt zich op de domeinparkeerindustrie. We presenteren een grondige verkenning van services die domeinparking aanbieden. We onderzoeken in het bijzonder deze monetiseringsstrategie van misleidende en parasitaire registraties en analyseren de schadelijke gevolgen voor gebruikers die ongewild op geparkeerde domeinen belanden.

Verder rapporteren we over twee nieuw ontdekte DNS-kwetsbaarheden waarmee aanvallers domeinnamen kunnen kapen via hun nameserver: “nameserver typosquatting” en “nameserver-bitsquatting”. Concreet richten we ons op de exploitatie van foute configuratie-instellingen en hardwarefouten bij nameservers. Deze beveiligingsproblemen, gesitueerd aan de serverkant, stellen aanvallers in staat controle te verkrijgen over DNS verbindingen gericht naar de nameserver

om zo domeinen te kapen.

Bijkomend presenteren we een grootschalige analyse gepresenteerd die resulteerde in een impactvolle onthulling omtrent DNS-gebaseerde cloud-beveiligingsdiensten. We hebben cloud-beschermde domeinnamen uitgebreid bestudeerd om de prevalentie van “origin-exposure te bepalen, d.i. de mogelijkheid om de cloudbeveiliging te omzeilen en de server van de klant rechtstreeks aan te vallen.

Het onderzoek naar DNS-beveiliging, zoals hierboven beschreven, levert een bredere reflectie op over het complexe en verspreide aanvalsoppervlak van domeinnamen. Op basis van de inzichten die we uit deze analyses hebben verzameld, kunnen we een agenda opstellen met de belangrijkste elementen die moeten worden geadresseerd om een veiliger DNS te bereiken.

Contents

Abstract iii

Beknopte samenvatting v

Contents vii

List of Figures xiii

List of Tables xvii

1 Introduction 1

1.1 Thesis statement 2

1.2 Contributions 3

1.3 Dissertation outline 4

2 Analysis of DNS abuse 7

2.1 Background 8

2.1.1 History 8

2.1.2 How does DNS work? 8

2.2 Current state of DNS abuse 13

2.2.1 DNS is used to attract traffic to malicious destinations . 13

2.2.2 Cybercrime depends on DNS to run their operations . . 21

3 Malicious registrations 27

3.1 Introduction 30

3.2 Datasets and initial findings 32

3.2.1 Registration data 32

3.2.2 Blacklists 33

3.2.3 Preliminary insights 33

3.3 Campaign identification experiment 34

3.3.1 Campaign identification process 35

3.3.2 General campaign observations 38

3.3.3 Validation of campaign selection criteria 38

3.4 Insights into malicious campaigns 41

3.5 Automating campaign identification 48

3.5.1 Clustering process 50

3.5.2 Results 51

3.6 Discussion and limitations 52

3.7 Related work 55

3.8 Conclusion 56

4 Domain Parking 57

4.1 Introduction 60

4.2 Domain Parking Ecosystem 62

4.2.1 What is Domain Parking? 62

4.2.2 How do users end up on parked domains? 63

4.2.3 Gathering Parked Domains 64

4.3 Analysis of Parked Domains 65

4.3.1 Parked Domain Owners 66

4.3.2 Advertisement Syndicators . . . 67
4.3.3 Typosquatting abuse . . . 69
4.3.4 Parking a Typosquatting Domain . . . 71
4.3.5 Trademark abuse . . . 72
4.3.6 Malicious redirections . . . 73
4.3.7 Detecting and Bypassing Ad-Blockers . . . 77
4.3.8 Summary of Findings . . . 79
4.4 Detecting Parked Domains . . . 80
4.4.1 Gathering data . . . 80
4.4.2 Feature set . . . 81
4.4.3 Classification . . . 84
4.5 Discussion . . . 87
4.6 Related Work . . . 88
4.7 Conclusion . . . 90

5 Hijacking domains through their nameserver 91

5.1 Introduction . . . 93
5.2 Problem statement . . . 95
5.2.1 Hijacking requests to nameservers . . . 95
5.2.2 Scope of study . . . 97
5.2.3 Nameserver dependencies . . . 97
5.3 Nameserver typosquatting . . . 99
5.3.1 Attack vector . . . 99
5.3.2 Amount of traffic affected . . . 100
5.3.3 Potential and current abuse . . . 102
5.3.4 Measuring vulnerable cases . . . 106
5.3.5 Summary . . . 111

5.4 Nameserver bitsquatting . . . 111
5.4.1 Attack vector . . . 111
5.4.2 Amount of traffic affected . . . 112
5.4.3 Assessing current abuse . . . 114
5.4.4 Measuring bit-flip occurrences . . . 115
5.4.5 Summary . . . 117
5.5 WHOIS email hijacking . . . 117
5.5.1 Attack vector . . . 117
5.5.2 Finding vulnerable nameservers . . . 118
5.5.3 Potential impact . . . 119
5.6 Security practices of nameservers . . . 120
5.6.1 Analysis . . . 121
5.7 Discussion . . . 122
5.8 Related work . . . 124
5.8.1 Hijacking domain names . . . 124
5.8.2 Abusing expired domains . . . 125
5.9 Conclusion . . . 126

6 Bypassing cloud-based security providers 127

6.1 Introduction . . . 131
6.2 Background . . . 132
6.2.1 Modus Operandi of CBSPs . . . 133
6.2.2 CDNs as CBSPs . . . 134
6.3 Potential Origin Exposure . . . 135
6.3.1 IP History . . . 136
6.3.2 Subdomains . . . 137
6.3.3 DNS records . . . 137

6.3.4 Temporary exposure 138
6.3.5 SSL certificates 138
6.3.6 Sensitive files 139
6.3.7 Origin in content 139
6.3.8 Outbound connections 139
6.4 Large-scale Analysis 140
6.4.1 CBSP Providers 140
6.4.2 Origin Verification 144
6.4.3 Ethical Considerations 145
6.4.4 Results 145
6.5 Discussion & countermeasures 152
6.6 Related Work 154
6.7 Conclusion 156
6.8 Availability 156

7 Conclusion **159**
7.1 Summary 159
7.2 Discussion of possible root causes 160
7.2.1 Lack of DNSSEC adoption 160
7.2.2 Reactive countermeasures are outpaced 162
7.2.3 Ambiguous roles and incentives 163
7.2.4 Non-uniform authoritative domain names 164
7.2.5 Implications of GDPR 165
7.3 DNS-dependent security mechanisms 166
7.4 Concluding thoughts 167

Bibliography **169**

List of publications **189**

List of presentations and trainings **190**

List of Figures

2.1 A fully-qualified domain name annotated with its hierarchical components 9

2.2 Excerpt of the .se zone file 9

2.3 DNS resolution process of a client that connects to host example.com for the first time. 10

2.4 The domain registration process. 11

2.5 DNSSEC adoption within large TLDs. 12

2.6 Safari's URL bar 16

2.7 DNS-based cloud security and direct-to-IP attacks 23

2.8 DNS amplification attack 25

3.1 The share of registrations made in each week 34

3.2 Daily percentage of malicious registrations, including and excluding campaign registrations. 35

3.3 Malicious registrations, grouped by email provider and country of the registrant. 36

3.4 Campaign duration and activity over time. 37

3.5 Extended false positive analysis of each campaign. 41

3.6 Daily share of all and malicious registrations between April 1, 2015 and April 20, 2015. 44

3.7 Times of registrations for campaign c_19. 45

3.8 Registrations per day and per registrar of campaign c_11. . . . 46
3.9 Mapping of the top 30 clusters to campaign instances. 53
3.10 Related temporal activity of clusters. 53

4.1 Overview of the operation model of domain parking. 63
4.2 Number of found domains for the 15 observed parking services. 67
4.3 The percentage of distinct owners and their cumulative share in possession of parked domains. 68
4.4 Different third-party JavaScript advertisement syndicators and their presence on parked domains. 69
4.5 Percentage of typosquatting domains present per service. . . . 70
4.6 Graph of redirections leading to malware recorded by the US crawler. 74
4.7 Screenshot of a scam website to which our crawler was redirected after visiting a parking page. 78
4.8 ROC curve of the parking detection model on the test dataset. 86

5.1 DNS resolution process of a client that connects to host example.com for the first time. 95
5.2 Fragment of the nameserver dependencies related to dynect.net. 99
5.3 The amount of domains an attacker can hijack by registering a number of available typosquatting NSDOMs. 104
5.4 Experimental setup for monitoring the resolutions to typosquatting nameservers. 108
5.5 Requests per minute to typosquatting nameserver for two different misconfigured domains. 108
5.6 Possible bit-flip events during recursive resolution. 113
5.7 The cumulative distribution of nameservers by the amount of days their BIND version is outdated. 122

6.1 Cloud-based security. 133

6.2 Bypassing cloud-based security. 135
6.3 The portion of domains protected by each company, across segments of the Alexa top 1 million. 141
6.4 For each 100K-rank segment in the Alexa top 1 million, the percentage of domains where the CBSP is bypassable. 146
6.5 The percentage of domains that is susceptible to one or more origin-exposing vectors. 147

List of Tables

2.1 Example of authoritative variations of a domain, next to malicious deceptions. 16

3.1 Attributes used to express the selection criteria of a campaign. 36

3.2 The different types of abuse, the blacklists and registration timing patterns per campaign. 43

3.3 Minor inconsistencies found in the registration details campaign domains. 45

3.4 The amount of registrars, phone numbers, email addresses and types of email providers used per campaign. 47

3.5 Top 3 most malicious registrars, email providers and registrants. 49

4.1 Summary of the observed parking services together with their required domain configuration. 66

4.2 The percentage of redirections that occurred per service per region. 72

5.1 First and last part of the top NSDOMs 96

5.2 Categories of registered typo/bitsquatting NSDOMs based on their responses to target DNS queries. 106

5.3 Web pages returned from the rogue IP addresses. 106

5.4 Registered nameserver typosquatting domains and the subdomains that were queried. 110

5.5 The most queried name for each typosquatting nameserver registration during 31 days. 110

5.6 Registered nameserver bitsquatting domains. 116

5.7 Observed nameserver bitsquatting occurrences. 117

5.8 NSDOMs with outdated WHOIS. 119

6.1 Cloud-based security providers, along with the DNS records that are adjusted by their clients, and the number of protected domains that were found in the Alexa top 1 million. 142

6.2 The percentage of domains that can be bypassed using each origin-exposing vector, for each cloud-based security provider's customers. 147

Chapter 1

Introduction

"J'aime bien repérer le petit détail
que personne ne verra jamais."

Amélie Poulain

Anno 2018, society is rapidly embracing the digital world as its primary means of communication. IT systems have become vital facilitators of most administrative and business operations. Meanwhile, the use of connected devices and software is seeping into all aspects of everyday life. This revolution is largely fueled by the omnipresence of powerful personal devices (e.g. smartphones, computers) and inexpensive, reliable data connectivity. Nowadays, WhatsApp –an internet-connected messaging platform– processes 60 billion messages, pictures and videos that are sent between its 1.5 billion users every day [42]. In addition, we have online social networks boasting billions of active users [41]. However, the Internet is not just about connecting people. In Europe, more than half of the adult population uses the Internet to handle financial transactions [66]. Furthermore, more than one billion people use an internet-connected app to navigate through traffic and find their next destination [104]; Ultimately, even driving itself will soon be fully in the hands of connected software [202]. Altogether, in a very brief time span, we have witnessed an enormous impact of the Internet on our social, business, everyday and even physical lifes.

One of the cornerstones of the Internet is the Domain Name System, or *DNS*. It essentially acts as the phone book of the Internet: mapping human-friendly names to numeric addresses that are used to route data across the entire network. When taking into account the examples given above, it is evident that the role the Internet, and by extension DNS, plays in society has dramatically changed. Surprisingly, DNS has been largely unaltered since its creation in the 1980s. Back then, when the Internet was still in its infancy and essentially a scientific experiment, there was no need for strong security. Hence, security was essentially not built into its specification. Interestingly, DNS has scaled remarkably well with the exponential increase in Internet usage. However, given its current critical importance, the need for security guarantees is now very apparent.

According to a report by Norton, in 2017, 44% of consumers were impacted by cybercrime, leading to a global loss 172 billion USD [146]. The most common attacks that consumers face are malware infections on their devices, financial fraud and compromised online accounts. In 2016, the world panicked when ransomware took the stage and held the data of business and individuals hostage. In 2017, the ransoms collected by these attacks alone are projected at 2 billion USD. Overall, the total cost of cybercrime is estimated at 600 billion USD [148].

Given the fundamental nature of DNS, it is involved in most stages of Internet-based attacks. From reconnaissance to delivery, as well as command and control: DNS is present all over the cyber kill chain. In this dissertation, we discuss several different aspects of DNS abuse through large-scale analyses. We provide insights into the modus operandi of DNS abuse, disclose novel attack vectors and perform measurements of the prevalence of vulnerabilities and abuse.

1.1 Thesis statement

Securing domain names and DNS resolutions becomes increasingly critical as more sensitive data and transactions are moved online. Moreover, additional security mechanisms are built on top of DNS, further relying on the integrity and authenticity of DNS resources. In this thesis, I exhibit the non-uniformity of threats and abuse against DNS by presenting intertwined security problems that affect different parts of the DNS ecosystem. I disclose novel types of DNS errors and misconfigurations that expose online entities to traffic hijacking and web attacks. Next to that, I explore multiple cybercriminal activities that register and monetize massive amounts of abusive domain names. Keeping in mind the lack of coordinated anti-abuse efforts in DNS due to its complex and distributed nature, I propose pragmatic countermeasures that administrators, users and online organizations can leverage to detect and also prevent DNS abuse.

1.2 Contributions

In this dissertation, we analyze a multitude of attacks and abuse concerning Internet domain names and vulnerabilities with DNS itself. The main contributions of this dissertation are:

- We provide a multifaceted view on *developments in DNS abuse.* We create a mapping between real-world exploitation techniques, and the different parties within the hierarchical DNS components. Additionally, we discuss how cybercriminals weaponize and employ DNS in different ways to facilitate their cyber operations. (Chapter 2)
- We study how malicious actors *register domain names* to be used in cybercriminal campaigns. We conduct an extensive analysis of their registration patterns and underlying strategies. We are able to identify several large malicious campaigns that are responsible for the vast majority of all blacklisted registrations. We observe that these campaigns continuously register thousands of short-lived domains that are mostly used in spam attacks. Using these insights, we introduce a method for registries to automatically identify malicious campaigns. (Chapter 3)
- We bring forward an ecosystem analysis of *domain name parking services*, mapping out the parties involved and discussing the role that they play. We find that parked services lead visitors to malicious content, including malware, scams and inappropriate content. We analyze how trademark infringing domain names are an integral part of the business model. To aid the community, we introduce an accurate classifier for detecting parked pages that can be leveraged to minimize and filter out exposure to parked domains. (Chapter 4)
- We present two novel DNS attacks: *nameserver typosquatting* and *nameserver bitsquatting.* These vulnerabilities are further understood and confirmed through in-depth analyses and large-scale measurements. We find instances of targeted abuse exploiting these weaknesses in the wild and demonstrate the prevalence of vulnerable domains. (Chapter 5)
- We deliver a comprehensive overview of *origin-exposing attacks* on *DNS-based cloud security services.* We report on the first large-scale measurement of this critical vulnerability and find that over 70% of domains using this security product are at risk. Starting from the insights of our study, we bring forward several suggestions to reduce the risk of bypassable cloud security. (Chapter 6)

Next to the main academic contributions that are fully included within this dissertation, the author of this thesis has been involved in practical deployments.

- As a further contribution to the malicious domain registration study presented in Chapter 3, the author worked on a detection system that aims to predict whether new incoming registrations will be used in malicious operations. Specifically, the author developed a custom distance metric to automatically identify clusters of malicious registrations. The similarity to these clusters help the system to assess the malicious intent of new registrations. This approach has been deployed as part of early warning system at the .eu registry. A patent application has been filed for this invention.
- Along with the publication of the results regarding the risk of origin exposure of cloud-protected domains (Chapter 6), the author developed a freely available *online vulnerability scanner* to allow website administrators to test their own exposure. The tool has been used to scan hundreds of domains.

1.3 Dissertation outline

The remainder of this dissertation is structured as follows. In Chapter 2, we discuss the domain name ecosystem and present an updated view on developments in DNS abuse. Next, we present an in-depth exploration of malicious registrations within the .eu TLD in Chapter 3. In Chapter 4, we discuss the ecosystem and security implications of domain parking services. Thereafter, Chapter 5 introduces new vulnerabilities in DNS caused by nameserver misconfigurations and hardware errors, followed by a large-scale security evaluation of DNS-based cloud security services in Chapter 6. Finally, in Chapter 7, we conclude the dissertation by summarizing the different findings throughout the dissertation and by reflecting on the root causes of unresolved DNS abuse and potential future directions for its mitigation.

Chapter 3, 4, 5 and 6 contain the contents of peer-reviewed papers published at international academic conferences. Apart from the paper content, each of these chapters encompasses a preamble that places the publication in context and provides post-factum reflections and developments.

Publication data of these papers:

- *Parking sensors: Analyzing and detecting parked domains.* [198]
  Vissers, T., Joosen, W., and Nikiforakis, N.
  NDSS 2015

- *Maneuvering around clouds: Bypassing cloud-based security providers.* [200]
  Vissers, T., Van Goethem, T., Joosen, W., and Nikiforakis, N.
  ACM CCS 2015

- *Exploring the ecosystem of malicious domain registrations in the .eu tld.* [199]
  Vissers, T., Spooren, J., Agten, P., Jumpertz, D., Janssen, P., Van Wesemael, M., Piessens, F., Joosen, W., and Desmet, L.
  RAID 2017

- *The wolf of name street: Hijacking domains through their nameservers.* [197]
  Vissers, T., Barron, T., Van Goethem, T., Joosen, W., and Nikiforakis, N.
  ACM CCS 2017

Chapter 2

Analysis of DNS abuse

The Domain Name System (DNS) is one of the key foundations of the Internet. First and foremost, it takes care of translating domain names into routable IP addresses. DNS is a massively applied and stable protocol –first introduced in the 80s– that has scaled exceptionally well with the explosive growth of the Internet. While security issues with DNS were off the radar for several years, we are now seeing the rise of a new generation of abuse. As a prime example, in October 2016, we witnessed one of the biggest Internet outage to date. Online giants such as Twitter, Netflix and Amazon were taken down all at once by an attack on their shared DNS infrastructure. However, it is not only the availability of online resources that is threatened by this rekindled security interest in DNS. These days, cyber criminals are hijacking Internet traffic by tampering with DNS resolutions, victimizing users with deceptive domain names and leveraging DNS to execute large-scale malicious operations.

This chapter first introduces the important entities within the entire DNS ecosystem and gives an overview of the workings of DNS. In the second part, we analyze the current developments in the abuse of domain names. We report how new attacks on DNS force traffic to malicious destinations and how DNS is used by cybercriminals as a standard service necessary to run their abusive operations.

2.1 Background

This section covers a brief overview of the history of the domain name system (DNS), which parties the domain ecosystem is comprised of, and how the name resolution process works.

2.1.1 History

In the 1970s, computer network technology was rapidly developing. People started accessing other machines on their local networks, and later on the Internet, back then still called ARPANET. Technically, computers communicate with each other using Internet Protocol (IP) addresses. However, these numeric addresses –while perfectly suited for computers– are hard to remember for humans. Networks started to grow rapidly and human-friendly references to IP addresses quickly became a necessity [120]. Initially, a file ("hosts") was periodically downloaded from a central location and listed all hostnames with their corresponding network address. However, soon problems arose with the scalability and synchronization of this file as networks continued to expand.

To solve this problem, Paul Mockapetris proposed a distributed and dynamic domain name database in 1983 [126]. Essentially, this was the first version of DNS. In 1987, DNS was officially born when an updated RFC became an IETF Internet Standard [127].

2.1.2 How does DNS work?

The DNS hierarchy

The domain name system (DNS) is essentially a hierarchical, distributed database. When decomposing a fully-qualified domain name (FQDN), as shown in Figure 2.1, we can distinguish between the following components, from right to left: the root zone, the top-level domain (TLD), second-level domain (2LD) and the optional subdomains.

One can think of the DNS hierarchy as a tree structure, that starts at the root. On the internet, the root zone is generally implied to be present in every FQDN, but can be represented by a dot at the most right position. The root zone is managed by the nonprofit organization ICANN[1]. Thirteen globally distributed root nameserver clusters are responsible for operating the root zone. The IP

[1]Internet Corporation for Assigned Names and Numbers

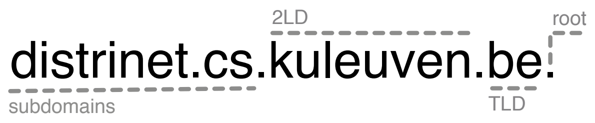


Figure 2.1: A fully-qualified domain name annotated with its hierarchical components

address of at least one of these servers has to be “hardcoded” into DNS resolvers in order to bootstrap DNS resolution.

The root is able to point resolvers to the nameservers of all possible top-level domains (e.g. “com”, “net”). Often, we make a distinction between generic TLDs (gTLDs) and country-code TLDs (ccTLDs), such as “.be” or “.uk”. Early 2013, there existed only about one hundred TLDs. Later that year, ICANN opened up the possibility for individuals and organizations to sponsor their own TLD (e.g. “.jewelery” or “.toyota”), which stimulated an explosive growth in the number of new gTLDs. As of March 2018, there are over 1,500 gTLDs listed under the root zone.

Each TLD is managed by a registry that is responsible for delegating resolutions for the 2LDs in their zone. Analogous to the upper hierarchy, the registries maintain a list (called *zone file*) of the “child” domains in their zone. The zone file lists the *authoritative nameservers* of these children. The authoritative nameservers of a 2LD serve as the authority for that domain. These servers hold the entire inventory of DNS records for that domain and are thus able to provide an answer to any query regarding the domain[2]. Listing 2.2 shows an excerpt of the .se TLD zone file [188].

Listing 2.2: Excerpt of the .se zone file [188]. Each line lists from left to right: the 2LD, the time-to-live value, record type and the record’s data: an authoritative nameserver for the 2LD.

```
music.se.           86400   NS      ns1.parkingcrew.net.
music.se.           86400   NS      ns2.parkingcrew.net.
music-art.se.       86400   NS      ns1.loopia.se.
music-art.se.       86400   NS      ns2.loopia.se.
music-city.se.      86400   NS      ns01.one.com.
music-city.se.      86400   NS      ns02.one.com.
music-corner.se.    86400   NS      ns01.one.com.
music-corner.se.    86400   NS      ns02.one.com.
```

[2]Alternatively, domain administrators may opt to further delegate subdomains to other nameservers.

The name resolution process

To understand how DNS resolutions work, we have to consider the three main parties involved. First, we have the *client* that wishes to resolve a domain name to an IP address. This could, for instance, be a web browser on the computer of a user. Next, we have the *recursive resolver*. Typically, Internet service providers (ISPs) set up recursive resolvers to do the heavy-lifting of DNS resolutions for all their clients. Alternatively, clients can use open DNS resolvers (e.g. OpenDNS, Google DNS). Lastly, we have the *authorative nameservers* at each level in the DNS hierarchy, which correspond to the hierarchical levels described previously. These servers hold the actual data: the DNS records.

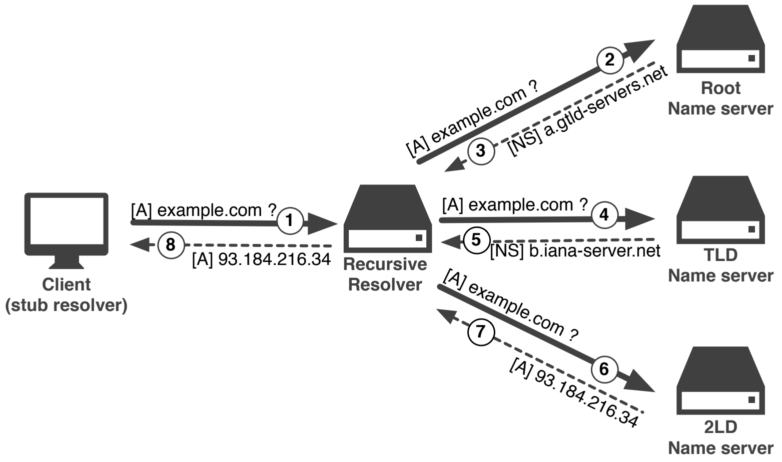


Figure 2.3: DNS resolution process of a client that connects to host example.com for the first time.

During the resolution of a name to an IP address, as illustrated in Figure 2.3, the client (or *stub resolver*), who wishes to resolve a domain name, sends out a DNS request to its recursive resolver (1). Subsequently, iterative queries are made to the hierarchically structured nameservers. Recursive resolvers (RR) work their way down the nameserver hierarchy. By asking the root server (2), the RR gets the location of the TLD's nameservers, in this case com's (3). After asking com's nameservers about example.com (4), the RR will be referred to the 2LD nameservers (5-6). Finally, the RR gets an authoritative response for example.com (7) and can forward it to the client (8). In practice, the RR caches the intermediate responses and will most likely already know where, for instance, the TLD nameservers are located because of a previous recursive query.

The domain registration ecosystem

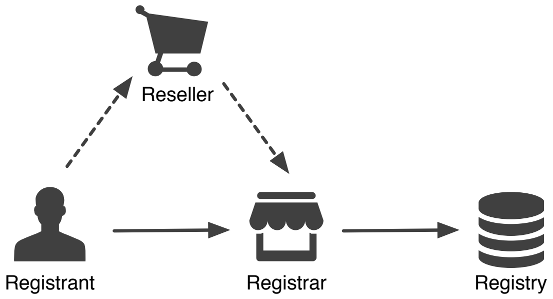


Figure 2.4: The domain registration process.

Under open[3] TLDs, second-level domains can be registered by any eligible entity (individuals, companies, etc.). We refer to these entities as *registrants*. Although the TLD zone files are managed by the *registries*, in most cases, registrants have to register domain names with a *registrar* (e.g. GoDaddy), as illustrated in Figure 2.4. Registrars are accredited by ICANN and essentially act as an interface between the registrant and the TLD registry. Most registrars partner with multiple registries and thus offer registrations under multiple TLDs. Many registrars also provide reseller programs to other companies, so these third-parties can also "sell" registrations using the registrar's accreditation. The yearly registration fee that the registrant pays to the registrar (or reseller) is distributed amongst the registrar, the registry and ICANN. The situation is different for registries of ccTLDs, who typically pay a yearly financial contribtion to ICANN based on, amongst others, the number of domain names under their administration [91].

DNS record types

Translating a domain name to an IP address is generally regarded as the primary functionality of DNS. This information is propagated through A or AAAA records in the case of IPv6. However, many more record types exist to facilitate the publication of other data regarding the domain. For instance, the location of the email server is listed in the domain's MX records. Also, as seen in Listing 2.2, the NS records point to the nameservers of certain zone and essentially establish the reference between parent and child zone. The SRV record is used to publish information regarding the host and port number of

[3]Not all TLDs allow registrations of 2LDs by external entities. Some TLDs are private or have specific requirements for the registrant.

certain services such as SIP [163] and XMPP [166]. Another important record is the TXT record, which allows free text entries. It enables many protocols and applications to specify records for their own purpose. For instance, today there are many examples of security protocols (e.g. SPF [102], DKIM [49], CAA [167], etc.) and domain verification mechanism (e.g. ACME [21]) that rely on TXT records.

DNSSEC

DNS by itself was designed with little security in mind. It lacks the security guarantees that should be expected for a system that is so crucial and integral as it is today.

To solve this problem, a first version of DNSSEC was published back in 1999 [62], but would see major revisions to improve in scalability in the subsequent years. DNSSEC is a security extension to DNS that provides authentication and integrity by allowing nameservers to add digital signatures of their resource records. As such, they establish a chain of trust starting from the root zone, all the way down to the domains's authoritative nameserver. This allows clients to validate whether the DNS responses they receive have not been tampered with, and that they originate from the legitimate owner of the domain name. Although DNSSEC has matured greatly and the demand for secure DNS transactions is more critical than ever, its adoption has remained surprisingly low. Figure 2.5 demonstrates this by showing the adoption rate of 2LDs domains in five popular TLDs, all of which are less than 1%.

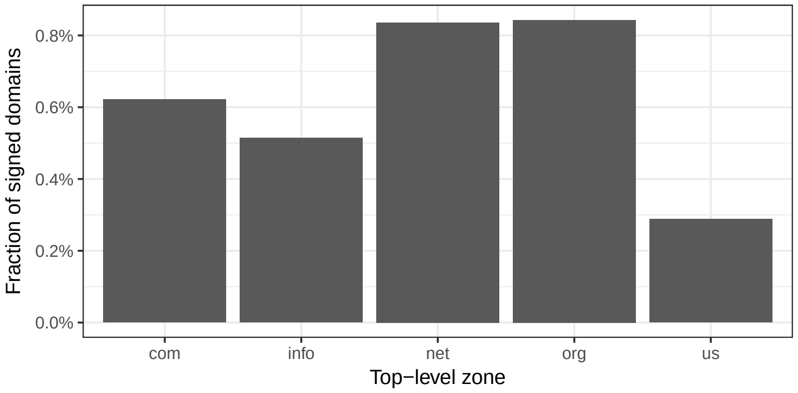


Figure 2.5: DNSSEC adoption within the five largest TLDs listed on StatDNS [180].

2.2 Current state of DNS abuse

In this section, we give an updated view on a variety of DNS abuse vectors. First, we consider techniques where DNS is exploited to lure traffic to malicious destinations. Afterwards, we discuss why cybercriminals need the general functionality of DNS to conduct their malicious operations.

2.2.1 DNS is used to attract traffic to malicious destinations

Attracting traffic is an integral part of most cybercriminal businesses. Infecting users with malware through drive-by downloads [30], stealing sensitive credentials through phishing pages [95], or selling counterfeit products in fake webshops [101] are all examples where users have to be lured to malicious setups. Given the crucial role that DNS plays in connecting entities on the Internet, malicious actors abuse it to victimize users. In this section, we discuss two categories of malicious tactics that use DNS to attract traffic. First, we elaborate on techniques that visually deceive users and secondly, we address technical attacks on DNS that hijack traffic from legitimate domains.

Users are deceived while browsing the web

When connecting to an online resource, the domain name is the most apparent aspect. Users type in the browser's URL bar, click on links and are redirected to other websites, while putting trust in the domain names they interact with. Attackers take advantage of this trust and deceive users by connecting them to visually similar domains and URLs.

Typosquatting Popular domains such as facebook.com and youtube.com are typed into browsers' URL bars millions of times each day. As a result, there is bound to be some users that make small typing mistakes while doing so. Miscreants anticipate on this by registering typographical variations of domain names. Instead of receiving an error messages that the domain they just typed in does not exist, users will now connect to an attacker-controlled domain name. This practice is called *typosquatting*. Generally, miscreants register domains based on commonly made typos, such as character permutations (e.g. f$\underline{\text{ae}}$cbook.com) and missing-dot typos (e.g. $\underline{\text{www}}$facebook.com) [204]. Overall, the most popular domains are targeted the most as they yield the highest potential to receive unintended traffic. In 2015, researchers of KU Leuven measured that for the 500 most popular domain names, there are more than 10,000 active typosquatting domains [8].

Evidently, typosquatters attempt to monetize the traffic coming from their misled visitors. To do so, the majority of typosquatters partner with *domain parking* services. These companies generate revenue from inactive domains by hosting web pages filled with advertisements, sharing the earnings with the domain owner. Moreover, a study by Vissers et al. [198] reported that parked domains are exposing their visitors not just to ads, but to all sorts of cybercriminal –and thus more lucrative– activities (discussed in Chapter 4). This monetization scheme is called *Pay-Per-Redirect*, and allows other miscreants to buy traffic to automatically forward visitors to their malicious destinations, such as scam and malware campaigns. Next to domain parking, some typosquatters set up more targeted monetization strategies for their trademark infringing domain names. In the case of *affiliate abuse*, attackers register typos of companies with an affiliate program (e.g. webshops such as Amazon) and immediately forward accidental visitors to the authoritative website through their own affiliate link. This allows the typosquatters to strike up a commission for the purchases that are made after the redirection, even though the visitor intended to go directly to the company's website.

Combosquatting Instead of anticipating on typos, other abusive strategies keep the original name intact, but combine it with other characters or keywords to form a new domain name (e.g. paypal$\overline{\text{-updates}}$.com). This practice is called *combosquatting.* In a study from 2017, Kintis et al. [101], discovered that the corpus of active combosquatting registrations is nearly a hundred times larger compared to typosquatting. In part, this can be explained by the virtually infinite amount of new variations that can be created through combosquatting. The most common use of combosquatting domains is trademark abuse, whereby cybercriminals lure interested users to their deceptive domains through, for example, paid advertisements. A typical example is a fake ecommerce website or one that sells counterfeit products. This trend is confirmed when looking at the top five keywords that are most frequently combined with abused trademarks: *free*, *online*, *code*, *store* and *sale.*

The combosquatting technique can also be leveraged to set up *phishing* pages that have visually similar URLs to authoritative pages. Take for instance the following domain name:

```
https://bankofamerica-comlogin-sys-update-online.com
```

It tries to deceive less tech-savvy users who cannot easily distinguish between the domain name and the path portions of a URL. The authoritative counterpart of this phishing URL looks like this:

```
https://bankofamerica.com/login/sys/update/online
```

The content of these web pages is typically visually identical to their legitimate counterpart, giving very little clues that the visited URL is malicious. Similarly, criminals create deceptive phishing URLs through elaborate subdomain sequences, wherein the 2LD that was actually registered becomes an inconspicuous part of the URL. As subdomains can be created freely under any registered domain name, attackers have a lot of leeway to create convincing deceptions. For instance, in the truncated, real-world example listed below, a user might believe that he is interacting with paypal.com, while in reality the domain in the URL is viemzaza.com [178].

```
https://paypal.com.us.cgi-bin.ebscr.cmd.home.general.dispatch.
[*].viemzaza.com/sas/cgi-bin/ias/A/1/FGT/ibd/IAS/presentation/pm_
token=C2886KJEHD89483JS03829ENDHU83920JD/
```

Inconsistent authenticity signals We argue that it is not straightforward for a user to determine the authenticity of a domain or URL. In part, this is caused by inconsistent placement of the trusted name in the URL, preventing a clear-cut, uniform assessment strategy.

The first problem is that large organizations legitimately use combosquatting techniques to set up company-related websites. It appears that in a lot of companies, hosting web pages on a subdomain or path of the official domain is a difficult or tedious process that is often avoided. Instead, combosquatting-like registrations are made to quickly set up websites for marketing campaigns or other secondary activities. The epitome of this confusing practice took place after Equifax suffered from a large breach in September 2017. To aid possible victims of the hack, Equifax had set up a website on an entirely new domain: equifaxsecurity2017.com. However, the company –in its own confusion– wrongly linked users to securityequifax2017.com via their Twitter page [54]. Arguably, companies should avoid making new registrations that include their name as this could train users to trust arbitrary domains that are not necessarily owned by the trademark holder.

Another layer of complexity in assessing the authenticity of a URL was introduced when ICANN launched the new gTLD program. Since late 2013, this program has spawned over a thousand new TLDs sponsored by companies, organizations and individuals. As a result, there now exist domains such as

Authoritative	`microsoft.com`
Authoritative	`home.microsoft`
Authoritative	`advertisewithmicrosoft.com`
Combosquatting	`microsoft-secure.com`
Deceptive subdomain	`microsoft.com.secure[*]updates.com`
Typosquatting	`mircosoft.com`
Bitsquatting	`mic2osoft.com`

Table 2.1: Example of authoritative variations of a domain, next to malicious deceptions.

home.jpmorgan and **jpmorgan.com** which are equally authentic domains of JPMorgan Chase & Co. Arguably, this introduces even more ambiguity when assessing the semantics of a URL as the registered name to trust can now legitimately appear in multiple places in the URL (at the top-level and the second-level). To illustrate the difficulty of assessing the legitimacy of a domain, Table 2.1 shows several authoritative variations of microsoft.com, next to deceptive variations of the domain name. To add to the confusion, in practice some 2LDs form an integral part of the TLD (e.g. co.uk or edu.in), making the third-level domain (3LD) the primary registered name. These longer "suffixes" are diverse and not always very obvious. To keep track, browser vendors even maintain a Public Suffix List to correctly determine the scope of cookies within a domain [135].

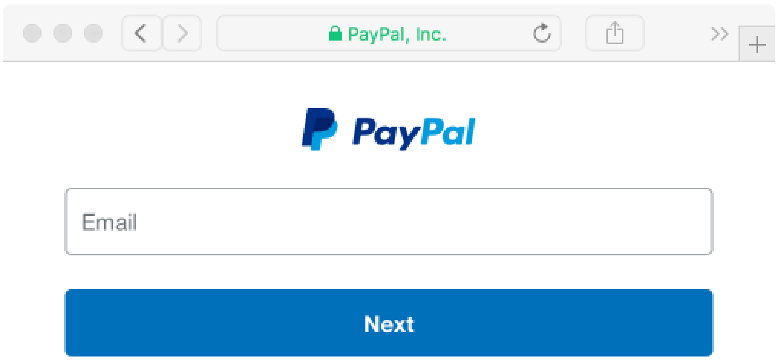


Figure 2.6: The URL bar in Safari only shows the name listed on the EV certificate presented by paypal.com.

As these kinds of deceptions mainly rely on the inability to distinguish between the registered domain name and other portions of the URL, several browser vendors are modifying their user interface to make this distinction more apparent. For instance, Apple's Safari browser does not longer show the path of a URL, they only visualize its domain or even just the company name listed on the domain's certificate (Figure 2.6). However, even this is not foolproof. Recently, security researchers showed how businesses with the same name as popular online services can be legally incorporated in other states or countries. This allows malicious actors to also obtain an Extended Validation (EV) certificate with that service's name on it. Consequently, they are once more able to deceive users of Safari and other browsers [72].

Taking down infringing domains An important legal weapon for trademark or brand owners is the Uniform Domain-Name Dispute-Resolution Policy (UDRP) [90], which allows trademark holders to file a case against confusingly similar domains. Although the UDRP leads to quicker decisions than legal cases, a resolution can still take up to two months [210]. Alternatively, some domain owners resort to *defensive registration*, a more proactive countermeasure. Here, the idea is that the domain owners registers the variations of its domain name himself, preventing others from doing so. For example, Microsoft owns wwwmicrosoft.com and simply redirects accidental visitors to the correct domain. Obviously, this can quickly become complex or expensive as the space for typo variations may be quite large. As an example, microsoft.com has 76 different variations, according to commonly used typo models. Defensive registrations are not a feasible tactic against combosquatters and deceptive subdomains. Simply because the possibilities for infringing URLs and domains are practically unbounded. To aid in the fight against cybersquatters, companies employ brand protection services such as MarkMonitor, Incopro and Netnames which actively monitor new potentially infringing registrations.

Traffic is also hijacked under the hood

Given the amount of deceptive links and domains, users are constantly instructed to rigorously verify the Internet address that they are interacting with. However, instead of deceiving users, attackers might also exploit technical vulnerabilities and misconfigurations of DNS itself. As the DNS resolution process happens entirely in the background, this allows miscreants to reroute users to a malicious destination, entirely transparent to the end user. In this section, we discuss existing –but still relevant– DNS attacks, alongside new vulnerabilities reported in recent publications.

Cache poisoning DNS heavily relies on caching to improve performance. As recursive resolvers are typically shared among many clients, a DNS response can be cached to later serve other clients that query for that same record. This is the mechanism that is targeted by cache poisoning attacks. The goal of this technique is to inject a rogue DNS record into the resolver's cache, leading to the false resolution of a –typically popular– domain name to the attacker's IP address. If successful, all clients behind the targeted resolver could be given that poisoned cache record and unintendedly connect to the attacker's server.

The main crux of a cache poisoning attack is to get resolvers to accept a rogue DNS response before the authentic response is able to reach the resolver. As DNS uses the connectionless UDP protocol, the attacker can forge a response that seems to come from the IP address of the authoritative nameserver. This technique is called *source address spoofing*. It allows the attacker to send a fake DNS response to the resolver from anywhere on the Internet. However, each DNS request has a randomly assigned transaction ID[4] and the attacker's fake response has to match this ID to get accepted by the resolver. Although an attacker can flood the server with thousands of packets to guess the correct transaction ID, the attack was long deemed practically impossible. There is namely only a very small window of opportunity to guess the ID, as the response from the authentic nameserver typically arrives within a matter of milliseconds. Moreover, attackers were waiting until the cache expired before they executed another poisoning attempt.

2018 marks the 10-year anniversary of the most famous DNS cache poisoning vulnerability: The Kaminsky Attack [96]. Dan Kaminsky came up with a practical technique to enable aggressive, repeated brute-force attempts to guess the correct transaction ID of a DNS request, enabling the attack to succeed within a matter of seconds. A collaborative patching effort introduced *UDP source port randomization*, preventing the attacker from knowing which port the DNS response was expected to be delivered to. Now attackers have to guess both the random source port *and* the random transaction ID. At that time, this new measure sufficiently reduced the feasibility of the attack. More recently, we see the adoption of 0x20-bit encoding to increase the entropy of DNS requests even further [52]. This feature randomly mixes the case of each character in the requested DNS name, resulting in queries such as `wWW.gOoGlE.cOM`. Although DNS mappings are case-insensitive, the DNS response packet must exactly mimic the requested characters in order to be accepted.

While these multiple workarounds made a cache poisoning attack impractical over the years, the problem was not fundamentally solved. DNSSEC is the proposed general solution against the technical tamperings described in this

[4] In the early days, the transaction ID was often even just a predictable sequential counter.

section. It provides authentication and integrity for DNS requests by a cryptographically signing responses. This allows users to fully validate the DNS responses they receive.

Dangling DNS records While cache poisoning is a problem inherent to the DNS protocol, negligent DNS administrators are also a cause of vulnerabilities. Over the last two years, multiple research groups have specifically looked at misconfigurations in DNS that expose domains to exploitation.

One such issue is that of *dangling DNS records* [119]. This vulnerability occurs when administrators forget to remove DNS records that point to abandoned resources, such as IP addresses of discontinued VMs in public clouds (e.g. Amazon Web Services or Azure). Cloud resources are very dynamic in nature as computing instances can be spawned and shut down continuously. Consequently, the amount of dangling DNS records is substantial: in 2018, researchers reported that over 700,000 unique domain names have DNS records pointing to freely available cloud IP addresses [26]. As an attacker, it is possible to cycle through the pool of available IP addresses of cloud providers at a very fast pace. They demonstrate that it is feasible to get a hold of a specific released IP address in a matter of minutes. This short time window is often even within the TTL of many DNS records, implying that exploitation could succeed even when the DNS record was properly removed but simply still active in the cache of clients.

When an attacker is able to get a hold of such an abandoned or reconfigured IP address, there are numerous options to exploit this situation. For instance, nowadays we see a lot services authenticating domain owners by asking to upload a unique file at a certain path on the domain. The attacker can pass these verification tests as he is now in control of resources hosted at that particular dangling (sub)domain. This gives the attacker potential access to a variety of privileges, such as the valid generation of TLS certificates for that domain (e.g. Let's Encrypt [113] using ACME verification [21]) and the administration of the domain in third-party applications (e.g. Google Search Console). Furthermore, when in control of a dangling DNS record, attackers are able to receive and send authenticated emails from that record, set up convincing phishing campaigns using the authentic domain, or even inject malicious code through stale script inclusions.

Misconfigured nameserver records Managed DNS providers (e.g Dyn, Neustar, DNSimple) are a popular choice in these days of cloud outsourcing. This results in large numbers of domains relying on the same services for their DNS operations. Consequently, there are many domains that have configured the same domain in their NS records. Another recently studied DNS misconfiguration is that of typos in those NS records. The NS record is a pivotal element of the hierarchical DNS structure. During the recursive resolution process, it effectively enables the parent zone (e.g. "com") to refer the resolver to the authoritative nameserver of the second-level domain (e.g. "google").

A domain with an erroneous NS record is vulnerable to extensive hijacking attacks. If an attacker is able to register the domain listed in the erroneous NS record, he will now be considered as an authoritative nameserver for that domain from the protocol's perspective. This gives him near full control of the domain name, as it allows him to configure DNS records that enable interception and rerouting of traffic, emails and essentially all communication to and from the domain name.

This newly discovered problem is an actively exploited threat, as attackers are actively exploiting hundreds of domains with misconfigured NS records (discussed in Chapter 5). We report on cyber criminals that spent hundreds of dollars to get a hold of a single nameserver domain that is able to exploit a highly-ranked misconfigured webshop. Analogous to regular typosquatting abuse, they found that proactive attackers register typographical variations of the most-used nameservers, i.e. those from large managed DNS providers. This strategy can be extremely efficient, as these NS records are configured by millions of domains, increasing the general odds of misconfigurations.

Bit-errors in DNS resolutions Faulty hardware, exposure to extreme temperatures and even cosmic radiation are known to cause bit-flip errors in the memory of computers and devices [187]. In DRAM, bit errors are typically mitigated with Error Correcting Codes (ECCs). Although the adoption of this kind of countermeasure is common, it is still often missing in consumer devices and even in DRAM-containing components of enterprise class systems, such as NICs and hard drives. Since DNS resolutions travel through a lot of different devices, they are prone to bit-flipping as well. If these bit-flips alter the in-memory representation of a domain name, it can effectively lead to a request to another domain name. To illustrate this, a bit-flip can cause an accidental connection to twitte2.com instead of twitter.com as the binary ASCII code for "2" is **0011 0010**, which is a single bit-flip away from **0111 0010**, the ASCII code for "r". A study by VeriSign [43], reported that about one in every $10^7 - 10^8$ DNS resolutions suffers from a bit-level error. This phenomenon inspired a cybersquatting attack, called *bitsquatting*, wherein miscreants register bit-flipped variants of often resolved domain names in order to intercept victims of these rare bit-flipped DNS resolutions. Bitsquatting was first introduced to the wide public at Blackhat in 2011 by Artem Dinaburg [55]. Recently, a light was also shed on the efficiency of bit-flipped resolutions on NS records.

