DNS-based detection of malicious activity

Dodopoulos, R.

Award date:
2015

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Section Security and Embedded Networked Systems (SENS)
Security Research Group (SEC)

DNS-based Detection of Malicious Activity

Master Thesis

Romanos Dodopoulos

Assessment committee:

Dr. Nicola Zannone (TU/e)
Roland van Rijswijk-Deij, M.Sc (SURFnet)
Dr. Rudolf Mak (TU/e)

version 1.0

Eindhoven, November 2015

Abstract

The Domain Name System (DNS) is the primary directory service of the Internet and an essential component of almost every network service. The DNS has made the Internet more accessible to regular users via the human-readable domain names. However, since it is well-known it is frequently exploited by attackers. Analyzing or monitoring DNS traffic is an efficient way to detect attacks. Cyber attacks are a highly significant threat for the reliability and operation of on-line services. Moreover, they threaten individuals, where privacy and personal information are at stake.

In this report, we focus on DNS-based detection of malicious activity. We examine and summarize dynamic methods that detect malicious activities and, especially, previously unknown malicious domain names. The methods are also classified according to a set of well-defined criteria. Next, we focus on the effective detection of infected computers within a network by taking advantage of DNS-based monitoring. To this end, we evaluate a detection system that combines DNS-based and flow-based monitoring. The basic approach utilizes DNS-based prefiltering before performing flow-based filtering. The goal is to increase performance, while maintaining a high level of accuracy. Our evaluation indicates that although the performance is increased, the accuracy is depleted and the method cannot be considered adequate. Finally, we examine the DNS-based monitoring as a standalone detection method. The performed measurements indicate that this detection method has low recall and precision rates. Therefore, it is unsuitable for live deployment at present.

Acknowledgments

- I would like to express my gratitude to Dr. Nicola Zannone, my internal supervisor at TU/e, who gave me invaluable suggestions and comments on my work.
- I would also like to express my gratitude to Roland M. van Rijswijk-Deij, my external/daily supervisor at SURFnet, for his guidance and assistance through the work of this thesis.
- I am grateful to Xander Jansen from SURFnet for the useful discussions and the information that he shared with me.
- I want to thank the Spamhaus Project for providing the blacklists the were used in the performed experiments.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

The Domain Name System (DNS) is the primary directory service of the Internet. Therefore, it is an essential component of almost every network service. The DNS has made the Internet more accessible to regular users via the human-readable domain names. Its main purpose is to translate (or resolve) domain names, such as "*www.example.com*", into IP addresses, such as "*93.184.216.34*". However, since it is well-known it is frequently exploited by attackers.

There are exploitation techniques that leverage its properties and actively abuse it. These techniques have an impact on the DNS by mainly exploiting flaws in the software. As a result, compromised servers might host fake information (e.g., cache poisoning attack). However, other types of attacks consider adequate to legitimately utilize domain names. These malicious activities do not directly rely on the DNS, although they misuse it for greater success.

Analyzing or monitoring DNS traffic is an efficient way to detect attacks. Cyber attacks are a highly significant threat for the reliability and operation of on-line services. Moreover, they threaten individuals, where privacy and personal information are at stake. When a node is compromised the only way to prevent further negative consequences is to detect it as soon as possible.

Security analysts are increasingly focusing on the DNS in order to detect malicious activities. It has also been argued that DNS-based monitoring is the most effective security countermeasure. Sensors that monitor DNS traffic have been deployed in large networks and operators. However, the successful identifications of malicious activities usually depend on manual investigation. This is a costly process that cannot be effective in large scale. Consequently, dynamic detection methods have been proposed in order to automate the process.

The generic research question of this report is the following: *can we detect malicious activity by observing DNS traffic?* In order to answer this broad question we break it down into three specific research (sub-)questions. Each answer to a research question contributes to the answer of the generic question. The specific research questions are the following:

Q 1 Can we dynamically detect new malicious activities by monitoring DNS traffic?

Q 2 Can we effectively detect infected computers within a network by taking advantage of DNS-based monitoring?

Q 3 Can DNS-based monitoring be a valid standalone detection method?

In order to answer the first question (Q1), we investigate proposed methods [29, 30, 31, 32] that detect malicious activities and, especially, previously unknown malicious domain names. These methods automatically identify entities that are controlled by attackers.