As discussed previously, globally speaking, the NS records of popular managed DNS providers occur very frequently in DNS resolutions. This increases the odds of those domains suffering from bit-errors, as will be further discussed in Chapter 5.

2.2.2 Cybercrime depends on DNS to run their operations

Malicious actors do not just use DNS to steal traffic away from legitimate domains. They actually employ DNS to faciliate their cybercriminal operations. In this section, we discuss how criminals register domain names to set up communication, perform reconnaissance on target domains and weaponize DNS servers for disruptive cyber attacks.

Criminals use DNS to set up the attack infrastructure

DNS is a key component across all phases of cybercriminal campaigns. Therefore, most cybercriminal operations rely heavily on functional domain names. To give some examples: spammers need domains to send email, host scam and phishing websites; cyber criminals infect hosts through malware distributing domains, after which the botnet uses rendezvous domains to connect to the command-and-control (C&C) server; and so forth.

DNS-related malicious activity is generally stopped by blacklists (i.e. blocking the resolution of that domain name for clients) or seizing control over the involved domain names with the help of law enforcement. To avoid being disrupted, cybercriminals adopted agile domain registrations to outpace these countermeasures. More specifically, botnets are now programmed with unpredictable domain generation algorithms (DGAs), spawning dozens of possible new rendez-vous domains per day. The botmaster is the only one who knows which domains will be generated in advance, allowing him to register a small subset right in time. This prevents the early setup of any domain-specific countermeasures. In turn, security researchers are now reverse engineering DGAs to be able to precompute domains that will be generated in the future in order to blacklist them in advance [153].

We see a similar short-lived strategy to avoid disruption by blacklists in the realm of email spam. Researchers have measured that the majority of spam domains are active for just one day after registration [83]. In other words, cyber criminals register, abuse and immediately abandon domains at a very fast rate.

These hit-and-run strategies are reflected in the ecosystem of malicious domain registrations, as attackers now need a constant supply of new domain names to ensure continuity of their activities. In 2017, Vissers et al. reported on that ecosystem in the .eu TLD [199] and were able to identify several massive, long-running campaigns that burned through thousands of domains (discussed in Chapter 3). Moreover, the authors find that just a small set of criminal actors was responsible for over 80% of all malicious registrations.

DNS is helpful for attackers while preparing for cyber attacks

When hackers target an organization, generally their first step is to gather information about the infrastructure under the domain. This allows hackers to explore their target and to determine possible points of entry. DNS plays a crucial role in the attacker's reconnaissance phase, as DNS records essentially hold an inventory of hosts and services within a domain.

Zone enumeration The most convenient method to retrieve information regarding a domain is to request all its DNS records through a zone transfer. Zone transfers are intended to synchronize the primary and secondary authoritative nameservers by sending over a copy of the zone file. Given the sensitivity of this information, unauthenticated zone transfers are generally considered a vulnerability and are typically closed down for outsiders these days.

When zone transfers are properly hardened, hackers resort to brute-force zone enumerations. By explicitly sending DNS requests for each potentially existing record, they try to reconstruct the entire zone file. As many thousands of requests have to be sent to achieve this, hackers typically make use of dictionaries and multiple resolvers to make this process more efficient and stealthy (e.g. SubBrute[5]).

DNSSEC has the concept of *authenticated denial-of-existence*, which makes defending against zone enumeration considerably harder. This mechanism is implemented through NSEC records and essentially creates a linked list of all names that exist in the zone. In response to a query for a non-existing name, the DNSSEC server returns the two alphabetically adjacent names that do exist in the zone. The client thus has proof that the name he queried does not exist, as it falls into the signed gap between those two names. Unfortunately, by stating the two adjacent DNS names, an attacker can now easily reconstruct the entire zone file by walking through this linked list of NSEC records. The issue is partly addressed by NSEC3, wherein hashed versions of the DNS names are used instead. However, given the typically short lengths of DNS names, it is

[5]https://github.com/TheRook/subbrute

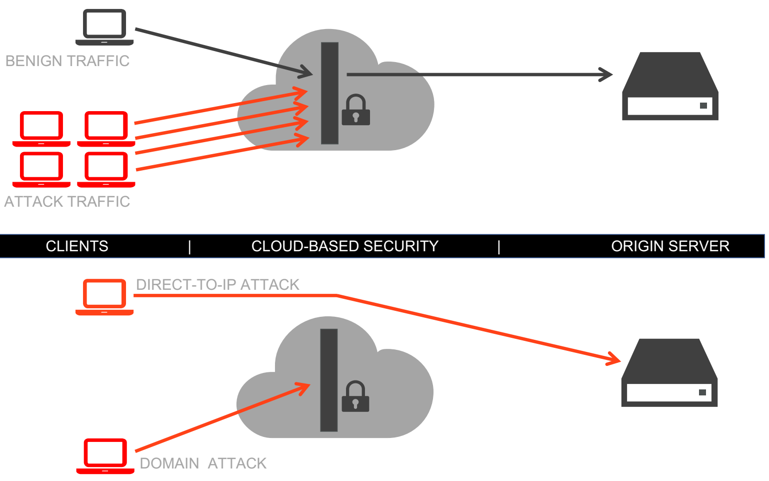


Figure 2.7: DNS-based cloud security acts as a reverse proxy for traffic destined to the origin server (Upper). Attackers can still directly reach the origin server if its IP address is exposed (Lower).

quite feasible to crack these hashes with the help of rainbow tables or dictionary attacks [203]. Ultimately, the only real solution for zone enumeration with DNSSEC currently is *online signing*. Meaning that one dynamically generates signed responses for each incoming non-existing name. This strategy has been implemented by some large DNS providers but comes at the cost of more computational power and the additional risk of constantly using the private keys to sign new responses. Since 2015, NSEC5 is being developed which aims to reduce the risk of online signing by using a separate key for NSEC5 records [70].

Origin exposure DNS is also an integral part of cloud security services, such as Incapsula or Cloudflare. These companies offer protection against distributed denial-of-service (DDoS) and web application attacks by filtering out malicious traffic in the cloud, before it is able to reach the protected server. To hide and protect their customer's servers (the *origin*), the customer's domain resolves to an IP address of the security company instead of the customer's own server. This enables the security company to act as a reverse proxy and to only forward benign traffic to the customer's server. As shown in Figure 2.7, it is crucial that the real IP address of the customer's server remains hidden to prevent hackers from bypassing the cloud protection and directly attacking it.

In reality, it is shown that hackers can easily retrieve the hidden IP address and attack the origin directly by carefully sifting through DNS data. In 2015, Vissers et al. [200] evaluated this on a large-scale and found that over 70% of cloud security customers were exposed (discussed in Chapter 6). One of the most common leaks was the presence of historical DNS information. Several companies constantly keep track of DNS changes. This historical data can be leveraged to find out to which IP address the protected domain resolved to prior to the cloud security deployment. Additionally, the domain's current DNS records may contain traces of the origin's IP address as well. Often there are subdomains configured, such as *ftp.* that simply directly resolve to the customer's IP address.

DDoS attacks heavily utilize DNS

Over the last years, denial-of-service attacks have become a very real and important threat. DDoS-for-hire services have made executing attacks cheap and as simple as pushing a button [97]. Typically, a DDoS attack involves a distributed botnet of infected computers that are simultaneously instructed to flood the network infrastructure of the victim, disrupting access to legitimate users. In many cases, a technique called *traffic amplification* is used to increase the amount data they are throwing at the victim, as illustrated in Figure 2.8. During an amplification attack, packets are not sent to the victim's infrastructure directly. Instead, attackers send requests to unrelated public UDP services (such as DNS), while spoofing the victim as the alleged sender of those requests. When the UDP services reply, they will direct their response to the victim. This traffic reflection hides the attacker's infrastructure from the victim and may even amplify the attacker's bandwidth in case there is an asymmetry between the request and response size of the UDP service. DNS servers are a popular choice to execute amplification attacks. A study by Rossow [165] finds 7.8 million available open DNS resolvers, which can amplify an attacker's traffic by a factor of 28.7, on average. Akamai's latest "State of The Internet Security" (Q4 2017) reports that DNS-based DDoS attacks are the most frequently observed [10].

DNS servers are not only enablers of denial-of-service attacks, they are also appealing targets. When a domain's authoritative nameservers are offline, clients cannot connect to any of the domain's resources. More specifically, as soon as cached records for that domain expire, the necessary IP addresses can no longer be retrieved and all clients are put in the dark on how to contact the domain. Essentially, the authoritative nameservers are a single point of failure, which explains why the provisioning of redundant nameservers is mandatory in the DNS specification. The situation is even more critical when a large number of domains share the same DNS infrastructure, as is the case with managed DNS providers. This is exactly what happened during the October 2016 Dyn attack, which is considered the largest Internet outage in history. Although

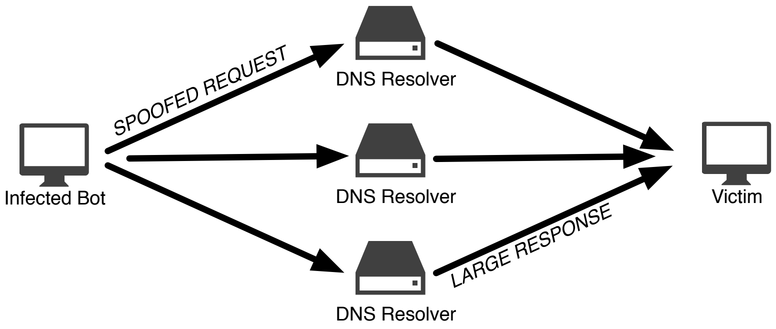


Figure 2.8: DNS amplification attack. The infected bot is instructed to send out spoofed requests to open DNS resolvers. This reflects and amplifies the attack traffic.

the company had a highly redundant and globally distributed infrastructure spread across 20 data centers, it was no match for the unprecedented massive DDoS attack of 1.2 Tbps [212]. In the end, the Dyn's nameservers were heavily disrupted, affecting many of its high-profile customers such as Spotify, Amazon, Netflix, Reddit and Twitter.

In this chapter, we covered the basic functionality of DNS and introduced the most important entities involved in registering and operating an Internet domain name. Thereafter, we gave an updated view on recent developments in DNS-related attacks. In the following four chapters, we present large-scale analyses that are concerned with several aspects of this varied abuse landscape.

Chapter 3

Malicious registrations

Preamble

This chapter reports on the findings of a joint research project between DistriNet, KU Leuven and EURid, the registry that operates and manages the .eu top-level domain. This collaboration was set up to reduce the abuse conducted by domain names registered within the .eu zone.

Typically, abusive domain names are blocked by third-party reputation providers, such as blacklists (e.g. Spamhaus [190] and SURBL [184]). However, these countermeasures do not dismantle the abusive domain, they only block access for their users. Furthermore, reputation providers mainly blacklist domains *after* abuse has already been reported or observed. In contrast, entities involved in the operational aspects of domain names (i.e. registries and registrars) are able to completely halt the workings of a malicious domain, preventing any further abuse. Moreover, as these entities handle the registration and delegation process of domain names, they are in a unique position to stop abuse in an early stage of a domain's lifetime. EURid leverages their role as a registry to enhance the security and trust in the .eu zone by detecting abusive domains at the time they are registered. This prevents these malicious domain names from becoming operational and cause harm in the first place.

Thomas Vissers was involved in practically all stages of the long-standing collaboration with EURid. Throughout the project there have been fruitful interactions between the research team and the management, security, legal and operations teams of EURid.

The project encompasses two phases. First, the abuse landscape within the .eu was mapped out to establish a thorough understanding of how cybercriminals register domain names and for which purposes. By identifying large-scale malicious campaigns, we could analyze the underlying modi operandi of the most prominent malicious actors. The insights of this study were first publicly presented at the RAID 2017 conference in Atlanta, USA. Later, the research team was invited to speak at several other events such as the 41th M3AAWG[1] meeting in Toronto, CA and the OWASP[2] BeNeLuX Day in Tilburg, NL.

In the second phase, predictive models were developed to enable EURid to detect malicious domain names. The cyber criminal motives and strategies observed during the first phase served as crucial insights to steer the feature engineering process of the machine learning models. This follow-up work, which is not further documented in this chapter, involved implementing techniques to automatically predict –at registration time– whether a domain name is bound to be used for abusive purposes. To that extent, two complementary models have been developed to classify new registrations as benign or malicious. The first model groups previous blacklisted registrations into clusters, by mainly focusing on similarities and patterns present in those registrations. Once related malicious domain names are grouped into clusters, new registrations can be predicted as being registered as part of existing malicious clusters. The second model is based on calculating reputation scores of historical registration data. These scores form the input to a binary classifier that learns to associate reputations with malicious intent. Both strategies are complementary and can be combined to improve the performance of prediction. Furthermore, the machine learning models are designed to operate as online learners, i.e. re-training with new data to generate new models on a regular basis. This ensures continuous adaptation to the changing strategies of intelligent attackers, contributing to a robust detection system.

The two predictive models are operational within EURid's production environment as of December 2017. This required the deployment of the research prototype into this environment, taking into account SLAs, real-time performance and integration with existing systems. Additionally, EURid has filed a patent for the prediction models. After 34 days of operation, the prediction system has processed 54,705 new registrations, flagging 3,889 with malicious intent. To evaluate these predictions, we consult other reputation providers and manually verify certain cases. These first results are promising: the system was able to predict the vast majority of blacklisted registrations (88.6% of them), with an acceptable low number of false positives (45 domains, or 0.09%).

[1]Messaging Malware Mobile Anti-Abuse Working Group
[2]Open Web Application Security Project

Once a registration is flagged as malicious by the prediction system, EURid has the opportunity to take action and prevent the domain name from causing immediate harm. As most malicious campaigns depend on a fast-paced, hit-and-run strategy this is an effective strategy. We are closely monitoring the impact of the prediction system on the ecosystem while counteractions are phased into operation. In particular, we are monitoring changes in the rate of abuse and the introduction of evasive strategies. Up until now, the first major impact on the abuse landscape was the removal of over 25,000 malicious registrations of a single campaign that was detected with the help of the reputation-based algorithm [64].

The further contents of this chapter are replicated from the paper titled "Exploring the Ecosystem of Malicious Domain Registrations in the. eu TLD" [199], which was published in the proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses (RAID), in 2017. This work was done with the collaboration of other authors from KU Leuven and EURid. Thomas Vissers and Jan Spooren are the lead authors of this paper.

Exploring the Ecosystem of Malicious Domain Registrations in the. eu TLD

Abstract

This chapter extensively scrutinizes 14 months of registration data to identify large-scale malicious campaigns present in the .eu TLD. We explore the ecosystem and modus operandi of elaborate cybercriminal entities that recurrently register large amounts of domains for one-shot, malicious use. Although these malicious domains are short-lived, by incorporating registrant information, we establish that at least 80.04% of them can be framed into 20 larger campaigns with varying duration and intensity. We further report on insights in the operational aspects of this business and observe, amongst other findings, that their processes are only partially automated. Finally, we apply a post-factum clustering process to validate the campaign identification process and to automate the ecosystem analysis of malicious registrations in a TLD zone.

3.1 Introduction

The Domain Name System (DNS) is one of the key technologies that has allowed the web to expand to its current dimensions. Virtually all communication on the web requires the resolution of domain names to IP addresses. Malicious activities are no exception, and attackers constantly depend upon functioning domain names to execute their abusive operations. For instance, phishing attacks, distributing spam emails, botnet command and control (C&C) connections and malware distribution: these activities all require domain names to operate.

Widely-used domain blacklists are curated and used to stop malicious domain names[3] shortly after abusive activities have been observed and reported. As a consequence, attackers changed to a hit-and-run strategy, in which malicious domain names are operational for only a very small time window after the initial registration, just for a single day in 60% of the cases [83]. Once domain names have fulfilled their purpose, attackers can simply abandon them and register a new set of domain names to ensure continuity of their criminal activities [201].

This strategy is economically viable to the attackers when the cost of registering a domain name is minimal. However, this approach requires repetitive and often automated domain name registrations. We refer to these series of malicious domain names registered by a single entity as *campaigns*. To obscure their actions, attackers often use fake registration details and need to switch between identities, registrars and resellers to avoid detection.

Moreover, we have observed that certain underground services pop up to facilitate the bulk domain registration process for abusive activities. For instance, on the darknet forum “AlphaBay”, we found several instances of “Domain and Email Registration as a Service”. In one example[4], cyber criminals register new domain names and create fresh, private email accounts that are sold to be used for illegal activities, such as carding.

The sheer volume of malicious domain names, as well as the fact that the registration process is being automated and monetized, illustrates the need for strong insights into the modus operandi of cybercriminals to produce effective countermeasures.

In this chapter, we focus on the malicious campaign ecosystem by extensively leveraging the registrant and registration details, with the goal to better understand how miscreants operate to acquire a constant stream of domain names. We rigorously investigate 14 months of .eu domain registrations, a top 10 ccTLD [138] for the European Economic Area. Overall, the dataset of this study contains 824,121 new domain registrations; 2.53% of which have been flagged as malicious by blacklisting services.

Among others, the following conclusions can be drawn from this in-depth assessment:

[3]We use the term *malicious domain name* whenever we refer to a domain name that is registered to be bound to a malicious service or activity.

[4]`http://pwoah7foa6au2pul.onion/forum/index.php?threads/%E2%96%84-%E2%96%88-%E2%98%85-paperghost-%E2%98%85-%E2%96%88-%E2%96%84-fresh-non-hacked-private-email-logins-lower-your-fraud-detection-score-2.71566`

1. While most malicious domains are short-lived, a large fraction of them can be attributed to a small set of malicious actors: 80.04% of the malicious registrations are part of just 20 long-running campaigns. We identified campaigns that were active for over a year, and campaigns that registered more than 2,000 blacklisted domains. (Section 3.3)
2. The campaign identification process suggests that 18.23% of malicious domains does not end up on a blacklist. (Section 3.3.3)
3. The malicious domain registration process is only partially automated: underground syndicates work along office hours, take holiday breaks and make human errors while registering domains. (Section 3.4)
4. Ecosystem analysis can be automated and reproduced by leveraging clustering algorithms. In our experiment, the 30 largest clusters formed by agglomerative clustering encompass 91.48% of blacklisted campaign registrations. These clusters exhibit a clear mapping with manually identified campaigns. (Section 3.5)

The remainder of this chapter is structured as follows. First, in Section 3.2, we introduce the data set used in this research, along with initial insights. Next, we perform a large scale experiment to manually identify malicious campaigns (Section 3.3), followed by several analyses to gather more insights (Section 3.4). In Section 3.5, we follow up with a method to automate campaign identification. We discuss applications and limitations in Section 3.6, followed by a summary of related work in Section 3.7. Lastly, we conclude this study in Section 3.8.

3.2 Datasets and initial findings

In this section, we present the data used in this research and give initial insights based on a first, high-level analysis.

3.2.1 Registration data

We analyzed 824,121 .eu domain registrations between April 1, 2015 and May 31, 2016. We inspected the following fields:

Basic registration information contains the domain name, the date and time of registration, and the registrar via which the registration happened.

Contact information of the registrant contains the company name, name, the language, email address, phone, fax, as well as postal address information.

We decomposed two additional attributes from the email address: the email account and the email provider.

Nameservers or glue records that are responsible for resolving entries within the domain. We enriched the nameserver data with their geographical location by resolving the NS records and adding IP geolocation data.

3.2.2 Blacklists

To capture whether or not a domain was used in malicious activity, a set of public blacklists was queried on a daily basis. Each new domain is monitored daily during 1 month after registration. Afterwards, all domains were checked once more 4 months after the last registration in our dataset. The following blacklist services have been used:

dbl.spamhaus.org blacklist [190]. This Spamhaus blacklist is queried using their DNS API, and provides indicators for botnet C&C domains, malware domains, phishing domains, and spam domains.

multi.surbl.org blacklist [184]. SURBL features a combination of different lists, such as abuse, phishing, malware, etc. The combined SURBL list is queried over DNS.

Google's Safe Browsing list [76]. Google's Safe Browsing list is queried via a Web API, and provides indicators for malware domains, phishing domains, and domains hosting unwanted software, as described in [77].

3.2.3 Preliminary insights

Given the data described above, we present a preliminary analysis to provide insights in the general trends and patterns of malicious registrations.

Observing the 824,121 registrations that occurred between April 1, 2015 and May 31, 2016, we find that 2.53% end up on a blacklist. This corresponds to a total of 20,870 registrations used by cyber criminals in the given 14 month time span. Figure 3.1 shows the weekly share of both malicious and all registrations over this period. The differences in intensity of malicious registrations are moderately correlated with those of all registrations ($\rho = 0.54$). However, the variance of malicious registrations is clearly much larger. Most of the increased malicious activity, for instance at the start of February 2016, can be attributed to a single malicious campaign. These cases are discussed in depth in Section 3.3.

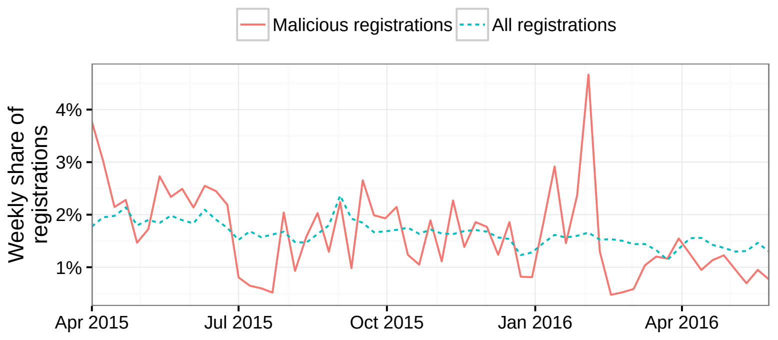


Figure 3.1: The share of registrations made in each week with respect to the total amount registrations that were made in one year. The red line represents these numbers solely for the set of malicious registrations.

The selected blacklists return metadata that encode the reason(s) why a particular domain name was flagged. In our records, 93.68% of the blacklisted domains in the dataset is labelled for spam, 2.09% for malware infrastructure, 0.57% for unwanted software, and 3.22% for phishing activities.

Most domains appear on blacklists very shortly after their registration. More specifically, 72.93% of malicious domains were flagged within 5 days of delegation. 98.57% of malicious registrations are listed on a blacklist in their first month.

3.3 Campaign identification experiment

Typically, illegal online activities do not occur in an isolated or dispersed fashion [67, 83]. Instead, malicious actors commonly set up campaigns that involve multiple, tightly related abusive strategies, techniques and targets. Through an in-depth, a posteriori analysis of the .eu dataset, we assessed whether such patterns can be identified between domain registrations and to what extent these registrations happen in bulk.

Ultimately, we manually identified 20 distinct campaigns responsible for the vast majority of malicious registrations. A campaign represents a series of registrations over time, with strong similarities in terms of registration data (e.g. the registrar, the registrant's address information, phone number or email address, and the set of nameservers). Moreover, a campaign can most probably be attributed to a single individual or organization. In this section, we first give a more thorough description on how these campaigns were identified, followed by some general insights into their characteristics.

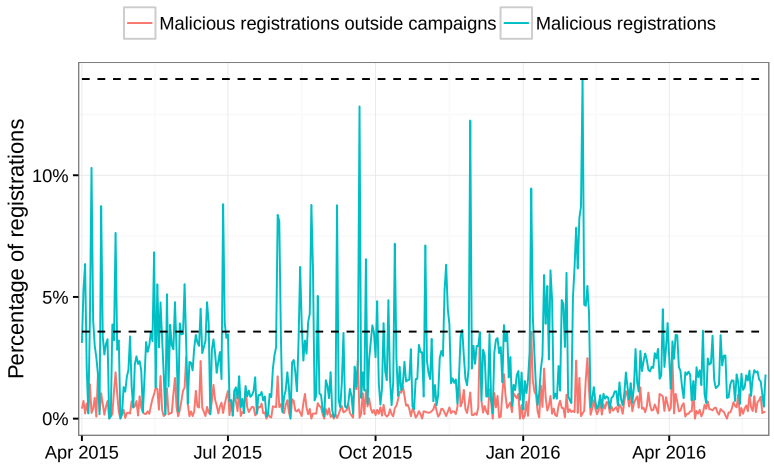


Figure 3.2: Daily percentage of malicious registrations, including and excluding campaign registrations. The dotted lines represent the highest daily concentration of both sets.

3.3.1 Campaign identification process

As malicious registrations often occur in batches [83, 82], high temporal concentrations can serve as a preliminary indicator of campaign activity. Figure 3.2 plots the relative amount of malicious registrations on each day. That graph can be used to identify the time periods in which the amount of malicious registrations was surging. If a campaign was responsible for a high concentration of malicious registrations, a substantial subset of registrations within that timeframe should be related to each other. Hence, all malicious registrations that occurred in that time span are examined to find common characteristics in the registration data. These can be recurring values or distinct patterns in the email address, the address info, the registrar, the registrant name, etc. To detect useful outliers, we visualized correlations between registration fields. For example, by plotting the email providers of the registrants versus the country listed in their street address (as shown in Figure 3.3), multiple hotspots of malicious registrations can be found that contribute to one or more campaigns. These unique combinations and patterns form the basis of the manually assigned campaign selection criteria. To evaluate these, we apply them to the full dataset, i.e. on both benign and blacklisted registrations, over all 14 months. If the criteria match multiple active days and contain a substantial number of blacklisted domains, they are withheld as a new campaign. This process was repeated iteratively, reducing the number of malicious concentrations

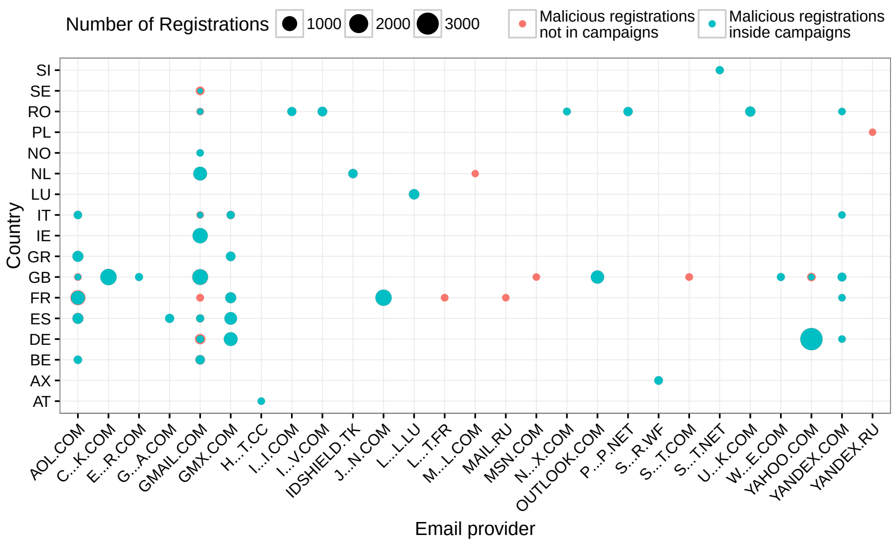


Figure 3.3: Malicious registrations, grouped by email provider and country of the registrant. For visibility, combinations with less that 50 registrations are left out of the figure. Moreover, email providers with less than 50 distinct email addresses in the dataset have been obfuscated for privacy reasons.

Table 3.1: Attributes used to express the selection criteria of a campaign. ● represents a string match, and ☆ a regular expression pattern.

	Campaign																			
Criteria	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
domain name	–	–	–	–	☆	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
registrar	–	–	–	●	–	–	–	–	●	–	–	●	–	–	●	–	–	–	–	●
nameservers	–	–	–	☆	–	–	–	●	–	–	–	–	–	–	☆	–	–	–	–	●
name	☆	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
address	–	●	●	☆	–	●	–	–	–	–	–	–	●	●	☆	●	–	–	–	–
organization	☆	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
email account	–	–	☆	☆	–	–	●	–	–	–	–	☆	–	–	–	–	–	–	●	–
email provider	●	–	●	●	●	–	●	–	●	●	●	–	–	–	☆	●	–	●	●	●

each time.

Over the complete dataset, we identify 20 distinct campaigns. A variety of attributes of the registration details have been used to characterize a campaign, the specifics for each campaign are listed in Table 3.1.

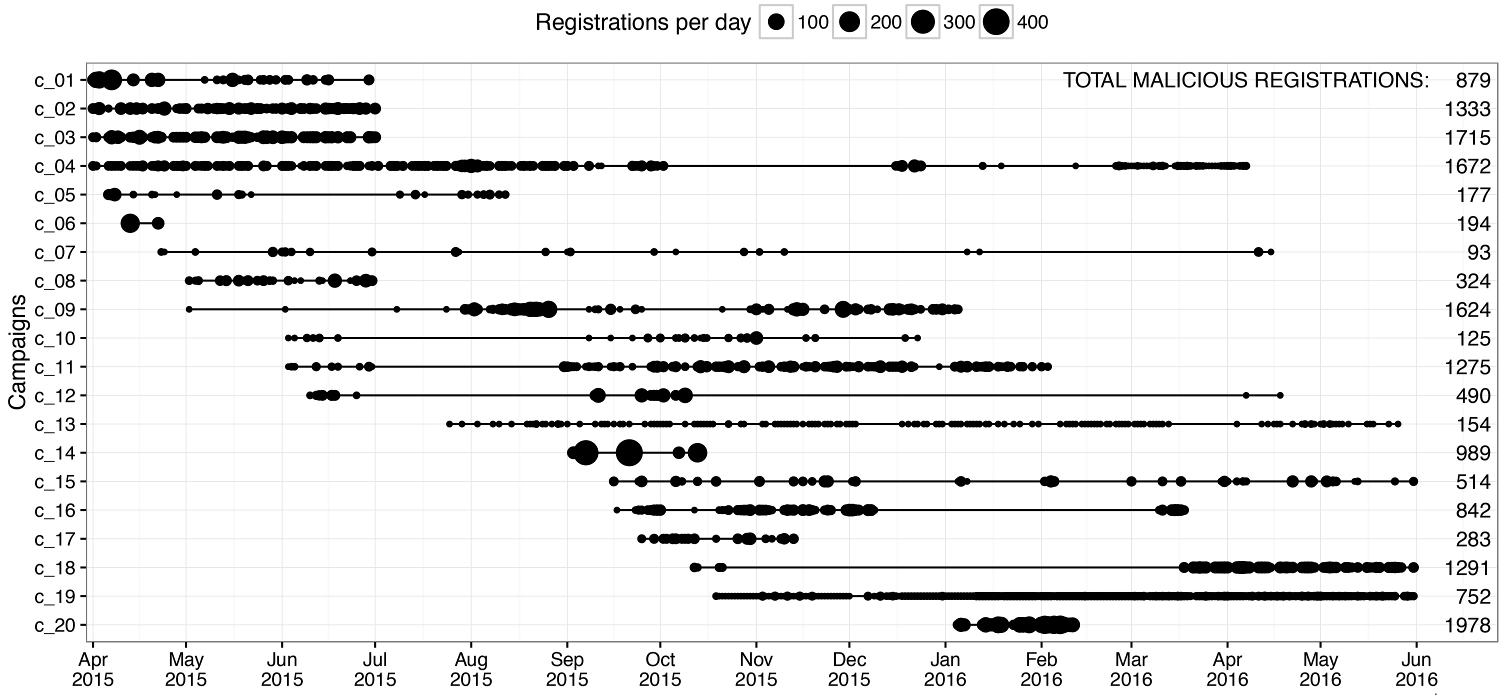


Figure 3.4: Campaign duration and activity over time. The black lines represent the overall duration of the campaign, while the black dots indicate the number of malicious registrations on that day.

3.3.2 General campaign observations

The activity of the 20 identified campaigns is depicted in Figure 3.4. A first observation is that most of the campaigns are long-living: only one campaign runs for less than a month, while some campaigns run up to a year and more[5].

Secondly, campaigns strongly vary in their activity patterns. Some campaigns are active on almost a daily basis (e.g. campaign c_19), whereas others only have a few distinct active days throughout their lifetime (e.g. campaign c_07). Similarly, campaigns vary in concentration. An intense, six week campaign was for instance responsible for almost 2,000 new registrations (c_20), whereas one steady campaign ran over 10 months and produced only 154 malicious registrations (c_13).

A third observation is that campaigns contribute to a large fraction of malicious registrations found in the .eu registration data. Together, the 20 campaigns cover 16,704 domain registrations, that appeared on blacklists. This represents 80.04% of the 20,780 blacklisted registrations in our dataset.

Lastly, not all registrations identified as part of a campaign are flagged as malicious. In total, 19.30% of the campaign registrations we identified are not known as abusive domains by blacklisting services. A more in-depth analysis of these potential false positives is discussed in Section 3.3.3. Note that to avoid any bias, Figure 3.4 only includes registrations that appeared on blacklists, and thus represent a lower bound of campaign activity.

3.3.3 Validation of campaign selection criteria

As briefly mentioned in the previous section, 19.30% of the registrations associated with malicious campaigns do not appear on blacklists. We expect that various reasons contribute to this mismatch:

1. **Incomplete coverage by blacklists.** As blacklists are not exhaustive oracles, we expect that certain domains in a campaign may simply not have been picked up by the specific set of blacklists used.
2. **Not abused.** It is possible that a number of campaign registrations simply has not been used for malicious purposes (yet).
3. **False positives.** Some of our campaign criteria might not be strict enough, introducing false positive matches.

[5]Note that some campaigns might be running even longer than 372 days, as they might have been active before the starting date of our dataset (campaigns c_01 - c_05) or they may still be active past the time span that is covered in our dataset.

Figure 3.5 depicts in red the percentage of registrations for each individual campaign that appears on a blacklist. There are three campaigns with less than 60% of their registrations blacklisted: c_05, c_11 and c_15. In the remainder of this section we validate the quality of the campaign selection criteria. We attempt to gauge the real false positive rate by inspecting domains belonging to campaigns, but do not appear on blacklists. A high false positive rate would imply that the selection criteria are imprecise and include a significant set of domains that were registered without any malicious intent. In contrast, a very high true positive rate implies that the selection criteria are substantially more exhaustive in defining domains with malicious intent compared to blacklisting services.

Transitive attribution.

To assess the prevalence of incomplete blacklists and not-active malicious domains, we examine the registrant data of false positives in order to find undeniable traces that connect them to malicious domains. We base this transitive attribution on phone numbers as these are uniquely assigned identifiers that were never used in our campaign selection criteria. Thus, if the registrant's phone number is identical to that of a blacklisted registration, we consider the domain name to be part of the malicious campaign and assume that it has either not been abused yet, or was not picked up by a blacklist. In total, 3,252 campaign domains are transitively considered as malicious. As shown in yellow in Figure 3.5, 14 of the 20 identified campaigns are thereby completely validated.

A threat to using phone numbers to identify malicious registrants arises when an attacker retrieves the WHOIS information of a legitimate .eu domain and falsely uses it for his own registration. With three small experiments, we try to invalidate the presence of this scenario in the transitive attributed set. Firstly, we measure the time interval between the registration time of a transitively attributed domain and of the blacklisted domain that it was associated with. We find that for 2,058 domains, the malicious registration (with the same phone number) occurred within 60 seconds of the transitively attributed registration. We argue that it is virtually impossible for an attacker to observe a new registration (which is non-public information in the .eu zone), query its WHOIS data and subsequently make a similar registration in that time interval. In a second experiment, we argue that an attacker would not exploit a benign registrant's information if those contact details are already tainted. In that regard, we filter out 965 of the remaining domains that were registered after a prior registration with the same phone number was already blacklisted. Lastly, we consult a phone number verification tool [192] and identify invalid phone numbers for 189 of the 229 remaining domains. We presume that a malicious actor would not steal benign registrant details with an invalid phone number while attempting to mimic a legitimate registrant. In the end, we observe one of these three indicators for 3,212 (98.77%) of the transitively attributed domains and conclude that this attribution is justified.

In-depth analysis of campaign c_15.

After the transitive attribution step, still 30.6% of the registrations in campaign c_15 remain potential false positives. This set of domain names is further investigated.

Within campaign c_15, all domain names are composed of concatenated Dutch words (mostly 2 words, but sometimes up to 4). The same words are frequently reused, indicating that a limited dictionary was used to generate the domain names. The remaining 583 potential false positives domain names were split up by a native Dutch speaker in segments of existing words. 396 of these unflagged domain names turned out to be exclusively constructed out of Dutch words used in blacklisted domains of the campaign. As this is a very specific pattern, these domains have been labeled as *validated true positive*. The remaining domain names had either one word segment in common (172 domains) or no common word at all (15 domains). Thereafter, a new iteration of transitive attribution strategy was applied on that remaining set. Hereby, 147 registrations shared a phone number with the previously validated registrations, reducing the potential false positives to just 40 registrations.

Interstingly, we find that 95 out of the 98 registrant names that are used in c_15 can be generated with one of the forks of the Laravel Faker generator tool [149] using the nl-NL language option.

Manual analysis of the remaining false positives.

After the transitive attribution and the analysis of c_15, the residual potential false positives in all campaigns were further investigated manually, by querying DNS records, visiting websites, and searching on blacklists (e.g. *URLVoid* [193]) and search engines. Only two additional domains could be validated as true positives: one registration in campaign c_04 was identified as a phishing website by FortiGuard [69], and one registration in campaign c_15 sent out unsolicited to a temporary email account on *email-fake.com.*

Summary of validation.

Of the 20,698 campaign registrations, 16,704 domains (80.73%) were flagged by blacklisting services, 3,252 registrations (15.71%) were linked to malicious domains via transitive attribution, and 552 (2.67%) have been manually validated as registered with malicious intent.

To conclude, the campaign selection criteria resulted in only 190 potential false positives (i.e. 0.92%). This is a strong indicator that the selection criteria are sufficiently accurate to perform a representative analysis and to give us the necessary insights into the malicious domain ecosystem.

3.4 Insights into malicious campaigns

In this section, we discuss several interesting observations regarding malicious campaigns, found during our assessment.

Abuse indicators and categories

Overall, the vast majority of blacklisted domains (93.68%) were associated with spam domains. As listed in Table 3.2, all campaigns follow this general distribution, except for c_19 where nearly 28% is linked to botnet C&C servers.

Spamhaus DBL and SURBL are the two abuse sources that cover the largest number of domains. While there is a considerable overlap, both are required to get an exhaustive coverage of all campaigns. In particular, c_1 and c_19

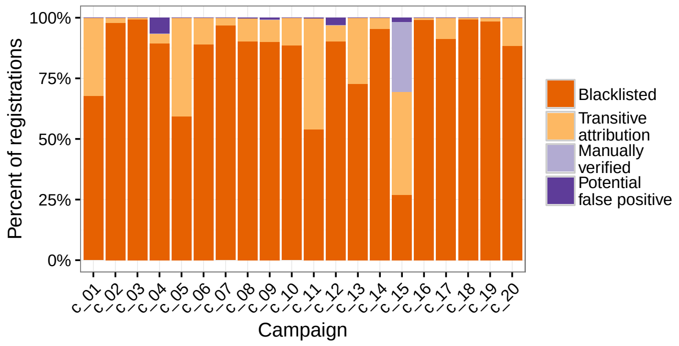


Figure 3.5: Extended false positive analysis of each campaign.

are exclusively flagged by just one of the two sources. Interestingly, Google Safe Browsing was not involved in flagging domains in any of the campaigns. Presumably, Safe Browsing focuses more on malware delivery, as opposed to malicious infrastructure.

Cross-campaign characteristics.

Some interesting characteristics exist across multiple campaigns. For instance, c_03, c_04 and c_20 generate the registrant's email address from its name followed with a numerical suffix. Similarly, the registered domain names in c_05 and c_11 follow clear character patterns with numerical suffixes. Another returning peculiarity is the discrepancy between the registrant's street address and his country. c_07, c_9, c_13 and c_14 use valid street addresses located outside of Europe (US and Panama) in combination with a European country (Norway, Ireland and others). Presumably, this is to partly confuse the residential requirements for registering a .eu domain. In the case of c_10, a fixed street address is listed throughout the campaign while 10 different countries are combined with it.

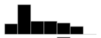
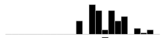
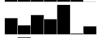
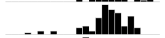
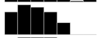
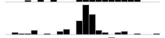
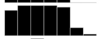
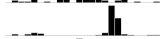
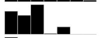
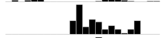
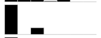
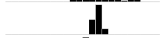
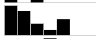
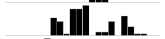
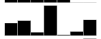
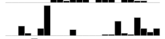
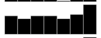
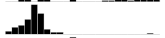
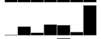
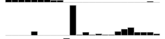
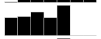
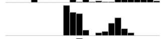
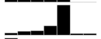
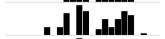
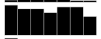
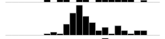
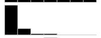
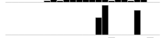
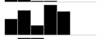
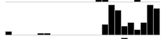
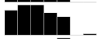
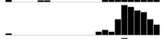
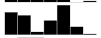
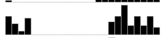
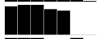
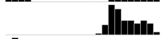
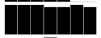
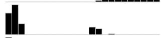
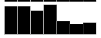
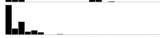
Registration process is not fully automated.

While performing the in-depth analysis of the malicious domain registrations, we found multiple indications that the malicious registration process in (at least some of) the campaigns is not fully automated: syndicates work along office hours and make human errors while registering domains.

Office hours and holiday breaks. As expected, the overall registrations in the .eu zone follow a weekly pattern. Figure 3.6 demonstrates this by zooming into 1 month of registrations. During weekends, a significantly smaller amount of registrations occurs than during the week. On average, week days have 2.34 times more registrations than weekend days. For blacklisted registrations, the difference is even more prominent. During weekend days, 3.85 times less malicious registrations occur as compared to weekdays. Moreover, several weekend days have no blacklisted registrations at all. Table 3.2 displays this behavior separately for each campaign.

As already mentioned in Section 3.2, the distribution over time of malicious registrations is much more fluctuating than those of all registrations (Figure 3.1). Interestingly, the longer drops in malicious registration activity coincide with holiday periods. The most significant one starts at the first week of July

Table 3.2: The different types of abuse, the blacklists and registration timing patterns per campaign. A small fraction of blacklisted domains has a missing abuse type. The max. burst represents the highest number of registrations that occurred within a 60-second time span.

	Abuse types					Blacklist sources			Registration timing patterns		
Campaign	Spam	Botnet	Malware	Phishing	Unwanted	Spamhaus	SURBL	Google SB	Day of week (Mon-Sun)	Hour of day (00-23h)	Max. burst
c_01	100.00%						100.00%				99
c_02	100.00%					100.00%	27.53%				59
c_03	100.00%					99.48%	86.82%				51
c_04	99.88%		0.12%	1.38%		99.64%	76.26%				28
c_05	83.05%					12.99%	77.97%				9
c_06	100.00%					87.63%	12.37%				3
c_07	91.40%					91.40%	1.08%				10
c_08	100.00%					100.00%	3.70%				19
c_09	99.63%		0.12%	1.97%		99.26%	28.45%				46
c_10	99.20%			1.60%		78.40%	90.40%				48
c_11	85.18%		0.08%			16.00%	77.02%				59
c_12	99.59%			0.20%		99.39%	74.29%				23
c_13	96.75%					81.82%	19.48%				1
c_14	100.00%					84.43%	86.05%				132
c_15	97.28%					73.35%	33.46%				13
c_16	100.00%			0.12%		100.00%	43.71%				8
c_17	100.00%					100.00%	8.83%				18
c_18	99.85%			0.15%		99.77%	28.04%				10
c_19	72.07%	27.93%				100.00%					5
c_20	99.29%		0.96%			99.14%	7.58%				19
All malicious	93.68%	1.27%	0.85%	3.22%	0.57%	81.07%	50.04%	1.81%			

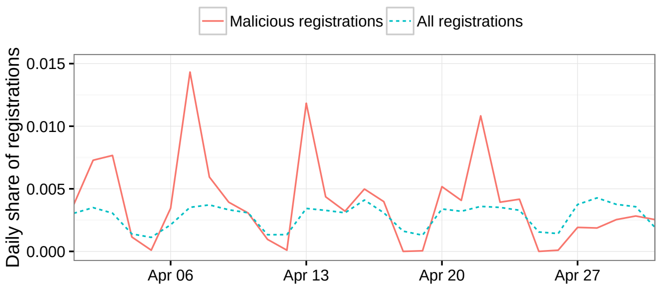


Figure 3.6: Daily share of all and malicious registrations between April 1, 2015 and April 20, 2015.

and continues for several weeks, concurring with the summer holidays. The other major periods of recess correspond to Labour day weekend (May 1), the Christmas holidays (last week of December) and the beginning of Lent or Carnival (mid-February).

There are multiple hypotheses to explain these registration patterns:

1. Malicious actors might deliberately mimic normal registration patterns to avoid detection.
2. There might be a lower demand for new malicious domains during holidays, when potential victims are less active online.
3. Cybercriminal activities could be managed as any other business and are therefore equally susceptible to vacation periods.

To substantiate the latter hypothesis, we also zoomed in into the variation in registration time per campaign. Interestingly, as shown in Table 3.2 displays this separately for each campaign, we identified that some of the campaigns clearly align with a typical day at the office. For instance, in campaign c_11 and c_18 syndicates are working 8 to 10 hours a day, and the daily pattern of c_11 even suggests that there is sufficient time to take a lunch break. In contrast, the daily registration pattern of campaign c_19, further illustrated in Figure 3.7, hints at a more automated process. The vast majority of registrations are made daily at midnight and 1 PM. Furthermore, campaigns such as c_14 are registering at a rate up to 132 new domains per minute, suggesting underlying automation.

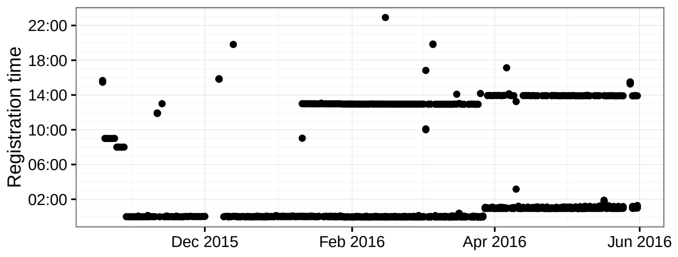


Figure 3.7: Times of registrations for campaign c_19. Note the impact of daylight saving time starting from the last Sunday of March.

Table 3.3: Minor inconsistencies found in the registration details campaign domains. Some registration details have been obfuscated for privacy reasons.

Attribute	Inconsistencies	
c_04 street	P.O BOX 3...4	P.O BOX 3...4,
c_11 city	AIX EN PROVENCE	AIX-EN-PROVENCE
c_11 street	1... ROUTE D AVIGNON	1... ROUTE D'AVIGNON
c_16 street	947 C...R	9457 C...R

Minor inconsistencies in the data. We observe a number of inconsistencies in several registration details of certain campaigns. These inconsistencies could be the consequence of small errors or typos, suggesting that some of the data has been manually entered into scripts or registration forms, or that different input validation rules have been applied by registrars or resellers.

As listed in Table 3.3, we encounter a few cases where registration fields belonging to the same registrant vary typographically inside a single campaign.

In campaign c_15, we also observed registrant names for which the name field has been filled inconsistently, leading to name patterns such as *Lastname Lastname* or *Firstname Firstname Lastname.*

Adaptive registration strategies.

Several campaigns alter their strategies throughout their lifetime. For instance, five campaigns have registered domains via multiple registrars: c_01, c_03, c_11, c_12 and c_16. Figure 3.8 illustrates how campaign c_11 sequentially changes between 4 registrars over the entire duration of the campaign. Malicious actors might change registrars for economic reasons (cheaper domain registrations) or to evade detection. Alternatively, the change in registrar can be triggered by an intermediate reseller that changes registrar.

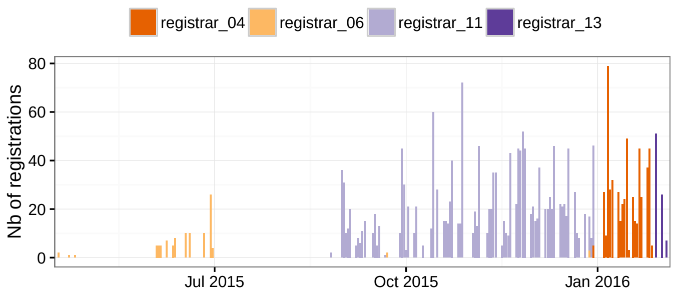


Figure 3.8: Registrations per day and per registrar of campaign c_11.

Table 3.4 lists for each campaign the amount of adaptive registration details that were used throughout its lifespan. While five campaigns use just a single phone number and email address, the large majority leverage multiple registration details. The email providers that are categorized as "Campaign" indicate that a domain name that was registered as part of the campaign, was later used as the email provider for a new registration.

As primary indicators of evasion sophistication, we list two metrics. Firstly, we give the maximum number of domains for which a campaign has reused a single phone number or email address. Secondly, we measure the longest period during which a registrant's phone or email address has been reused. c_15, c_12 and c_8 demonstrate the highest sophistication in terms of minimizing the reuse of registrant details. However, c_15 uses many different self-registered email providers and only reuses details sparsely over a long period. In other words, they leverage a more elaborate strategy than c_12 and c_8, where registrant details seem to be automatically generated in a hit-and-run fashion The success of c_15's strategy is supported by its low blacklist presence. In contrast, c_2, c_11 and c_18 deploy exhibit more simple and high-volume strategies.

Related campaigns.

By searching for overlaps between campaigns in their registrants' details, as well as temporal characteristics (simultaneous or chained activity), we have identified that several campaigns are likely related to each other:

- c_02 and c_03 have registrants with the same phone number
- c_08 and c_12 have registrants with the same phone number, email and address
- c_16 and c_18 have registrants with the same address

Table 3.4: The amount of registrars, phone numbers, email addresses and types of email providers used per campaign.

		Campaign																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	Nb of registrars	3	1	2	1	1	1	1	1	1	1	4	2	1	1	1	3	1	1	1	1
	Nb of phones	4	3	19	54	1	2	1	29	14	1	2	29	1	1	97	8	1	4	1	13
	Max domains per phone	338	1026	385	169	177	158	93	20	590	125	1220	24	154	989	16	372	283	1265	752	237
	Max phone usage (days)	90	71	69	276	129	1	359	2	155	204	246	15	307	41	232	147	50	75	226	35
	Nb of email addresses	6	18	71	54	177	2	1	29	13	1	2	29	29	1	98	8	1	4	1	14
	Max domains per email	263	103	68	169	1	158	93	20	590	125	1240	24	126	989	16	373	283	1265	752	237
	Max email usage (days)	50	8	14	267	–	1	359	2	155	204	157	15	255	41	232	147	50	75	226	35
Email Providers	Public	–	1	1	2	–	–	–	6	1	–	–	1	–	1	–	3	1	1	1	1
	Private	5	–	–	–	–	2	1	–	–	1	1	–	1	–	–	–	–	–	–	–
	Campaign	–	–	–	–	–	–	–	–	–	–	–	–	28	–	98	–	–	–	–	–
	WHOIS privacy	–	–	–	–	1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Similarly, the abrupt ending of campaigns c_01, c_02 and c_03 suggest that these campaigns might be of the same actor, or depend on the same reseller or registrar that ended their service.

Most active malicious actors.

Table 3.5 gives the highest represented malicious registrars, registrants and email providers in our dataset. Most surprisingly, 49.6% of all the malicious domain names are registered with one single registrar. Furthermore, it used by half of all the campaigns we identified. We argue that this registrar is either very *flexible* in accepting registrations, or has the most interesting price setting for bulk registrants. Note that this registrar only accounts for 2.27% of all benign registrations. This observation confirms earlier findings in [114, 83] that a handful of registrars accounts for the majority of spammer domains.

The most used malicious email providers are all popular public webmail providers. The situation is different compared to the registrars as gmail.com has the largest share of malicious registrations but also well-represented in benign registrations. In contrast, aol.com and yahoo.com do have a large fraction of malicious registrations.

Over 3,000 malicious registrations can be attributed to just 3 registrants who are predominantly malicious. Related to the reasoning in Section 3.3.3, we suspect that non-blacklisted registrations of these registrants are likely malicious as well.

3.5 Automating campaign identification

In the previous section, we discussed a large-scale experiment in which we manually identified large campaigns from a corpus of malicious registrations. The criteria that defined these campaigns were mainly recurring registrant and nameserver details. In this section, we use that knowledge to automate the campaign identification process by using a clustering algorithm. The results serve to both validate the manual experiment, as well as to demonstrate the capabilities of automatic campaign identification to aid ecosystem analyses in TLDs.

Table 3.5: Top 3 most malicious registrars, email providers and registrants. For each entry, we list their contribution to all malicious and benign registrations, their toxicity and the campaigns that are associated with them. The toxicity expresses the percentage of malicious registrations within that entity.

	Nb of	Contribution			
	malicious	Malicious	Benign	Toxicity	Associated campaigns
1. registrar_5	10,353	49.61%	2.27%	36.25%	1 2 3 4 9 10 12 13 14 17
2. registrar_3	3,004	14.39%	2.64%	12.41%	3 7 8 12 16 18
3. registrar_7	2,327	11.15%	0.46%	38.67%	20
1. gmail.com	4,221	20.23%	24.79%	2.08%	4 14 19
2. yahoo.com	3,348	16.04%	1.49%	21.85%	2 3 4 8 20
3. aol.com	2,134	10.23%	0.31%	46.28%	8 9
1. m...s@c...k.com	1,265	6.06%	0.00%	99.37%	18
2. abuse@j...n.com	1,240	5.94%	0.12%	54.89%	11
3. n...t@gmail.com	989	4.74%	0.01%	95.37%	14

3.5.1 Clustering process

Algorithm.

Agglomerative clustering is chosen as the basis to perform automatic campaign identification. It is a hierarchical clustering algorithm that works by iteratively merging the two clusters that are closest to each other [84]. We adopt the complete linkage criterion to determine the distance between clusters. Using this criterion, the distance is equal to that of the most dissimilar instances of both clusters, promoting a high density. There are two main reasons for opting for agglomerative clustering.

1. The algorithm does not require a predetermined number of clusters, allowing us to statistically evaluate the optimal number of clusters afterwards.
2. Given the results from Section 3.3, we presume that about 80% of malicious domains can be grouped into clusters. Agglomerative clustering allows the remaining independent domains to have their own singleton cluster, without necessarily polluting the large clusters.

Feature set.

For each of the 20,870 blacklisted registrations, we extract 13 features. There are two general **registration features**, *domain length* and *registrar.* Next, we have ten **registrant features**: *name, street, city, region, country, zip code, phone number, email account* and *email provider.* Lastly, two **nameserver features** were included, the *nameserver domain names* and their *geographical location.*

Agglomerative clustering uses the Euclidean distance measure to calculate the distance between two instances. However, except for *domain length* and *address score*, all features in our set are categorical, not numeric. In order to accommodate these features, we apply one-hot encoding [171]: for each possible category in our set, a new binary feature is created. Each instance that carried that value will receive a value of 1 in the new binary encoded feature, all others are set to 0. Naturally, one-hot encoding dramatically increases the number of features, more specifically from 13 to 30,843.

Cutoff selection.

Agglomerative clustering has no predefined stopping criteria and merges clusters until only one remains. Using the campaign labels from the manual analysis in Section 3.3, we calculate the *V-measure* after each merging step to statistically express the mapping between clusters and campaigns. The V-measure is the harmonic mean of the *homogeneity* and *completeness score* [172]. The former is a metric that represents how homogeneous each cluster is in terms of campaign labels, the latter measures whether the instances of a certain label are all assigned to the same cluster[6]. The highest V-measure is observed at a cutoff of 432 clusters, where the homogeneity is 0.90 and the completeness score 0.86.

3.5.2 Results

In the selected model, very large clusters have formed. Namely, 80% of domains reside in the 39 largest clusters, while a long tail of 227 clusters consisting out of only 5 registrations or less. In other words, the clustering algorithm forms a Pareto distribution similar to the manual campaign identification in Section 3.4. Furthermore, the top 30 clusters represent 91.48% of blacklisted registrations that reside within the 20 manually identified campaigns.

Using Figure 3.9, we analyze the top 30 largest clusters and their correlation to the campaign labels from the manual analysis. The clustering algorithm largely aligns with the manual campaign identification, with most clusters mapping to a single campaign. The notable exceptions being the two largest clusters. The first cluster encompasses 2,052 domains of both c_02 and c_03. This is in line with our previous speculation (Section 3.4) that c_02 and c_03 are related given their synchronized ending and the fact that they share registrants with the same phone number. The same is true for the second cluster, as both c_16 and c_18 clearly share registrants with the same address.

Cluster 16 is the only automatically identified cluster that solely exists out of domains without campaign labels. When inspecting those domains, we find that this cluster is likely related to or part of c_20. More specifically, their active days align and the same registrar is used for all registrations in both sets, as shown in the bottom part of Figure 3.10.

Several clusters also contain a small amount of instances without a campaign label. We distinguish two cases: instances that closely align to a campaign, but were not selected because of too narrow selection criteria; and instances that

[6]For instances without campaign labels, the registrant's phone numbers are set as their label.

have no campaign affinity, but are most probably merged because the clustering algorithm has executed too many merges. The former are labeled as (Related) in Figure 3.9, the latter as (Unrelated).

18 of the 20 manually identified campaigns are represented in the top 30 clusters. The smallest identified campaign, c_07, is not found in this subset of clusters because it is simply too small. Cluster 30 contains 110 domains, more than c_07 encompasses as a whole. However, we find that c_07 is completely and homogeneously represented by the 35th cluster. The second campaign that is missing is c_15. As mentioned in Section 3.3.3, this campaign was selected by a unique and complex address formatting pattern. Since the clustering algorithm only performs binary matches on these fields, it is less effective at detecting these more advanced similarities. As shown in the top part of Figure 3.10, c_15 is spread out over 18 clusters, that essentially represent 18 different registrants that are reused throughout the campaign. The affinity between those clusters is clear when considering their active days.

In conclusion, the manual and automatic campaign identification results align to a large extent. We find that, when performing automatic detection using clustering, we achieve a more exhaustive identification of clear similarities as opposed to manual identification (e.g. cluster 16). However, the automatic approach has difficulties to detect more advanced similarity patterns (e.g. c_15). In future work, more sophisticated techniques, such as n-grams, can be integrated into the clustering algorithm to detect more advanced similarity patterns.

In general, the outcome of the clustering algorithm both validates the approach of the manual analysis, as well as demonstrates the capabilities of automatic and reproducible campaign identification using registrant and nameserver details.

3.6 Discussion and limitations

In this section, we want to discuss the relevance and applications, as well as the limitations of our study.

Applications.

Given the exploratory nature of this research, we anticipate several applications and next steps.

The relevance of this work is not limited to .eu domains. Presumably, malicious actors do not restrict their potential to a single TLD. Furthermore, bulk

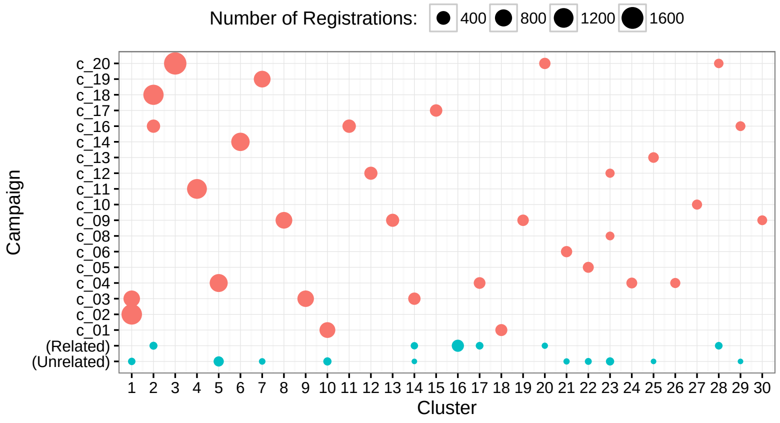


Figure 3.9: Mapping of the top 30 clusters to campaign instances. The bottom two rows represent domains without a campaign label: the (Related) row groups the registrations that closely align with campaigns, the (Unrelated) groups registrations without campaign affinity. The clusters are ranked from large to small.

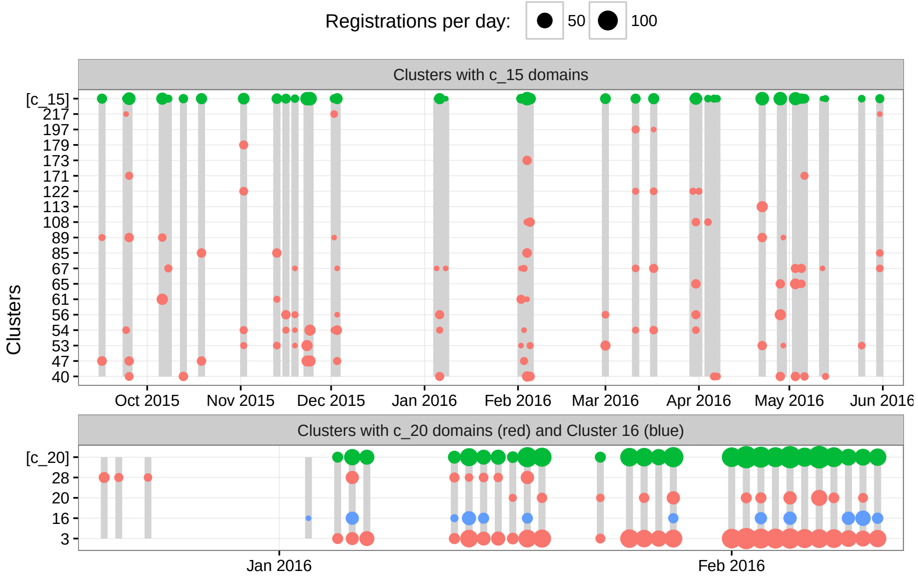


Figure 3.10: Related temporal activity of clusters highlighted in gray. **Top**: Domains of campaign c_15 are spread over many clusters. **Bottom**: Cluster 16 maps to clusters of c_20 domains.

registrations can be made across multiple TLDs using the same registrar. Therefore, the findings and methods described in this work can most likely be applied to other or across TLDs. To reproduce this study, access to registrant and nameserver details of registrations is required. This data can generally be obtained by downloading zone files and WHOIS data.

Additionally, we demonstrate that automatic campaign identification using clustering is a feasible strategy. Moreover, 18.38% of registrations in the identified campaigns are not present on blacklists. This entices interesting opportunities to extend the coverage of blacklists. Although the proposed system relies on a post-factum analysis, it could create opportunities to stop ongoing campaigns.

Limitations.

We note four limitations and potential validity threats.

Firstly, the main subjects of this research are domain names that are registered with malicious intent. However, backlists also contain legitimate registrations that have been compromised later on. We argue that the prevalence of these cases is minimal, since 98.57% of blacklisted registrations were already flagged within the first 30 days of registration. Furthermore, compromised benign domains would appear as outliers in our data and could thus hardly pollute campaign analyses.

Secondly, both the manual and automatic identification rely on patterns in the registration data. Malicious actors can leverage this dependency by constantly using different registration data and patterns. However, the cost for attackers would increase to achieve this higher level of circumvention. Specifically, certain registration details are more costly to obtain. For instance, frequently using different email and name server domains requires the registration or compromise of additional domain names. As another example, using different registrars necessitates the creation of new registrar accounts and new registration automating programs. Ultimately, it is also hard not to exhibit any patterns when performing bulk registrations (timing patterns, fake identity generating tools,...).

Additionally, several registrars offer anonymization services to their customers, obscuring the registrant contact information to the registry. Evidently, this diminishes the ability to differentiate between registrations and conceals information that can be used to identify domains registered by the same entity. In the case of .eu, the use of such obfuscation services is not allowed by the registry's terms and conditions. During our analysis, we find that such services were only deployed by c_05 which could have impacted this campaign.

Finally, our research is based on a set of publicly available blacklists that are, at least to some extent, incomplete. A more complete ground truth would likely improve the performance of our approach.

3.7 Related work

Prior to our research, Hao et al. [83] studied the domain registration behavior of spammers. They reported that most spam domains are very short-lived. More specifically, 60% of these domains were active for only a single day. Spammers are registering many "single-shot" domains to minimize interference by blacklists. To counter this strategy, the authors explore various features on which spam domains exhibit distinctive behavior. For instance, in contrast with benign registrations, they find that malicious domains are more often registered in batches. Recently, Hao et al. implemented many features discussed in that prior work to create a machine learning-based predictor capable of detecting malicious domains at time-of-registration [82]. The three most dominant features of their classifier are authoritative nameservers, trigrams in domain names and the IP addresses of nameservers.

While both papers approach malicious domains as a two-class problem (benign vs. malicious registrations), many of their features essentially depend on returning characteristics of different underlying malicious campaigns. In this work, we are the first to shift the focus to the campaigns itself, exploring their modus operandi and different identifying characteristics.

A method related to ours was proposed by Felegyhzi et al. [67], who investigated the feasibility of proactive domain blacklisting, by inferring other malicious registrations from known-bad domains through shared nameservers and identical registration times. The proposed system shortens the time required to blacklist malicious domains, while providing important insights regarding the similarities of registrations within campaigns. Additionally, Cova et al. [45] indentified different rogue antivirus campaigns by looking at the hosting infrastructure and registration details (including the registrant's email) of different domains.

Related studies concentrate on DNS traffic of newly registered domains to characterize malicious behaviour [22, 81, 17, 18, 132]. These systems mainly focus on the initial operational DNS patterns of domain names.

Other important efforts regarding malicious domains come from the study of domain generation algorithms (DGAs). Recent work by Plohmann et al. [153]

demonstrates the increasing importance of understanding DGAs to thwart C&C communication. Using reimplementations of these algorithms, the authors execute forward generation of domain lists, which enables proactive identification of C&C domains.

3.8 Conclusion

In this study, we analyzed the maliciously-flagged .eu registrations over a 14-month period. This research is the first to extensively dissect the underbelly of malicious registrations using registrant details to identify its operational components, namely campaigns. We explored the ecosystem and modus operandi of elaborate malicious actors that register massive amounts of domains for short-lived, malicious use.

By searching for shared characteristics, we established that at least 80.04% of all malicious registrations can be attributed to 20 campaigns with varying duration and intensity. Moreover, the coverage of blacklists can be extended by 19.30%, using the information from the campaign identification. After a rigorous evaluation, we are able to confirm that the vast majority of these previously undetected registrations are genuinely related to malicious activity; at most 0.92% are false positive registrations.

Our study demonstrates the potential to leverage the registrant details and other registration characteristics to identify large campaigns. Aided by an automatic identification process, this insight can be used to easily track and interfere with massive, long-running campaigns and to preemptively extend blacklists with malicious domains that have yet to be actively used by a cybercriminal.

Chapter 4

Domain Parking

Preamble

This chapter presents a study dedicated to security issues related to the *domain parking* industry. A domain can be "parked" when the owner is not actively using it. Often, domain owners employ domain parking services to take care of hosting their idle domain. Typically, these services render contentless pages filled with ads with the aim of generating a passive revenue stream for the domain owner.

Domain parking became a subject of interest to the security community as researchers repeatedly encountered parked domains when studying the behavior and infrastructure of other cybercriminal activities. One prominent example is cybersquatting, as multiple analyses identified that domain parking is the most prevalent strategy to monetize trademark infringing domain names [8, 185]. Other studies found that malware-related domains generated by DGAs [153] and domains used by malicious traffic distribution systems [117] frequently employed domain parking services. Additionally, the domain parking business itself became known for fraudulent practices. One study gathered evidence of click fraud schemes, exploiting both the domain owners as well as their advertisers [16]

The work in this chapter shifted the focus on the abuse against users landing on parked domains and the damage done to trademark owners. The study gathers an understanding of the reluctant role that the domain parking industry plays in the cybercriminal ecosystems. Together with the study, we presented a machine learning model that accurately detects parked pages. After publication, our model and data was freely shared with, amongst others, HubSpot: a company that creates inbound marketing and sales software. HubSpot wanted to develop a model similar to ours to filter out parked pages from their web scraping data.

Several findings presented in this chapter have, in the meanwhile, been followed-up by new developments. For instance, we reported that 6% of the parked domains that were abusing trademarks, showed ads for competitors of that trademark. One year later, counteractions for this sort of abusive scenario received a juridical precedent when *Enterprise*, a car rental company, won a UDRP against the owner of MyEnterprise.com [14]. Although “enterprise” can be regarded as a non-infringing generic word, the domain name was parked and its web page displayed ads related to car rental companies. Although it was the parking service that generated these ads (most likely with an automatic keyword matching algorithm), the owner was still held responsible for the content on his domain. Consequently, he lost the domain to *Enterprise Holdings Inc* and now MyEnterprise.com simply redirects to Enterprise.com.

Our study also reported on *technical support scams*, specifically triggered by malvertising present on parked domains. Afterwards, this phenomenon was further systematically studied by Miramirkhani et al. [125], taking parked domains as their main source for tracking down this kind of scams.

The further contents of this chapter are replicated from the paper titled “Parking Sensors: Analyzing and Detecting Parked Domains” [198], which was published in the proceedings of the 22nd Network and Distributed System Security Symposium (NDSS) in 2015. This work was done with the collaboration of other authors from KU Leuven. Thomas Vissers is the lead author of this paper. Later, the results of this study were also summarized in a blog post on *Internet Society* [9].

Parking Sensors: Analyzing and Detecting Parked Domains

Abstract

A parked domain is an undeveloped domain which has no content other than automatically computed advertising banners and links, used to generate profit. Despite the apparent popularity of this practice, little is known about parked domains and domain parking services that assist domain owners in parking and monetizing unused domains.

This chapter presents and in-depth exploration of the ecosystem of domain parking services from a security point of view, focusing mostly on the consequences for everyday users who land on parked pages. By collecting data from over 8 million parked domains, we are able to map out the entities that constitute the ecosystem, thus allowing us to analyze the domain owners, parking services, and advertisement syndicators involved. We show that users who land on parked websites are exposed to malware, inappropriate content, and elaborate scams, such as fake antivirus warnings and costly remote "technicians". At the same time, we find a significant number of parked domains to be abusing popular names and trademarks through typosquatting and through domain names confusingly similar to authoritative ones. Given the extent of observed abuse, we propose a set of features that are representative of parked pages and build a robust client-side classifier which achieves high accuracy with a negligible percentage of false positives.

4.1 Introduction

Up until twenty years ago, domain names were available for free on a first-come, first-serve basis. Network Solutions, the company that was contracted by the US Defense Information Systems Agency to operate the DNS registry, was imposing at the time a one domain for each person/company limitation. Because of the high demand, in 1995, Network Solutions switched to a paying model which has been preserved till today.

Apart from the rapid expansion of the web due to the fact that every person and company desired to have an online presence, the high demand for domain names was also due to people buying large amounts of domains in bulk and creating domain portfolios. These people, later called “domainers”, had no interest in setting up companies and using their purchased domains. They had, instead, realized that domains would soon become a very valuable commodity. As such, they bought hundreds or thousands of domains under the premise that real companies would later pay them a large amount of money in exchange for a desirable domain from their extensive portfolios. Popular successful examples of this strategy include domains like `wine.com` and `casino.com` both sold for millions of dollars. Even with the advent of new Top Level Domains (TLDs), short and generic domains are still occasionally sold for exorbitant prices [211].

Initially, the owners of large domain portfolios did little more than wait until an interested buyer contacted them for one of their domains. Eventually, some people realized that instead of just idly waiting, domain portfolios could be put to better use. These people struck deals with other websites and included banners and “favorite links” for a flat monthly fee [99]. At the same time, however, a new type of advertising started appearing on the web, namely, Pay-Per-Click advertising. As the name implies, in this new scheme, a publisher would only get paid by the advertiser if a user would click on one of the ads.

Undeveloped domains were a natural fit for this type of ads. Services started appearing that made it easy for domain owners to incorporate ads on their domains without worrying about finding advertisers and setting up contracts. These services were called *domain parking* services, since domain owners would simply “park” their domains at these service providers, and then the rest would be done automatically. Domain parking was so successful that owners of large domain portfolios stopped worrying about selling their domains and instead started making profit simply because of the commission they got from the ads that were clicked [99].

Despite of the popularity of the domain parking phenomenon, domain parking services have received little attention from the security community. Up until recently, these services were only mentioned in papers studying cybersquatting, showing that domain parking is, by far, the most popular monetization strategy for cybersquatters [204, 131, 144, 185, 8, 87, 141]. In recent research, Alrwais et al. studied domain parking from the point of view of advertisers and domain owners [16]. They showed that the majority of domain parking companies employ shady practices, such as, hiding clicks from domain owners (thereby not sharing click commissions), conducting click fraud, and sending unrelated traffic to advertisers who pay for traffic with very specific demographics.

In this chapter, we study domain parking services mainly from the point of view of everyday web users and thus orthogonally to the work of Alrwais et al. [16]. We identify 15 popular parking services and retrieve a corpus of more than 8 million parked domains. Through a series of automatic and manual experiments, we identify the presence of a wide range of fraud and abuse, targeting users as well as companies. Among others, we show that typical users landing on parked pages are exposed to inappropriate ads, malware, elaborate scams involving "technicians" getting access to a user's machine and scripts that detect and bypass advertising blockers. We also challenge the stance of the most popular ad syndicator who provides the domain parking ecosystem with the necessary advertising infrastructure, while at the same time telling its users that parked pages are *spam* and are thus hidden from that syndicator's search engine.

Given the extent of the discovered abuse, we design and build a robust classifier for detecting parked domains which does not rely on hard coded signatures but on distinguishing features of parked domains with a true-positive rate of 97.9% and a false-positive rate of only 0.5%. This classifier can be straightforwardly incorporated in a browser through its extension system, and alert users whenever they land on a parked domain.

Our main contributions are:

- We perform the first thorough study of domain parking services, mapping out the entire ecosystem while focusing on the abuse affecting everyday web users.
- We show that parked pages expose the user to a series of dangers including malware, scams and inappropriate content
- We verify the presence of an unacceptably large number of typosquatting domains parked with popular domain parking services, and demonstrate the lack of control when it comes to parking an obviously typosquatting domain
- We propose a performant classifier for detecting parked pages, utilizing robust features that are telling of a website's nature

The remainder of the chapter is organized as follows. Section 4.2 describes the ecosystem of the domain parking industry. In Section 4.3, we analyze the domains parked with these services and we study several of their abusive practices. Afterwards, Section 4.4 details how we designed a classifier to detect parked pages. Thereafter, the observations and results of this research are further discussed in Section 4.5, followed by an overview of the related work in Section 4.6. Lastly, the chapter concludes in Section 4.7.

4.2 Domain Parking Ecosystem

In this section, we first describe the modus operandi of domain parking services and the various entities involved in the domain parking ecosystem. We then report on the discovery of 8 million parked domains, hosted with 15 different parking services, in an effort to identify how domain parking is used today and the extent of abuse in terms of trademark infringements, typosquatting and malicious redirections.

4.2.1 What is Domain Parking?

The ecosystem of domain parking consists of four different parties: domain owners, parking services, advertisement syndicators, and advertisers. Domain owners (or domainers) usually own a large portfolio of undeveloped domain names, which they wish to monetize. As long as the monetization strategy returns more money than the cost of owning and managing a domain portfolio, this strategy probably is financially attractive.

Parking services provide hosting and generated content for domain owners who wish to monetize their domain names. As the name suggests, the domain owners "park" their domain names with domain parking services who then manage these domains and, in return, give the domain owners a share of their profits. Parking services typically collaborate with advertisement syndicators to serve purportedly relevant pay-per-click (PPC) ads from one of their advertising clients.

The operation model of domain parking is depicted in Figure 4.1. When a domain owner selects a service to park his domains, he configures the DNS settings of the domains to use the name servers of that parking service (1). When a web user later visits that domain, a parking page is displayed with content that is dynamically generated by the parking service. In addition, the parking service includes JavaScript code from an ad syndicator on the parked page. The syndicator's code will attempt to fetch and display ads from a plethora of advertisers, based on relevant keywords derived from the domain name (2). For example, if a user visits the parked domain cheapgas.com the parked page eventually displays ads that, in principle, are relevant to "cheap gas." If a user clicks on one of these ads, he will be sent to the respective advertiser's website, through the syndicator's tracking and redirection mechanisms (3). The advertiser will pay the syndicator for the visitor, who, in turn, will pay the parking service for the delivered click. Finally, the domain owner is given his share for supplying the domain on which the click was generated (4).

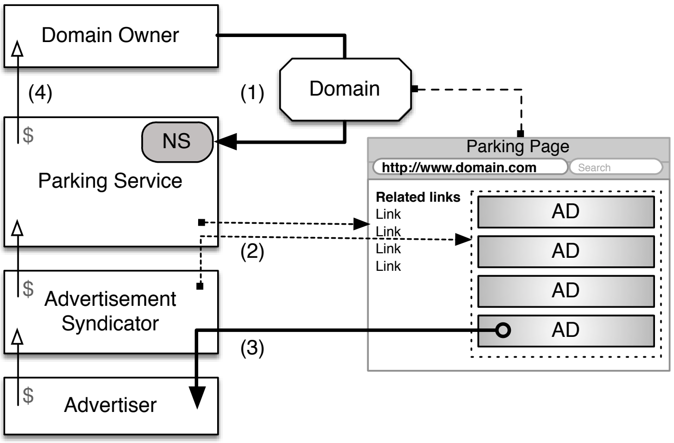


Figure 4.1: Overview of the operation model of domain parking. (1) Domain owner hosts domain with parking service. (2) Parking page is dynamically generated. (3) Visitor clicks on an ad and is sent to the advertiser's website. (4) Payment flows from the advertiser to every entity up the chain.

4.2.2 How do users end up on parked domains?

A detail that is missing from the above description is the process through which users end up on parked domains. Since parked services take, by definition, advantage of undeveloped domain names, it is unlikely that a user landed on a parked page by finding and following a link in a search engine, or on a trustworthy domain. As such, users end up on parked domains through alternative means.

While today's search engines are tightly coupled in a user's browser (either in a dedicated text field next to the address bar, or piggybacking on the address bar itself) that was not always the case. In the past, a user would have to either know about a specific domain name, or explicitly visit the website of a search engine and search for relevant content. During that time, browsing the web involved significantly more typing of domain names in a browser's URL bar than it does today.

For known sites, users would typically need to memorize the domain name of a website and manually type it, in full, in their browsers' address bars. Opportunists noticed that as users type in long URLs, there is a possibility of a typing mistake that will go by undetected, e.g., typing and requesting `wikipdeia.org` instead of `wikipedia.org`, and started registering these typo-including domain names. This practice was called *typosquatting* and, as prior research has shown, the preferred monetization strategy of typosquatters is parked domains [131, 204, 8, 185].

Next to typographical mistakes, users would also try to "guess" the domain names of websites relevant to their needs. Thus, a user who is interested in finding "cheap gas" could either visit a search engine and search for that phrase or, alternatively, concatenate the two words, append the most popular TLD and attempt to visit the `cheapgas.com` website. It is also worth noting that, at the time, browsers were trying to "help" users by appending TLDs before giving up on a domain. That is, if the user typed in `cheapgas` and the domain did not resolve, the browser would automatically append the `.com` TLD and try again [136].

In both scenarios, a user lands on a parked page by attempting to type the address of a website. The traffic resulting from this action was appropriately named "type-in" traffic and is, historically, the reason why domain parking services exist. Next to this type of traffic, researchers also recently noticed that domains belonging to malware distribution sites and C&C servers, when taken down, are usually repurposed as parked pages [117]. Thus, in addition to type-in traffic, users can land on parked pages by visiting infected sites that forward these users to a parked domain, instead of forwarding them to a browser-exploiting page

4.2.3 Gathering Parked Domains

In order to establish a representative analysis of the domain parking industry, four different sources were used to assemble a list of domain parking services: a survey amongst domainers [13], the top results of Alexa.com and Google.com when querying for *domain parking*, and a thread from a popular domaining forum [139]. Next, we select all services that are listed in more than one source, resulting in a collection of 15 services on which we focus for the rest of this chapter.

The next step is to find domains that are parked with the aforementioned domain parking services. Since it is unlikely that the services will voluntarily provide us with the domain portfolios that they manage, we must identify parked domains in an alternative way. The strategy for this task is a rather straightforward one: given any domain name, if its configuration matches the configuration of any of the 15 studied services, then that domain is indeed operated by a domain parking service.

To that extent, we registered ourselves as domain owners with each service and took note of the configuration instructions. The parking configuration of a domain usually involves the setting of DNS records to point to the parking service's servers. For the vast majority of cases, this involved pointing the appropriate NS record of your domain to their name servers. In some cases, services did not accept us as their clients. This was usually due to our domain portfolio not being "large enough". For these services, we manually identified existing parked domains and extracted the appropriate information from their DNS records. Table 4.1 summarizes the configurations we found across all services.

We started the process of collecting parked domains by searching the records of the DNS Census dataset [56], which contains about 2.5 billion DNS records gathered in 2012 and 2013. We extracted all domains that matched the configurations of Table 4.1 and subsequently queried its domain's DNS records to confirm whether they were still parked with that particular parking service. This resulted in a total of 8,064,914 actively parked domains from the 15 observed parking services. Since the DNS Census is outdated (and likely incomplete), this means that at the time of this writing, there exist *at least* 8 million domains whose sole purpose is to serve ads when they are visited.

Figure 4.2 shows the distribution of the gathered domains with each service. One can quickly notice that even though many parking services exist, only a handful of them are responsible for managing the majority of parked domains. That is, 60% of the discovered domains are parked with the three most popular services: *Sedo Parking*, *Internet Traffic* and *Cash Parking*.

4.3 Analysis of Parked Domains

In this section we study several characteristics and practices of parked domains. We start by inspecting the 8 million gathered domain names and map out their typosquatting abuse. Next, we randomly sample several thousand parked domains which we crawl automatically using PhantomJS [152], an instrumented browser, while saving the HTML of every loaded frame, logging all HTTP

service	Setting	Address
SedoParking	NS	sedoparking.com
InternetTraffic*	NS	internettraffic.com
CashParking*	NS	cashparking.com
Fabulous*	NS	fabulous.com
DomainSponsor	NS	dsredirection.com
Above[1]	NS	above.com
ParkingCrew	NS	parkingcrew.net
	A	62.116.181.25
	CNAME	parkingcrew.net
Skenzo*	NS	ztomy.com
NameDrive	NS	fastpark.net
Voodoo*	NS	voodoo.com
RookMedia	NS	rookdns.com
Bodis	NS	bodis.com
	CNAME	parking.bodis.com
DomainApps	NS	domainapps.com
TrafficZ*	NS	trafficz.com
	A	198.202.142.246
	A	198.202.143.246
TheParkingPlace	NS	pql.net
	Redirect	putoppose.net/d/domain

Table 4.1: Summary of the observed parking services together with their required domain configuration. Entries marked with an asterisk were found through external analysis.

requests and taking a screenshot. Furthermore, we collected WHOIS data for these domains. This data is then used to map out advertising networks, domain owners, trademark abuse, malicious redirects and ad-blocker detection mechanisms.

4.3.1 Parked Domain Owners

To get an understanding of parked domain ownership, we request WHOIS data for 3000 randomly-selected parked domains. We parse the WHOIS records using Ruby Whois [32] and filter out the records that are anonymized or unparseable. From the remaining 1,582 domains, we extract the registrant, administrator, and technical contact details and group together the domains that list the same name, email address and organization. As a final step, we manually merge

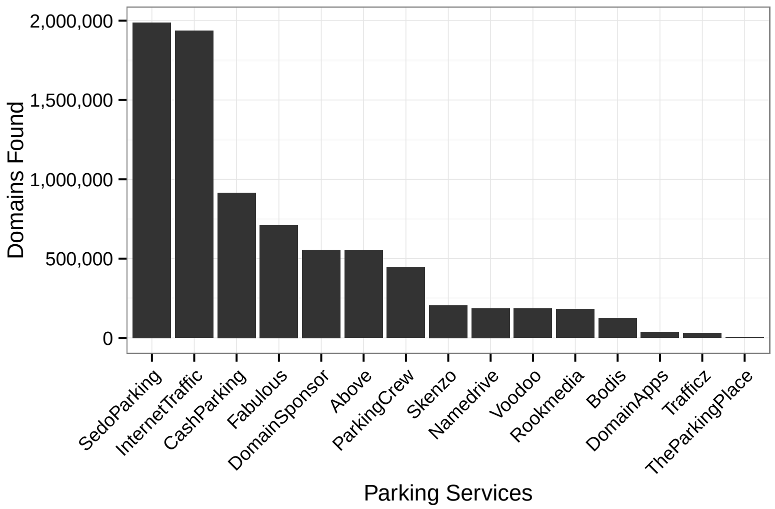


Figure 4.2: Number of found domains for the 15 observed parking services.

clusters of domains for which we are certain belong to the same individual or organization. In total, we find 910 distinct domain owners to which the 1,582 domains belong. Figure 4.3, shows that a small number of them is responsible for the majority of parked domains. For instance, 50% of domains is owned by 15.6% (142) of owners. This means that next to owners who posses a couple of domains and use domain parking services, there are individuals with very large collections of domains, all of which are registered to simply serve ads.

4.3.2 Advertisement Syndicators

Advertising is the lifeblood of domain parking services. In order to generate revenue from parked domains, parking services usually serve pay-per-click (PPC) advertisements to visitors. Every time that a user clicks on an advertisement situated on a parked page, both the domain owner, as well as the domain parking service itself are paid a small commission.

While a domain parking service could, in principle, directly accept ads from people who wish to advertise their products and services and then display them on their own parked pages, it is easier and more scalable for them to use existing advertising infrastructure of third-party ad syndicators.

The integration of these syndicators involves little more than the inclusion of remote JavaScript libraries from the servers of the syndicator. These libraries

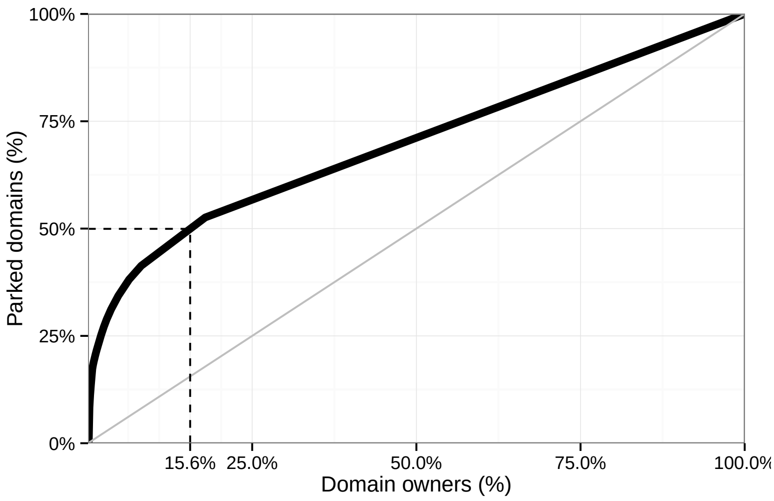


Figure 4.3: The percentage of distinct owners and their cumulative share in possession of parked domains.

are responsible for assessing the content of the page, fetch and display ads from other third-party servers, deliver the user who clicks on an ad to the advertiser who paid for that ad and register that action so that the publishing site (in this case the domain parking service) can receive the appropriate commission.

In search for these syndicators, we sampled 3,000 parked domains, crawled them, and inspected their source code for ad-related remote JavaScript inclusions. In total, we found only four advertisement syndicators that provided ad-related JavaScript code, *Adsense, Doubleclick, Media.net* and *Chango*, as shown in Figure 4.4. Their aggregate presence reaches 91%±1% of the parked websites. The other 9%±1% were either redirects (7%±0.9%) (described later in Section 4.3.6) or had no identifiable advertising code (2%±0.05%). Since domain parking services depend on advertising, we assume that the latter was due to some temporal server-side misconfiguration.

Doubleclick and Adsense, both Google products, were present in 90%±1.1% and 88%±1.2% of parked websites respectively. While, given the ubiquitous nature of Google in the modern web, this comes as no surprise, there is an interesting back story that is worth mentioning.

Google used to manage their own domain parking service, that was operating in a fashion similar to the 15 services studied in this work. In 2012, Google ceased their own hosted parking service [73], possibly due to typosquatting lawsuits such as *Vulcan Golf, LLC v. Google, Inc.* [24]. In this lawsuit, the plaintiffs

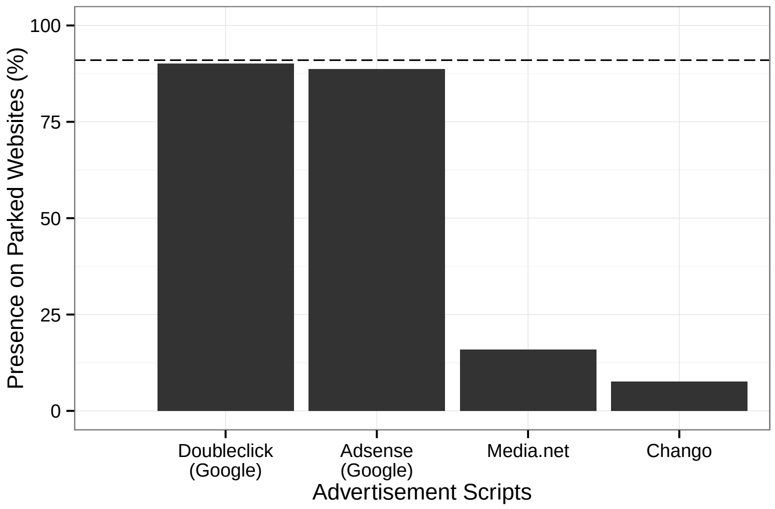


Figure 4.4: Different third-party JavaScript advertisement syndicators and their presence on parked domains. The horizontal dashed line represents the amount of websites that included at least one third-party ad syndicator

claimed the Google's parking program as a *"massive scheme to generate billions of advertising dollars through the parking of domain names that are the same as or substantially and confusingly similar to the plaintiffs' distinctive trade names or trademarks."* In addition to stopping their own domain parking products, Google is, at the time of this writing, stating on their *fighting spam* page that: *"Parked domains are placeholder sites with little unique content, so Google doesn't typically include them in search results."* [74]

One can easily spot the contradiction of these statements when compared to the pervasiveness of Google as an ad-syndicator for domain parking services. We thus find it hard to reconcile Google's decision to keep parked pages away from its search results, while still profiting by being the most popular advertising syndicator in known domain parking services.

4.3.3 Typosquatting abuse

Prior research investigating the phenomenon of typosquatting has established that the preferred monetization approach of domain squatters is the use of parking services [131, 204]. Yet typosquatting sites are illegal under the Anti-Cybersquatting Consumer Protection Act (ACPA), which prohibits registration and use of domain names that are identical or confusingly similar to a trademark.

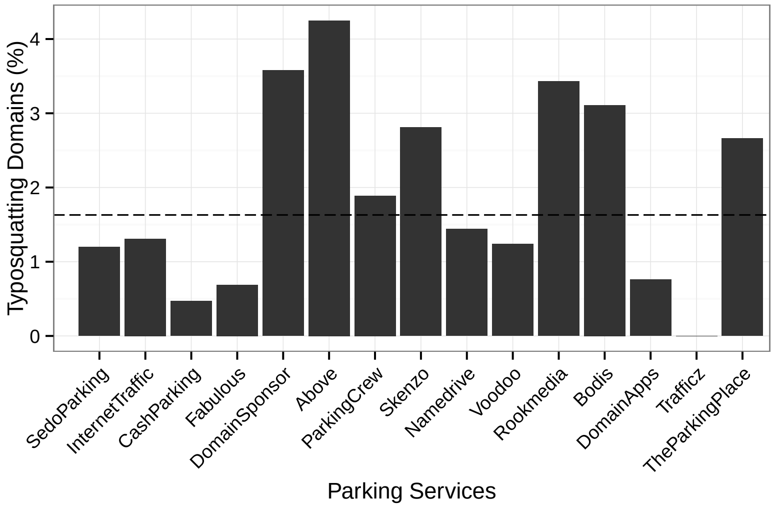


Figure 4.5: Percentage of typosquatting domains present per service. The dashed line represents the relative amount of typosquatting domains present in all 8 million domains (1.63%).

As such, a domain parking service should, in principle, always be wary of people trying to park typosquatting domains.

In order to measure the prevalence of typosquatting in the investigated parking services, we perform several "reverse-typo" transformations on the parked domains and attempt to discover whether these transformation result in an authoritative domain. We define an authoritative domain, as a domain that is ranked higher on the Alexa global list than its potential typosquatted version. Consider, for instance, the currently parked domain `vacebook.com`. This website does not belong in Alexa's top 1 million sites thus it automatically receives the lowest possible rank. One of our reverse-typo transformations will produce `facebook.com`, the domain belonging to the popular social network and ranked as the second most popular website of the Internet. Since this domain is ranked higher than the parked `vacebook.com`, we automatically mark the latter as a typosquatting domain.

Our transformation models were mainly inspired by Wang et al. [204], which introduced missing-dot, character-omission, character-permutation, adjacent character-substitution and character-insertion typo models. For some models, we had to develop the reverse operation, because of their non-bidirectional properties, such as for the missing-dot and character-insertion typo models. For the missing-dot model, we used heuristics to identify the place where a dot has to be added (e.g. add a dot in `wwwfacebook.com` *after* the “www” sequence of characters). Additionally, for the character-insertion model we defined a subset, namely, the character-duplication model, which only takes into account the accidental repetition of a character in the domain name. In order to reverse the character-duplication model, we had to identify a sequence of repeated characters and deduplicate them. Furthermore, we chose to discard the “reverse-typo” character-omission model due to high chances of false positives. Finally, in addition to the aforementioned models, we also introduced a TLD-substitution model which changes the top-level domain (TLD) with another popular one, for example from `com` to `us`.

We applied the reverse-typo transformations to the 8 million parked domains and found a total of 131,673 typosquatting domains (1.63%). The distribution, however, of these domains across parking services is anything but uniform. Figure 4.5 shows the percentage of typosquatting abuse for each studied service. *TrafficZ* is the only service with no typosquatting domains in its observed portfolio. The service with the biggest relative presence is *Above* with over 4%, followed by *DomainSponsor* and *RookMedia*. A positive finding is that the most popular parking services, as listed in Fig. 4.2 are also the services with less than average typosquatting abuse.

4.3.4 Parking a Typosquatting Domain

In the previous section we established that unfortunately, typosquatting domains are by no means rare on the investigated domain parking services. In this section, we approach the problem from the opposite side. Instead of trying to identify parked typosquatting domains, we want to quantify the “hurdles” that domain squatters have to go through in order to successfully park their domain names, i.e., we try to assess the parking services’ selectiveness in terms of excluding obviously abusive domains.

With explicit permission of the authoritative company, we register a typosquatting domain of a high profile and well known website, *stackoverflow.com*. At the time of writing (July 2014), this website is the 53rd most popular website in the world, according to Alexa. More specifically, we registered a character-permutation typo domain, namely *st<u>ca</u>koverflow.com* which we then attempted to park with each service on which we managed to register an account. During this process, we discovered that the domain name was owned by a typosquatter in the past: it was registered from August 2012 until it expired a year later. During that time, the domain was parked with four of the services under analysis. Since these services had already record of this domain name belonging to a different user, they required an extra “verification” step. The verification

	United States				Europe			
Service	Redirections	Malware	Scams	Adult	Redirections	Malware	Scams	Adult
Parking Service 1	0.4%	66.7%	-	-	-	-	-	-
Parking Service 2	1.3%	11.1%	-	-	0.4%	-	-	-
Parking Service 3	1.9%	-	-	-	2.0%	-	42.9%	21.4%
Parking Service 4	2.6%	44.4%	-	-	3.0%	-	-	38.1%
Parking Service 5	5.0%	-	-	(60.0%)	5.0%	-	-	(60.0%)
Parking Service 6	8.6%	3.3%	21.7%	-	2.6%	-	-	(50.0%)
Parking Service 7	12.4%	60.9%	1.2%	-	12.0%	-	26.2%	10.7%
Parking Service 8	19.4%	42.7%	6.6%	-	10.9%	-	26.3%	2.6%
Parking Service 9	34.6%	9.1%	2.1%	-	34.6%	0.4%	46.3%	0.8%
Parking Service 10	65.4%	21.0%	7.4%	-	66.0%	-	54.5%	27.7%

Table 4.2: The percentage of redirections that occurred per service per region. Of these redirections, the relative malicious amount is given per category: malware, scams and adult. Redirections leading to adult content that originated from adult-oriented parked domain names are put in parenthesis.

simply involved emailing the support and optionally sending a screenshot of our registrar's account details. Even with the extra attention of a verification step involving a human operator verifying our screenshots, not a single service denied the submission of this abusive domain. Every service hosted the typosquatting domain for at least a week, until we transferred it to the next one. According to the services' statistics, the parking page was receiving visitors daily.

While our experience with the process of parking a single typosquatting domain does not necessarily generalize to the parking of hundreds of abusive domains, they are in line with the findings of the aforementioned domain-squatting studies, namely, that for domain parking services, taking advantage of squatting domains appears to be part of their everyday operations.

4.3.5 Trademark abuse

While typosquatting domains belong to the trademark abuse category, not every domain that abuses trademarks is necessarily a typosquatting domain. Consider, for example, the currently parked domain `facebookonline.com`. This domain clearly abuses on Facebook's trademark, yet it would never be automatically generated by the aforementioned typosquatting models. Historically, such abusive domains have been called "cousin domains" [95] and have been often associated with phishing since it is unlikely that an everyday user of the web will think that `facebookonline.com` is *not* associated with `facebook.com`.

As done in the earlier sections, we again make use of sampling to cope with the large amount of data. That is, we randomly selected 500 parked domains, manually extracted each domain's distinct keywords (e.g. the words `facebook` and `online` for the aforementioned example) and queried a popular search engine for these keywords. If the results revealed the presence of an obviously similarly named website or organization, we mark the domain as abusive.

Out of the 500 investigated parked domains, 79 (16%±3.2%) domains were clearly abusing trademarks of existing companies and websites.

More specifically, in 44 (9%±2.5%) of the cases, the domains were found to be typosquatting, while in the remaining 35 (7%±2.2%), the domains obviously contained trademarks. The percentage of typosquatting domains is significantly larger than the one that we automatically calculated in Section 4.3.3 suggesting that our typosquatting models provided a very conservative estimate of the magnitude of the problem. The reason for this discrepancy is that the typosquatting models that we used, do not cover all typographical errors abused in the wild. For instance, in our manual sample we noticed the abuse of homonyms, e.g., `theheneryford.org` abusing the authoritative `thehenryford.org` as well as character-omission typos which were excluded from our earlier analysis.

To gain insights on what type of advertisements end up on trademark-abusing domains, we also looked closely at the ads captured by our crawlers. There, we found that 29 (6%±2%) of domains abusing existing trademarks, displayed advertisements of a competitor. This means that when a user lands on such a trademark-abusive domain, not only would the trademark holder "lose" that user's visit, but the user could potentially end up on a website of a competing company. Given the presence of advertising syndicators and automatically computed advertisements based on the keywords in a domain name, we can relatively safely conclude that the blame here is with the domain parking service which did not check whether the domain was abusing trademarks, rather than with the competitor who ended up receiving the user.

4.3.6 Malicious redirections

Domain parking companies state that they provide a legitimate service that help visitors by showing them relevant advertising links [99]. While this can be true for domains without any trademark or typosquatting issues, there is another phenomenon that makes us even more skeptical about the goodwill of domain parking services.

Out of our initial 3,000 randomly sampled pages, our instrumented browser was redirected to a different domain in 7%±0.9% of the cases, a feature of parking services that is called Pay-Per-Redirect (PPR) [16]. Note that our crawler never

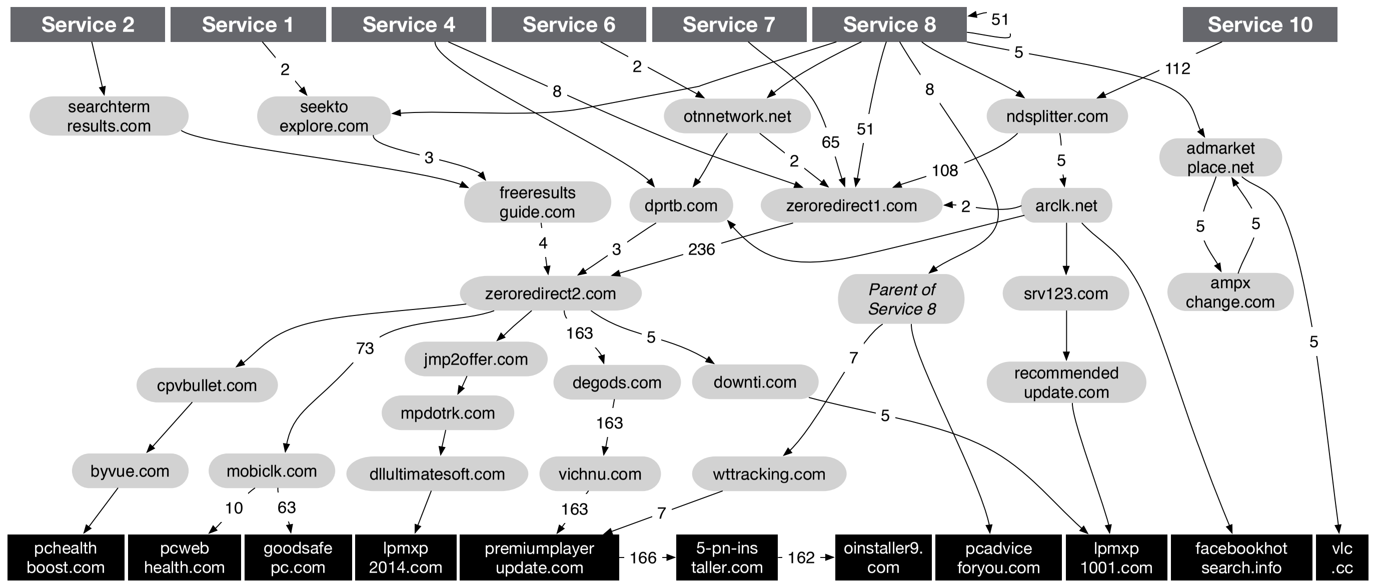


Figure 4.6: Graph of redirections leading to malware recorded by the US crawler. The graph shows a page hosted by a parking service (top row), following redirections throughout various nodes, towards domains hosting malware (bottom row). The graph excludes Parking Service 9, because it had no common nodes with the other services.

clicks on any advertising links thus the redirection can be fully attributed to the domain parking service. In a preliminary examination of these redirections, we witness the landing on dubious websites including, among others, malware, scams, and affiliate abuse.

In order to examine this phenomenon more thoroughly, we sampled 100 domains from each parking service and setup an additional crawler which crawled them daily for a week. During this crawl, we recorded the entire redirection chain, visited and downloaded all links present on the final page, and kept track of all downloaded data and files. To assess any geographical differences, we performed this crawl in parallel from the United States and one country from Europe. Table 4.2 shows that ten out of fifteen studied services are conducting redirects[2]. Parking Service 10 tops the list by redirecting in 2/3 of the cases, followed by Parking Service 9, which redirected about 1/3 of the visits. Some services redirect far more often than others, but there is no clear trend in relation to the size of the service's domain portfolio.

We distinguish between three categories of malicious redirections: malware, scams and adult material. In order to identify different campaigns, we automatically clustered malicious websites based on their screenshots through the use of perceptual hashing [128], a technique that generates distinct hash values that are robust to small image changes. After this process, we manually corrected any wrong clustering, labeled the resulting clusters, and extracted the chain of redirect domains for each campaign.

A website was classified as containing malware, if it hosted any downloadable executables that are flagged as malware by at least one antivirus engine of the VirusTotal service [196]. The redirections that contained malware were hosted on 16 different domains, most of them trying to convince the visitor to download a malicious update for their Flash player or browser. Interestingly, as shown in Table 4.2, these malware campaigns seem to aim almost exclusively on American traffic, since we encountered them mostly on our crawler behind an IP address located in the US. One plausible explanation for this phenomenon is the high cost of advertising targeted at US traffic, compared to the rest of the world. As such, malicious advertisers may be much more aggressive so that they can recuperate their high advertising costs. Related to this is the fact that compromised machines are worth much more in the US than the rest of the world [29, 140], presumably because of their increased trustworthiness and, in extent, ability to be monetized better through spam and other malicious operations. The effect of geographical location on the type of abusive ads was

[2]Previous research has shown that malicious advertisements can occasionally be delivered by legitimate advertising networks [214]. For this reason, we decided to anonymize the names of the domain parking services which delivered malicious or inappropriate content during our measurements.

also recently observed Nikiforakis et al. [143] who studied malicious advertising in the context of ad-based URL shortening services.

Figure 4.6 summarizes the redirection chains of seven services that have been found to redirect to malware-laden websites. The graph shows that many intermediate parties are involved in leading a visitor to a malicious website and several nodes account for redirections of multiple services, such as `zeroredirect2.com`. Possibly, these complex chains are the consequence of a process similar to *ad arbitration*, a widely adopted practice performed by most ad syndicators [214]. During this process, the syndicator bids on available ad slots of other publishers or syndicators, allowing them to resell these slots to the next bidder. Often, ad slots are subjected to multiple iterations of this reselling process. As a consequence, ad slots are no longer under control of the syndicator that the original publisher partnered with. All these interactions and intermediate parties have the potential to blur the direct involvement of the parking service in serving malware. In some cases, however, we also see malware being delivered more directly, for example, by the parent company of Parking Service 8.

In terms of scams, we encountered several different kinds of campaigns. In one campaign, spanning eight different domains, the advertiser was trying to persuade users to hand over highly sensitive information, such as their Social Security Number, to allegedly retrieve the user's creditworthiness. In one particular case, typosquatting domains, such as `banjofamerica.com`, were used to increase the credibility of the scam. The displayed page had a special notice for "Bank of America visitors" warning them to urgently check their credit score because of a security breach.

Another kind of scam, of which we found two campaigns residing on three domains, claims that the visitor's computer is infected with malware. The web page insists on calling a given phone number for support. An example of this is depicted in Figure 4.7, which abuses the Norton logo to increase its trustworthiness. We called two of these numbers, pretending to be a clueless victim. Both "support lines" offered us assistance for the bogus infection. They required us to install a remote desktop application and inspected our machine for malware. For the purposes of this experiment, we setup a virtual machine running Microsoft Windows XP and installed a handful of popular desktop applications. Since we had just created these virtual machines for our experiment, we are confident that no malware was present on our systems.

The two "technicians" who were given remote access to our machine, used similar social engineering techniques to try to convince us that our computer was infected with malware. More specifically, they inspected our list of processes and showed us completely benign warnings from the Windows Event Viewer log, which they claimed were there because of malware. One service offered to remove this malware for 150 USD, the other asked 250 USD, plus another 250 USD to install antivirus software. Other scams involved the user's participation in surveys which convince users to disclose their sensitive information by promising them high value coupons for big box stores which, somewhat predictably, are never delivered.

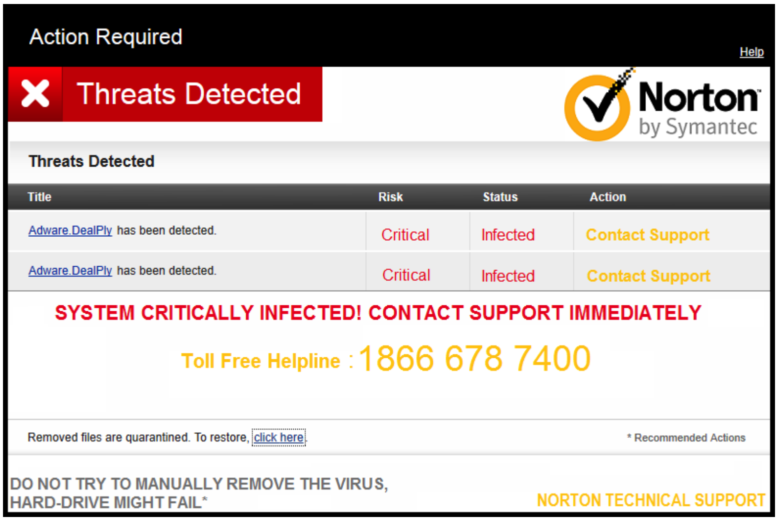
When one considers both regions, all ten services were found to redirect to malware or scams at some point during those seven days.

Finally, the last category of observed malicious redirections leads to adult websites. We encountered seven domains hosting pornographic content. As seen in Table 4.2, such redirections were most prevalent in our European crawler. Seven of the parked domains that redirected to adult material originated from adult-oriented parked domain names, specifically those parked with Parking Service 5 and 6. We assume users expect to be exposed to adult content when typing in such domain names, therefore, these redirections were considered non-malicious and put in parenthesis. However, the other 42 parked domains that automatically redirected to pages with sexually explicit content were either unrelated or even completely inappropriate, such as `southwestairlanes.com` and `arabianmarriage.com` and thus considered as malicious redirections.

Overall, while malicious advertisers are ultimately responsible for malicious ads, one cannot fully excuse the domain parking services involved in putting users in harm's way. The large percentages of discovered abuse suggest either a complete lack of countermeasures against malicious advertising, or the presence of very ineffective ones. Moreover, given the way that some of these scams were set up, we think that it would be really hard to successfully press charges against these services, since, for instance, in the scam involving virus warnings, the user voluntarily calls their support centers and gives them access to his machine. These kinds of scams show that legislation is not always sufficient to protect or compensate users, and thus highlights the necessity of technological countermeasures and user education.

4.3.7 Detecting and Bypassing Ad-Blockers

Today, some of the most popular extensions in browser markets are ad-blocking extensions. These extensions typically operate by scanning each web page against a blacklist of advertising-related regular expressions and prevent the matched content from being loaded. Since domain parking services mostly rely on visitors clicking on, usually omnipresent, advertisements for their revenue, the adoption of these tools is a big threat to their business model. For each investigated service, we tested whether the domains parked with them tried to



Action Required

Help

Threats Detected

Norton by Symantec

Threats Detected

Title	Risk	Status	Action
Adware.DealPly has been detected.	Critical	Infected	Contact Support
Adware.DealPly has been detected.	Critical	Infected	Contact Support

SYSTEM CRITICALLY INFECTED! CONTACT SUPPORT IMMEDIATELY

Toll Free Helpline : 1866 678 7400

Removed files are quarantined. To restore, click here

* Recommended Actions

DO NOT TRY TO MANUALLY REMOVE THE VIRUS, HARD-DRIVE MIGHT FAIL*

NORTON TECHNICAL SUPPORT

Figure 4.7: Screenshot of a scam website to which our crawler was redirected after visiting a parking page.

detect or bypass the workings of ad-blocking extensions. Note that since these services have full control of their parked domains, the discovery of a parked domain being involved in a specific practice equates to the parking service being involved in that same practice. Overall, we discovered that 2 out of the 15 studied services attempted to detect and bypass advertising blockers.

NameDrive NameDrive ships their parked websites with an additional detection mechanism for ad blockers. They include an external file called *advertisements.js*. However, unlike the file name suggests, this file does not contain any code related to advertisements: it is just a single statement that sets a variable to `false`. This file is basically a honeypot for ad blockers: it deliberately attempts to trigger the blocking of resources by exposing a very obvious advertising-related filename for its JavaScript code. The file is detectable by the `"/advertisements."` present in the *EasyList*[5] blacklist, one of the most common blacklists used by a range of ad-blocking extensions, including the popular AdBlock Plus. Later on, another script included on the page verifies if the variable was actually set to `false` or not. If not, the user is automatically redirected to a completely different website for PPR monetization, as described in Section 4.3.6, since no money can be made from this user from regular PPC advertising schemes. As such, we expect the number of redirections to be higher for users browsing with ad-blocking extensions.

Fabulous.com Websites parked with Fabulous contain JavaScript code that verifies whether the JavaScript object named *google* exists. This object is normally initialized by a Google Adsense script included in the parked page. If this object does not exist, one can reasonably assume that advertisements are blocked. Subsequently, the web page reacts by creating iframes with content generated from other Fabulous pages. The generated content contains internal advertisement links, which when clicked, eventually redirect the user to the websites of advertisers.

4.3.8 Summary of Findings

By analyzing more than 8 million parked domains, hosted with 15 different services, we found that the lion's share of the domain parking ecosystem is under the control of a small fraction of the involved entities. For instance, we discovered that the majority of domains are in the hands of a small fraction (16%) of all domain owners, and that just the three most popular parking services are accommodating over 60% of the examined domains. This trend continues when it comes to the monetization through advertisements, as a single, popular, advertising syndicator provides PPC ads for 90%±1.1% of all visits to parked domains. Since only a handful of parties is responsible for the majority of the ecosystem's monetization chain, we argue that a change in policy or the business model of these large players, could drastically and effectively influence the domain parking scene.

In terms of abuse, we found that only one parking service did not include typosquatting domains in their managing portfolios, with all other services accepting and profiting from these abusive domains. Specifically, through automatic measurements, we conservatively estimated that a service's domain portfolio may contain over 4% typosquatting domains. In a more in-depth manual sample analysis, however, we found 16%±3.2% of parked domain names to be either trademark or typosquatting abuse. A reasonable assumption is that typosquatting domains receive more visitors than most generic parked domain names, thus contributing to a large extent to the profits generated in the domain parking industry. As such, foregoing the profits associated with accepting typosquatting, and other cybersquatting domains, is likely not something that these services will do voluntarily.

Furthermore, we examined the Pay-Per-Redirect (PPR) phenomenon, which is used as a secondary monetization source, performed in 7%±0.9% of the visits to parked domains. All ten services deploying this redirection strategy have been found to unexpectedly send their visitors to malware-laden websites, scams or explicit adult content. This phenomenon shows that parking services are not reluctant to deploy malicious monetization strategies at the expense of user safety.

4.4 Detecting Parked Domains

The previous section analyzed the ecosystem of domain parking and mapped out the practices involved in their business model. Based on our findings, we do not consider it much of a stretch to claim that, at their current state-of-practice, domain parking services act in a parasitic way. As such, it is important to implement countermeasures in order to reduce their prevalence or at least minimize the user's exposure to them. We approach this problem by proposing and developing a classification model that is able to detect parked pages. This model is meant to be robust, meaning that it does not rely on any parking services' specifics, such as their specific name servers, but rather relies on features inherent to the conceptual operations of parked domains. Applications for this model exist on various levels, e.g., it could be part of a search engine crawler, where the detection could be used to discard parked pages from their search results, or part of a browser extension that detects and blocks parked pages when a browsing user encounters one.

In this section, we walk through the construction of the classification model by describing how the data was gathered, which features were used, how the model was tuned, and how the classifier was trained and evaluated.

4.4.1 Gathering data

We begun this process by first obtaining a random sample of 3,000 verified parked pages and 3,000 pages from the Alexa top 1 million. We automatically verified that the sampled pages from Alexa are not parked by examining their name servers and ensuring that they do not match the name servers of any of the studied parking services. Next, we crawled all 6,000 pages and collected data from several different sources. More specifically, we gathered the HTML source code of every loaded frame recursively, recorded a trace of all HTTP requests initiated by the web page (HAR), as well as the redirection chains of the main page and every frame. Finally, we inspected properties of the domain itself, such as typosquatting occurrence and WHOIS records. From this data, we can extract discriminative features that can serve as input for our classifier.

4.4.2 Feature set

When creating features to detect parked pages, we try to target the inherent nature of the parking services' operation model. This approach results in more robust detection, as opposed to searching for traces of specific parking services or looking for fixed keywords.

We focus on detecting the omnipresence of third-party advertising, dynamic and on-the-fly page generation, lack of content, malicious redirections, and abusive domains. In total, we construct eleven HTML features, five HAR features, four frame features and one domain feature. We elaborate on the extraction of each feature and our rational for choosing them in the following paragraphs:

HTML Features: The HTML features are extracted from the source code of every loaded frame. From this code we can analyze the content and the scripts deployed on the page.

- **Average and maximum link length.** We count the number of `<a>` elements present on the page and measure the string length of the destination addresses. From these numbers, we can calculate the average and maximum link length of the page. The rationale behind this feature is that advertisement links, which usually form the majority on domain parked websites, pass more and longer parameters along with the link in order to track the click on the PPC ad. They might, for example, include the publisher's identifier, the final link destination, tokens, timestamps, etc.
- **Average source length.** Similar to the previous feature, source addresses for banners and other advertisement media, tend to pass parameters of campaigns, image dimensions, etc. We expect non-parked websites to have more static media sources and thus shorter address lengths.
- **External link and external source ratio.** We define an external link or source as one with an address pointing to a another domain. Links and media generated by third-party advertisement syndicators will generally reside on domains of that syndicator. We expect non-parked websites to have a lower ratio of external links, because they commonly also have links to pages and media hosted on the same domain.
- **Website directory presence.** Since parked domains are undeveloped websites that display content that is generated on-the-fly, it is uncommon for them to have dedicated directories on their website. We search within the HTML source and link addresses for the presence of a directory and use this as a boolean feature.

- **Link-to-global text ratio.** Many parked pages have hardly any text on their page that is not part of a link. On a typical parked page, text is either part of an ad or part of the "Related links". To assess this characteristic, we extract all text from the HTML pages with Python's Natural Language Toolkit [23], which omits the HTML tags and returns the textual content. We compare the amount of text that resides within links (`<a>` elements and their child nodes) to the global amount of text present on the page.

- **Amount of non-link characters.** To more robustly test the characteristic of the previous feature, we incorporate an additional feature that counts the actual amount of characters not belonging to any link element, instead of solely relying on the ratio.

- **Text-to-HTML ratio.** We also measure the ratio of text to the total amount of characters in the HTML file. This feature focuses more on the dynamic generation of content.

- **Redirection mechanisms** Parked pages use redirection mechanisms to lead visitors to other pages or domains. Although non-parked pages might also deploy such mechanisms, we still believe that this feature, when considered together with other ones, can assist classification. We detect two different redirection methods. One feature records the presence of JavaScript redirection code by searching for `window.location`, while the other finds HTML meta refreshes by looking for `http-equiv="refresh"`.

HAR Features: These features are derived from the HTTP archive (HAR) that is constructed while loading a page. We focus on the following discriminative characteristics of HTTP requests:

- **Third-party requests ratio.** We extract the number of HTTP requests to third-parties (other domains) and the total amount of requests. Next, we calculate the ratio between those two. This feature is motivated by the amount of third-party content and media generated on parked pages. In addition, HTTP requests conducted after redirecting to a different domain, are all considered third-party requests, with respect to the initial domain.

- **Third-party data ratio.** Similarly, we calculate the ratio between data (number of bytes) coming from third-party sources and all incoming data.

- **Third-party HTML content ratio.** This feature further incorporates the characteristics of third-party content. We expect most third-party content on regular websites to be generally JavaScript libraries and media files. Parked websites, however, are known to include `html/text` content pulled from third-party services, such as through the use of iframes generated by ad syndicators. For this reason, we include a specific feature that represents the ratio of HTML content brought in by third-party requests.

- **Initial response size and ratio.** For this feature, we first record the size of the initial response when making the first request to the web page. Next, we compare this with the total amount of received data after completely loading the website. This feature attempts to capture the dynamic generation of content on parked pages which is a core concept of the modus operandi of domain parking services. With this feature, we expect to identify the initial lightweight page skeleton, which stands in contrast with the final amount of received data.

Frame Features: The following frame features are extracted by tracking every loaded frame on the web page.

- **Amount of frames.** While manually inspecting the structure of parked pages, we found that the presence of iframes is very common. In order to take this into account, we recursively count all frames and iframes present on a page and its child frames.
- **Main frame and iframe redirections.** The redirection chain of every frame was tracked when we crawled the domains. For every chain, we extract the number of redirections that occur on the main frame as well as all other frames. As noted in previous sections, malicious redirects initiating at parked pages contain many different traffic distributors and redirection hops, e.g., as shown in Figure 4.6. Thus, we expect a benign redirection chain to consist of a limited amount of intermediate steps.
- **Different final domain.** This feature checks if the main frame (i.e. the frame of which the address is visible in the browser's URL bar) was redirected to a different domain. It excludes internal redirections, such as from `www.domain.com` to `blog.domain.com`, which is a more common redirection process on regular websites.

Domain name feature: This feature focus on characteristics inferred from the domain name itself.

- **Typosquatting domain.** The current domain name is checked for typosquatting abuse with the algorithm described in 4.3.3. Regular, authoritative websites should not be flagged by this feature.

4.4.3 Classification

Our objective is to construct a classifier that can reliably detect parked websites when visiting them. When loading a web page, the aforementioned features are extracted and are treated as the features of a particular *instance*. The classification model's goal is to take an unknown instance as input, process the features, and assign a probability of that instance belonging to a certain class. More specifically, the model calculates whether or not any given web page is likely a parked one. Given these probabilities, a threshold value can be used to actually classify the instance as either parked or non-parked. This threshold can then be appropriately varied to tune the sensitivity of the model.

In order to build the classifier, we first select an appropriate learning algorithm for our model. Afterwards, this model needs to be given a sufficient amount of parked and non-parked instances for it to learn from. Once a model is learned, we can evaluate the performance of the classifier with unseen test instances.

Learning method

We aim for high interpretability of our classifier, as it is important to comprehend the prediction of the classifier for further improvement and adaptability. This quality of a classifier is also important if our model is to be incorporated in a browser and be used by users. Therefore, we opt for the Random Forest algorithm, as it combines the strength of ensemble learning with the interpretability qualities of decision trees. Furthermore, decision trees tend to be robust with regard to outliers, while the ensemble technique of Random Forest protects the model against overfitting. Moreover, after the trees have been constructed in the learning phase, the classification is usually very quick in the detection phase.

Dataset handling

Train and test dataset The set of instances used in the learning phase of the model is referred to as the *training set*, for which we reserve 2/3 of our dataset of 6,000 pages. As follows, 1/3 of the set is isolated and used after the model is built, in order to adequately evaluate our classifier. Since these instances are used to test a classifier, they are referred to as the *test set*.

Data transformation In order not to overfit the classifier, we remove outliers and extreme values from our training set using an interquartile range filter, which is configured to operate on a per feature basis.

Omitted features Our initial set of features contained two additional features, which, however, were omitted after manual and algorithmic selection.

One of these features was based on the detection of WHOIS entries that make use of anonymizing services, such as WhoisGuard [208]. We expected parked domain owners to have a higher incentive to anonymize their personal association with their domains, as opposed to regular domain owners. However, when manually inspecting the features of the training set, we found that this feature was not very discriminative for parked domains. Furthermore, since it required searching for specific strings related to anonymizing services, the feature is also less robust. For these reasons, the feature was removed from our final model.

To further improve our feature selection, a backwards greedy stepwise search was conducted on the remaining feature set. The search started with the complete feature set, and removed features one by one until a better classification result was achieved. The algorithm found that another feature was reducing the performance of the classifier. This particular feature counted the number of times the parked domain name was passed along with HTTP request parameters. The reasoning behind this feature was that, often, the domain name is sent as a parameter in a request to a third-party. Parked domains do this in order to enable the parking service to return content or ads relevant to the given domain, as described by Wang et al. [204]. Nevertheless, the feature was found to confuse our classifier and was thus omitted from the model.

Evaluation

After transforming the data and selecting the best performing features, we built a model from our training set using Random Forest with 10-fold cross validation. As a model's performance is not determined by its effectiveness on the training set, but on its ability to classify unknown instances, our test set was used for evaluation purposes. The trained model processed 1,000 unseen parked and 1,000 unseen benign instances. Next, the ROC curve, Figure 4.8, was generated by varying the classification threshold and keeping track of the resulting True Positive Rate (TPR) and False Positive Rate (FPR). With a 99.65% area under the curve (AUC), the model proves to generalize very well, resulting in a performant classifier.

The default threshold, set at 0.5, results in a FPR of 1.5% and a TPR of 98.6%. Since we consider the cost of a false positive to be higher than a false negative (i.e., it is worse to classify a normal website as a parked one, than vice-versa),

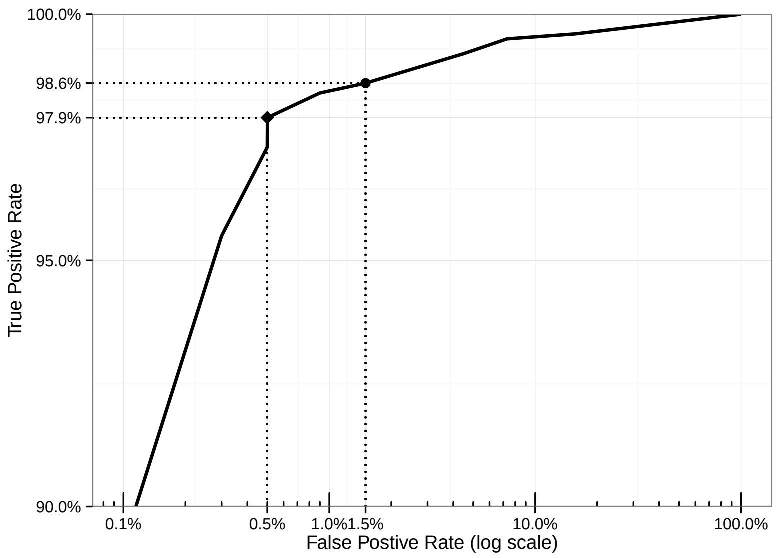


Figure 4.8: ROC curve of the parking detection model on the test dataset. The model achieves 99.65% AUC. Two threshold points (0.5 and 0.7) indicated on the curve with their resulting FPR and TPR.

we attempt to increase the specificity of the model. As one can see from the ROC curve, it is possible to reduce the false positive rate without sacrificing too much sensitivity. With the threshold set at 0.7, a FPR of 0.5% is achieved (i.e. 5 out of 1,000 benign domains were falsely classified as parked domains) together with a TPR of 97.9%, which results in an overall classifier accuracy of 98.7%. Both thresholds and their resulting FPR and TPR are indicated on Figure 4.8.

Looking at the performance statistics, we consider the classifier highly capable of protecting a browser, search engine, or other system by detecting parked websites on the web.

Detection Evasion

It is reasonable to assume that once a classifier is deployed at a large enough scale, the affected parties will attempt to evade it. The presented classification model, however, purposely focuses on features inherent to the parking services' workings. As such, an evasion sufficient to purposefully misclassify a parking page is non-trivial. Domain parking services would face all sorts of obstacles while attempting to bypass the features of the classification model. For instance, they would need to substantially decrease the relative presence of third-party advertising by either providing a large portion of first-party content for each domain, or removing third-party syndicators all together. Both options require major refactoring of their current operations. Additionally, redirections chains and mechanisms are tracked in every frame, so the PPR monetization would need to be abandoned. Likewise, typosquatting domains could be excluded from parked domain portfolios to reduce detection, but this would be a “welcoming” evasion since it would effectively reduce the level of abuse present in domain parking services.

In other words, we opine that if the classifier would be widely deployed, the domain parking industry would either be forced to shift to a different, hopefully more legislated, business model, or lose its remaining exposure to users, both effectively mitigating current abusive monetization techniques.

4.5 Discussion

Although parking services have been around for over a decade, they have always been controversial, mainly because of the limited added value they contribute to the web, which stands in contrast with the millions of domains associated with them. The people who disagree with this opinion, are usually domainers and the services themselves who argue that domain parking is of legitimate help to users by assisting them in finding relevant content [99].

While the business climate for domainers was extremely satisfactory in the early 2000s and a lot of money was made with parking services, the circumstances have changed substantially since then. If we look at the annual reports of the Sedo Holding [173], the service managing the largest domain portfolio, we observe the sales revenue of their “Domain Marketing” segment dropping year after year. Since 2007, they have been losing up to 17.9% yearly, now summing up to a 34.9 million EUR (56%) decline in sales revenue in 2013. The report explains this profit shrinkage, and the overall downtrend of the parking industry, as further impacted by “Advances in browser technologies.” We postulate that these “advances” might refer to the deep integration of search engines in the browser, as described in Section 4.2.2. Most likely, this integration heavily reduced type-in traffic for parking pages. Furthermore, the adoption of ad-blocking tools is another browser technology that might be causing a huge setback for their monetization capabilities, as explained in Section 4.3.7. Lastly, one more obstacle for the industry was introduced by an update to Google’s search engine in December 2011 [134] where Google decided to exclude parked domains from Google's search results.

We speculate that parking services (and domainers) have become desperate to counter this problematic regression, and are therefore resorting to shadier or even downright malicious ways to attract and monetize traffic. It is clear that typosquatting domains are still a regular part of their business model, as these are probably one of the only remaining types of domains that reliably receive type-in traffic. Next to abuses of trademarks, we encountered parked domains that redirected to malware, scams and adult material, for which we can reasonably assume that they pay more than legitimate redirections.

A similar observation is made in concurrent work, where the involvement of parking services in click fraud towards their advertisers, and the reluctancy to distribute the rightful commission to the domain owners, is attributed to the decrease in revenue [16].

If we look at the number of services involved in fraud, malicious activities and monetization of abusive domains, we opine that domain parking has become a threat for all users on the web as well as for the (legitimate) advertising industry.

4.6 Related Work

In concurrent work, Alrwais et al. [16] investigated the domain parking industry, focusing on the fraudulent practices in their monetization chain. In their work, the authors registered themselves both as domain owners and advertisers, effectively operating at both ends of the domain parking ecosystem. This allowed them to connect the dots and detect any discrepancies between what they knew to be true and what the domain parking services claimed as true. Using this method, the authors uncovered several fraudulent practices including the presence of click fraud, where their advertising campaigns were charged for receiving visitors through clicks, even though the visitors were the researchers' crawlers which were configured to never "click" on any ads.

While Alrwais et al. focus their investigation on the abuse against advertisers, we instead focus on the abuse against users landing on parked domains, as well as domain owners whose trademarks are being diluted because of the absence of trademark-infringing checks from the domain parking services. Though our work is orthogonal to their work, we both arrive at the same conclusion: Domain parking services are currently unlegislated and that allows for a lot of abuse, towards all third-parties of the domain parking ecosystem: users landing on parked pages, advertisers paying for ads, and holders of popular trademarks and domain names. To that extent, a client-side countermeasure, like the parked-page classifier we propose in this chapter, can protect the user-part of this ecosystem while we hopefully transit to a more legislated domain parking industry.

In 2010, Almishari et al. [15] developed a classifier for “ads-portal” domains, which they used to identify typosquatting abuse of domain parking services. They observed that in 2008, 50% of parked pages were residing on typosquatting domains. Now, 6 years later, we witness that parking services have a significantly smaller, albeit still substantial, typosquatting portfolio. The classifier they developed leveraged several HTML features similar to the ones proposed in this chapter. For instance, they focused on the dominant presence of anchor and image elements and the numerous parameters passed along with these URLs. However, they did not incorporate HAR features, which are able to identify the dynamic and external nature of parking pages. Furthermore, they did not take into account redirections and no in-depth frame analysis was made. In terms of performance, they incorporated keyword-based features to increase the accuracy of the classifier, a strategy that we deliberately avoided in order to ensure robustness.

Domain parking was also recently mentioned by Li et al. who studied generic malicious web infrastructures [117]. Interestingly, the authors observed that domains hosting malicious Traffic Distribution Systems (TDS), were monetized through parking services after the TDS had been taken down. Even after a take down, these domains keep on receiving a large amount of traffic since numerous malicious doorways (typically compromised websites) are still redirecting users to the TDS domain.

Prior to the aforementioned research, domain parking was only mentioned in research regarding cybersquatting. One of the oldest studies of typosquatting by Wang et al. [204] proposed a series of typosquatting models and showed that, in 2006, 51% of all the possible typosquatting domains of websites in the Alexa top 10,000 were registered and active. The authors uncovered that 59% of the active typosquatting domains were hosted at 6 major domain parking companies and reported that inappropriate adult content was encountered on typos of children’s websites.

In a later study, Moore and Edelman tried to identify the entities responsible for typosquatting [131]. The researchers observed that name servers belonging to parking companies have up to 4 times more typosquatting domains than average. In terms of advertising, they identified Google as the prime source for PPC ads on typosquatted domains. Furthermore, they encountered abusive domains to be serving ads for the authoritative domain (self-advertising) and for competitors. Additionally, they measured a similar concentration phenomenon as to what we noted in the domain parking ecosystem: 63% of typosquatting domains that were using Google PPC ads, belonged to only 5 different publisher IDs. In our study, we observed that not only Google maintains almost complete control of advertising in domain parking services, but that they do so while claiming that domain parking pages are spam [74].

In a very recent content-based typosquatting study, Agten et al. [8] verified that parked pages are still the most prominent monetization strategy for typosquatting domains. More specifically, ad parking was witnessed in 51% of observations. Furthermore, they detected malicious redirections towards malware, scams and adult content. Taking into account the observations from our research, it is likely that these redirections were initiated by parking services that hosted the typosquatting domains.

4.7 Conclusion

Despite existing for over ten years, managing millions of domain names, and making yearly revenues of multiple millions of dollars, domain parking services have received little attention so far.

In this chapter, we mapped the ecosystem of domain parking services by identifying popular parking services, the types of parked domains, the owners of these parked domains, and the advertising content that users are exposed to when they land on the pages of parked domains. In this process, we identified multiple types of abuse including malware, inappropriate advertising, and scams that could cost users their personal details and hundreds of dollars. Next to the abuse affecting users, we also discovered a significant fraction of typosquatting domains being monetized through domain parking, and witnessed the lack of controls by successfully parking an obviously abusive domain with all domain parking services on which we could get a parking account.

Motivated by the discovered abuse, we designed and built a parked-page classifier which can be used to, among others, block parked pages, or alert users that they are currently interacting with a parked page. Instead of hardcoding parking signatures, we compiled a list of generic and robust features that are characteristic of parked pages and showed that our classifier has a very high accuracy with only minimal false positives.

Chapter 5

Hijacking domains through their nameserver

Preamble

We previously discussed typosquatting and bitsquatting in Chapter 2 and 4. In these attacks, malicious actors anticipate on resolution errors affecting the domain name the client intends to visit. For this chapter, we analyzed whether such vulnerabilities are also applicable to other parts of the recursive DNS resolution process. Specifically, we focus on requests to authoritative nameservers. This approach turns the classical typosquatting and bitsquatting problem around by shifting the focus from the DNS clients to the DNS *servers*. As is discussed in this chapter, the implications of these exploitable errors differ greatly from the previously studied issues.

One of the primary themes in this chapter is the critical, but inconspicuous role of DNS nameservers. We study negligence and deployment errors related to these key components of the Internet's distributed and hierarchical naming infrastructure. When looking at the larger perspective of nameserver misconfigurations, one could argue that, in essence, this a *dependency management* problem. Domains namely rely on a chain of trust, established by different nameservers, at different levels in the DNS hierarchy. Often, dependency management is discussed in the context of software, i.e. external libraries and plugins that are included in the application. Software maintainers have to keep track of all these dependencies. At any point in time, a vulnerability might arise in a piece of external code that the application relies upon.

A recent paper studied the dependencies of client-side Javascript libraries in the top websites [111]. They find that 37% of websites depend on inclusions that have known vulnerabilities. Moreover, they elaborate on *indirect* dependencies, i.e. vulnerable libraries that are transitively included in the web application via code introduced by other dependencies (e.g. advertising, tracking or social media widgets). Interestingly, these indirect dependencies have a higher rate of introducing vulnerabilities into the web application than direct inclusions. The inability to transparently evaluate changes and vulnerabilities of multi-level dependencies also plays an important role in this chapter.

As will be discussed in Section 5.2.3, the "textbook" DNS resolution process does not hold in practice. Typically, the recursive resolution path is not straight down from the root to the 2LD zone. This is caused by 2LD nameservers that, in turn, depend on other nameservers that might be in different zones. In practice, an administrator directly appoints about 2 nameservers for his domain. However, its resolution process is found to –indirectly– depend on 46 different nameservers [158]. Similar to software dependency management, getting a transparent and continuous overview of all the DNS infrastructure and configurations your domain is exposed to, is not very straightforward.

Thomas Vissers started this project in the fall of 2016 as a visiting researcher at the PragSec lab at Stony Brook University (New York, USA) under the supervision of Prof. Nick Nikiforakis. The further contents of this chapter are replicated from the paper titled "The Wolf of Name Street: Hijacking Domains Through Their Nameservers" [197], which was published in the proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS). This work was done with the collaboration of other authors from KU Leuven and Stony Brook University. Thomas Vissers is the lead author of this paper.

The Wolf of Name Street: Hijacking Domains Through Their Nameservers

Abstract

The functionality and security of all domain names are contingent upon their nameservers. When these nameservers, or requests to them, are compromised, all domains that rely on them are affected. In this chapter, we study the exploitation of configuration issues (typosquatting and outdated WHOIS records) and hardware errors (bitsquatting) to seize control over nameservers' requests to hijack domains. We perform a large-scale analysis of 10,000 popular nameserver domains, in which we map out existing abuse and vulnerable entities. We confirm the capabilities of these attacks through real-world measurements. Overall, we find that over 12,000 domains are susceptible to near-immediate compromise, while 52.8M domains are being targeted by nameserver bitsquatters that respond with rogue IP addresses. Additionally, we determine that 1.28M domains are at risk of a denial-of-service attack by relying on an outdated nameserver.

5.1 Introduction

The Domain Name System (DNS) is one of the most important protocols of today's Internet, seamlessly converting human-readable domain names to machine-routable IP addresses. From a branding perspective, domain names are important because they are essentially the brands which users recognize and interact with. Even though new TLDs are constantly introduced, short and generic domains in the traditional TLDs are still sold for millions of dollars [211]. From a security perspective, domain names and their properties are implicitly and explicitly trusted by users and programs alike. Users are constantly instructed to look at the domain name of websites that they visit and inspect the domain part of the sender's email address when they suspect that they have received a malicious email. Websites will send reset-password links to the mail servers listed in a domain's MX records and intrusion detection systems will treat links as less likely to be malicious if they point to long-existing domain names, rather than newly created ones.

In recent years, researchers have identified that attackers will often register old domains that were allowed to expire in order to capitalize on the residual trust of these domains. This trust has been abused to host malware on the domains of old financial institutions [130], masquerade the communication of C&C malware as traffic to and from long-established domains [115], and even hijack entire autonomous systems [169, 168]. In some cases, attackers do not even have to wait for domains to expire. In addition to guessing/stealing a domain owner's registrar password and moving that domain to a new registry [176], researchers have shown that, under the right conditions, attackers could hijack control of live domains by abusing the dangling links to stale IP addresses that arise because of environment idiosyncrasies of public clouds and managed DNS services [27, 28, 119].

In this work, instead of focusing on individual domain names, we perform the first, large-scale investigation into the "hijackability" of nameservers and, consequently, of all the domain names that trust these nameservers for resolution purposes. More specifically, we focus on exploiting configuration issues and hardware errors to gain control over DNS requests to nameservers.

Targeting the nameserver substantially increases the attacker's potential. As the actual requested domain name remains unaltered in the DNS resolution, extreme stealthy offenses are possible. For instance, invasive MITM attacks enable miscreants to take full control over the victimized domain and its incoming traffic. Furthermore, compromising a nameserver is very efficient, as a single attack targets all domains relying on that nameserver simultaneously.

The main contributions of this study are:

- Through extensive analysis and measurements, we describe and confirm the presence of typosquatting and bitsquatting vulnerabilities, specifically applied to nameservers.
- We identify instances of targeted exploitation of both nameserver squatting attacks. Meanwhile, we find a large corpus of domains that remain vulnerable for immediate exploitation by making just a few registrations.

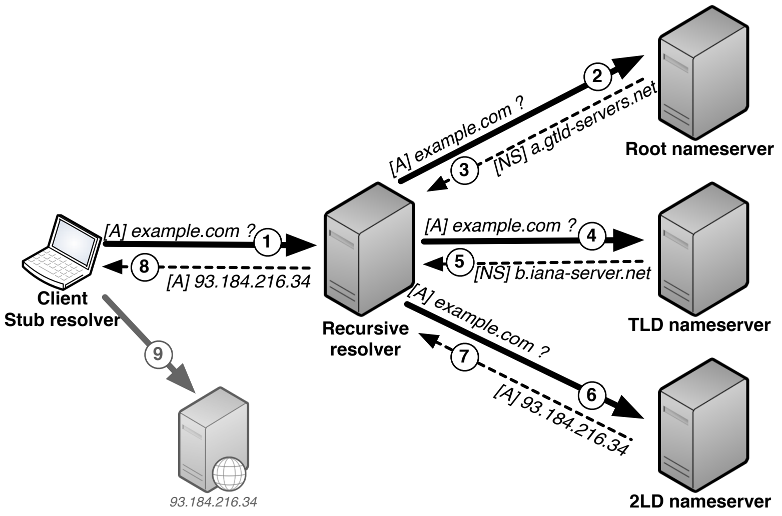


Figure 5.1: DNS resolution process of a client that connects to host example.com for the first time. Steps 2-7 depict the recursive resolution process. Step 9 illustrates the connection that is made to the IP address after the DNS resolution has taken place.

- We evaluate outdated WHOIS email records of nameserver domains and find several thousand domains at risk of being compromised due to negligence of their nameserver provider.
- We analyze the security practices of widely used nameservers and find that over a million domains are dependent on 8-year-old vulnerable BIND versions.

5.2 Problem statement

In this section, we introduce the general concept behind the nameserver hijacking attacks and define the scope of this study. Furthermore, we discuss nameserver dependencies which greatly influences the impact of the presented attacks.

5.2.1 Hijacking requests to nameservers

When clients want to connect to a certain domain name, this domain name first needs to be resolved into an IP address. This resolution process, shown

Rank	Nameserver domain	Domains
1	domaincontrol.com	39,674,597
2	hichina.com	4,975,760
3	dnspod.net	2,832,233
	⋮	⋮
10,000	ptserve.info	399

Table 5.1: First and last part of the top NSDOMs

in Figure 5.1, will typically be executed by a recursive resolver who will first contact the root and TLD nameservers, and in a last step obtain the IP address from the second-level domain nameserver (2LD nameserver). In our evaluation, we investigate various techniques that allow an adversary to take control over such a 2LD nameserver. As virtually any type of online application or service makes use of DNS, most without realizing it, the potential consequences are widespread. In this section, we provide a brief overview of scenarios that are made possible by exploiting any one of the attacks described in this chapter. It is important to note that in this overview, we only consider attacks against the most common software, and therefore the list of described attack scenarios is by no means an exhaustive one.

Man-in-the-Middle (MITM) As soon as an attacker has taken control over a domain's authoritative nameserver, clients wanting to connect to the victimized domain will send requests for the A record to the attacker's nameserver. By replying with an IP address under his control, the adversary can relay and possibly alter the traffic between the client and the domain it intended to contact. The specific consequences of such an attack will largely depend on the type of service that is being accessed. For instance, in the case of a webserver, the attacker can secretly intercept sensitive information, such as credentials and session tokens. In contrast to a MITM attack on a local network, which may only be able to target a limited number of clients, a MITM attack based on hijacking the nameserver affects *all* clients.

Domain-ownership verification In the case of a MITM attack, the time period during which the adversary can cause harm to the domain or its clients is limited to the time he has control over the nameserver. In addition to such attacks, an adversary can also perform actions that may have a more long-lasting effect. A number of such actions are related to the proof of ownership for a domain. More precisely, a number of services require (website) administrators to verify they are in fact in control of a domain, e.g. by serving a randomly-generated file at a predefined location. For many Certificate Authorities, including Let's Encrypt [113], such a verification is the only requirement in order to obtain a certificate for a domain. This means that even with only temporary control over a domain's nameserver, an adversary can obtain a certificate, which may be valid for multiple years. Moreover, as the issuance was invoked by the attacker, the domain owner does not have access to the associated private key, and thus cannot revoke the certificate. In addition to issuing SSL certificates, there are many other services that provide domain owners with more permanent access to restricted features. For instance, Google Webmaster Tools gives domain owners exclusive access to a number of features, such as the removal of pages from search results.

E-mail In addition to the aforementioned attacks, miscreants may also leverage other types of DNS records. For instance, by returning rogue MX records, an adversary can intercept emails destined to the targeted domain. With carefully chosen TXT records, he can spoof e-mail messages from the domain, even in the most secured setups where SPF, DKIM and DMARC records are verified.

5.2.2 Scope of study

To evaluate the risk of hijacking domains through their nameservers, we focus on the most prominent 2LD nameservers. More specifically, we consider the top 10,000 nameserver domains that are authoritative for the largest number of domain names. To determine this set of nameservers, on December 15, 2016, we obtained the zone files of the top five gTLDs (`com`, `net`, `org`, `xyz` and `info`) with respect to the number of second-level domains present in their zones [58]. For each domain name in each zone file, we extract the `NS` records. Overall, we collect the nameserver information of over 164 million domains.

Next, we derive the *nameserver domain* (NSDOM) for each observed nameserver, e.g we extract `dnspod.net` from the NS record listing `ns2.dnspod.net`. Then we determine the largest NSDOMs in terms of the number of domains that have NS records pointing to them. Finally, we select the top 10,000 NSDOMs as the starting point of our analyses. An excerpt of this list is shown in Table 5.1.

5.2.3 Nameserver dependencies

An important aspect of this study is the dependencies that exist between nameservers. We define a NSDOM as independent when its own NS records are in-bailiwick (e.g. the NS record for `hichina.com` is `ns1.hichina.com`) and thus the TLD nameserver will directly return the IP address of the nameserver with a glue record. In contrast, NSDOMs can be dependent on out-of-bailiwick 2LD nameservers. For instance, when querying `ns1.hostgator.sg`, we find that the NS records of the nameserver point to hosts under `dynect.net`. Since no glue record can be provided for those nameservers, an additional lookup must be made to resolve the nameserver under `dynect.net`. Only thereafter, a resolver can query `ns1.hostgator.sg` to retrieve the DNS records of a certain domain name[1]. In other words, `ns1.hostgator.sg` is completely dependent on another 2LD nameserver, and by extension, all domains relying on `ns1.hostgator.sg` are as well.

This kind of dependency is quite common. In fact, 36.4% of the top 10,000 NSDOMs are dependent on at least one out-of-bailiwick nameserver. To further illustrate these dependencies, Figure 5.2 maps out the NSDOMs that are dependent on `dynect.net`, the managed nameserver provider that suffered from a massive DDoS attack in October 2016, rendering offline multiple of its high-profile customers [212]. We find that `dynect.net` is the "direct" 2LD nameserver for 191,068 distinct domains. But if we take into account the other NSDOMs that are, at least partially, dependent on `dynect.net`, we come to a total of 9,242,256 domains "indirectly" relying on `dynect.net` (a 48-fold increase). Moreover, `dynect.com` is in turn dependent on a higher-level nameserver. Many of these relationships we observe are *full dependencies*, i.e. when a NSDOM is completely and solely dependent on a single external NSDOM. In contrast, other NSDOMs, are only partially relying on others. These nameservers usually employ multiple managed DNS providers to prevent a single point-of-failure.

We find certain instances where long chains of nameserver dependencies emerge. In other words, there are domains that rely on out-of-bailiwick nameservers, who in turn are dependent on other out-of-bailiwick nameservers, and so forth. We call these *nameserver dependency chains*. As an example, some of the chains depicted in Figure 5.2 go down to 5 levels (the figure only shows up to 4 levels for visibility reasons). Moreover, we find that one NSDOM in our dataset had 8 levels of nameserver dependencies. If any of the nameservers (or the requests) involved in such a dependency chain would be compromised, the requests to all of the dependent 2LD nameservers down the chain would be affected. By extent, the attacker then has the potential of compromising all domains relying on those nameservers as well. A similar observation was made by Ramasubramanian et al. in 2008 [158]. They measured that the resolution of a domain name is, on

[1]This assumes no caching has taken place. Furthermore, this scenario may be different in terms of glue records when the domains are in the same TLD zone. Additionally, some TLD nameservers reply with non-glue records in the additional section for performance improvements [206].

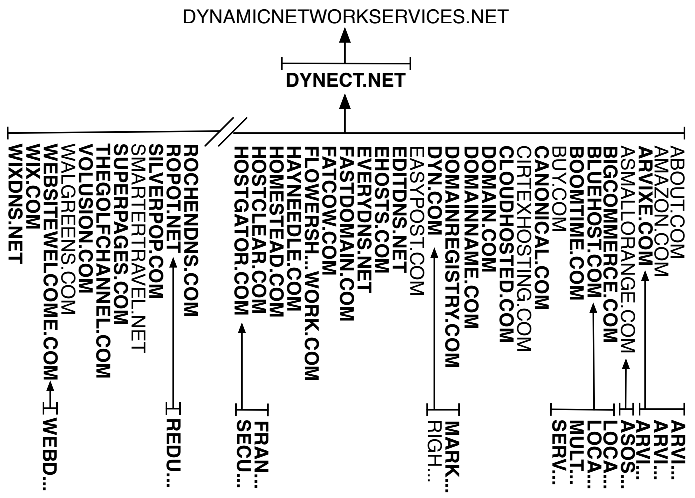


Figure 5.2: Fragment of the nameserver dependencies related to dynect.net. An arrow symbolizes a dependency on another nameserver. Fully dependent nameservers are marked in bold.

average, dependent on 46 different nameservers, while, on average, only 2.2 of those are directly appointed by the domain owner.

5.3 Nameserver typosquatting

In this section we describe the main idea of hijacking domains via typos in nameserver records and present our measurements on the potential and actual abuse of this phenomenon in the wild.

5.3.1 Attack vector

Typosquatting is the act of registering domain names that are typographical errors of authoritative domains. The malicious actors registering these domains, called typosquatters, attempt to attract accidental visitors that mistype a domain name in their browser's URL bar. As an example, a typosquatter has registered `twittre.com` in the hopes of getting a share of `twitter.com`'s massive amount of traffic.

Listing 5.1: The NS records of polishop.com according to a TLD nameserver. All .com's TLD nameservers return this answer.

```
$ dig NS polishop.com @a.gtld-servers.net
...
;; AUTHORITY SECTION:
polishop.com. 172800 IN NS ns-310.awsdns-38.com.
polishop.com. 172800 IN NS ns-1156.awsdns-16.org.
polishop.com. 172800 IN NS ns-1974.awsdns-54.co.uk.
polishop.com. 172800 IN NS ns-566.awsdns-06.ne.
...
```

Typosquatting is a well-studied problem [204, 8, 185, 131, 100], however it has been limited to the scenario where a visitor of a website is making the typographical error in his browser's URL bar. In this chapter, we analyze the yet uncharted phenomenon of *nameserver typosquatting*. In this scenario, the administrator of the domain mistypes the `NS` records while setting up the DNS configuration of the domain which usually happens through a web control panel or API offered by the registrar. To illustrate this, we take the case of `polishop.com`, a popular Brazilian web shop, which has a misconfigured (last verified on May 15, 2017) `NS` record (Listing 5.1).

The DNS administrator of `polishop.com` mistyped the `NS` record for `ns-566.awsdns-06.net` while configuring his entries in the registry's zone file through his registrar. More specifically, he missed the last character of `.net` and typed `.ne` instead. Although this record is wrong, the result is still a valid domain name that can be registered and resolved (`.ne` is the ccTLD of Niger). We can verify that this domain is in fact an accidental error by querying the other authoritative nameservers of `polishop.com`. Listing 5.2 confirms this, as `ns-310.awsdns-38.com` returns the `NS` record that correctly ends in `.net`. Because of the presence of redundant nameservers, an administrator will likely not notice when a single NS record is broken.

5.3.2 Amount of traffic affected

In the classic typosquatting scenario, only those visitors that actually make a typographical mistake in their browser are affected. Furthermore, that single mistake impacts that visitor only once. In contrast, the impact of nameserver typosquatting is persistent for as long the misconfigured `NS` record is present. It is, however, far from trivial to determine the exact amount of traffic an attacker is able to control once he exploits a single misconfigured `NS`

Listing 5.2: The NS records of polishop.com according to any of the domain's authoritative nameservers. All 2LD nameservers return this answer.

```
$ dig NS polishop.com @ns-310.awsdns-38.com.
...
;; ANSWER SECTION:
polishop.com. 172800 IN NS ns-1156.awsdns-16.org.
polishop.com. 172800 IN NS ns-1974.awsdns-54.co.uk.
polishop.com. 172800 IN NS ns-310.awsdns-38.com.
polishop.com. 172800 IN NS ns-566.awsdns-06.net.
...
```

Listing 5.3: The A record of polishop.com according to one of the domain's authoritative nameservers

```
$ dig A polishop.com @ns-310.awsdns-38.com.
...
;; ANSWER SECTION:
polishop.com. 300 IN A 54.207.32.165
...
```

record. We could simplistically assume that the ratio of DNS requests going to the attacker's nameserver is equal to the ratio of nameservers the attacker now controls. In the example of `polishop.com`, this would imply that the attacker sees one-fourth of the DNS requests. This case holds when one of the nameservers is chosen randomly for every uncached request. This happens when either the TLD nameserver randomizes the returned `NS` records, or when the client's local resolver randomly chooses which nameserver to query. There exist, however, other possibilities [174] including one where local resolvers use the best performing nameserver or query all nameservers in parallel accepting the fastest response. In these scenarios, an attacker can increase his impact by achieving faster response times than the authoritative nameservers. Attackers could attempt to launch a DoS attack on the authoritative nameservers in order to force the clients to use the attacker's nameserver, however, we assume this approach is of limited value since it trades the ability to conduct long-term stealthy attacks for a temporary increase in traffic.

To increase the amount of traffic they can manipulate, attackers can also set a higher TTL value on the rogue DNS records that they return thereby extending their cached lifetime, e.g. as shown in Listing 5.4 the malicious nameserver sets the TTL of its rogue records to more than ten times higher than the

Listing 5.4: The A records of polishop.com according to the attacker's nameserver

```
$ dig A polishop.com @ns-566.awsdns-06.ne
...
;; ANSWER SECTION:
polishop.com. 3600 IN A 185.53.177.31
...
```

authoritative records (Listing 5.3). Most administrators favor a short TTL to allow for more rapid adjustments to their infrastructure, however the default maximum cache time accepted by BIND, the most popular DNS software, is 7 days [217]. This allows potential attackers to drastically increase their impact since their rogue records can be cached thousands of times longer than authoritative ones.

It is clear that nameserver typosquatting poses an entirely different, complex, and more invasive threat than the traditional typosquatting attacks. An example that demonstrates this difference is that `polishop.com` nameserver typosquatters are willing to pay over 400 USD for the price of a single valuable `.ne` domain [2], a price that is about 40 times higher than the common gTLDs.

5.3.3 Potential and current abuse

Dataset

We generated 926,742 typo variations of the top 10,000 NSDOMs and their dependencies using the typo models described by Wang et al. [204]. These models include character omission, permutation, substitution and insertion. The substitutions and insertions are based on the set of characters adjacent to the given character on a QWERTY keyboard. Additionally, there is the missing-dot typo model, where we collected the subdomains present in NS records (e.g. ns1, ns2) and directly concatenated it with the NSDOM. Overall, we find that 95% of the generated typo NSDOMs were available for registration using the Domainr API [57].

Available typos

Of the 882,653 available typosquatting NSDOMs, 2,276 were actively used as nameservers by 6,213 misconfigured domains. Essentially, they are *unexploited* typosquatting NSDOMs, i.e. an attacker can simply register those NSDOMs and instantly compromise affected domains. As shown in Figure 5.3, registering just 6 typosquatting NSDOMs allows for the immediate compromise of over 2,000 domains, demonstrating the high impact of these attacks. 23 out of 6,213 domains are present within the Alexa top 1 million. Regardless of their Alexa ranking, all of them remain attractive targets for abuse of residual trust [130, 115, 169, 168].

One of the misconfigured domains is `protect-ns.com`. However, this domain serves as a nameserver for other domains as well. Thus, when we take into account nameserver dependencies as described in Section 5.2.3, an attacker could compromise 682 additional domains that rely on a misconfigured nameserver. Unlike the 6,213 vulnerable domains, these domains have not misconfigured their own NS records but are nevertheless vulnerable due to a mistake by a third party. The indirect nature of this error makes it particularly hard for these domain owners to, not only realize their domains can be hijacked, but also to fix the issue since the error happens at the nameserver which they trust but do not control.

Registered typos

We separate the 44,089 registered typosquatting NSDOMs into two categories based on whether they appear in NS records. 3,233 (7%) of the registered typosquatting NSDOMs are actively used by domains as a nameserver. These may be *exploited* misconfigured domains or false positives where the registered typo is coincidentally similar to a domain in the top 10K NSDOMs, but is in fact the intended authoritative domain. In Section 5.3.3 we will further investigate to determine how many of these registrations are malicious. The other 40,856 (93%) registered typosquatting NSDOMs were not currently used as a nameserver by a domain in our dataset. These may also be false positives where the similarity to top NSDOMs is a coincidence or a defensive registration, however, they could potentially be *proactive* nameserver typosquatting attacks. That is, a typosquatting NSDOM is predatorily registered, waiting for a domain to be misconfigured in the future. Given the number of new customers served by some of the largest nameservers, a well chosen proactive registration could pay off in the long term.

Assessing current abuse

To determine whether the registered typosquatting domains mentioned above are truly malicious or false positives we send specific DNS queries to each one

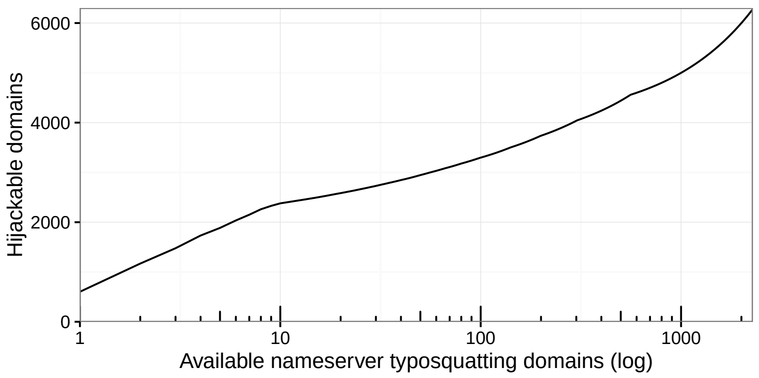


Figure 5.3: The amount of domains an attacker can hijack by registering a number of available typosquatting NSDOMs.

and analyze the responses. More specifically, we request the A record for a target domain from both the typosquatting nameserver and the target's authoritative nameserver and compare the responses. The typosquatting nameserver can either reply with a *Rogue* IP (i.e. one that differs from the one given by an authoritative nameserver), a *Matching* IP (the same one given by the authoritative nameserver), or *No Response*. Cases where the authoritative nameserver does not respond, but the typosquatting one does are ignored since we are left without a point of comparison. We argue that a rogue response suggests active abuse.

We further analyze these responses from the *Rogue* category by making an HTTP request to the rogue IP addresses with the Host header set to the target domain, effectively mimicking a user accidentally ending up at the page due to a nameserver misconfiguration. This allows us to categorize the types of abuse used by the malicious nameservers. This was a semi-manual process. First, we established a category for a certain webpage, and afterwards we gathered other instances that lead to the same or very similar page (by grouping them by URL and IP address).

Exploitive registrations We choose a target domain for each of the 3,233 potentially exploitive typosquatting NSDOMs by selecting the highest ranked domain (according to Alexa) among all those configured to use that NSDOM.

To reduce false positives, we conservatively consider only those typosquatting NSDOMs where the target domain has NS records for both the authoritative, as well as the typosquatting NSDOM. Hence, we exclude the cases where the target domain is only configured to use typosquatting NS records. The reasoning here is that a domain would not correctly resolve if all its NS records are erroneous and domain owners would notice the mistake immediately. A possible exception to this would be if an attacker had set up a stealthy proactive typosquatting NSDOM as a recursive resolver to keep newly misconfigured domains fully operational. Nevertheless, we decide to consider these cases as likely false positives. Additionally, this filtering step also ensures that we can compare the responses of a typo and authoritative nameserver during our DNS tests.

There are 86 typosquatting nameservers serving rogue replies as shown in Table 5.2. These 86 malicious nameservers are capable of hijacking traffic from 423 domains including dependencies. After close inspection we find that 26 of those nameserver typosquatting registrations are all related to the same actor that performs the targeted nameserver hijacking attack on `polishop.com`. These nameservers allowed zone transfers and by probing with selective AXFR queries we find that they solely contained zone files for misconfigured domains, with every domain's A record pointing to the same IP address. This demonstrates that these malicious setups are specifically targeting those domains with erroneous `NS` records. When making an HTTP request to this rogue IP, our instrumented browser was shown parking pages (Table 5.3). Although parking pages are already known to be potentially harmful to end users [198], these can also be a front for dormant malicious activity [117].

The 391 typo domains that did not respond may not be acting maliciously at the time of our resolutions, but there is a clear security risk to the misconfigured domains since they are pointing to a third party that is not their intended authoritative domain. For the 35 nameservers with matching responses, while they appear benign, there is always the potential for attackers to lay dormant, purposefully returning the appropriate IP address, thereby avoiding detection of the hijacked nameserver until a time of their choosing.

Proactive registrations To test the 40,856 unused typosquatting NSDOMs, we choose the target domain by selecting the highest ranked domain using the squatted authoritative NSDOM from which the typo was derived. While there was no response from most of these domains, among the 3.6% nameservers that replied, 86% of them served rogue responses for the target domain (Table 5.2). HTTP requests to the rogue IPs, resulted in a wide variety of observations (Table 5.3). The most frequent cases were parking, empty, error and scam pages. By looking at WHOIS data, we also encounter one defensive registration though it is unclear whether it was registered to protect the website of the NSDOM, the nameserver itself, or both.

Since the typosquatting NSDOMs in this category are not found in any NS records in our dataset we assume they are not authoritative for any domain, however, 204 actually returned the same IP address as the authoritative domain.

	Rogue	Matching	No Response	Other
Typo (Exploited)	86	35	366	25
Typo (Proactive)	1,295	204	39,218	139
Bitsquatting	522	85	19,141	108

Table 5.2: Categories of registered typo/bitsquatting NSDOMs based on their responses to target DNS queries.

	Empty page	Defensive	Security Co.	For Sale	Other	Parking page	Scam page	Error page	Redirection
Type (Exploited)	1	-	-	-	-	77	8	-	-
Type (Proactive)	210	1	15	29	7	914	48	64	7
Bitsquatting	72	1	115	21	5	265	5	36	4

Table 5.3: Web pages returned from the rogue IP addresses.

Since there is little incentive for a typical nameserver to answer queries for domains outside its zone, opening that server up to DoS attacks, this is suspicious behavior which may indicate the type of stealthy proactive attackers who wait for typos to be made and avoid detection until they choose to initiate an attack. We do not expect these to be defensive registrations because it is more likely that a defensive domain would either not respond or delegate to the correct nameserver rather than answering with the correct IP address itself. Finally, while it is possible that some of these typosquatting NSDOMs are used by domains outside of the 5 TLDs in our dataset, we consider it suspicious that they answer correctly for our target domains.

5.3.4 Measuring vulnerable cases

In order to assess the potential impact of nameserver typosquatting from an attacker's perspective, we registered six nameserver typosquatting domains, as listed in Table 5.4. We partly anonymize the presented domain names in order to prevent exposure of vulnerable entities. Four of these were known to be unexploited. More specifically, we were aware of 47 domains that were currently misconfigured to use these four NSDOMs. Therefore, we expected to nearly instantly receive DNS requests to these nameservers. We also made two proactive registrations. For these NSDOMs, we had no record of them being used as nameservers in the gathered TLD xzone files.

Experimental setup

Our experiment mainly aims to gauge the prevalence of hijack-able DNS resolutions. First, we intend to measure the number of DNS requests that are made to typosquatting NSDOMs. Second, we aim to determine which misconfigured domains names are effectively resolved by contacting our nameserver in error. Meanwhile, we want to minimize the impact of our measurements for the clients resolving those domains.

In order to obtain the necessary data, we adopt a specific setup, as illustrated in Figure 5.4. To explain this setup, assume we have registered a typosquatting NSDOM, `typo-ns.com`, and there exists a domain name, `misconfigured.com` that has listed `ns.typo-ns.com` in its NS records. Therefore, when a recursive resolver tries to resolve `misconfigured.com`, the `com` TLD nameserver will point the resolver to `ns.typo-ns.com` (1). Instead of simply setting up `ns1.typo-ns.com` with a glue record, we introduce an additional nameserver under our control on a different TLD, namely `ns.m1.xyz`, which we refer to as NS M1. As a consequence, the resolver has to launch a request to NS M1 (3). This effectively creates the nameserver dependency scenario described in Section 5.2.3. As NS M1 is authoritative for the `typo-ns.com` zone, the resolver is forced to query it to get the IP address of `ns.typo-ns.com` (5), allowing us to log that a request for `ns.typo-ns.com` has been made. Afterwards, the resolver finally obtains the IP address of `misconfigured.com`'s nameserver (NS M2) and will subsequently make a request to it (7). At NS M2, we are able to log that a request for `misconfigured.com` is made, completing the log for that resolution.

In order to gather information concerning the clients behind recursive resolvers, we enable ECS (EDNS Client Subnet) [43] on both NS M1 and M2.

Ethical Considerations. To minimize the negative impact of our experiments we set the TTL of records for the domain names we registered to only 5 seconds. We also chose not to respond to requests for domains names we did not control. As a result, the final request to the M2 nameserver for `misconfigured.com`'s IP address will timeout, just as it would have when the typo was unexploited. We used ECS in our experiments to obtain IP information of incoming requests, but this only allowed us to observe the /24 subnet for a small number of queries, maintaining clients' anonymity.

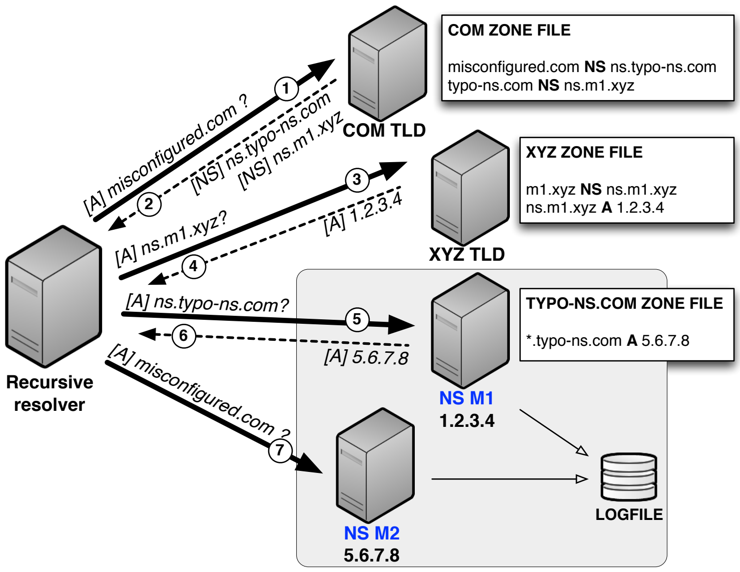


Figure 5.4: Experimental setup for monitoring the resolutions to typosquatting nameservers. The servers in the gray area are under our control.

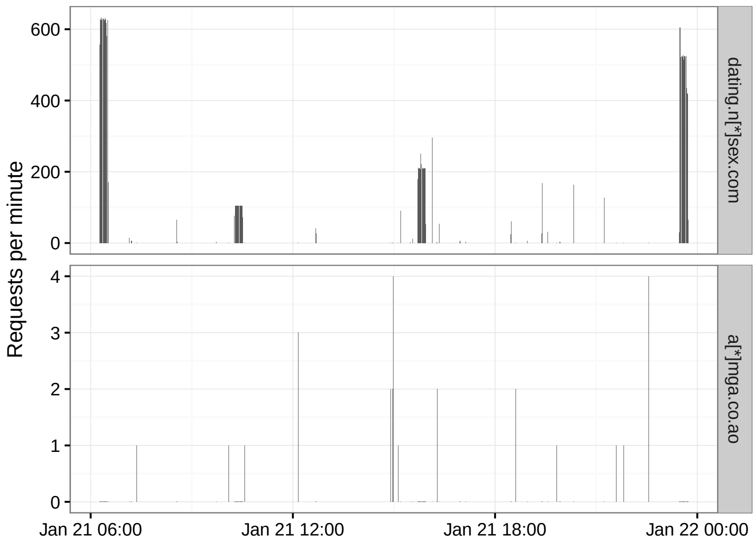


Figure 5.5: Requests per minute to typosquatting nameserver for two different misconfigured domains.

Findings

Over a one month period (Dec 22, 2016 - Jan 22, 2017), we received 734,300 DNS requests on NS M1 for all six registered typosquatting nameservers domains (step 5 in Figure 5.4). For the "missing-dot" typos (e.g. `ns[*]luehost.com`), there is generally only one nameserver queried, as that typo is specific to a particular subdomain. For the other cases, as shown in Table 5.4, we find that multiple nameservers on different subdomains are queried for a single typosquatting domain.

We previously determined that there were 47 domains in our dataset that were erroneously using one of our registered typosquatting NSDOMs. On NS M2, we logged resolutions for all of these expected victim domains, confirming that a typosquatting nameserver can effectively compromise all misconfigured domains. More specifically, we logged 3,013,420 "follow-up" DNS requests (step 7 in Figure 5.4) for those 47 domains, averaging to over 2,000 DNS requests per domain per day. The difference in the number of requests logged at NS M1 and M2 is influenced by the TTL and other factors previously discussed in Section 5.3.2. Interestingly, one of the two proactive registrations (`domaincon[*].com`) did receive requests, either for domains from different TLDs or for domains that were misconfigured afterwards. Other typo NSDOMs also observed requests for additional domains that suffered from temporal misconfiguration. For example, we recorded 342 queries for `p[*]hex.com` over the course of four days (Jan 18-21) while one of its NS records was mistakenly configured to `ns[*]luehost.com`.

We further record requests for a plethora of services and subdomains. For instance, we received 46 requests for DKIM public keys and 79 requests for DMARC records.

We want to note that the six experimental nameserver typosquatting registrations in this experiment were not chosen to simulate the maximum impact of an attacker, but rather to obtain diverse and representative measurements. An attacker could target more profitable cases, as described in Section 5.3.3.

The most frequently resolved FQDN for each registered typosquatting NSDOM is shown in Table 5.5. Based on WHOIS data, at least the five most resolved domains using `ns2.[*]tal.co.uk` are all owned by the same entity. We further analyzed the requests of one of these domains, `dating.n[*]sex.com`, on January 21, 2017, the day we recorded the most queries. Several abnormal characteristics come to light. As displayed in Figure 5.5, we witnessed several intense bursts of requests lasting for exactly 15 minutes each time. The request rate stays nearly constant during such a burst, but varies from 100 to over 600 requests per minute overall. Moreover, if we look at ECS information supplied by some requests

Authoritative NSDOM	Typosquatting registration	N° of expected victim domains	Queried subdomains
uniregistrymarket.link	ns[*]niregistrymarket.link	19	-
krystal.co.uk	[*]tal.co.uk	11	ns1, ns2
hostgator.com	ns[*]ostgator.com	16	-
bluehost.com	ns[*]luehost.com	1	-
domaincontrol.com	domaincon[*].com	0	ns50, ns74, ns78
dnspod.net	f1[*]nspod.net	0	-

Table 5.4: Registered nameserver typosquatting domains and the subdomains that were queried.

Requested name	Typo NS record	Requests
www.o[*]mes.net.	ns2.[*]tal.co.uk	738,581
[*].40.12.in-addr.arpa	ns74.domaincon[*].com	81,964
g[*]ong.com	ns[*]niregistrymarket.link	36,285
a[*]mga.co.ao	ns[*]luehost.com	1,177
p[*]tor.xyz	ns[*]ostgator.com	92
-	f1[*]nspod.net	-

Table 5.5: The most queried name for each typosquatting nameserver registration during 31 days.

(only 1%), we find that 83% of queries were made from IP address ranges belonging to 9 different hosting and cloud infrastructure companies. In other words, these requests are not coming from human website visitors, but from hosted servers. This kind of automated, coordinated and distributed suggests a misconfigured botnet infrastructure. In contrast, the bottom part of Figure 5.5 shows the requests pattern of a regular domain that was misconfigured.

Interestingly, the most requested name for `domaincon[*].com` is an inverse address. The typo is present in the zone file for an IP address space managed by AT&T of which the reverse DNS lookups are partially delegated in error to `ns74.domaincon[*].com`. This peculiar case involves different possibilities than a regular DNS query. It would allow an attacker to return false hostnames for IP address owned by another organization, allowing for instance denial-of-service attacks by associating the IP address with blacklisted domain names connected to malware or spam.

5.3.5 Summary

In this section we explored the potential exploitation of nameserver typosquatting. We found 6,213 unexploited misconfigured domains available in the wild and showed that a large number of them could be compromised with less than ten typosquatting registrations. 682 additional domains were found to be exploitable not through any fault of their own, but because the nameservers they rely on made typos. 86 currently registered typosquatting NSDOMs actively reply with rogue IP addresses, impacting 423 misconfigured domains. Moreover, we discovered that there exist many more proactive typosquatting registrations with 1,295 of them also responding with rogue IP addresses.

By registering 6 of our own typosquatting NSDOMs we successfully hijacked traffic from 100% of the 47 misconfigured domains pointing to our nameservers, recording more than 3 million DNS requests for those domains over a one-month period. We also found evidence of new temporary misconfigurations during this period, proving that there is value to proactive typo registrations.

5.4 Nameserver bitsquatting

The second attack described in this chapter, nameserver bitsquatting, is related to the typosquatting attack. However, the main premise of this attack is not human error, but hardware malfunction. As in Section 5.3, we first describe the attack vector and its impact, followed by an analysis of registered bitsquatting NSDOMs and an experiment to measure bit-flipped DNS resolutions to nameservers.

5.4.1 Attack vector

Bitsquatting is the act of registering domain names to receive unintentional traffic caused by random bit-flip errors in the memory of devices and computers. These bit-flips occur due to faulty hardware, extreme temperatures or radiation, and thus are by nature rare and unpredictable. However, bitsquatting is a documented phenomenon and multiple studies have been published reporting on its impact [55, 144], as well as conditions and causes [187, 170]. In DRAM, bit errors are typically mitigated with Error Correcting Codes (ECCs). Although the adoption of these techniques is common, they are still often missing in consumer devices and even in DRAM-containing components of enterprise class systems such as NICs and hard drives [55].

If these bit-flips alter the in-memory representation of a domain name, it can effectively lead to a request to another domain name. For instance, a bit-flip can cause a computer to accidentally connect to `twitte2.com` instead of `twitter.com` (the binary ASCII code for “2” is **0011 0010**, which is a single bit-flip away from **0111 0010**, the ASCII code for “r”). A study by VeriSign [205], reported that about one in every 10^7 – 10^8 DNS resolutions suffers from a bit-level error.

In previous studies, researchers observed requests to bitsquatting domain names that occurred before, as well as during DNS resolution. However, these studies focussed on bitsquatting connections to a web server’s domain name. In this chapter, we analyze the possibility of bitsquatted DNS requests to nameservers. NSDOMs are involved in more DNS requests than “regular” domain names, making them statistically more exposed to bit-flips. Furthermore, the impact of nameserver bitsquatting is potentially larger due to cache poisoning. We identify three specific requirements for a bitsquatting nameserver attack to enfold:

1. The bit-flip must corrupt the domain in a NS record that is or will be accepted by the recursive resolver.
2. The attack can only occur during a DNS resolution of a domain name whose nameserver is in another TLD zone. When they are in the same TLD zone, the nameserver’s IP address is returned immediately via glue records and no actual lookup for the NS records is made.
3. The bit-flip cannot occur during transmission, since a mismatch between the DNS request and response in the question section will be rejected by the resolver [3].

5.4.2 Amount of traffic affected

Previously studied bitsquatting attacks, as first described by Dinarburg [55], affect only a single domain name at a time. When a rogue IP address for a domain name is cached, it can affect multiple clients for a prolonged period. Although gauging the probability of bitsquatting vectors is extremely hard, we argue that nameserver bitsquatting could be more prevalent and more impactful than its previously studied counterpart.

First, as NSDOMs are often shared by many domains, NS records are, on a global scale, involved in a lot more DNS requests than a single domain name. Thus, a

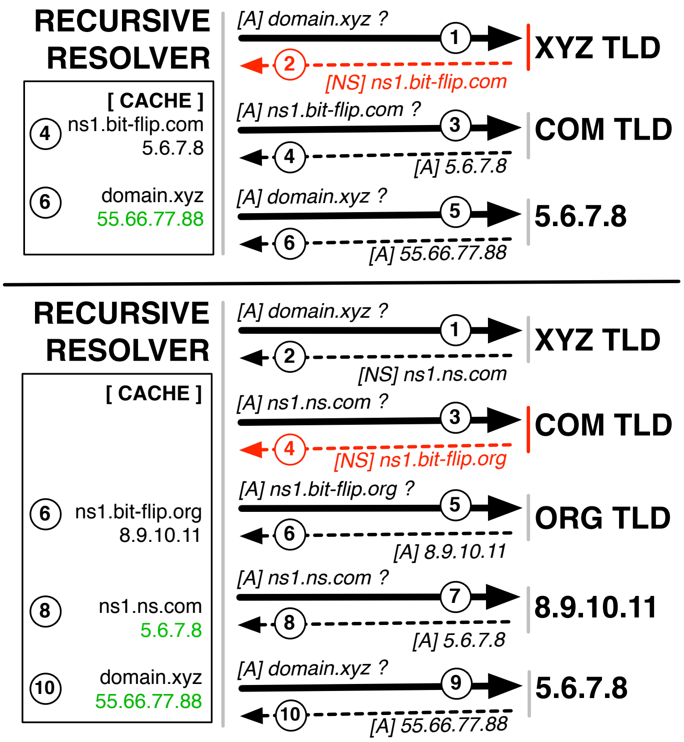


Figure 5.6: Possible bit-flip events during recursive resolution involving an independent (top) and a dependent nameserver (bottom). Red indicates where bit-flips occur and green signifies poisoned cache entries.

bit error is in general more likely to corrupt the in-memory representation of a widely-used nameserver than that of a website's domain name.

Second, instead of just poisoning the cache entry of a domain name, the entry of a nameserver can be poisoned. In that case, the attack will affect all domains of that victimized nameserver (for all the clients of the poisoned recursive resolver). However, this is only possible in the dependent nameserver scenario, as presented in Section 5.2.3. More specifically, as shown in Figure 5.6, when a second nameserver has to be queried (step 5) to retrieve the IP address of the dependent nameserver (7), an opportunity arises to poison the cache entry for the dependent nameserver (8).

5.4.3 Assessing current abuse

Dataset

We generated 605,965 domain bit-flips from the top 10,000 NSDOMs and their dependencies as in the work by Dinaburg [55]. As in Section 5.3.3, we included the subdomains of the NSDOMS since the first dot (**0010 1110**) may bit-flip to an ‘n’ (**0110 1110**) creating a new second level domain. 586,109 (97%) of bit-flipped domains were available for registration.

Finding malicious cases

For the 19,856 registered bitsquatting domains we investigate how many of them are malicious bitsquatting domains and how many are false positives. The bitsquatting scenario is similar to the proactive typosquatting in that the NSDOM is not necessarily actively used by any domains, but the attacker is betting that there will be bit-flips which will lead to their NSDOM. Therefore, we use the same methodology as in Section 5.3.3 to test the bitsquatting domains. The results of the DNS queries for the target domains are shown in Table 5.2. We found the categories are proportionally similar between bitsquatting and proactive typosquatting with 3.1% of domains set up as nameservers and 86% of those nameservers serving rogue IP addresses. There is some overlap of NSDOMs which were both bitsquatting and proactive typosquatting domains, but 433 of the 522 *Rogue* NSDOMs were uniquely bitsquatting names. This indicates that attackers value bitsquatting in addition to typosquatting despite its less predictable nature. These 522 malicious NSDOMs are capable of capitalizing on potential bit-flips from at least 52,888,224 distinct domains (not taking into account dependencies).

Table 5.3 shows the results of HTTP requests (with the host header set to the target domain) to the rogue IP addresses served by the malicious bitsquatting NSDOMs. Compared with the same categories for proactive typos, the number of domains associated with a security company stands out. All 115 of these NSDOMs were registered by the same person which is a significant investment in bitsquatting.

As we discussed for proactive typos, it is suspicious behavior for a nameserver to respond with the correct IP if it not listed in any NS records. We find that 48 of the 85 *Matching* bitsquatting NSDOMs do not have any NS records pointing to them and therefore fall into this suspicious category.

5.4.4 Measuring bit-flip occurrences

We registered ten distinct bitsquatting variations of popular NSDOMs, as listed in Table 5.6. Nine of these have other nameservers dependent on them, creating an opportunity for cache poisoning the nameserver entry, as described in above.

In order to monitor which bitsquatting variations of nameservers are contacted and log the domains that are being resolved using them, we deploy the same experimental setup that was used for the nameserver typosquatting measurements (Section 5.3.4), involving two measurement nameservers NS M1 and M2. At NS M1 we receive requests for the bitsquatting nameserver, while at NS M2 we record requests for domains using that nameserver. We evaluate the data for a one-month period (Dec 22, 2016 - Jan 22, 2017).

Ethical Considerations. The same measures that were applied in the experiments of Section 5.3.3 were used again here to minimize the impact of our experiments. We set the TTL of our responses to only 5 seconds to prevent long term cache poisoning, and we did not respond to requests for domain names we did not own, instead allowing them to timeout as they would in the case of an unexploited bit-flip.

Findings

We witness resolutions for each bitsquatting NSDOM on NS M1, though the vast majority are queries for the second-level domain or common subdomains, such as `mail` or `www`, presumably made by crawlers and DNS scanners. For 3 out of 10 bitsquatting registrations however, we receive requests to very specific subdomains on which nameservers reside on the authoritative NSDOM. For instance, we observed resolvers requesting the A record of `dns9.hi[*].com` and `ns4.p18.dy[*].net`. The authoritative counterpart of those NS records are used by 3,210,418 and 9,658 domains respectively. In total we received 33 requests to specific nameserver subdomains on the bitsquatting NSDOMs over the one-month experiment, averaging to about one per day. For most requests we did not receive a follow-up request on NS M2. We assume that either a correct nameserver was queried in parallel and delivered a faster response than us, or that our response was rejected due to a question section mismatch at the resolver's side.

For three requests, however, we did receive a follow-up DNS request on NS M2 i.e., an attempt to resolve a certain domain name using the bitsquatting nameserver. These observations are shown in Table 5.7. The first case occurred on December 22, 2016. An IP address of a Pakistani ISP requested two

Authoritative NSDOM	Bitsquatting registration	Dependants
domaincontrol.com	domain[*].com	●
dynect.net	dy[*].net	●
hichina.com	hi[*].com	●
1and1-dns.org	[*]-dns.org	-
ui-dns.org	[*]ns.org	●
dnsv2.com	d[*].com	●
dynamicnetworkservices.net	dynamicnetwor[*]s.net	●
ultradns.org	ult[*].org	●
verisigndns.com	veri[*]s.com	●
worldnic.com	[*]nic.com	●

Table 5.6: Registered nameserver bitsquatting domains.

nameserver subdomains of `domain[*].com` . The first is `pdns03`, where its authoritative counterpart is configured as a nameserver by 194,594 domains. We subsequently receive a follow-up request for `odin.g[*]oo.mx`, on NS M2. The domain name `g[*]oo.mx` does indeed have NS records pointing to `pdns03.domaincontrol.com` and `pdns04.domaincontrol.com`, confirming that the resolution was caused by a bit-flip. Concerning the second subdomain that was queried, `pd.304`, we deduce that this is a query for the second nameserver (`pdns04`), but containing two additional bit-flips ("n" to "." and "s" to "3").

The next two cases are very similar to each other and occurred on January 17 and 21, 2017. In both observations, we received a query for a nameserver subdomain of `domain[*].com` made by an IP address of Google's public DNS service. Afterwards, we observed three consecutive queries for a domain name on M2. As we do not respond to these queries, presumably, these are two retries of the same query. Although the source IP address differs for each of these requests, they all belong to the same Google DNS infrastructure located in Singapore [79]. Moreover, the ECS information provided in the initial, as well as the follow-up requests all match up, further confirming that all requests are part of a single DNS resolution. In both cases, the final requested domain names (`u[*]ock.global` and `s[*]ppy.global`) are using the authoritative counterpart of the bitsquatting nameserver.

For all three observations, the requested domain name is on a different TLD than its nameserver, satisfying the criteria for a successful nameserver bitsquatting hijack (Section 5.4.1). Since we are minimizing the impact of our measurements by not replying to the final requests and setting the TTL of the nameserver to just 5 seconds, we are unable to observe the true impact of cache poisoning.

Time	From	ECS (Hash)	NS	Requested name	
19:02:11.4	202.[*].[*].33	-	M1	A	pdns03.domain[*].com.
19:02:11.7	202.[*].[*].33	-	M1	A	pd.304.domain[*].com.
19:02:11.9	202.[*].[*].33	-	M2	A	odin.g[*]oo.mx.
06:58:37.1	74.125.190.132	0baf1a2 /24	M1	A	ns34.domain[*].com.
06:58:37.3	74.125.190.147	0baf1a2 /24	M2	MX	u[*]ock.global.
06:58:39.0	74.125.190.145	0baf1a2 /24	M2	MX	u[*]ock.global.
06:58:40.7	74.125.190.12	0baf1a2 /24	M2	MX	u[*]ock.global.
04:03:40.5	74.125.190.141	e814a06 /24	M1	A	ns11.domain[*].com.
04:03:40.7	74.125.190.8	e814a06 /24	M2	A	s[*]ppy.global.
04:03:42.4	74.125.190.16	e814a06 /24	M2	A	s[*]ppy.global.
04:03:44.1	74.125.190.143	e814a06 /24	M2	A	s[*]ppy.global.

Table 5.7: Observed nameserver bitsquatting occurrences.

5.4.5 Summary

In this section we investigated the potential of nameserver bitsquatting. We found 522 currently registered bitsquatting NSDOMs responding with rogue IPs with the potential to abuse bit-flips that occur from 52,888,224 domains.

By registering 10 bitsquatting NSDOMs we were able to verify that bit-flipped requests, while rare, do occur. Within one month we observed 3 legitimate bit-flipped requests which would allow for hijacking and cache poisoning of the requested domain name.

5.5 WHOIS email hijacking

In this section, we introduce the techniques allowing for take-overs of entire NSDOMs by targeting email addresses listed in the WHOIS records, and evaluate their applicability.

5.5.1 Attack vector

Nameserver domains can be hijacked by abusing out-of-date and inaccurate information in the WHOIS records. The idea is that either access can be gained to the registrar's web control panel, or an ownership transfer of the victim domain name can be issued. Both cases allow an attacker to set up a malicious nameserver using the victim's domain. Consequently, the attacker will be able to hijack all domains dependent on that nameserver. The WHOIS field that is the most ripe for abuse is that of email contacts. Typically, the registrant contact is the person who created the account with the registrar and their email is trusted for retrieving forgotten usernames and resetting forgotten passwords.

An attacker can hijack the email accounts listed in a WHOIS record in two ways. First, some webmail providers will expire an account and make the address available again when a user does not log in for a long period of time. If the email listed in the WHOIS records is an expired webmail account, then the attacker can merely register that address again with the webmail provider. There are known cases of this type of attack. For instance, in 2009, an attacker was able to steal internal documents of Twitter by re-registering an expired Hotmail account as a way of gaining access to a Twitter employee's primary GMail account [50].

Second, if the email account listed in the WHOIS resides on a domain which has been allowed to expire, then an attacker can register that domain name and set up a mail server to receive emails destined for that domain. As soon as attackers control the email address they can initiate a password reset with the registrar and set a new password through the link sent to the stolen email address. If two-factor authentication is not set up, the attacker will gain access and have full control over the nameserver domain.

An attacker can make it more difficult for the original owner to regain control of their domain by transferring it to a different registrar. Once a domain has been transferred away, the original owner is left with little recourse [176]. In order to transfer a domain, an attacker needs to provide an authorization code (also called an EPP code) which is obtained from the original registrar either via a web-accessible control panel or through email from the admin email contact. ICANN requires registrars to respond to such email requests within five days, but the registrar may still force the owner to log in to obtain the auth code. Once the attacker has the auth code, they can provide it to the new registrar to initiate the transfer process. The new registrar will send an email to the admin contact in the WHOIS and expect a response to verify consent to the transfer. Auth codes are required for any TLDs managed by ICANN [92]. ccTLDs (managed by registries in each country and not by ICANN) may have more or less restrictive policies regarding transfers, but `.fr` and `.ca`, the two ccTLDs in our list of vulnerable domains, do require auth codes [6][35].

5.5.2 Finding vulnerable nameservers

To find nameservers vulnerable to email-based hijacking, we began by obtaining the WHOIS records for the top 10,000 NSDOMs and their dependencies using the Whoxy API [209]. From these records, we extracted the email addresses

High Risk		Medium Risk		Low Risk	
scs[*]ver.info	394	fsi[*]ebs.net	461	pul[*]ion.fr	3,642
log[*]rks.net	565	bla[*]sun.ca	5,542	max[*]ech.com	1,912
nic[*]rup.com	1,934	[*].amsterdam	2,594	ube[*]tor.com	2,205
idc[*]com.net	689			web[*]ost.net	546
iqn[*]ion.com	1,019				
par[*]ost.net	1,425				
Affected	6,021		8,596		8,302
Dependents	29		16		112
Total	6,050		8,612		8,414

Table 5.8: NSDOMs with outdated WHOIS records and the number of domains dependent on them, categorized by their risk of being hijacked.

for the registrant, administrator, technical, and billing contacts. Using the Domainr API [57], we found that 11 of the domains used in these email addresses were available for registration. To find expired webmail accounts we used the Email-Hippo [63] validation API to filter active email addresses. For each email account that Email-Hippo flagged as "undeliverable", we checked whether it was available for re-registration. To that end, we developed a Selenium-based crawler that attempts to create a new email account using, as our address of choice, each of the flagged emails. If a webmail service did not present us with an availability error, that meant that that email address was available for registration. Note that in our experiments we took advantage of the UI present in the registration pages of all modern webmail providers which, through the use of appropriate AJAX calls, provides immediate feedback to the user as to whether the selected email address is available and not taken. As such, we do not need to actually register an email account in order to verify whether it is available. This allows us to ethically quantify the abuse potential of this attack vector without exploiting it and without creating any accounts on webmail providers. We found two such cases of previously existing addresses, both on `hotmail.com`, which had expired and were available to re-register.

5.5.3 Potential impact

In total, we found 13 NSDOMs with vulnerable WHOIS emails. We split them into 3 categories based on severity. Table 5.8 shows the nameserver domains by category. For each nameserver, the number of domains which use it in an NS record is given.

Over 6,000 domains could be impacted by hijacking the six domains in the

High Risk category. The High Risk category includes all domains where the vulnerable email address was the registrant contact. If an attacker uses the registrant email to gain access to the registrar's control panel then they have full control over the domain including the ability to change all other email contacts in the WHOIS record.

The *Medium Risk* category includes domains with a vulnerable admin email, but not a vulnerable registrant email. Even if it does not directly grant access to the account, control of the admin email could be used in an attempt to request an auth code from the registrar. Depending on how strict the registrar is about obtaining auth codes, this may require some amount of social engineering. Control of the admin email provides the appearance of authority which would aid such an attempt. Since the admin email is the first point of contact for domain transfers, an attacker could transfer the domain if they are able to obtain an auth code or if they are dealing with registries which do not require auth codes for transfers of particular TLDs.

The *Low Risk* category includes domains with vulnerable emails which are not admin or registrant contacts. It is unlikely that these emails could be used to gain access to the account or transfer the domain. However, there is still some amount of trust that comes along with being listed in a domain's WHOIS. For example, when obtaining an SSL certificate for a domain, certificate authorities, such as StartSSL [179], allow one to prove ownership of the domain using email addresses found in WHOIS. This assumption that the owner of an email in the WHOIS must be the owner of the domain makes any of these emails useful for social engineering. Therefore, even if attackers are not able to altogether hijack these Low-Risk domains, they could certainly request SSL certificates for them and abuse them in MITM scenarios.

Ethical Considerations While we identify vulnerable NSDOMS, we do not register their emails or attempt to compromise any of them. We have reported the WHOIS inaccuracies for the expired emails to ICANN [93] who will forward them to the appropriate registrars.

5.6 Security practices of nameservers

Following the idea that a domain name's security is entirely jeopardized when (the connection to) the nameserver is compromised, we set out to explore the security risks of the most widely used nameservers. To this end, we evaluated the patching practices of 312,304 nameservers (i.e., all hosts behind the fully-qualified domain names of the top 10K NSDOMs and the parent servers on which they depend), using patching as a proxy variable for a server's overall security. This decision is based on the assumption that a security-conscious administrator will be determined to update the DNS software to a version for which there are no known vulnerabilities.

5.6.1 Analysis

To determine whether the deployed DNS software is up-to-date, we obtained version information that is being exposed through the banners on port 53, both for TCP as well as UDP. By analyzing these banners, we found that, by far, BIND is the most popular software for DNS servers – out of the 165,012 nameservers for which we received a non-empty banner, 78.33% were using BIND. Because of this uneven distribution of DNS software in the domain name ecosystem, we focus our analysis on the patching practices in BIND.

Leveraging the information extracted from the banner, we tried to determine the exact version of BIND that was used. Surprisingly, only 9,032 nameservers (6.99% of all BIND servers) reported version information. Most likely, this is because it is considered a best practice to hide this data from attackers, making it harder for them to determine which exploit they could use. For the servers where we could extract the version information, we determined the release date of the employed installation, along with the number of days it had been outdated. As a point of reference, we used the release date of the latest vulnerability-free versions that were available at the time of our scan (versions 9.9.9-P6, 9.10.4-P6, and 9.11.0-P3). Using this information, we mapped out the distribution of nameservers by the number of days they were outdated, as shown in Figure 5.7. This graph clearly shows that the vast majority of the nameservers for which we could determine the version are running an outdated version of BIND. More precisely, 7,703 evaluated nameservers are vulnerable to a denial-of-service attack (CVE-2016-2776), for which an exploit is publicly known [186]. Even when being more conservative with regards to considering a version out of date, we still find 7,214 nameservers (79.87% of the BIND servers that returned version information) that are vulnerable to a second denial-of-service attack (CVE-2015-5477), for which an exploit is readily available in the Metasploit framework [159].

Lastly, we want to point out that because nameservers are a common building block typically shared among thousands or even millions of domain names, all these domains are directly affected by the security of their nameservers. The 7,214 nameservers we found to be vulnerable to the DoS exploit in Metasploit, are directly jeopardizing the availability of at least 1.28M unique domain names, out of which 514 operate as nameserver themselves. As a case in point, the

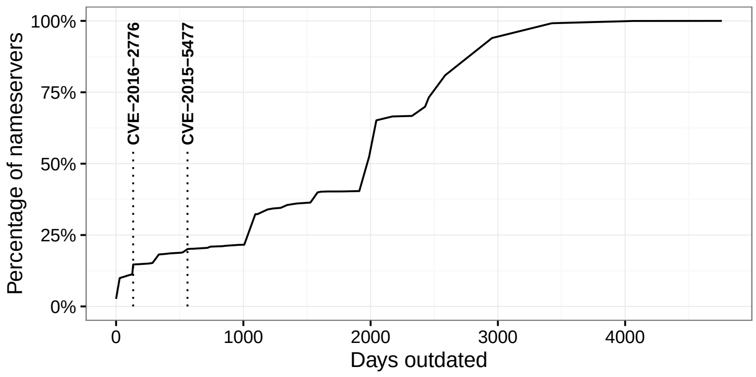


Figure 5.7: The cumulative distribution of nameservers by the amount of days their BIND version is outdated.

nameservers `yns1.yahoo.com` and `yns2.yahoo.com` report to use BIND version 9.4.3-P3, which was released in July 2009, making the software almost 8 years old. Unless the reported version is incorrect – we have no reason to believe so, as this would make the server more likely to attract unwarranted attacks – more than 646,290 domain names are put at risk by having these nameservers as their sole authoritative nameservers.

Ethical Considerations and limitations. The choice to obtain nameserver versions by reading their banners provided a non-invasive method to explore their security. This has a minimal impact on the nameservers and avoids the risk of more in depth security tests on live third-party systems. It comes at the cost of relying on some potentially inaccurate self-reported version information.

5.7 Discussion

Summary of findings. Hijacking domains through their nameservers is an extremely stealthy and powerful attack vector, capable of compromising domains en masse through, among others, MITM, domain-ownership verification and email attacks. In this study, we presented, for the first time, three nameserver attacks based on configuration errors and hardware issues that were evaluated on the top 10,000 nameserver domains.

We found that 6,213 domains can be hijacked, where 2,000 can be compromised with just six targeted registrations. Moreover, we raise the issue of nameserver dependencies and identify that 682 additional domains could be exploited due to a typographical error made by a third party, preventing the victims to directly locate and resolve the issue themselves. Furthermore, by evaluating the possibility of re-registering email addresses present in outdated WHOIS records of nameserver domains, we discovered that at least 6,050 additional domains are at high risk of compromise. In total, we *conservatively* estimate that 12,945 domains are directly or indirectly exposed to being hijacked through a configuration error related to their nameserver. In terms of current exploitation in the wild, we discover that attackers are already aware of these issues and register domains to exploit typos and bit-flip errors in NS records.

Lastly, our study of security practices of nameservers revealed that 7,214 nameservers are susceptible to an 8-year-old exploitable nameserver DoS vulnerability. Thereby, they are exposing 1.28M domains, enabling a large-scale denial-of-service similar to the October 2016 Dyn attack [212] without even requiring a botnet.

DNSSEC. DNSSEC is an extension to DNS which provides integrity to DNS by allowing nameservers to add digital signatures for their resource records and establishing chains of trust from the root zone to the authoritative nameserver. DNSSEC, when deployed properly, is capable of defending against the attacks described in this chapter.

We refer the reader to a more complete overview of DNSSEC [39], but for the purposes of this research the most important component is the DS record which is added to the domain's parent zone. This record tells the DNS resolver to expect signed responses from the next nameserver in the chain and contains a hash of the public key signing key for the next zone which is used to verify the source of the signed responses. When an administrator creates the DS record, they are adding a secondary reference to the correct nameserver beyond the standard NS record. If a victim domain points to a malicious nameserver, regardless of whether it was due to a mistyped NS record, a bit-flip, or stolen control of the nameserver domain, the attacker will be unable to correctly sign its responses. Without a proper signature generated by the key pairs that match the hashed public key in the DS record, a DNSSEC validating resolver will reject any response from the malicious nameserver.

However, in order for a full DNSSEC deployment to work properly there are several requirements involving responsibility and/or cooperation between domain owners, nameserver owners, registries, and ISPs. The complexity of deployment has led to slow adoption despite the age of DNSSEC [53]. For instance, in the `com` zone, only 0.56% of domains are signed at the time of writing [1].

Other defenses. Next to DNSSEC, we suggest the need for additional defenses requiring less cooperation between parties that can be adopted faster than

DNSSEC.

To reduce the number of misconfigured domains, registrars can check for typos by comparing all NS records that administrators are entering into the registrar's control panel. A warning could be shown when two records fit one of the typo models proposed by Wang et al. [204], extended with our specific adjustments for NS records (Section 5.3.3). Alternatively, registrars could require administrators to enter new NS records twice, similar to creating a new password. Known typosquatting and bitsquatting defenses, such as large-scale defensive registrations, the use of ECC-enabled DRAM, and filing abuse complaints, are also applicable in the nameserver realm. These kinds of countermeasures are especially interesting for large managed nameserver providers as they are most often victimized and have the means to execute them.

Regarding outdated WHOIS information, we suggest that registrars periodically verify the email addresses listed in the WHOIS records. To prevent validation of stolen email accounts, the verification process should involve the registrant authenticating with the registrar after clicking a link received on the email account. Additionally, we encourage the adoption of two-factor authentication for access to a registrar's control panel.

Finally, we argue that many of the problems discussed in this chapter are due to the inconspicuous nature of nameservers. While they are not directly visible to end users and often not even administrators, they do play an extremely crucial and security sensitive role for all Internet services.

5.8 Related work

To the best of our knowledge, this work is the first one that investigates the threat of hijacking domain names through nameservers by taking advantage of configuration errors and hardware issues. At the same time, in recent years, the research community has exhibited a rekindled interest in the Domain Name System because of DNS' central involvement in carrying out attacks.

5.8.1 Hijacking domain names

In 2015, Bryant showed that one could hijack domain names by iteratively requesting public IP addresses from AWS and identifying the domain names that were still pointing to these IP addresses because their owners had once utilized AWS for hosting purposes but had forgotten to update their DNS records after shutting down their virtual machines [27]. Liu et al. showed that these techniques could be abused to attack more public clouds and presented additional cases where websites could be hijacked by dangling DNS records [119]. Even though the authors position their work as capable of identifying all types of dangling DNS records, including dangling nameserver records (the subject of this chapter), they were only able to find four confirmed cases of dangling NS records in the Alexa top 1 million list. Contrastingly, in this study, we follow a top-down methodology where we start with popular nameservers (as defined by the number of domains utilizing them for resolutions) and identify not only the domains with dangling records, but also the current name squatting abuse of misconfigured domains. Furthermore, we consider the important role that nameserver dependencies play regarding these issues and highlight the ability to hijack nameserver domains via expired WHOIS email accounts.

In recent work, Bryant identified another type of dangling DNS vulnerability related to managed DNS providers [28] showing that he could hijack control of more than 120K domain names using the managed DNS services of public cloud providers while their owners had stopped using the hosting services of the aforementioned companies. While Bryant's techniques could be straightforwardly incorporated to identify more hijack-able nameservers, we chose to focus on techniques that were hoster-agnostic i.e., techniques that do not rely on the use of specific cloud providers.

5.8.2 Abusing expired domains

In 2012, Nikiforakis et al. discovered that popular websites contained stale, remote script inclusions that were referring to domains that had expired [142] allowing attackers to register them and deliver malicious JavaScript code. Starov et al. investigated the ecosystem of malicious web shells discovering that some webshells were requesting remote resources from expired domains which allowed researchers (or competing hacking groups) to learn about each new shell deployment and hijack their deployed shells [177].

In 2014, Moore and Clayton investigated the use of old domain names that belonged to US banks and financial institutions and were left to expire after merges or after the companies went out of business [130]. The authors discovered that these domains were often re-registered by attackers who abused the residual trust associated with these domains for SEO activities and malware spreading. Lever et al. analyzed six years of domain data and, among others, discovered that 8.7% of the domains that appear in public blacklists are re-registered after their former owners allow them to expire [115]. Schlamp et al. took the abuse of expired domains even further by showing that attackers can (and already have [168]) hijack entire autonomous systems by re-registering the appropriate expired domains present in the databases of Regional Internet Registrars, such as RIPE and ARIN [169].

5.9 Conclusion

In this chapter, we investigated the applicability of issues that are commonly thought of as end-host issues, to nameservers. We found that typosquatting, bitsquatting, and the expiration of email addresses can all be abused to hijack thousands of domain names through their nameserver records. By registering our own typosquatting and bitsquatting domains, we showed how attackers can receive millions of DNS requests by merely registering the appropriate domains. We quantified the thousands of BIND DNS servers that are running outdated software with known vulnerabilities and publicly-available exploits. Lastly we explained why poorly-adopted DNSSEC can defend against most of our described attacks, and suggested pragmatic approaches that registrars could adopt to reduce the likelihood of misconfigurations in the short-term.

Chapter 6

Bypassing cloud-based security providers

Preamble

This chapter discusses an important vulnerability of DNS-based cloud security services. This line of products offers protection against distributed denial-of-service (DDoS) and web application attacks such as SQL injections and cross-site scripting by filtering out malicious traffic in massive cloud infrastructures ("scrubbing centers").

One of the strengths of DNS-based cloud security products is that its deployment is particularly quick and convenient. Typically, it requires no more than a single change in the customer's DNS configuration to be deployed. These products have seen a massive adoption-rate which continues up to today. One of the most well-known company in this market, *Cloudflare*, was protecting over 7 million customers in December 2017 [38], 1.5 million more than the year before [71]. Other prominent players in this field are *Imperva Incapsula* and the cloud security products from *Akamai* [25].

The main crux of DNS-based cloud security is to protect and hide the *origin*, i.e. the customer's server that hosts the website. In essence, they employ a security through obscurity model: the customer's domain name no longer resolves to the IP address of the origin, it points to an IP address of the cloud security provider. This allows the large-scale cloud infrastructure to function as a reverse proxy for all incoming requests and filter out the malicious traffic before forwarding it to the origin. The Achilles heel of this approach is that the IP address of the origin must remain hidden. If an attacker is able to expose the actual location of the origin, he can directly attack the server, effectively bypassing the entire cloud security setup ("direct-to-IP attacks").

In 2015, we published a paper [200] in ACM CCS that evaluated and measured the security issues of *origin-exposing attacks* at a large scale. The outcome was alarming: we could retrieve the origin for over 70% of the websites protected by this type of products. The results of this paper received substantial media coverage [150, 133, 98, 44, 31] and responses of cloud security companies. These responses tried to raise awareness to their customers [215, 11], or argumented to switch to other security products not vulnerable to origin exposure [94].

Together with the paper, we created *cloudpiercer.org*, a free online vulnerability scanner that allows administrators to track down origin exposure for their own websites. As of March 20, 2018, our tool received 5,514 sign-ups for distinct domain names. 855 of which completed the entire verification procedure and successfully launched a scan that resulted in the delivery of origin exposure report. For 279 domains, we received more than one scan request. We assume that, in these cases, the administrators attempted to fix certain vulnerabilities that were found by the initial scan. Next, they verified whether the exposure was effectively mitigated by requesting an updated report. At the time of writing, the tool still regularly finds origins of cloud-protected websites.

Following the results of our research, several companies made changes to their cloud security products in line with our recommendations. For instance, Cloudflare stopped setting up a direct-to-origin subdomain by default for new customers soon after the publication. Instead, Cloudflare's control panel now actually warns users if one of their DNS records exposes the origin.

As for Incapsula, they ensured in their initial response to our research results that they were in the process of developing a solution called "IP Protection". This product was later released in March 2016 [175]. In essence, this solution follows the recommendation to assign a dedicated IP address of the cloud security provider to each domain they protect. This allows non-HTTP traffic to be routed through the cloud infrastructure, which –as confirmed by our study– greatly reduces the risk of origin exposure. To accomplish this, Incapsula uses a two-way GRE tunnel between their cloud infrastructure and the customer's origin, enabling the origin to transparently send and receive all sorts of packets on all ports.

More recently, in September 2017, Cloudflare announced their next step in origin protection, named "Warp" [80], now rebranded to "Argo Tunnel" [40]. This beta feature ships with an application that automates the setup process on the origin server. The tool installs a TLS certificate on the origin; establishes the tunnel by initiating a long-lived encrypted HTTP2 connection to Cloudflare's infrastructure; and configures the client's DNS settings (Cloudflare also serves as the managed DNS provider). Apart from the encrypted tunnel, the added benefit of this approach is that the origin does not have to be directly reachable via the Internet. Instead, the origin server can sit behind a firewall or NAT as the tunnel is initiated by an outbound connection from the origin. This enables more opportunities to adequately hide the origin server. The downside compared to solutions such as "IP Protection" is the lack of support for non-HTTP protocols.

In conclusion, our study raised awareness of this previously underestimated issue, while cloudpiercer.org helped administrators pinpoint their own vulnerabilities. Moreover, over the past two years, cloud security providers have been moving in the right direction and concrete initiatives have improved the possibilities to protect against origin exposure. However, although most ingredients are currently available, the DNS-based cloud security sector still lacks a comprehensive solution that addresses both non-HTTP traffic and non-public retrievable origins.

The further contents of this chapter are replicated from the paper titled "Maneuvering Around Clouds: Bypassing Cloud-based Security Providers" [200], which was published in the proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), in 2015. This work was done with the collaboration of other authors from KU Leuven and Stony Brook University. Thomas Vissers is the lead author of this paper.

Maneuvering around clouds: Bypassing cloud-based security providers

Abstract

The increase of Distributed Denial-of-Service (DDoS) attacks in volume, frequency, and complexity, combined with the constant required alertness for mitigating web application threats, has caused many website owners to turn to Cloud-based Security Providers (CBSPs) to protect their infrastructure. These solutions typically involve the rerouting of traffic from the original website through the CBSP's network, where malicious traffic can be detected and absorbed before it ever reaches the servers of the protected website. The most popular Cloud-based Security Providers do not require the purchase of dedicated traffic-rerouting hardware, but rely solely on changing the DNS settings of a domain name to reroute a website's traffic through their security infrastructure. Consequently, this rerouting mechanism can be completely circumvented by directly attacking the website's hosting IP address. Therefore, it is crucial for the security and availability of these websites that their real IP address remains hidden from potential attackers.

In this chapter, we discuss existing, as well as novel "origin-exposing" attack vectors which attackers can leverage to discover the IP address of the server where a website protected by a CBSP is hosted. To assess the impact of the discussed origin-exposing vectors on the security of CBSP-protected websites, we consolidate all vectors into CLOUDPIERCER, an automated origin-exposing tool, which we then use to conduct the first large-scale analysis of the effectiveness of the origin-exposing vectors.

6.1 Introduction

Although Distributed Denial-of-Service (DDoS) attacks have threatened the availability of online services for years, attacks are rapidly increasing in volume, complexity and frequency. Early 2014, the Network Time Protocol (NTP) was exploited in order to conduct amplification attacks[165] of previously unseen magnitudes, leading to multiple record-breaking volumetric attacks that reached up to 500 Gbps [195, 147]. Unfortunately, these powerful attacks are no longer exceptional cases. For instance, in 2014, there were four times as many attacks that crossed the 100 Gbps barrier as compared to 2013 [20]. Consequently, these massive attacks are now regarded as "the new normal" [105], an observation further confirmed by the frequent news reports of high-profile websites and web applications that become victims of such attacks [68].

Aside from advancing in strength and complexity, DDoS attacks are becoming increasingly accessible to the general public. The main cause is the rising popularity of websites offering DDoS attacks as a service, which enable non-technical users to launch DDoS attacks with the click of a button. These services, often called *booters* or *stressers*, allow their customers to orchestrate powerful DDoS attacks for just a few dollars through convenient, and user-friendly web interfaces [97].

To cope with the elevated risk and increased difficulty in fending off large DDoS attacks, several companies engineered highly capable, globally distributed networks that are able to deal with with DDoS traffic and malicious web requests. The resulting cloud-based defense infrastructure is then shared among the companies' customers. It is safe to assume that not all customers will be suffering from a large DDoS attack simultaneously, and thus companies can dedicate enough bandwidth and processing power to clients that are, at any given point, under attack.

Since several of these Cloud-based Security Providers (CBSPs) solely rely on changing the DNS settings of a domain name to reroute a website's traffic through their security infrastructure, the rerouting mechanism can be, in principle, completely circumvented by directly attacking the website's hosting IP address. Therefore, it is crucial for the security and availability of these websites that their real IP address, referred to as the *origin*, remains hidden from potential attackers. Past reports have claimed that the origin of CBSP customers can potentially be acquired through various methods, such as querying historical DNS data for a domain, and searching for subdomains that directly resolve to a server's real IP address [145]. Although these origin-exposing attack vectors have been known since 2013, the global extent of this issue has not yet been evaluated.

In this chapter, we assess the magnitude of this problem on a large scale, i.e., we evaluate the number of protected domains whose CBSP-based protection can be bypassed. First, we discuss eight existing as well as novel vectors that have the potential to expose the underlying IP address of a CBSP-protected web server. Next, we consolidate these vectors into CLOUDPIERCER, an automated origin-exposing tool. We deploy CLOUDPIERCER and conduct the first large-scale experiment where we evaluate 17,877 domains that are protected by five different CBSPs. CLOUDPIERCER uses a novel verification method to ensure that an IP address retrieved by the vectors is indeed the real origin of a website. After this verification step, we find that over 70% of protected domains expose their real IP address and, as a consequence, can be attacked directly, rendering the cloud-based protection service useless. Furthermore, we elaborate on the impact and prevalence of each exposing vector and discuss the feasibility of remediating the problem.

The main contributions of this research are the following:

- We provide a comprehensive overview of novel and previously known origin-exposing vectors that allow attackers to bypass CBSPs.
- We report on the first large-scale measurement of this crucial security issue and conclude that the majority of CBSP clients are at risk, while providing insights into which vectors are most widespread.
- We discuss the difficulties of mitigating origin exposure, while suggesting several effective countermeasures that can vastly remediate the problem.

6.2 Background

As Distributed Denial-of-Service (DDoS) attacks are becoming increasingly powerful, it becomes infeasible for websites to protect their own infrastructure. Even advanced, on-site, defense systems are rendered useless when the amount of traffic exceeds the processing capabilities of upstream devices or simply saturates the entire network connection. Furthermore, with the constant evolution of web application threats, there is also a need for increasing resources to fend off breaches. As a result, website owners turn to Cloud-based Security Providers (CBSPs) to protect their infrastructure. These companies reroute traffic from the original website through their network where malicious traffic is filtered before it ever reaches the network of their customer.

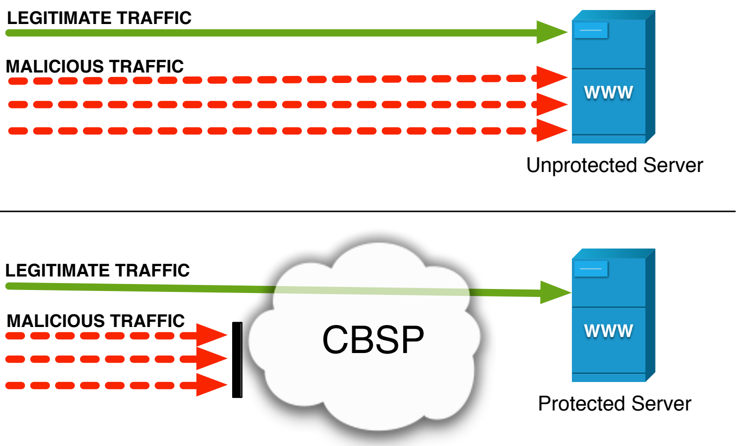


Figure 6.1: Cloud-based security. An unprotected server receives malicious traffic, potentially breaching the web server or denying service to the legitimate traffic (upper). Malicious traffic heading towards the protected server is absorbed by the CBSP, only allowing legitimate traffic to pass through (lower).

6.2.1 Modus Operandi of CBSPs

CBSPs act as reverse proxies for the web servers they are protecting. They inspect incoming traffic for various clients simultaneously, by routing it through their own distributed infrastructure. These cloud-based infrastructures, often referred to as *scrubbing centers*, act as highly-available traffic filters that are capable of absorbing extremely large volumetric DDoS attacks. Furthermore, they often integrate Web Application Firewalls (WAFs) to filter out malicious web application traffic, such as application-layer DDoS attempts, SQL injections and XSS attacks.

As depicted in Figure 6.1, all traffic towards a CBSP-protected web server, often referred to as the *origin*, is redirected through cloud-based scrubbing centers. After inspection of the incoming requests, only "clean traffic" is forwarded to the web server, effectively stopping attacks before they even reach the customer's premises.

Rerouting mechanisms

Several different strategies exist to route a web server's traffic through the cloud-based infrastructure. For instance, a website administrator can either opt for an *always-on* or for an *on-demand* strategy. The former redirects all traffic through the scrubbing centers on a permanent basis. The latter only starts redirecting traffic when necessary. Usually, this requires customer-premises equipment (CPE), that locally monitors incoming traffic. In case an attack is detected, this device initiates the redirection mechanism.

When traffic-redirection is active, there are two mechanisms to reroute traffic through the scrubbing centers. The first option is *DNS rerouting*, where an administrator changes the DNS settings of his website's domain name so that it resolves to an IP address that belongs to the CBSP. Normally, when a visitor requests a webpage, e.g., from `example.com`, his computer will first make a request to a DNS server to discover the corresponding IP address. Next, the visitor's browser can request the page from `example.com`'s web server using the discovered IP address. In the case of CBSPs, the visitors of the protected domain will receive an IP address of the CBSP's scrubbing center from the DNS server. Hence, the visitor will direct his requests to the scrubbing center, which in turn will transparently forward the legitimate requests to the origin, i.e., the actual web server of `example.com`.

Alternatively, a technique called *BGP rerouting* can be adopted. When the entity managing the website controls an entire /24 IP block, it can withdraw the BGP announcements for that block from its own routers. At this point, the CBSP can initiate BGP announcements for that IP range from their own network. Consequently, all traffic destined for the web server's IP address will start flowing through the CBSP's scrubbing centers. Since BGP rerouting is only available to entities that manage entire IP blocks and are able to install dedicated hardware, DNS rerouting has become the cloud-based security alternative for the masses [36].

6.2.2 CDNs as CBSPs

At their core, Content Distribution Networks (CDNs) are globally deployed services that increase the performance of websites by bringing static web content closer to users. The network usually consists out of a large set of geographically-distributed cache servers. This allows a CDN to quickly serve cached content from a server that is near a particular user. This setup reduces response times, load, and bandwidth of a website's main web server.

Similar to CBSPs, a CDN intercepts requests to a web server, which enables it to inspect incoming requests and selectively decide whether to serve cached content or forward the request to the web server for a dynamically generated response. Therefore, traffic towards the web server has to be constantly redirected through the CDN. To achieve this, CDNs either opt for URL rewriting or

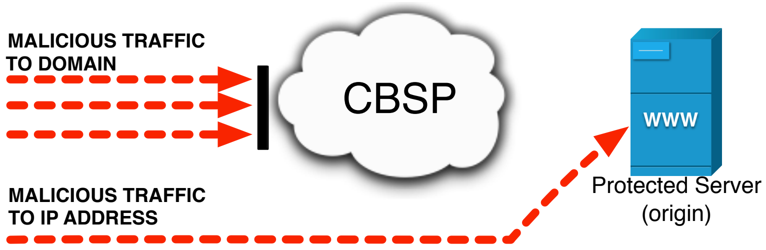


Figure 6.2: Bypassing cloud-based security. In the case of DNS rerouting, only traffic that uses the domain name is diverted through the CBSP's network. Traffic that uses the IP address of the protected server can reach the web server directly.

DNS rerouting [118]. Considering that a CDN's infrastructure is inherently capable of inspecting requests to leverage intelligent caching techniques, they are ideally placed to provide cloud-based security as well. Since traffic is already being redirected through their CDN, scrubbing centers and WAFs can be conveniently chained in the infrastructure. Moreover, in terms of volumetric DDoS attacks, a CDN is an ideal fit for mitigation strategies due to their geographically distributed and highly-available network. By using Anycast [4], servers spread across the globe can each process a small portion of the distributed attack, effectively making it feasible to absorb large amounts of malicious traffic.

As a result from this overlapping feature set, a significant share of CBSPs has emerged from CDN providers that started offering security services on top of their existing platform. Similarly, several security-focused companies that provided cloud-based services, have also started incorporating caching features to their infrastructure. Consequently, the line between CDNs and CBSPs is blurred. As such, the origin-exposing vectors that we discuss in Section 6.3 are applicable to CBSPs as well as to CDNs with security extensions.

6.3 Potential Origin Exposure

While CBSPs have become really popular because of their ability to stop real, large DDoS [155] and web application attacks, there are concerns about their DNS rerouting mechanisms. The concept of cloud-based security relies on keeping the underlying web server, the so-called origin, secret and inaccessible by direct traffic. However, in the case of DNS rerouting, this is achieved by hiding the origin's IP address and relying on redirection through the use of the website's domain name. Consequently, as illustrated in Figure 6.2, the website is *only* protected against traffic that uses the *domain name* to initiate the connection. So, in principle, if attackers are able to discover the real IP address of the origin, they can target traffic to the web server directly, thereby circumventing all security mechanisms present in the CBSP's network.

We refer to this security concern as the risk of *origin exposure*. This issue, which is specific to DNS rerouting, has been raised several times before [145, 124], and has, in the past, received some attention by the press, followed by several reactions from the security companies [207, 183, 116]. Many different potential vulnerabilities exist that might expose a CBSP-protected website's origin. We refer to these potential vulnerabilities as *origin-exposing vectors*. In the remainder of this section, we discuss eight origin-exposing vectors, of wich four have been reported previously, as well as four novel vectors, namely Temporary DNS exposure, SSL Certificates and specific instances of Sensitive Files and Outbound Connection Triggering. All vectors combined later form the basis of our automated scanning tool, CLOUDPIERCER.

6.3.1 IP History

When setting up cloud-based security, website administrators are required to change the DNS settings for their domain. From that point on, the origin's IP address is no longer listed in the DNS records of the domain name. As already mentioned in earlier sections, this secrecy is crucial for preventing origin exposure. However, if the origin is still assigned the same IP address as before the adoption of a CBSP, the server can be exposed through historical knowledge of the domain and its corresponding IP address.

Several companies specialize in harvesting data about domain names by continually tracking their DNS configuration. This allows them to build a vast database of historical DNS records, mainly used for domain marketing research, which can also be leveraged to track down an origin's IP address.

Accessing these databases is almost effortless and some of these services even offer a small number of free queries. However, these databases do not cover all existing domains as some TLDs do not share their zone files, making it harder to discover and monitor some domains. At the same time, domains that are not indexed in these databases are certainly not guaranteed to be safe from IP history vulnerabilities. For instance, if an attacker has been targeting a particular victim for a prolonged period, he could have manually gathered information about the domain and its origin before it was protected by the CBSP.

Because of the multitude of parties that could be collecting historical information about websites and their IP addresses, several CBSPs recommend administrators to assign a new IP address to their web server after migrating their DNS records to the CBSP [183].

6.3.2 Subdomains

Since the CBSP acts as a reverse proxy for multiple clients simultaneously, it relies on information available in HTTP requests to distinguish between requests intended for different clients. More specifically, by looking at the domain listed in the HTTP `Host` header, the CBSP can correctly forward incoming traffic to the intended origin. An unfortunate side-effects is that protocols that do not contain host information, such as FTP and SSH, cannot be properly handled by the CBSPs' proxies and are thus, by default, broken.

There are two ways around this problem: first, instead of using the domain name, an administrator can directly specify the origin's IP address when working with non-web protocols. This, however, lacks the flexibility of a domain-name-based solution since the IP address must be either hardcoded in scripts and program profiles, or remembered by a website's administrator.

Alternatively, administrators can create a specific subdomain, such as `origin.example.com`, that directly resolves to the origin's IP address. This provides a convenient tool for non-web protocols to bypass the CBSP and establish a direct connection with the origin. However, since this workaround effectively creates a direct path to a website's origin, it is a potential backdoor that, if discovered, can be abused by attackers. In the absence of misconfigured DNS servers allowing unauthenticated Zone Transfers, subdomains are not directly visible when querying the DNS records of the main domain name. An attacker can, however, perform a dictionary attack by trying to guess valid subdomains, using dictionaries of words popularly used in subdomains.

6.3.3 DNS records

Once a website is protected, the DNS `A` record of its domain name points to an IP address of the CBSP instead of directly to the origin. However, it is possible that traces of the origin are still present in other DNS records.

For instance, `MX` records reference the mail servers that are responsible for accepting email messages that are destined for mailboxes on a given domain. When only HTTP traffic is forwarded by the CBSP, SMTP needs to be able to establish a direct connection with the mail server. Therefore, the `MX` records should directly resolve to the mail server's IP address in order to keep email services operational. This can lead to origin exposure, especially when the mail server is listening on the same network interface as the origin's web server.

Another potentially problematic case are `TXT` records, often used for mechanisms such as the Sender Policy Framework (SPF) [102]. This framework aims to counter email address spoofing by validating the IP address of the sender against a list of approved IP addresses. The list of addresses from which emails may be sent, has to be placed in an `TXT` record of the domain [122]. Thus, if one wants the origin server to be able to send out emails using the SPF mechanism, they are forced to expose its IP address in the appropriate `TXT` record. Note that the solution to this problem is not obvious; an administrator has to choose to either abandon the Sender Policy Framework (thereby opening himself to email abuse), or accept that the protected web server cannot send verified emails.

The origin exposure, unfortunately, is not limited to `TXT` and `MX` records. Especially when a CBSP does not manage the DNS records of its customers' domains, exposure from other records may be overlooked by the customer. For instance, if the origin is accessible through IPv6, `AAAA` records are present. If the CBSP's setup instructions only cover the change of the `A` record of the domain, the `AAAA` record might be left unchanged, effectively keeping the origin exposed through its IPv6 address.

6.3.4 Temporary exposure

Administrators might temporarily pause the cloud-based security service, e.g., for maintenance or server migrations. During this time, the domain might temporarily resolve to its origin, effectively leading to a brief origin exposure. Temporary leaks can occur in many DNS record types, including `MX`, `CNAMEs`, and `TXT`. Attackers who are closely monitoring their victim might be able to witness a temporary exposure. Once the origin is known, the web server remains vulnerable even after the leak has disappeared, since the attacker can keep reusing the leaked IP address. The leak will only be remediated when the administrator decides to, yet again, change the IP address associated with the victim website.

6.3.5 SSL certificates

If administrators want to enable HTTPS for their website while under the protection of a CBSP, they can let the CBSP set up a certificate for their domain. This enables the CBSP to take care of securing the front-end connection between their own cloud infrastructure and a visitor. Alternatively, the administrator can hand over the private key of their origin's certificate to the CBSP. In this case, the CBSP can set up the front-end SSL connection with the website's own certificate. In order to secure the back-end connection between the CBSP and the origin, the origin must present a certificate. However, this certificate lists the domain name as the subject, and therefore identifies itself as the origin. In other words, if an attacker is able to scan all IP addresses and retrieve all SSL certificates, he can find the IP addresses of hosts with certificates that are associated with the domain he is trying to expose. Because of recent advancements in network scanners, performing such a massive scan has become quite feasible. For example, using ZMAP [61] and an appropriately fast network connection, allows an attacker to conduct a scan of the entire IPv4 address space on a single port in 45 minutes.

6.3.6 Sensitive files

Sensitive files located on the server form another vector through which a server's IP address can be exposed. For instance, files that were created during the development or configuration phase, in order to aid the administrator, can be used to expose a server's origin, especially when they show detailed information regarding the server. Furthermore, as already explained by Akamai [116], verbose error pages and log files can also disclose the origin that is meant to be kept secret. Usually, these types of files are meant to be removed or given proper access restrictions once a website goes into production, but presumably this is not always done correctly.

6.3.7 Origin in content

Instead of using a domain name to link to content, a webpage is free to use the IP address of the server directly. For example, a developer might use the IP address directly in the HTML of a page during an early development phase of the website. Although this is probably rather uncommon, it does form a potential origin-exposing vector. Furthermore, the IP address might be listed in the HTML as part of server-side software calculating web server statistics.

6.3.8 Outbound connections

Although a web server's incoming connections are rerouted through the CBSP's infrastructure, this is not the case for outbound connections. When a web server initiates an outgoing connection on its own accord, the CBSP is not used as a proxy. Consequently, the origin establishes a direct connection with an external host, effectively exposing its IP address to that particular host.

In order to exploit this phenomenon, an attacker can attempt to deliberately trigger the origin to initiate outgoing connections. Many possibilities exist in this regard and these are usually very specific to the applications running on the web server. Some examples include the possibility to upload a file via a URL, or link back mechanisms such as PingBack [110], which retrieve external webpages to verify whether a claimed link to their website is real or not.

6.4 Large-scale Analysis

To assess the magnitude of the origin-exposure problem, we conduct a large-scale analysis in which we attempt to uncover the origin of CBSP-protected domains. First, we consolidate the eight origin-exposing vectors into one automated origin-exposing system called CLOUDPIERCER. Then, we assemble a list of clients from five CBSP companies by studying their DNS configurations and obtaining their adoption rate across the Alexa top 1 million websites. Starting from these client lists, we use CLOUDPIERCER to evaluate 17,877 long-term, CBSP-protected domains against origin exposure. In the final step of CLOUDPIERCER, all collected candidate IP addresses are validated with a novel verification method to assess whether each discovered IP address is indeed the one of a protected website. Using CLOUDPIERCER, we are not only able to measure the amount of bypassable domains but also to gauge which origin-exposing vectors are the most prevalent.

6.4.1 CBSP Providers

For our purposes, we are interested in analyzing various *always-on, DNS rerouting* CBSPs. As mentioned in Section 6.2.2, several CBSPs are CDNs that offer additional security services, and vice versa. Since it is not straightforward to externally distinguish between clients that only use the CDN capabilities from those who are specifically paying for a plan that includes security, we selected five well-known providers that have a *specific focus on security*, i.e., at least some form of cloud-based security is present by default in all of these provider's pricing plans. The selected providers are *CloudFlare, Incapsula, DOSarrest, Prolexic (PLXedge)* and *Sucuri (Cloud Proxy)*. We gather a list of clients from each provider, enabling us to study their necessary configurations and their adoption by popular websites.

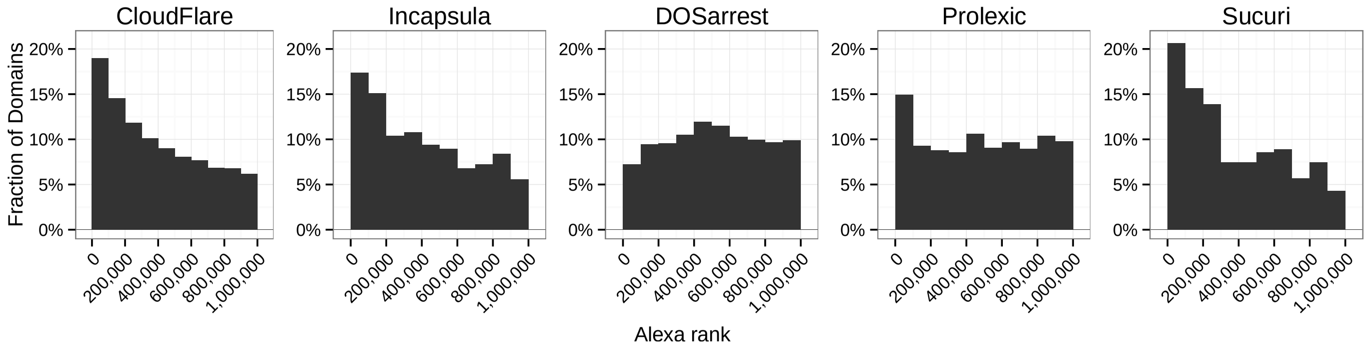


Figure 6.3: The portion of domains protected by each company, across segments of the Alexa top 1 million. For example, about 15% of the domains protected by Incapsula are situated between rank 100,000 and 200,000.

Security Provider	DNS Configuration	Domains
CloudFlare	`NS`	35,552
Incapsula	`A, CNAME`	1,841
DOSarrest	`A`	1,295
Prolexic	`A`	829
Sucuri Cloud Proxy	`A`	281

Table 6.1: Cloud-based security providers, along with the DNS records that are adjusted by their clients, and the number of protected domains that were found in the Alexa top 1 million.

Clients

In order to identify protected clients, we need to be aware of the different DNS configurations that are required by each of the CBSPs. To retrieve this information, we first attempted to subscribe to each company, and took note of the set up process. If we were not able to register, e.g., due to the absence of trial or free service tiers, we searched for publicly available instructions or retrieved representative configuration settings by manually finding other existing clients.

Generally, we found two different types of DNS configurations that are used to reroute a website's traffic: changing the `NS` records and changing the `A` records. Incapsula, DOSarrest, Sucuri and Prolexic instruct their clients to change their domain's `A` record to a specific IP address, that is under the CBSP's control. In some cases, the `CNAME` or `A` record of the `www` subdomain is configured as well.

When the `NS` records of the domain have to be changed, as it is the case with CloudFlare, the entire DNS records of the domain name become actively managed by the CBSP's name servers. Consequently, all DNS records of the domain and its subdomains fall under their direct authority. This enables the CBSP to provide all the necessary DNS records in order for rerouting to take place. However, the configuration of additional custom records, such as the `MX` records to identify the domain's mail server, has to be managed through the CBSP's own custom interface where these additional records need to be added by the client.

Adoption

To assess the adoption of CBSPs, we analyze the top 1 million most popular websites, according to Alexa [12]. By retrieving each domain's DNS records and comparing them with the collected CBSP configurations, we can straightforwardly compile a list of the most popular CBSP-protected domains. Table 6.1 lists the number of clients found for each company, along with the type of DNS configuration that was used for identification.

When we evaluate the adoption of cloud-based security, we find that 4% of the web's most popular 1 million websites are protected by one of the five companies under analysis. Moreover, cloud-based security services appear to be a more prominent solution amongst the more popular websites, since, if we restrict our search to the top 10K websites, the CBSP adoption increases to 9%. To further quantify the relationship between CBSP adoption and website popularity, Figure 6.3 shows the distribution of each company's client list across rankings. Four out of five companies have a significantly higher portion of domains in the top 100K segment, further strengthening the correlation between CBSP adoption and website popularity. More specifically, CloudFlare, Incapsula and Sucuri have visibly less clients coming from the lower parts of the Alexa ranking. DOSarrest and Prolexic do not show this kind of correlation. However, we found that both companies have large domain parking services as one of their customers. These parking services are responsible for a large number of relatively unpopular undeveloped domains that are placed under protection by the CBSP thereby affecting the ranking distribution of the clients of these CBSPs.

Evaluated domains

For our large-scale analysis, we subjected the entire list of clients in the Alexa top 1 million of Incapsula, DOSarrest, Prolexic and Sucuri, as input to Cloudpiercer. Because of the disproportional popularity of CloudFlare, we decided to test a random sample of only half of their clients. This sample is small enough to allow us to conduct our experiments in a reasonable amount of time and large enough so that any conclusions can be safely generalized to the population of CloudFlare's clients. In addition, we limited the experiment to domains that remained customers of a CBSP during, at least, our 6-month monitoring period. Through this filtering process, we aim to remove negative bias, by excluding customers that were simply trying-out a CBSP and were, perhaps, not interested enough to take all necessary precautionary steps and eliminate origin-exposure vectors.

Overall, this process resulted in a final list of 17,877 domains, which we refer to as the *evaluation set*.

6.4.2 Origin Verification

To determine whether a discovered IP address is the actual origin of a CBSP-protected website, we evaluate whether we can retrieve the website's landing page using that IP address. First, we ensure that the IP address is a valid candidate by verifying that it does not belong to an address block owned by a CBSP. Then, our verification starts by visiting the website through its CBSP-protected domain name to retrieve the URL of the landing page. For example, when issuing a regular HTTP request to `http://example.com`, the browser might be redirected to a landing page with a different scheme, host and path, such as `https://blog.example.com/about_me.html`. Next, we use PhantomJS [152], an instrumented browser, to initiate an HTTP request to the candidate IP address, incorporating the previously extracted scheme, host and path of the landing page. If the candidate IP address is the actual origin of the website, this HTTP request should return the same webpage as the request using the domain name, as both requests are identical from the web server's perspective.

Determining, however, whether two webpages are identical is not as straightforward as executing a simple string comparison. For instance, when loaded twice, the same page can result in different HTML as dynamically generated content may be included in the website's response, such as, rotating articles and advertisements. In addition, timestamps and random values embedded in a webpage can also alter the resulting HTML. Moreover, several CBSPs inject content into the displayed page, such as, analytics scripts, which will not be present in a direct response from the origin.

To account for this natural variability, we designed a more intelligent and robust HTML comparison technique. Instead of comparing strings, we examine the structure of the DOM (Document Object Model). We parse both HTML strings with LXML and BeautifulSoup [162] into a tree representation of the nodes in the DOM. Next, we determine the difference between the two trees by calculating the Zhang-Shasha's tree edit distance [216], which counts the number of edit operations (insertions, deletions and substitutions of nodes) to get from one tree to the other. Furthermore, we extended the implementation [85] by incorporating normalization which is necessary to meaningfully compare the measured distances between tree-pairs of different sizes. Normalization is achieved by dividing the calculated edit distance by the sum of the tree sizes. We refer to this metric as the Normalized DOM-edit Distance (NDD).

Prior to our large-scale analysis, we measured the inter-page and intra-page NDD distributions from a random set of domains from the Alexa top 1 million, enabling us to calculate an optimal threshold. Additionally, we evaluated the effect of a more coarse-grained tree comparison by pruning the DOM trees to a certain maximum nesting depth. We measured the NDD between 3,500 pairs of different website's landing pages, as well as between 3,500 pairs of the same landing page loaded twice. Furthermore, we conducted this test for different tree pruning levels. Afterwards, we used this data to choose an optimal threshold that is used to decide whether two different HTML documents are, in fact, the same webpage. When evaluating thresholds, we focussed primarily on limiting false positives (two different webpages that are falsely marked as identical). At the end of this process, we found that a threshold of 0.18, at a maximum nesting depth of 5 levels, results in zero false positives and only 0.36% false negatives.

6.4.3 Ethical Considerations

To realistically assess the magnitude of the origin-exposing problem in the wild, one cannot avoid scanning real online websites and web applications. During our analysis and the development of CLOUDPIERCER, we took all appropriate steps to ensure that neither the origins, nor the CBSPs, were negatively impacted by our measurements. In addition, we only used publicly available webpages and data from publicly available sources.

Since the evaluated domain names are a subset of the most visited websites in the world, their infrastructure is capable of processing an abundant amount of requests on a daily basis. Nevertheless, we took several steps to minimize the impact of our analysis. For instance, the number of contacted PingBack endpoints was limited to three per domain, although, often, many more were present. Furthermore, web requests and DNS queries were adequately spaced in time in order to minimize impact on servers. Overall, we believe that the effects of our experiment on each individual site were minimal and we are confident that for the majority of websites, the extra traffic generated by our requests was just part of the expected traffic variability.

6.4.4 Results

All 17,877 domains in the evaluation set were subjected, by CLOUDPIERCER, to each of the eight origin-exposing vectors. Afterwards, CLOUDPIERCER used the origin verification algorithm to determine which IP addresses were the actual websites' hosting IP addresses. These results allow us to measure, both the origin-exposing power of each attack vector, as well as the overall risk of the origin being exposed. We manually inspected a sample of 250 exposed origins and saw that there were no false-positive verifications.

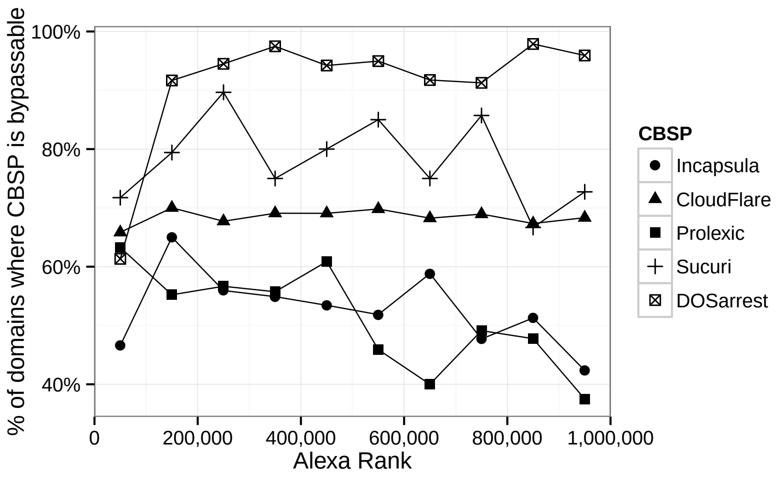


Figure 6.4: For each 100K-rank segment in the Alexa top 1 million, the percentage of domains where the CBSP is bypassable.

Overall, we found that 71.5% of protected domains is bypassable by combining the effect of all origin-exposing vectors. Table 6.2 lists the success-rate of each individual vector for the client domains of the different CBSPs. Subdomains and IP history are clearly the major vulnerabilities, both compromising the origin of more than 40% of domains. Figure 6.4 sheds light on the differences in the bypass-ratio between highly-ranked and less popular domains. We observe that for four out of five companies, domains in the top 100K are less susceptible to being exposed. A possible explanation is that higher ranked websites are more security conscious and more concerned with preventing origin exposure. A similar phenomenon was also observed in [194], where the top 100K-ranked websites were found to adopt significantly more web security mechanisms. Conversely, the risk of being exposed through SSL certificates is up to 3.6 times higher in that same top segment, presumably due to a higher SSL adoption-rate amongst these security conscious websites. Except for that first segment, there are no clear global trends across the remaining lower ranks.

As shown in Figure 6.5, 42% of domains are susceptible to exposure by more than one origin-exposing vector. More specifically, 19% of domains need to patch at least three origin-exposing vectors before they are safe. These numbers indicate that the problem is prevalent as well as multifaceted.

In the following paragraphs we discuss the results in more detail by examining the specifics of each origin-exposing vector.

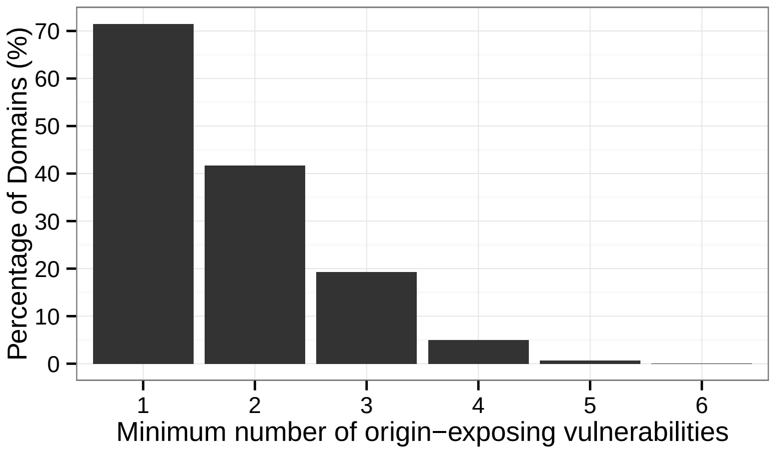


Figure 6.5: The percentage of domains that is susceptible to one or more origin-exposing vectors.

Provider	CloudFlare	Incapsula	DOSarrest	Prolexic	Sucuri	All Providers
IP History	37.0%	36.4%	88.8%	40.4%	66.7%	40.5%
Subdomains	48.9%	31.7%	3.3%	7.3%	51.5%	43.4%
DNS records	32.6%	11.2%	0.9%	1.2%	29.0%	27.9%
Temporary DNS	4.1%	0.8%	0.6%	2.0%	0.9%	3.6%
SSL Certificates	9.4%	10.7%	2.5%	6.7%	17.3%	9.1%
Sensitive files	6.4%	1.5%	0.4%	0.2%	8.2%	5.4%
Origin in content	1.2%	0.4%	-	0.9%	2.2%	1.0%
PingBack (OC)	8.2%	2.2%	0.3%	-	3.9%	6.9%
RefBack (OC)	0.5%	0.1%	-	0.3%	-	0.5%
All Combined	72.5%	53.8%	92.0%	52.0%	77.9%	71.5%

Table 6.2: The percentage of domains that can be bypassed using each origin-exposing vector, for each cloud-based security provider's customers.

Subdomain exposure

Overall, the most prevalent attack vector is the existence of origin-exposing subdomains. The feasibility of an attacker finding origin-exposing subdomains was tested by trying a list 5000 possible subdomains, provided by DNS Recon [151] on each domain in the evaluation set. If an entry existed for one or more of the tested subdomains, we verified whether the IP address to which it resolved was the origin. Our results indicate that CloudFlare's and Sucuri's customers are particularly vulnerable, with respectively 48.9% and 51.5% of domains disclosing their real IP address through subdomains.

When we take a closer look at which specific subdomains are responsible for the exposure, we find that the `ftp` subdomain is the most dominant problem, with 3,952 out of 17,877 domains having this "backdoor." This result implies a strong desire by website administrators to be able to use an FTP client that is able to connect to the server through a subdomain. Other subdomains that frequently reveal the origin are often related to email services, such as `mail` (3,203 domains), `webmail` (1,662) and `smtp` (258). Furthermore, a large number of exposing subdomains is related to cPanel, a hosting control panel that provides a web interface to help administrators configure their websites [47]. The discovered, origin-exposing subdomains are: `cpanel` (1,456 domains), `webdisk` (1,645) and `whm` (1,359). These subdomains are tied to particular services and interfaces incorporated into cPanel. Although these are HTTP services, they have to be accessed through non-standard ports which are often inaccessible when standard firewall policies are used. Therefore, cPanel creates these "proxy subdomains" that are directly linked to a specific port on the origin [46]. Despite the effort of some CBSPs to support some typical ports used by these control panels [154], these origin-exposing proxy subdomains are still frequently used.

The second-most dominating origin-exposing subdomain, namely `direct` (3,583 domains), is rather specific to CloudFlare customers. This subdomain was, in the past, given as an example when a user first configures his domain on CloudFlare's web interface [37]. Apparently, a large number of these clients used the company's instructions to the letter and thus kept this example subdomain bypass to link directly to their origin.

Interestingly, DOSarrest and Prolexic customers are less prone to subdomain exposure, with only 3.3% and 7.3% of exposed domains respectively. This is most likely due to the fact that each of their customers receives a dedicated IP address. This one-to-one mapping between an IP address of the CBSP and an IP address of the origin enables the CBSP to simply forward certain ports to the correct origin, without requiring any additional information to identify the intended host.

IP history exposure

To assess the number of domains that are still accessible through a previously documented IP address, we used two domain tracking services, DomainTools [59] and MyIP.ms [137]. We queried these databases for every domain in our evaluation set and all historical IP addresses were marked as candidates.

After the verification of the collected, "historical" IP addresses, it is evident that exposure though historical data is severe. Across all providers, we find that 40.5% of the domains are vulnerable to being exposed by consulting IP

History databases. Furthermore, the issue is prevalent with all five provider's customers. This implies that, in general, CBSP's customers often fail to configure a new IP address after setting up their cloud-based security service. This, in turn, indicates that customers are either unaware of this attack vector, or are neglecting the CBSP's recommendation to change their IP address, possibly because of operational or infrastructural barriers. Regarding DOSarrest and Prolexic, it should be noted that the misconfiguration of a single client is greatly influencing the global number of IP history bypasses. Namely, 92% of DOSarrest's domains that were vulnerable to IP history exposure, were caused by domains that belonged to a single domain parking service. For Prolexic, a similar parking service is responsible for 86% of their historically exposed subdomains.

DNS record exposure

DNS records are arguably the most trivial and practical attack vector that we studied. To assess whether they reveal a domain's origin, we simply retrieved all records for each domain at a single point in time. From each record we extracted all IP addresses and marked them as candidates for the origin. Additionally, if a domain was present in the DNS record, we resolved the domain and marked the resulting IP address as a candidate.

Despite its simplicity, we find that a significant number of domains is exposed by this vector. More specifically, the origin of CloudFlare-protected domains is exposed by DNS records in 32.6% of the cases. For Sucuri and Incapsula, this is 29.0% and 11.2% respectively. Most of these domains are leaking their IP address through their `MX` record (4,390 domains), followed by `TXT` records (1,134), where SPF is often the reason, as described in Section 6.3.3. The frequent exposure through these two records suggests that web servers that send and receive email are responsible for a substantial fraction of the discovered origin exposure. Interestingly, we also found 16 domains that were exposed through their `A` records. In these cases, both the origin and the CBSP's IP address were present in the domain's `A` records. We speculate that in this situation, the client has created an additional `A` record for the CBSP's IP address, while forgetting to remove the existing record that pointed to the origin. For the domains under the protection of CloudFlare, the DNS records are managed by the CBSP. Therefore, we excluded CloudFlare customers that were exposed through their `A` record, as this indicates that the administrator has deliberately paused the CBSP rerouting through CloudFlare's web interface.

Temporary exposure

To determine whether origins were temporarily exposed due to an interruption of the cloud-based security service or due to a transient leak in another DNS record, we repeatedly retrieved, on a daily basis, all DNS records of protected domains for a period of 10 weeks (Sucuri and Prolexic) or 16 weeks (CloudFlare, Incapsula and DOSarrest). We excluded the domains that were already exposed by the one-time DNS records retrieval, described in the previous paragraph. This allows us to isolate the domains that only temporarily exposed their origin. The number of temporal exposures is considerable. We discover that more than 3% of domains transiently revealed their origin through their DNS records during a 10 or 16-week period. The vast majority of them were exposed through their `A` record, indicating a brief disabling of the protection system.

SSL certificate exposure

In order to find IP addresses hosting SSL certificates associated with the domains in the evaluation set, we made use of the publicly available data of Rapid7's Project Sonar [161]. This project uses ZMAP [61] to periodically conduct scans of the entire IPv4 address range in search for, among other things, SSL certificates. We used their certificate data [160] and extracted all IP addresses that presented certificates related to the domains in the evaluation set. According to Durumeric et al. [60], 129,695 of the domains in the Alexa top 1 million (13%) possess browser-trusted certificates. This appears to be in line with the 9% of origins that we discovered by looking for IP addresses that presented a certificate for those domains. If the origin desires to secure the back-end connection (the one between the CBSP and the origin) with HTTPS, a certificate for its domain has to be presented by the origin. Paradoxically, this, in turn, makes the entire set up less secure by introducing the risk of origin exposure.

Sensitive files

We limit our search of sensitive files to the so-called *phpinfo* files. These files execute the PHP function `phpinfo()` [189], which outputs a large amount of data regarding the server, the execution environment, PHP compilation options, etc. This function is particularly interesting because it dynamically retrieves all this data each time it is called. Furthermore, it usually displays the server's IP address in the `SERVER_ADDR` field.

We attempted to find files that execute this function by trying several obvious file names, namely `phpinfo.php`, `info.php`, `test.php` and `phpMyAdmin/phpinfo.php`. Overall, we found that 5.4% of domains had at least one of these files accessible and exposed their origin in this manner. Presumably, the files are a remainder of the development setup, which the developers forgot to remove.

Origin in content

For the vectors involving analyzing the HTML content of pages, we crawled each domain in the evaluation set. First, we queried Bing for each domain using `site:example.com` to retrieve an initial seed of 50 webpages. Starting from this seed, we crawled additional pages by visiting internal links, up to a maximum of 500 pages per domain. On average we retrieved 328 pages per domain in the evaluation set.

To detect whether the origin was present on the website's pages, we searched the HTML source code of every crawled page for the presence of IP addresses. We found only a small number of domains (1%) that included the real IP address of their web server in one of their pages, making it one of the least effective origin-exposing vectors.

Outbound connections

Since triggering outbound connections is closely tied to the applications that run on any given web server, it is near impossible to get a comprehensive measurement of the associated risk. In order to get an impression of what is possible, we chose to conduct two experiments on potentially widespread mechanisms. The first one revolved around the Pingback mechanism, which is mostly found on WordPress [213], the most wide-spread blogging software [157]. The second experiment focussed on the verification of the HTTP referrer header, which is being used, e.g., by RefBack [182], to discover incoming links.

Pingback exposure. Pingback is a protocol that allows website owners to get notified when one of their pages or articles is mentioned on another website. When a server receives a notification, Pingback should automatically visit the other website to verify whether it actually contains a valid hyperlink. This verification procedure can be leveraged to trigger an outbound connection from the origin. For the *Origin in content* vector, every domain in the evaluation set was crawled. During this process, we simultaneously searched for Pingback enabled webpages. Next, we made an XML-RPC request to the Pingback endpoints, in which we included a URL of a page on our server that contained a unique token for each domain. As a result, we could extract candidate origin IP addresses by monitoring the incoming requests on our server and recording which IP addresses accessed which domain-specific, tokenized URLs.

Essentially, Pingbacks allow a third party to force a web server to issue a request to an arbitrary host. In the past, this had lead to the creation of entire WordPress botnets, which were abused to conduct DDoS attacks on websites [34]. As a consequence, awareness about Pingback abuse was increased, encouraging many security companies and administrators to take steps towards preventing it from happening again [129, 181]. During our analysis, we often found that websites and CBSPs were actively blocking our Pingback requests, or refrained from initiating any outbound connection to our server. However, we were still able to confirm that 6.9% of protected domains expose their origin's IP address through the Pingback mechanism.

Referrer verification exposure. In order to test exposure through referrer verification, we set the HTTP Referrer header to a tokenized URL during the entire domain crawling process. Similar to our Pingback approach, we monitored whether there were any connections made to our unique URLs, potentially by a web application of the origin that wanted to inspect the referrer page that had lead a user to the origin's website. Our results indicate that this vector poses only a minor risk for origin exposure. Only 0.5% of domains were exposed by making an outbound connection from their origin to the referrer of a visitor on their website. Our server was, however, contacted by a plethora of other IP addresses which mostly belonged to web spiders, such as, Googlebot [75] and Proximic [156].

6.5 Discussion & countermeasures

Our findings categorically demonstrate that a comprehensive adoption of CBSPs is harder than just changing DNS records. Multiple origin-exposing vectors are highly prevalent and they generally involve different underlying causes, making the problem complex and multifaceted. Additionally, the results of our large-scale analysis are lower bounds. In the wild, an attacker can go to a greater extent to discover the origin of a particular targeted victim. For instance, if an attacker has found an IP address associated with a website through one of the origin-exposing vectors, he could scan the entire IP address block to which it belongs in further search for the origin. This can be effective when a victim has requested a new, "clean" IP address, but that address is possibly close to the previous one, since it is distributed by the same ISP. Similarly, when associated servers, such as mail servers, are discovered through subdomains or DNS records, it is a reasonable assumption that the origin is located at a nearby address. Furthermore, attackers can manually analyze the website to trigger outbound connections, search for specific configuration files, test for more subdomains, and perform much more intrusive tests than those included in CLOUDPIERCER.

Ultimately, unlike us, an attacker is not necessarily bound to origin verification. As noted in [145], an attacker can deduce the location of the origin by starting a DDoS attack on one or more plausible IP addresses and observing the effect it has on the CBSP-protected website. If the origin is taken down by this attack, the CBSP will display either a static cached copy of the offline website, or a 404-like error message.

Countermeasures

Complete mitigation of origin exposure is hard, as administrators are required to fully understand the potential risks and comprehensively address all vulnerabilities in order to fully prevent an attacker from circumventing the CBSP. However, a tool similar to CLOUDPIERCER could be deployed by CBSPs to proactively scan their client's domains for exposed origins, creating awareness and helping administrators fix specific vulnerabilities.

Apart from countering each origin-exposing vector, the logical first line of defense is a proper firewall configuration that blocks all connections except those originating from the CBSP. This will significantly complicate the life of an attacker who will not be able to tell whether an IP address is unreachable, or whether it, in fact, is the origin of a target website. Together with requesting a new IP address, this firewall configuration should be standard practice when cloud-based security is utilized. We can safely assume that the vast majority of customers are currently not adopting such a strategy, since, if they did, our origin verification method would have failed. It appears that administrators are either uninformed about the risks, or are deterred by the complications of properly whitelisting all IP addresses necessary to keep the website operational. We conducted a small-scale survey asking vulnerable websites about the missing firewall configurations and their CBSP-related security expectations but we, unfortunately, received no responses.

CBSPs could actively monitor whether their client's domain was assigned a fresh IP address, and whether the client's web server is blocking requests coming from outside of the CBSP's network. This information could then further be used to explicitly warn and motivate administrators to take the necessary measures to prevent exposure.

Another beneficial strategy for CBSPs is to assign a unique IP address to each customer, which is already the case with Prolexic and DOSarrest. As our results showed, this has a significant effect on the number of subdomains and DNS records exposures. If the necessary ports can be forwarded to the origin, there is no need to set up subdomains or `MX` records that connect directly to the origin's IP address. We expect that as the adoption of IPv6 expands, this defense mechanism will become increasingly more practical, even for very large CBSPs, such as CloudFlare.

Possibly, some larger websites that possess entire /24 IP blocks might be able to initiate BGP rerouting once the origin has been attacked directly. However, relying on this fallback scenario defeats the benefits of the always-on strategy and eliminates the protection against web application attacks.

6.6 Related Work

To the best of our knowledge, our research is the first to review existing origin-exposing attack vectors for the bypassing of CBSPs, propose new ones, and systematically assess the magnitude of the exposure problem in the wild.

Over the years, a plethora of DDoS defense systems have been proposed. However, destination-based systems are usually rendered ineffective against large volumetric attacks that are able to saturate a site's uplink. Additionally, according to Huang et al. [88], systems that seek cooperation of many different parties usually face deployment issues. The authors argue that a lack of incentive prevents these cooperative systems from being widely deployed across the Internet's infrastructure. For instance, the profit of transit providers greatly depends on the amount of traffic they forward. Hence, these providers are cautious with implementing filtering systems that might negatively impact their business. In constrast, recent publications [51, 108] have documented the decline of the NTP DDoS attacks, impacted by a large-scale collaborative effort amongst ISPs, CERTs and academia.

A feasible non-collaborative solution for a victimized autonomous system (AS) was introduced in 2003 by Argawal et al. [7]. The concept is to reroute DDoS traffic through off-site cleaning centers that are dedicated to filtering and absorbing malicious attack traffic. The authors studied various network-layer techniques for diverting DDoS traffic to cleaning centers and, afterwards, redirecting the clean traffic to the protected web server. This work later became the inspiration for the patents of several popular DDoS mitigation services, such as Prolexic [121]. The use of rerouting techniques, such as BGP diversion and GRE tunnelling, resurfaced in Shield by Kline et al. [103]. In that paper, the authors focus on leveraging the off-site DDoS mitigation as an insurance model to solve the incentive problem. The authors also note that CDNs can be leveraged as DDoS defense systems in a similar fashion. In 2007, Lee et al. [112] already studied the inherent DDoS resilience of CDNs and proposed a novel scheme to further improve their robustness against attacks.

As CDNs further incorporated security features into their products, their business extended increasingly into cloud-based security providers. Thereupon, various studies evaluated these security components and several problems were uncovered. For instance, Liang et al. [118] analyzed how HTTPS was implemented within the context of CDNs. Inherently, a CDN is a man-in-the-middle (MITM) between the website and its visitors. This allows them to inspect incoming requests for the purpose of serving cached content and filtering out malicious requests. However, as HTTPS is intended for end-to-end encryption, this introduces various complications. In their study, the authors report on several implementation issues, including private key sharing, insecure back-end communication and numerous issues with invalid, stale and revoked certificates.

Another issue, discovered by Triukose et al. [191], allows CDNs to be abused to conduct a bandwidth amplification DDoS attack against their own customers. The vulnerability leveraged the fact that requests to CDN-enabled websites typically involve two decoupled TCP connections, with the CDN as a MITM. However, once the CDN forwards an attacker's request to the origin, the attacker can cleverly break off his own TCP connection with the CDN. Thereupon, the origin will waste bandwidth by sending a response to the CDN that will no longer be forwarded to the attacker.

Finally, in 2013, Nixon et al. [145] and McDonald [124] raised awareness of origin-exposing vectors that could enable attackers to bypass CBSPs and CDNs. We extend their work by proposing novel origin-exposing vectors and combining them into one automated origin-exposing tool with origin-verification capabilities, which we then deployed to conduct the first large-scale assessment of the issue. DOM-based similarities, which we leveraged for origin-verification, were previously used by [164] to detect phishing attempts.

6.7 Conclusion

Cloud-based security is a popular solution to counter the increasing threat of DDoS and web application attacks. CBSPs that use proxying via DNS are adopted by at least 9% of the 10K most popular websites. Presumably, the trivial setup without infrastructural investments, combined with the benefit of an always-on protection service, attracts a large user base. The mechanism itself, however, suffers from a critical weakness. The entire mitigation service is completely dependent on the secrecy of the website's hosting IP address, the so-called origin. Moreover, several vulnerabilities are reported that have the potential to expose this origin.

In this chapter, we discussed eight origin-exposing vectors, including various novel vulnerabilities. We consolidated all vectors into CLOUDPIERCER, an automated origin-exposing tool, which we then used to conduct the first large-scale analysis to measure the global risk of origin exposure. Our results demonstrate that the problem is severe: 71.5% of the 17,877 CBSP-protected websites that we tested, exposed their real IP address through at least one of the evaluated vectors.

Taking into account the severe consequences of an exposed origin and its prevalence amongst CBSP-protected websites, we opine that the problem is currently inadequately addressed. However, the findings of our research can be used both by CBSPs to encourage better practices regarding the adoption of their security infrastructure, as well as by administrators of CBSP-protected websites who can verify and remediate their own origin-exposing vulnerabilities. All five CBSPs have been notified of our findings prior to publication.

Finally, a silver lining of our findings is that a tool like CLOUDPIERCER can, in principle, be used by law enforcement. It is well known that miscreants use CBSPs to hide their real hosting location [48], making it harder to track and shut them down. Consequently, the discussed vectors and their reported effectiveness can be leveraged by the appropriate institutions to react quicker against malicious online activities.

6.8 Availability

CLOUDPIERCER will be made available as a web service on `https://distrinet.cs.kuleuven.be/software/cloudpiercer/`, where users of CBSPs, after proving ownership of their websites, will be able to submit their URLs for scanning and get a detailed report on all the origin-exposing vectors that

Cloudpiercer was able to find. We hope that the community will benefit from this service by allowing administrators to discover and eliminate vulnerabilities on their websites, before they are discovered and abused by attackers.

Chapter 7

Conclusion

To conclude this dissertation, we first present a summary of the work. Next, we reflect on potential root causes of DNS abuse and the shortcomings of current countermeasures. Lastly, we end with some final concluding thoughts.

7.1 Summary

In this dissertation, we reported on various analyses of DNS attacks and abuse.

First, after describing the workings of and the parties involved in DNS, we analyzed current developments of DNS abuse in Chapter 2. In particular, we discussed deceptive domain names, technical DNS hijacking attacks and how cybercriminals use DNS to run their operations.

Secondly, we analyzed two ecosystems that monetize of DNS in an abusive manner. In Chapter 3, we studied malicious domain registrations in the .eu TLD. We identified the modus operandi of cybercriminal entities that register large amounts of domains for one-shot, malicious use. We presented insights into the operational aspects of cybercriminal businesses and proposed a system that automatically clusters malicious registrations into campaigns.

In Chapter 4, we conducted an extensive analysis of the domain parking ecosystem. We concluded that this industry abuses both trademark holders, as well as visitors. Specifically, we confirmed that the domain parking industry hosts a large body of typosquatting domains for financial gain. Additionally, we observed that accidental visitors to parked pages are aggressively monetized by exposing them to malware and elaborate scams.

In the following two chapters, we explored DNS-related attacks and conducted large-scale measurements to assess their prevalence. Chapter 5 introduced two novel cybersquatting attack vectors that target hierarchical DNS resolutions: nameserver typosquatting and nameserver bitsquatting. The former is caused by DNS configuration errors, while the latter exists due to memory errors. Both attacks are capable of hijacking requests to a large number of 2LDs by making just a few targeted registrations.

In Chapter 6, we presented a comprehensive study of origin-exposing attacks on DNS-based cloud security services. We reported on a large-scale study of origin exposure, a vulnerability that allows attackers to bypass the entire cloud security infrastructure.

7.2 Discussion of possible root causes

In this section, we reflect on the difficulties of fighting DNS abuse. Taking into account the current developments and the insights gathered throughout the chapters of this dissertation, we discuss several root causes of unsuccessful countermeasures and suggest several possible directions for improvement.

7.2.1 Lack of DNSSEC adoption

DNSSEC [62] is a security extension to DNS that provides authentication and integrity by allowing name servers to add digital signatures of their resource records to establish a chain of trust from the root zone, all the way down to the domains's authoritative name servers. This allows clients to validate whether the DNS responses they receive have not been tampered with, and if they originate from the legitimate owner of the domain name.

DNSSEC is a powerful security mechanism. When deployed properly, it is capable of defending against most technical DNS attacks, such as cache poisoning and name server request hijacking. DNSSEC has been around for two decades, however, its adoption has remained minimal. More specifically, currently only 0.6% of domains in .com are signed [180]. On the resolver side of adoption, a study finds that while 83% of resolvers request DNSSEC records, only 12% of them are currently properly validating them [19].

When assessing the possible causes for the minimal presence of DNSSEC, at first sight, there are various undesirable side-effects caused by its deployment. One of these is the zone file exposure, previously discussed in Chapter 2.2.2, which can currently only be mitigated by employing online signing. Unfortunately, online signing is more computationally expensive and increases key exposure. Additionally, DNSSEC tends to produce much larger responses, which increases the load on the network and impacts the scalability of DNS. Moreover, the larger asymmetry between the request and response size makes DNSSEC a more attractive weapon for DDoS amplification attacks in comparison with regular DNS.

Deployment difficulties However, the root cause of low DNSSEC adoption probably lies in its complex and error-prone deployment process that requires multiple parties to cooperate. A research paper from 2017 looked into DNSSEC's deployment complications and finds that the dependency on other parties is a major culprit [33]. The Delegation Signer (DS) record is especially problematic. This record has to be transferred *out-of-band* to a parent zone to establish the entire chain of trust. In other words, if a domain owner wishes to deploy DNSSEC, he needs cooperation from both his registrar and his registry. Moreover, additional parties such as the registrar's resellers or the managed DNS provider that handles DNSSEC for the domain owner may come into play as well. Unfortunately, collaboration of these parties is lacking. Currently, 9 out of the 20 most popular registrars do not provide *any* mechanism to transfer the domain's DS record to the registry. This causes a broken chain of trust for one-third of the DNSSEC-enabled domains. Furthermore, the DS record transfer mechanism is not standardized and –if present– is usually implemented through uploads over a web interface or simply via email. This ad-hoc approach has already led to security critical human errors and successful social engineering attacks. Moreover, DNSSEC is not a "set it and forget it" installation, as old keys expire while new keys have to be rolled out. On the bright side, there are some efforts to partially automate the replacement of DS records during a key rollover (e.g. CDS [109]). However, this will not alleviate us from the manual process of initial deployment. Unfortunately, any failures in the intricate setup cause record validation to break, resulting in a self-inflicted denial of service. Over the past years, many major outages have occurred due to problems with DNSSEC configurations in large second-level, top-level, as well as root zones [89].

Incentivization In all fairness, most security initiatives come at a finanical or convenience cost. It is only when its benefits outweigh its costs that adoption is favored. In that light, one could say that the use of DNSSEC is not adequately incentivized. In comparison, if we look at HTTPS, users are actively notified by their browser that a domain is "*secure*" when a valid TLS certificate is presented. However, such a widely-adopted mechanism does not exist for DNSSEC. From a commercial perspective, this gives little reason for deployment as users will not be able to appreciate the added trust when connecting to a domain with DNSSEC. However, such incentives can be created by the appropriate influential organizations and products. For instance, data shows that registrars and registries have the power to steer the economic incentives of DNSSEC. The national registries of Sweden and the Netherlands (the .se and .nl TLDs) offer a yearly discount to DNSSEC-enabled domains. This incentivizes both registrars and domain owners to setup and complete a proper DNSSEC deployment process. As a result of these discounts, the DNSSEC adoption rates within registrars that are selling .nl and .se domains are spectacular (up to 75-95%). Interestingly, while these registrars invested in a good DNSSEC deployment process for .se or .nl, they did not bother to port this process to other TLDs where the incentive was not offered.

7.2.2 Reactive countermeasures are outpaced

As discussed in Chapter 3, malicious actors employ a hit-and-run strategy when using domain names in their cybercriminal operations. Domain names are phased out of operation within the first days or even hours of their existence. This requires a constant supply of new registrations to ensure the continuity of their operations. This strategy severely limits after-the-fact and per-domain countermeasures.

Here, we revisit two reactive countermeasures, the UDRP and domain blacklists, and elaborate on their inability to timely act against cybercriminal operations using short-lived domains.

UDRP The Uniform Domain-Name Dispute-Resolution Policy (UDRP) protects trademark holders against domains that abuse their name (e.g. typosquatting or combosquatting) [90]. Under the UDRP, trademark owners can file a complaint against infringing registrations. This complaint will be handled by a dispute-resolution service. These services assign a panel of experts that decides whether the domain should be transferred to the trademark holder or not. All ICANN accredited registrars are expected to follow and enforce the outcome of a dispute resolution. Overall, this entire process can take up to 60 days [210], which is typically much faster than regular legal action. Although UDRP helps fighting against, for instance, long-lived typosquatting domains, it is definitely not agile enough to counter short-lived domains abusing trademarks for their spam, phishing or scam campaigns.

Domain blacklists Generally, blacklists provide reputation scores for existing domain names. Users of these blacklists consult the reputation for domains they interact with and may decide to block communication with it. In most cases, domains end up on blacklists after their involvement with malicious activity has been reported or observed by honeypots, such as spam traps. Blacklists are typically able to act rather quickly. As shown in Chapter 3.2, 73% of domain names within .eu end up on a blacklist within 5 days of their existence. However, we also know that 60% of spam domains are active for just a single day after registration [83]. This gives us an indication that many domains are only blacklisted after, or at the very end, of their malicious lifetime. This is obviously not ideal, as most damage may already have been done. Lastly, blacklists are commercial products only available to their select number of clients. They do not dismantle malicious domain names.

Given that abuse often takes place immediately after registration, countermeasures would be more effective if they could block malicious domain names *before* they are active. At this time, several initiatives are concerned with this goal. In 2016, Hoa et al. [82] proposed PREDATOR, a system that only uses features that are available at the time of registration to assess a domain's reputation. Similarly, in collaboration with EURid, we developed predictive models to detect malicious domain names before they are operational (discussed in the preamble of Chapter 3). Ultimately, for these initiatives to succeed, cooperation of parties involved in the registration process is required. This is further discussed in the following section.

7.2.3 Ambiguous roles and incentives

In the discussion of the low DNSSEC adoption (Section 7.2.1), we already argued that a lack of incentive was one of the main causes. Similar issues arise in the realm of blocking malicious domain names.

As previously suggested, new malicious registrations have to be stopped very early in their lifetime to properly counter today's fast-paced abuse. The organizations that have the earliest visibility in the inception of malicious domains are those involved in the registration process, i.e. resellers, registrars and registries. This thesis specifically highlighted such an opportunity in Chapter 3.

However, it is not straightforward, nor clearly established how these organizations should play the double role of providing services for their clients, while policing them at the same time. Furthermore, arguably these parties might even be disincentivized, as they are financially dependent on the registrations they operate. For instance, it could hurt a registrar's business when it is the only one in the market performing extra screenings on clients and their domains. There might be, however, indirect benefits of preventing malicious or trademark-infringing registrations. It could increase the trustworthiness of domains registered under the registrar's administration. Additionally, processing complaints and disputes can be costly. If these parties facilitate less abusive domains, the amount of complaints –and thus costs– would be reduced. However, given the prevalence of ambiguous roles and the lack of incentives throught the DNS ecosystem, we argue that introducing globally-enforced policies and stimuli, established by governmental or regulatory instances such as ICANN, are necessary to further aid the prevention of abusive registrations.

7.2.4 Non-uniform authoritative domain names

Successfully preventing deceptive domain attacks is largely dependent on the ability of users to assess the trustworthiness of the fully qualified domain name (FQDN). Therefore, users are constantly instructed to verify their browser's URL bar and confirm that they are communicating with an authoritative domain. Moreover, companies now regularly hold anti-phishing training sessions to further educate their employees to detect fraudulent links or email senders. However, we argue that currently there is a lack of uniformity in the semantics of authoritative domain names. This creates confusion and prevents users to straightforwardly assess the legitimacy of a URL. We believe that more uniform practices for domain registrations are mandatory to empower users to successfully judge a domain name.

We argue that there are at least two culprits that confuse users, namely benign combosquatting and non-uniform name placement in the FQDN.

Benign combosquatting These days, it is very common for trademark owners to register benign combosquatting domains, i.e. entirely new 2LD registrations that combine the trademark with other keywords (e.g. paypal-prepaid.com is legitimately affiliated with paypal.com). By introducing benign combosquatting domains, there is no longer a single trustworthy authoritative domain. This is problematic, as it severely diminishes the ability to differentiate between a benign and a malicious (combosquatting) domain.

Non-uniform name placement We argue that training users to distinguish between a malicious and a benign domain is particularly hard, as the authoritative name can currently legitimately appear in three places in the FQDN. First, with the advent of the new gTLDs, an organization's name can now be in the TLD (e.g. **usa.bnpparisbas**). Next, arguably the most common scenario is when the name is registered as a 2LD (e.g. **bnpparisbas.be**). Lastly, due to different registry policies, the registered name may be present in the third-level domain (e.g. **bnpparisbas.com.au**). This last phenomenon is so extensive and scattered out that browser vendors collaboratively maintain a list of public suffixes so they can correctly and uniformly assess a FQDN to scope their cookies [135]. Obviously, we cannot expect the average user to be fully aware of this intricate list when he is evaluating a domain at first sight.

To counter this issue, we suggest efforts to be made on two fronts. First, we need a more uniform organization-name to Internet-domain-name mapping that non-technical users will be able to understand and assess. This would require the cooperation of all domain-registering organizations and entities within the DNS ecosystem. While this would be a non-trivial effort, having the trademark legitimately present at several locations in a domain name, often combined with other keywords, essentially trains users to trust almost any FQDN that contains their trademark somewhere. In our opinion, this significantly contributes to the large economic damages from cybercrime we see today. Secondly, we believe that user interfaces of browsers and email clients could further assist users in assessing the authority of the domain name they are interacting with. As an example, Mozilla's Firefox browser visually aids the user by showing the registered name and public suffix in a bold typeface in their URL bar.

7.2.5 Implications of GDPR

As of May 2018 the European Union's (EU) General Data Protection Regulation (GDPR) [65] will uniformly will be placed into effect across all the union's countries. GDPR aims to strengthen privacy and data protection for all individuals within the EU. Evidently, this also impacts DNS and, as a consequence, the fight against DNS abuse. WHOIS is the prime example of DNS related aspects that are bound to be changed to comply with the new regulations. WHOIS is a publicly available service that enables access to the personal data of domain registrants, such as their names and addresses. As making this data accessible to essentially anyone is not compliant with the GDPR, a complete overhaul of the system is necessary. Apart from the complexities of designing and deploying a compliant replacement model [123], the security community is worried about losing a valuable source of information to fight cybercrime [106]. This argument is confirmed by the role WHOIS plays in this thesis: the analyses presented in Chapters 3, 4 and 5 all use registrant data. While better privacy and data regulations are a positive development, we should be prepared for the future impact on the fight against cybercrime.

7.3 DNS-dependent security mechanisms

Despite the lack of effective defenses against DNS abuse, and in particular the low adoption of DNSSEC, we are seeing a rapid rise of security mechanisms and policies that rely on DNS records. Interestingly, without DNSSEC, the parties relying on these security mechanisms have no integrity guarantees of the data presented in those DNS records.

Email security is one important example that heavily depends on DNS records. For instance, domain owners use the widely-adopted Sender Policy Framework (SPF) to whitelist the IP addresses that are allowed to send email for the domain [102]. Similarly, DKIM publishes TXT records containing a public key that is used to cryptographically sign email headers [87]. Since 2012, further trust is placed in these DNS-based security policies with the introduction of DMARC [107]. This mechanism publishes DNS records that enable strict policy enforcement regarding email authentication mechanisms, effectively preventing email spoofing of that domain.

Additionally, many services rely on domain name based authentication, i.e. a unique challenge that has to be uploaded in the domain's DNS records or made accessible via a specific path under the domain to verify ownership. For instance, the ACME protocol (used by e.g. *Let's encrypt* [113]) works with a DNS-based challenge. A similar mechanism is employed by Google's Search Console [78] and also by the CLOUDPIERCER vulnerability scanner, introduced in Chapter 6.

A recent addition to DNS-dependent security is the Certification Authority Authorization (CAA) record, introduced in September 2017 [167]. This record enables an administrator to limit the certificate authorities (CAs) that are allowed to issue a certificate for his domain. This minimizes the risk of misissuances by, for example, a compromised CA.

Lastly, another important protocol is DNS-based Authentication of Named Entities (DANE) [86]. Essentially, this protocol replaces CAs by publishing the domain's certificate's fingerprint in its own DNS records. It allows clients to authenticate TLS connections by querying for a DNS record that is simply maintained by the domain owner. DANE is the only protocol mentioned in this section that strictly requires the use of DNSSEC.

Overall, it is a concerning development that DNS-dependent security mechanisms are on the rise, while only a fraction of DNS resolutions are secure. When the integrity of DNS response cannot be guaranteed, the security mechanisms depending on them can essentially not be relied on.

7.4 Concluding thoughts

The Domain Name System (DNS) is one of the most essential components of today's Internet. DNS is inherently hierarchical and distributed, which makes it highly scalable and reliable. However, it also brings forward a complex ecosystem with different parties that are involved in its operational, managerial and regulatory aspects on separate levels in the hierarchy. Under the surface, DNS is an intricate, loosely-coupled system of non-transparent dependencies. This disperse setting makes it particularly hard to fight abuse, while creating a wide attack surface. Deployment of security mechanisms requires the cooperation of multiple parties within this ecosystem, with usually some parties not directly benefitting from the investment. Without proper incentivization and policies, security initiatives that require the entire ecosystem to cooperate face adoption challenges.

As shown throughout the analyses presented in this dissertation, the security issues concerning Internet domain names are diverse and multifaceted. We hope that our work further drives research and initiatives to lift DNS security to a state that is in accordance with its crucial role in the digital society.

Bibliography

[1] DNSSEC Deployment Report. https://rick.eng.br/dnssecstat/, 2017.

[2] 101domain GRS Limited. .Ne Domain Registration. https://www.101domain.com/ne.htm, 2017.

[3] A Hubert, R van Mook. Measures for Making DNS More Resilient Against Forged Answers. https://tools.ietf.org/html/rfc5452, 2009.

[4] Abley, J., and Lindqvist, K. E. Operation of Anycast Services.

[5] AdBlock Plus. AdBlock Plus - EasyList. https://easylist-downloads.adblockplus.org/easylist.txt, 2014.

[6] AFNIC. Changing Registrars. https://www.afnic.fr/en/your-domain-name/manage-your-domain-name/changing-registrars-3.html, 2017.

[7] Agarwal, S., Dawson, T., and Tryfonas, C. DDoS Mitigation via Regional Cleaning Centers. Tech. rep.

[8] Agten, P., Joosen, W., Piessens, F., and Nikiforakis, N. Seven Months' Worth of Mistakes: A Longitudinal Study of Typosquatting Abuse. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)* (2015), Internet Society.

[9] Agten, P., and Vissers, T. Domain Name Abuse Is Still Widespread. http://internetsociety.ong/blog/tech-matters/2015/02/ndss-guest-blog-post-domain-name-abuse-still-widespread, 2015.

[10] Akamai. State of the Internet: Security Q4 2017 Report. https://www.akamai.com/us/en/multimedia/documents/state-of-the-

internet/q4-2017-state-of-the-internet-security-report.pdf, 2018.

[11] Akamai SIRT. Cloudpiercer Discovery Tool. https://blogs.akamai.com/2015/10/cloudpiercer-discovery-tool.html, 2015.

[12] Alexa. Alexa - Actionable Analytics for the Web. http://www.alexa.com/, 2014.

[13] Allemann, A. Survey: Domainers Have a New Favorite Domain Parking Company. http://domainnamewire.com/2013/02/19/survey-domainers-have-a-new-favorite-domain-parking-company/, 2013.

[14] Allemann, A. Parking Page Ads Lead to a UDRP Loss. https://domainnamewire.com/2016/02/01/parking-page-ads-lead-to-a-udrp-loss/, 2016.

[15] Almishari, M., and Yang, X. Ads-Portal Domains: Identification and Measurements. *ACM Transactions on the Web (TWEB) 4*, 2 (2010), 4.

[16] Alrwais, S., Yuan, K., Alowaisheq, E., Li, Z., and Wang, X. Understanding the Dark Side of Domain Parking. In *Proceedings of the 23rd USENIX Security Symposium* (2014).

[17] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., and Feamster, N. Building a Dynamic Reputation System for DNS. In *Proceedings of the 19th USENIX Conference on Security* (2010), pp. 18–18.

[18] Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou, II, N., and Dagon, D. Detecting Malware Domains at the Upper DNS Hierarchy. In *Proceedings of the 20th USENIX Conference on Security*, pp. 27–27.

[19] APNIC Labs. DNSSEC Measurement Maps. https://stats.labs.apnic.net/dnssec, 2018.

[20] Arbor Networks. Worldwide Infrastructure Security Report. http://pages.arbornetworks.com/rs/arbor/images/WISR2014_EN2014.pdf, 2015.

[21] Barnes, R., Hoffman-Andrews, J., McCarney, D., and Kasten, J. Automatic Certificate Management Environment (ACME). Internet-Draft draft-ietf-acme-acme-12, IETF Secretariat, April 2018. http://www.ietf.org/internet-drafts/draft-ietf-acme-acme-12.txt.

[22] Bilge, L., Sen, S., Balzarotti, D., Kirda, E., and Kruegel, C. EXPOSURE: A Passive DNS Analysis Service to Detect and Report Malicious Domains. *ACM Transactions on Information and System Security (TISSEC) 16*, 4 (2014), 14.

[23] Bird, S. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions* (Stroudsburg, PA, USA, 2006), COLING-ACL '06, Association for Computational Linguistics, pp. 69–72.

[24] Blanche Manning. Vulcan Golf, LLC v. Google, Inc. https://law.justia.com/cases/federal/district-courts/illinois/ilndce/1:2007cv03371/210005/236/, 2008.

[25] Blankenship, J. The Forrester Wave: DDoS Mitigation Solutions, Q4 2017, 2017.

[26] Borgolte, K., Fiebig, T., Hao, S., Kruegel, C., and Vigna, G. Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates. In *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS)* (2018).

[27] Bryant, M. Fishing the AWS IP Pool for Dangling Domains. http://www.bishopfox.com/blog/2015/10/fishing-the-aws-ip-pool-for-dangling-domains/, 2015.

[28] Bryant, M. The Orphaned Internet: Taking Over 120K Domains via a DNS Vulnerability in AWS, Google Cloud, Rackspace and Digital Ocean. https://thehackerblog.com/the-orphaned-internet-taking-over-120k-domains-via-a-dns-vulnerability-in-aws-google-cloud-rackspace-and-digital-ocean/, 2016.

[29] Caballero, J., Grier, C., Kreibich, C., and Paxson, V. Measuring Pay-per-Install: The Commoditization of Malware Distribution. In *Proceedings of the the 20th USENIX Security Symposium* (San Francisco, CA, August 2011).

[30] Canali, D., Cova, M., Vigna, G., and Kruegel, C. Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages. In *Proceedings of the 20th international conference on World wide web* (2011), ACM, pp. 197–206.

[31] Cappella, N. Cloudpiercer Tool Beats DNS Redirection Security. https://thestack.com/cloud/2016/04/12/cloudpiercer-tool-beats-dns-redirection-security/, 2016.

[32] Carletti, S. Ruby Whois. http://ruby-whois.org/, 2014.

[33] Chung, T., van Rijswijk-Deij, R., Choffnes, D., Levin, D., Maggs, B. M., Mislove, A., and Wilson, C. Understanding the

Role of Registrars in DNSSEC Deployment. In *Proceedings of the 2017 Internet Measurement Conference* (2017), ACM, pp. 369–383.

[34] Cid, D. More Than 162,000 WordPress Sites Used for Distributed Denial of Service Attack. http://blog.sucuri.net/2014/03/more-than-162000-wordpress-sites-used-for-distributed-denial-of-service-attack.html, 2014.

[35] CIRA. Register Your .CA. https://cira.ca/ca-domains/register-your-ca, 2017.

[36] Cloudflare. CloudFlare Sees Explosive Growth in 2013. http://www.marketwired.com/press-release/cloudflare-sees-explosive-growth-2013-passes-15-million-customers-revenue-up-450-network-1862981.htm, 2013.

[37] Cloudflare. Sign Up | CloudFlare | the Web Performance and Security Company. https://www.cloudflare.com/sign-up, 2015.

[38] Cloudflare. Cloudflare Grows Enterprise Business More Than 80 Percent Year Over Year Amid Significant Product Rollouts. https://www.cloudflare.com/press-releases/2017/cloudflare-grows-enterprise-business-more-than-80-percent-year-over-year-amid-significant-product-rollouts/, 2017.

[39] Cloudflare. How DNSSEC Works. https://www.cloudflare.com/dns/dnssec/how-dnssec-works/, 2017.

[40] Cloudflare. Welcome - Argo Tunnel. https://developers.cloudflare.com/argo-tunnel, 2017.

[41] Constine, J. Facebook Survives Q4 Despite Slowest Daily User Growth Ever. https://techcrunch.com/2018/01/31/facebook-q4-2017-earnings/, 2018.

[42] Constine, J. WhatsApp Hits 1.5 Billion Monthly Users. $19B? Not So Bad. https://techcrunch.com/2018/01/31/whatsapp-hits-1-5-billion-monthly-users-19b-not-so-bad/, 2018.

[43] Contavalli, C., Kumari, W., and Gaast, W. v. d. RFC7871: Client Subnet in DNS Queries. https://tools.ietf.org/html/rfc7871, 2016.

[44] Correa, D. CloudPiercer Tool Discloses DDoS Defence Providers. https://www.scmagazineuk.com/cloudpiercer-tool-discloses-ddos-defence-providers/article/534478/, 2015.

[45] Cova, M., Leita, C., Thonnard, O., Keromytis, A. D., and Dacier, M. An Analysis of Rogue AV Campaigns. In *International Workshop on Recent Advances in Intrusion Detection* (2010), Springer, pp. 442–463.

[46] cPanel. Tweak Settings - Domains. https://documentation.cpanel.net/display/ALD/Tweak+Settings+-+Domains#TweakSettings-Domains-Proxysubdomains, 2015.

[47] cPanel, Inc. cPanel and WHM. http://cpanel.net/, 2015.

[48] CrimeFlare. CloudFlare Watch. http://www.crimeflare.com/, 2015.

[49] Crocker, D., Hansen, T., and Kucherawy, M. DomainKeys Identified Mail (DKIM) Signatures. STD 76, RFC Editor, September 2011. http://www.rfc-editor.org/rfc/rfc6376.txt.

[50] Cubrilovic, N. The Anatomy of the Twitter Attack. https://techcrunch.com/2009/07/19/the-anatomy-of-the-twitter-attack/, 2009.

[51] Czyz, J., Kallitsis, M., Gharaibeh, M., Papadopoulos, C., Bailey, M., and Karir, M. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), ACM, pp. 435–448.

[52] Dagon, D., Antonakakis, M., Vixie, P., Jinmei, T., and Lee, W. Increased DNS Forgery Resistance Through 0x20-Bit Encoding: Security via Leet Queries. In *Proceedings of the 15th ACM conference on Computer and communications security* (2008), ACM, pp. 211–222.

[53] Dan York. DNSSEC Statistics. http://www.internetsociety.org/deploy360/dnssec/statistics/, 2011.

[54] Deahl, D., and Carman, A. For Weeks, Equifax Customer Service Has Been Directing Victims to a Fake Phishing Site. https://www.theverge.com/2017/9/20/16339612/equifax-tweet-wrong-website-phishing-identity-monitoring, 2017.

[55] Dinaburg, A. Bitsquatting: DNS Hijacking Without Exploitation. In *Proceedings of BlackHat Security* (2011).

[56] DNS Census. DNS Census 2013. http://dnscensus2013.neocities.org/, 2013.

[57] Domainr. Domainr Developer API. https://domainr.build/, 2017.

[58] DomainTools. Domain Count Statistics for TLDs. http://research.domaintools.com/statistics/tld-counts/, 2016.

[59] DomainTools, LLC. Domain Whois Lookup, Whois API and DNS Data Research - DomainTools. http://www.domaintools.com/, 2015.

[60] Durumeric, Z., Kasten, J., Bailey, M., and Halderman, J. A. Analysis of the HTTPS Certificate Ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference* (2013), ACM, pp. 291–304.

[61] Durumeric, Z., Wustrow, E., and Halderman, J. A. ZMap: Fast Internet-Wide Scanning and Its Security Applications. In *USENIX Security* (2013), Citeseer, pp. 605–620.

[62] Eastlake, D. Domain Name System Security Extensions. RFC 2535, RFC Editor, March 1999. http://www.rfc-editor.org/rfc/rfc2535.txt.

[63] Email-Hippo. Email Validation Online Service. https://www.emailhippo.com/en-US, 2017.

[64] EURid VZW. Over 25 000 Domain Names Suspended With Ties to Identity Fraud. https://eurid.eu/en/news/25-000-dn-suspended-identity-fraud, 2018.

[65] European Parliamant and Council. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *OJ L 119* (2016-05-04), 1–88.

[66] Eurostat. Internet Banking on the Rise. http://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20180115-1, 2018.

[67] Felegyhazi, M., Kreibich, C., and Paxson, V. On the Potential of Proactive Domain Blacklisting. In *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More* (2010), pp. 6–6.

[68] Fiveash, K. PlayStation Clambers Back Online Days After DDoS Attack PARALYSED Network. http://www.theregister.co.uk/2014/12/27/playstation_clambers_back_online/, 2014.

[69] FortiGuard Center. Antispam - IP & Signature Lookup. https://www.fortiguard.com/more/antispam, 2017.

[70] Goldberg, S., Naor, M., Papadopoulos, D., Reyzin, L., Vasant, S., and Ziv, A. NSEC5: Provably Preventing DNSSEC Zone Enumeration. In *NDSS* (2015).

[71] Goldstein, J. Cloudflare – Making Your Website Fast, Safe, and Accessible Everywhere in the World. https://hostadvice.com/blog/cloudflare-making-website-fast-safe-accessible-everywhere-world, 2016.

[72] Goodin, D. Nope, This Isn't the HTTPS-validated Stripe Website You Think It Is. https://arstechnica.com/information-technology/2017/12/nope-this-isnt-the-https-validated-stripe-website-you-think-it-is/, 2017.

[73] Google. Hosted Domains Have Been Retired. https://support.google.com/adsense/answer/2456470, 2012.

[74] Google. Fighting Spam. http://web.archive.org/web/20140301191752/http://www.google.com/insidesearch/howsearchworks/fighting-spam.html, 2014.

[75] Google. Googlebot. https://support.google.com/webmasters/answer/182072?hl=en, 2015.

[76] Google. Google Safe Browsing. https://developers.google.com/safe-browsing/, 2016.

[77] Google. Unwanted Software Policy. https://www.google.com/about/company/unwanted-software-policy.html, 2016.

[78] Google. Verify Your Site Ownership - Search Console Help. https://support.google.com/webmasters/answer/35179?hl=en, 2018.

[79] Google Public DNS. Where Are Your Servers Currently Located? https://developers.google.com/speed/public-dns/faq#locations, 2017.

[80] Grant, D. Introducing Cloudflare Warp: Hide Behind the Edge. https://blog.cloudflare.com/introducing-cloudflare-warp, 2017.

[81] Hao, S., Feamster, N., and Pandrangi, R. Monitoring the Initial DNS Behavior of Malicious Domains. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (2011), ACM, pp. 269–278.

[82] Hao, S., Kantchelian, A., Miller, B., Paxson, V., and Feamster, N. PREDATOR: Proactive Recognition and Elimination of Domain Abuse at Time-Of-Registration.

[83] Hao, S., Thomas, M., Paxson, V., Feamster, N., Kreibich, C., Grier, C., and Hollenbeck, S. Understanding the Domain Registration Behavior of Spammers. In *Proceedings of the 2013 Conference on Internet Measurement Conference* (2013), pp. 63–76.

[84] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning.* Springer Series in Statistics. 2001.

[85] Henderson, T., and Johnson, S. Zhang-Shasha: Tree Edit Distance in Python. https://github.com/timtadh/zhang-shasha, 2014.

[86] Hoffman, P., and Schlyter, J. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, RFC Editor, August 2012. http://www.rfc-editor.org/rfc/rfc6698.txt.

[87] Holgers, T., Watson, D. E., and Gribble, S. D. Cutting Through the Confusion: A Measurement Study of Homograph Attacks. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference* (Berkeley, CA, USA, 2006), ATEC '06, USENIX Association.

[88] Huang, Y., Geng, X., and Whinston, A. B. Defeating DDoS Attacks by Fixing the Incentive Chain. *ACM Transactions on Internet Technology (TOIT) 7*, 1 (2007), 5.

[89] IANIX. Major DNSSEC Outages and Validation Failures. https://ianix.com/pub/dnssec-outages.html, 2018.

[90] ICANN. Uniform Domain-Name Dispute-Resolution Policy. https://www.icann.org/resources/pages/help/dndr/udrp-en, 1999.

[91] ICANN. Model ccTLD Sponsorship Agreement—Triangular Situation. https://archive.icann.org/en/cctlds/model-tscsa-31jan02.htm, 2002.

[92] ICANN. Transfer Policy. https://www.icann.org/resources/pages/transfer-policy-2016-06-01-en, 2016.

[93] ICANN. Whois Inaccuracy Complaint Form. https://forms.icann.org/en/resources/compliance/complaints/whois/inaccuracy-form, 2017.

[94] Jain, H. Fear of a Filled Pipe - The Origin Exposed. https://www.fortinet.com/blog/industry-trends/fear-of-a-filled-pipe-the-origin-exposed.html, 2015.

[95] Jakobsson, M., and Myers, S. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft.* John Wiley & Sons, 2006.

[96] Kaminsky, D. Black Ops 2008: It's the End of the Cache as We Know It. *Black Hat USA* (2008).

[97] Karami, M., and McCoy, D. Understanding the Emerging Threat of DDoS-as-a-Service. In *LEET* (2013).

[98] Kassner, M. DDoS Mitigation May Leave Your Site Vulnerable. `https://www.techrepublic.com/article/ddos-mitigation-may-leave-your-site-even-more-vulnerable/`, 2015.

[99] Kesmodel, D. *The Domain Game: How People Get Rich from Internet Domain Names.* Xlibris Corporation, 2008.

[100] Khan, M. T., Huo, X., Li, Z., and Kanich, C. Every Second Counts: Quantifying the Negative Externalities of Cybercrime via Typosquatting. In *Proceedings of the 36th IEEE Symposium on Security and Privacy* (2015).

[101] Kintis, P., Miramirkhani, N., Lever, C., Chen, Y., Romero-Gómez, R., Pitropakis, N., Nikiforakis, N., and Antonakakis, M. Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 569–586.

[102] Kitterman, S. Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1. RFC 7208, RFC Editor, April 2014. `http://www.rfc-editor.org/rfc/rfc7208.txt`.

[103] Kline, E., Afanasyev, A., and Reiher, P. Shield: DoS Filtering Using Traffic Deflecting. In *Network Protocols (ICNP), 2011 19th IEEE International Conference on* (2011), IEEE, pp. 37–42.

[104] KRAUSE, R. Google Trumpets Platform User Base vs. Apple, Facebook, Amazon. `https://www.investors.com/news/technology/google-trumpets-platform-user-base-vs-apple-facebook-amazon/`, 2017.

[105] Krebs, B. The New Normal: 200-400 Gbps DDoS Attacks. `http://krebsonsecurity.com/2014/02/the-new-normal-200-400-gbps-ddos-attacks/`, 2014.

[106] KrebsOnSecurity. New EU Privacy Law May Weaken Security. `https://krebsonsecurity.com/2018/02/new-eu-privacy-law-may-weaken-security/`, 2018.

[107] Kucherawy, M., and Zwicky, E. Domain-Based Message Authentication, Reporting, and Conformance (DMARC). RFC 7489, RFC Editor, March 2015.

[108] Kührer, M., Hupperich, T., Rossow, C., and Holz, T. Exit From Hell? Reducing the Impact of Amplification Ddos Attacks. In *USENIX Security Symposium* (2014).

[109] Kumari, W., Gudmundsson, O., and Barwood, G. Automating DNSSEC Delegation Trust Maintenance. RFC 7344, RFC Editor, September 2014.

[110] Langridge, S., and Hickson, I. Pingback 1.0. `http://www.hixie.ch/specs/pingback/pingback`, 2002.

[111] Lauinger, T., Chaabane, A., Arshad, S., Robertson, W., Wilson, C., and Kirda, E. Thou Shalt Not Depend on Me: Analysing the Use of Outdated Javascript Libraries on the Web. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium (NDSS 2017). The Internet Society* (2017).

[112] Lee, K.-W., Chari, S., Shaikh, A., Sahu, S., and Cheng, P.-C. Improving the Resilience of Content Distribution Networks to Large Scale Distributed Denial of Service Attacks. *Computer Networks 51*, 10 (2007), 2753–2770.

[113] Let's Encrypt. How It Works. `https://letsencrypt.org/how-it-works/`, 2017.

[114] Levchenko, K., Pitsillidis, A., Chachra, N., Enright, B., Félegyházi, M., Grier, C., Halvorson, T., Kanich, C., Kreibich, C., Liu, H., McCoy, D., Weaver, N., Paxson, V., Voelker, G. M., and Savage, S. Click Trajectories: End-To-End Analysis of the Spam Value Chain. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2011), SP '11, IEEE Computer Society, pp. 431–446.

[115] Lever, C., Walls, R., Nadji, Y., Dagon, D., McDaniel, P., and Antonakakis, M. Domain-Z: 28 Registrations Later. In *Proceedings of the 37th IEEE Symposium on Security and Privacy* (2016).

[116] Lewis, D. Bypassing Content Delivery Security. `https://blogs.akamai.com/2013/08/bypassing-content-delivery-security.html`, 2013.

[117] Li, Z., Alrwais, S., Xie, Y., Yu, F., and Wang, X. Finding the Linchpins of the Dark Web: A Study on Topologically Dedicated Hosts

on Malicious Web Infrastructures. In *Security and Privacy (SP), 2013 IEEE Symposium on* (2013), IEEE, pp. 112–126.

[118] Liang, J., Jiang, J., Duan, H., Li, K., Wan, T., and Wu, J. When HTTPS Meets CDN: A Case of Authentication in Delegated Service. In *Security and Privacy (SP), 2014 IEEE Symposium on* (2014), IEEE, pp. 67–82.

[119] Liu, D., Hao, S., and Wang, H. All Your DNS Records Point to Us: Understanding the Security Threats of Dangling DNS Records. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), ACM, pp. 1414–1425.

[120] Living Internet. Domain Name System (DNS) History. https://www.livinginternet.com/i/iw_dns_history.htm, 2017.

[121] Lyon, B. Network Overload Detection and Mitigation System and Method, Jan. 13 2009. US Patent 7,478,429.

[122] Martens, K., Mehnle, J., and Kitterman, S. SPF Record Syntax. http://www.openspf.org/SPF_Record_Syntax, 2008.

[123] McCarthy, K. Europe Fires Back at ICANN's Delusional Efforts to Fix Whois for GDPR by Next, Er, Year. https://www.theregister.co.uk/2018/04/27/europe_icann_whois_gdpr/, 2018.

[124] McDonald, D. The Pentesters Guide to Akamai. https://www.nccgroup.com/media/230388/the_pentesters_guide_to_akamai.pdf, 2013.

[125] Miramirkhani, N., Starov, O., and Nikiforakis, N. Dial One for Scam: Analyzing and Detecting Technical Support Scams. *arXiv preprint arXiv:1607.06891* (2016).

[126] Mockapetris, P. Domain Names: Concepts and Facilities. RFC 882, RFC Editor, November 1983. http://www.rfc-editor.org/rfc/rfc882.txt.

[127] Mockapetris, P. Domain Names - Concepts and Facilities. STD 13, RFC Editor, November 1987. http://www.rfc-editor.org/rfc/rfc1034.txt.

[128] Monga, V., and Evans, B. L. Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE Transactions on Image Processing 15*, 11 (2006).

[129] Moore, S. WordPress Pingback Attacks and Our WAF. https://blog.cloudflare.com/wordpress-pingback-attacks-and-our-waf/, 2014.

[130] Moore, T., and Clayton, R. The Ghosts of Banking Past: Empirical Analysis of Closed Bank Websites. In *Financial Cryptography and Data Security*. Springer, 2014, pp. 33–48.

[131] Moore, T., and Edelman, B. Measuring the Perpetrators and Funders of Typosquatting. In *International Conference on Financial Cryptography and Data Security* (2010), Springer, pp. 175–191.

[132] Moura, G. C., Müller, M., Wullink, M., and Hesselman, C. nDEWS: A New Domains Early Warning System for TLDs. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium* (2016), IEEE, pp. 1061–1066.

[133] Moyle, E. DDoS Prevention for Attacks That Bypass DNS Rerouting.

[134] Moz, Inc. Google Algorithm Change History. http://moz.com/google-algorithm-change#2011, 2014.

[135] Mozilla Foundation. Public Suffix List. https://publicsuffix.org/, 2018.

[136] Mozilla.org. Domain Guessing. http://www-archive.mozilla.org/docs/end-user/domain-guessing.html, 2006.

[137] MYIP.MS. My IP Address - Shows IPv4 and IPv6 | Blacklist IP Check - Hosting Info. http://myip.ms/, 2015.

[138] Myles, P. DomainWire Global TLD Report 2016/2. https://www.centr.org/library/statistics-report/domainwire-global-tld-report-2016-2.html, 2016.

[139] NamePros. Domain Parking Companies by Payout. https://www.namepros.com/parking-and-traffic-monetization/783661-domain-parking-companies-by-payout.html, 2013.

[140] Naone, E. MIT Technology Review: Get Paid to Install Malware. http://www.technologyreview.com/view/417354/get-paid-to-install-malware/, 2010.

[141] Nikiforakis, N., Balduzzi, M., Desmet, L., Piessens, F., and Joosen, W. Soundsquatting: Uncovering the Use of Homophones in Domain Squatting. In *Proceedings of the 17th Information Security Conference (ISC)* (2014), pp. 291–308.

[142] Nikiforakis, N., Invernizzi, L., Kapravelos, A., Acker, S. V., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)* (2012), pp. 736–747.

[143] Nikiforakis, N., Maggi, F., Stringhini, G., Rafique, M. Z., Joosen, W., Kruegel, C., Piessens, F., Vigna, G., and Zanero, S. Stranger Danger: Exploring the Ecosystem of Ad-Based URL Shortening Services. In *Proceedings of the 23rd International Conference on World Wide Web* (2014), WWW '14, pp. 51–62.

[144] Nikiforakis, N., Van Acker, S., Meert, W., Desmet, L., Piessens, F., and Joosen, W. Bitsquatting: Exploiting Bit-Flips for Fun, or Profit? In *Proceedings of the 22nd international conference on World Wide Web* (2013), ACM, pp. 989–998.

[145] Nixon, A., and Camejo, C. DDoS Protection Bypass Techniques. *Black Hat USA* (2013).

[146] Norton. Norton Cyber Security Insights Report 2017 Global Results. https://www.symantec.com/content/dam/symantec/docs/about/2017-ncsir-global-results-en.pdf, 2018.

[147] Olson, P. The Largest Cyber Attack in History Has Been Hitting Hong Kong Sites. http://www.forbes.com/sites/parmyolson/2014/11/20/the-largest-cyber-attack-in-history-has-been-hitting-hong-kong-sites/, 2014.

[148] Patrick Howell, O. Report: Cybercrime Causes Over $600 Billion in Damage Annually. https://www.cyberscoop.com/report-cybercrime-causes-600-billion-damages-annually/, 2018.

[149] Pattanai. Faker Is a PHP Library That Generates Fake Data for You. https://github.com/teepluss/laravel-faker.

[150] Pauli, D. DDoS Defences Spiked by CloudPiercer Tool - Paper. http://www.theregister.co.uk/2015/10/08/cloudpiercer_tool_lifts_ddos_protection_cloak_from_70_percent_of_sites/, 2015.

[151] Perez, C. DNS Recon. https://github.com/darkoperator/dnsrecon, 2015.

[152] PhantomJS. PhantomJS - A Headless WebKit Scriptable With a JavaScript API. http://phantomjs.org, 2014.

[153] Plohmann, D., Yakdan, K., Klatt, M., Bader, J., and Gerhards-Padilla, E. A Comprehensive Measurement Study of Domain Generating Malware. In *25th USENIX Security Symposium* (2016), pp. 263–278.

[154] Prince, M. CloudFlare Now Supporting More Ports. https://blog.cloudflare.com/cloudflare-now-supporting-more-ports/, 2012.

[155] Prince, M. The DDoS That Almost Broke the Internet. https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/, 2013.

[156] Proximic. Proximic Spider. http://www.proximic.com/spider.html, 2015.

[157] Q-Success. Market Share Trends for Content Management Systems for Websites. http://w3techs.com/technologies/history_overview/content_management, 2015.

[158] Ramasubramanian, V., and Sirer, E. G. Perils of Transitive Trust in the Domain Name System. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement* (2005), USENIX Association, pp. 35–35.

[159] RAPID7. Vulnerability and Exploit Database: BIND TKEY Query Denial of Service. https://www.rapid7.com/db/modules/auxiliary/dos/dns/bind_tkey, 2015.

[160] Rapid7 Labs. Internet-Wide Scan Data Repository. Project Sonar: IPv4 SSL Certificates. https://sonar.labs.rapid7.com/, 2015.

[161] Rapid7 Labs. Project Sonar. https://sonar.labs.rapid7.com/, 2015.

[162] Richardson, L. Beautiful Soup. http://www.crummy.com/software/BeautifulSoup/, 2013.

[163] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. SIP: Session Initiation Protocol. Tech. rep., 2002.

[164] Rosiello, A. P., Kirda, E., Kruegel, C., and Ferrandi, F. A Layout-Similarity-Based Approach for Detecting Phishing Pages. In *Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on* (2007), IEEE, pp. 454–463.

[165] Rossow, C. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium* (February 2014).

[166] Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Core.

[167] Scheitle, Q., Chung, T., Hiller, J., Gasser, O., Naab, J., van Rijswijk-Deij, R., Hohlfeld, O., Holz, R., Choffnes, D., Mislove, A., et al. A First Look at Certification Authority Authorization (CAA). *ACM SIGCOMM Computer Communication Review 48*, 2 (2018).

[168] Schlamp, J., Carle, G., and Biersack, E. W. A Forensic Case Study on as Hijacking: The Attacker's Perspective. *ACM SIGCOMM Computer Communication Review 43*, 2 (2013), 5–12.

[169] Schlamp, J., Gustafsson, J., Wählisch, M., Schmidt, T. C., and Carle, G. The Abandoned Side of the Internet: Hijacking Internet Resources When Domain Names Expire. In *International Workshop on Traffic Monitoring and Analysis* (2015), Springer, pp. 188–201.

[170] Schroeder, B., Pinheiro, E., and Weber, W.-D. DRAM Errors in the Wild: A Large-Scale Field Study. In *ACM SIGMETRICS Performance Evaluation Review* (2009), vol. 37, ACM, pp. 193–204.

[171] Scikit-learn developers. Encoding Categorical Features. http://scikit-learn.org/stable/modules/preprocessing.html, 2017.

[172] Scikit-learn developers. Homogeneity, Completeness and V-Measure. http://scikit-learn.org/stable/modules/clustering.html, 2017.

[173] Sedo Holding AG. Reports: Sedo Holding AG. http://web.archive.org/web/20140420140532/http://www.sedoholding.com/en/investors/reports/, 2014.

[174] Serverfault. How Is DNS Lookup Order Determined? http://serverfault.com/questions/355414/how-is-dns-lookup-order-determined, 2012.

[175] Shapira, D. Imperva Incapsula IP Protection: The First in Protecting Single IPs. https://www.incapsula.com/blog/ip-protection-generally-available.html, 2016.

[176] Smith, G. When Hackers Steal a Web Address, Few Owners Ever Get It Back. http://www.huffingtonpost.com/2014/09/29/domain-theft_n_5877510.html, 2014.

[177] Starov, O., Dahse, J., Ahmad, S. S., Holz, T., and Nikiforakis, N. No Honor Among Thieves: A Large-Scale Analysis of Malicious

Web Shells. In *Proceedings of the 25th International World Wide Web Conference (WWW)* (2016).

[178] Starr, J. PayPal Phishing Spam. https://perishablepress.com/paypal-phishing-spam/, 2012.

[179] StartCom. StartCom Certificate Policy and Practice Statements. https://www.startcomca.com/policy.pdf, 2017.

[180] StatDNS. TLD Zone File Statistics - April 2018 Reports. https://www.statdns.com/, 2018.

[181] Stephenson, MaAnna. Disable XML-RPC in WordPress to Prevent DDoS Attack. http://www.blogaid.net/disable-xml-rpc-in-wordpress-to-prevent-ddos-attack, 2014.

[182] Story, H., and Sambra, A. Friending on the Social Web. http://bblfish.net/tmp/2011/05/09/, 2011.

[183] Sullivan, N. DDoS Prevention: Protecting the Origin. https://blog.cloudflare.com/ddos-prevention-protecting-the-origin/, 2013.

[184] SURBL. URI Reputation Data. http://www.surbl.org, 2016.

[185] Szurdi, J., Kocso, B., Cseh, G., Felegyhazi, M., and Kanich, C. The Long "Taile" of Typosquatting Domain Names. In *Proceedings of the 23rd USENIX conference on Security Symposium* (2014), USENIX Association, pp. 191–206.

[186] Tartarelli, M. A Tale of a DNS Packet (CVE-2016-2776). http://blog.infobytesec.com/2016/10/a-tale-of-dns-packet-cve-2016-2776.html, Oct 2016.

[187] Tezzaron Semiconductor. Soft Errors in Electronic Memory – A White Paper. https://tezzaron.com/media/soft_errors_1_1_secure.pdf, 2004.

[188] The Internet Foundation in Sweden. IIS Zone Data. https://zonedata.iis.se, 2018.

[189] The PHP Group. PHP: Phpinfo - Manual. http://php.net/manual/en/function.phpinfo.php, 2015.

[190] The Spamhaus Project Ltd. The Domain Block List. https://www.spamhaus.org/dbl/, 2016.

[191] Triukose, S., Al-Qudah, Z., and Rabinovich, M. Content Delivery Networks: Protection or Threat? In *Computer Security–ESORICS 2009*. Springer, 2009, pp. 371–389.

[192] Twilio. Lookup. https://www.twilio.com/lookup, 2017.

[193] URL Void. Website Reputation Checker Tool. http://www.urlvoid.com/, 2016.

[194] Van Goethem, T., Chen, P., Nikiforakis, N., Desmet, L., and Joosen, W. Large-Scale Security Analysis of the Web: Challenges and Findings. In *Trust and Trustworthy Computing* (2014), vol. 7, Springer, pp. 110–125.

[195] Vaughan-Nichols, S. J. Worst DDoS Attack of All Time Hits French Site. http://www.zdnet.com/article/worst-ddos-attack-of-all-time-hits-french-site/, 2014.

[196] VirusTotal. Free Online Virus, Malware and URL Scanner. https://virustotal.com, 2014.

[197] Vissers, T., Barron, T., Van Goethem, T., Joosen, W., and Nikiforakis, N. The Wolf of Name Street: Hijacking Domains Through Their Nameservers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 957–970.

[198] Vissers, T., Joosen, W., and Nikiforakis, N. Parking Sensors: Analyzing and Detecting Parked Domains. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)* (2015), Internet Society, pp. 53–53.

[199] Vissers, T., Spooren, J., Agten, P., Jumpertz, D., Janssen, P., Van Wesemael, M., Piessens, F., Joosen, W., and Desmet, L. Exploring the Ecosystem of Malicious Domain Registrations in The. Eu TLD. In *International Symposium on Research in Attacks, Intrusions, and Defenses* (2017), Springer, pp. 472–493.

[200] Vissers, T., Van Goethem, T., Joosen, W., and Nikiforakis, N. Maneuvering Around Clouds: Bypassing Cloud-Based Security Providers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), ACM, pp. 1530–1541.

[201] Vixie, P. Domain Name Abuse: How Cheap New Domain Names Fuel the eCrime Economy. Presentation at RSA Conference 2015, 2015.

[202] Walker, J. The Self-Driving Car Timeline – Predictions From the Top 11 Global Automakers. https://www.techemergence.com/self-driving-car-timeline-themselves-top-11-automakers/, 2017.

[203] Wander, M., Schwittmann, L., Boelmann, C., and Weis, T. GPU-Based NSEC3 Hash Breaking. In *Network Computing and Applications (NCA), 2014 IEEE 13th International Symposium on* (2014), IEEE, pp. 137–144.

[204] Wang, Y.-M., Beck, D., Wang, J., Verbowski, C., and Daniels, B. Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting. In *Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2* (Berkeley, CA, USA, 2006), SRUTI'06, USENIX Association.

[205] Wessels, D. Evidence of Bitsquatting in COM/NET Queries. https://www.nanog.org/meetings/nanog54/presentations/Tuesday/Wessels.pdf, 2012.

[206] Wessels, D. http://serverfault.com/a/819858, 2016.

[207] Westervelt, R. Cloud-Based DDoS Protection Is Easily Bypassed, Says Researcher. http://www.crn.com/news/security/240159295/cloud-based-ddos-protection-is-easily-bypassed-says-researcher.htm, 2013.

[208] WhoisGuard, Inc. WhoisGuard: Protect Your Privacy Using WhoisGuard (Whois Protection). http://www.whoisguard.com/, 2013.

[209] Whoxy. Whois Lookup API. https://www.whoxy.com/#api, 2017.

[210] WIPO - World Intellectual Property Organization. WIPO Guide to the Uniform Domain Name Dispute Resolution Policy (UDRP). http://www.wipo.int/amc/en/domains/guide/index.html, 2018.

[211] Woods, B. 15 of the Most Expensive Domains of All Time. https://thenextweb.com/shareables/2013/08/13/15-of-the-most-expensive-domains-of-all-time/, 2013.

[212] Woolf, N. DDoS Attack That Disrupted Internet Was Largest of Its Kind in History, Experts Say. https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

[213] WordPress. WordPress: Blog Tool, Publishing Platform, and CMS. http://wordpress.org, 2015.

[214] Zarras, A., Kapravelos, A., Stringhini, G., Holz, T., Kruegel, C., and Vigna, G. The Dark Alleys of Madison Avenue: Understanding Malicious Advertisements. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), ACM, pp. 373–380.

[215] Zeifman, I. How to Prevent “Origin Exposing” Attacks (CloudPiercer Study). https://www.incapsula.com/blog/cloudpiercer-origin-ddos-attack.html, 2015.

[216] Zhang, K., and Shasha, D. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM journal on computing 18*, 6 (1989), 1245–1262.

[217] ZyTrax, Inc. DNS BIND Operations Statements: Max-Cache-Ttl. http://www.zytrax.com/books/dns/ch7/hkpng.html#max-cache-ttl, 2015.

List of publications

- Vissers, T., Nikiforakis, N., Bielova, N., and Joosen, W.
  "Crying wolf? on the price discrimination of online airline tickets."
  In: *Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETS 2014).*

- Vissers, T., Joosen, W., and Nikiforakis, N.
  "Parking sensors: Analyzing and detecting parked domains."
  In: *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*, Internet Society, pp. 53–53.

- Vissers, T., Van Goethem, T., Joosen, W., and Nikiforakis, N.
  "Maneuvering around clouds: Bypassing cloud-based security providers."
  In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (2015)*, ACM, pp. 1530–1541.

- Vissers, T., Spooren, J., Agten, P., Jumpertz, D., Janssen, P., Van Wesemael, M., Piessens, F., Joosen, W., and Desmet, L.
  "Exploring the ecosystem of malicious domain registrations in the. eu tld."
  In: *Proceedings of the 20th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID 2017)*, Springer, pp. 472–493.

- Vissers, T., Barron, T., Van Goethem, T., Joosen, W., and Nikiforakis, N.
  "The wolf of name street: Hijacking domains through their nameservers."
  In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*, ACM, pp. 957–970.

List of presentations and trainings

- Nov 2017 - Dallas, USA.
  ACM CCS 2017
  Presenting academic paper *“The Wolf of Name Street: Hijacking Domains Through Their Nameservers”*
- Oct 2017 - Toronto, CA.
  M3AAWG, 41st General Meeeting
  Invited speaker on *“Exploring the ecosystem of malicious domain registrations in the .eu TLD”*
- Sep 2017 - Atlanta, USA.
  RAID 2017
  Presenting academic paper *“Exploring the ecosystem of malicious domain registrations in the .eu TLD”’*
- Feb 2017 - Leuven, BE.
  Secure Application Development (SecAppDev 2017)
  Training on *“Denial-of-Service Attacks: Solutions and Pitfalls”*
- Sep 2016 - Zaventem, BE.
  OWASP Chapter Meeting
  Invited speaker on *“CloudPiecer: Bypassing Cloud-based Security”*
- Oct 2015 - Denver, USA.
  ACM CCS 2015
  Presenting academic paper *“Maneuvering around clouds: Bypassing cloud-based security providers”*
- Feb 2015 - Leuven, BE.
  International Cyber Security Strategy Congress (ICSS 2015)
  Invited speaker on *“Web-based Device Fingerprinting”*

- Feb 2015 - San Diego, USA.
  NDSS 2015
  Presenting academic paper *"Parking sensors: Analyzing and detecting parked domains"*

- Jul 2014 - Amsterdam, NL.
  HotPETS 2014
  Presenting academic paper *"Crying Wolf? On the Price Discrimination of Online Airline Tickets"*

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DISTRINET
Celestijnenlaan 200A box 2402
B-3001 Leuven
thomas.vissers@cs.kuleuven.be
https://distrinet.cs.kuleuven.be