However, our literature survey reveals that designing dynamic systems to detect new malicious activities remains a difficult task. Most of the existing methods have limited real-life applications and have not actually evolved further. The main obstacle of the proposed methods is their false positive rate. False positives are the pitfall of a dynamic system because it loses its reliability.

Due to the lack of resources and available implementations, further comparison of the methods is not feasible.

The second research question (Q2) focuses on reliably detecting existing threats within a network. The infected nodes are insider threats because they have already penetrated the network. After they are detected, ideally, network administrators put them in quarantine until they are cleaned. In addition to DNS-based detection, commercial and widely adopted systems detect infections by monitoring network flows. These systems traditionally leverage historical information of known malicious destinations (blacklists) in order to identify suspicious connections. The strong advantage of DNS-based monitoring is performance because the volume of DNS traffic is just a fraction of the network flows. On the other hand, an observed flow can increase the confidence of a DNS-based alert, which by itself is not a conclusive evidence that the computer contacted the malicious destination.

In order to answer the second research question (Q2), we examine the effectiveness of a single detection system which combines DNS-based and flow-based filtering. The intuition is that observing an established connection (flow) increases the confidence of a DNS-based alert. The system aims at effectively reducing a given blacklist using DNS-based prefiltering to the minimum possible. Next, the output, the intermediate blacklist, is fed to the flow-based filtering process. We define two scenarios with alternative designs in order to find the optimal setup. **Scenario 1** places the DNS sensor between the resolver and the remote name servers. **Scenario 2** places the DNS sensor between the clients and the resolver. The second has the advantage that there is not cache involved and all the queries of the users are visible. On the other hand, the plain queries compose privacy-sensitive data and should be treated with respect.

Furthermore, we define **Scenario 3** that examines the effectiveness of exclusively monitoring DNS traffic, without considering the network flows. This can answer the third research question (Q3). A valid method should have high precision, recall and accuracy rates. Specifically, we want to identify how much information does the DNS-based detection miss in comparison to flow-based detection.

Outline The remainder of this report is structured as follows.

Chapter 2 explains the Domain Name Systems.

Chapter 3 introduces the challenges regarding the dynamic detection of malicious activities.

Chapter 4 classifies the related literature according to a set of well-defined criteria.

Chapter 5 presents the proposed method that combines flow-based and DNS-based monitoring in order to increase performance and accuracy.

Chapter 6 documents the methodology and the measurements to evaluate the proposed method.

Chapter 7 contains the results and the evaluation.

Chapter 8 concludes this report.

Chapter 2

Domain Name System

The Domain Name System (DNS) is the primary directory service of the Internet. Therefore, it is an essential component of almost every network service. In this chapter, we present an overview of the DNS along with the domain name hierarchy and its basic components. We describe the functionality of the system and, finally, specific advanced technologies.

2.1 Overview

The DNS has been used for many years in order to make the Internet more accessible to regular users. Its purpose is to function as the "phone book" of the Internet. Computers actually communicate using the Internet Protocol (IP) which is sufficient for establishing an active connection. However, humans cannot easily memorize this structure of numbers, and, besides, IP addresses are often interchanged. Domain names are essential for the memorization of digital locations. Moreover, it makes it possible to assign a changeless reference without reserving a costly static IP address.

The DNS is a distributed database system. Its role is to correlate any domain name with at least one IP address. Clients send DNS queries including domain names to initiate the protocol and receive DNS answers (or replies) with the corresponding IP addresses. Domain names are usually constant and associated with the trademark of a company or an organization. Normally, registering a domain is easier and cheaper because the collection of IP addresses (v4) is limited.

The necessity of domain names was already realized during the ARPANET era[1]. Initially, a centralized location was providing a single "hosts.txt" file containing the mapping of human-friendly names to numerical addresses. Meanwhile, the number of interconnected hosts had been increasingly growing. As a result, the Domain Name System was published in 1983 [1, 2].

2.2 Domain name space

An important part of implementing and operating domain name facilities is the definition of the hierarchical name space. During the ARPANET era, system administrators were setting their own host names. The Network Information Center (NIC) had to manually approve them before inserting them into the "hosts.txt" file. Later on, organizations wanted the ability to register additional domain names under the existing ones, with some form of local hierarchy. This hierarchy would roughly represent the organizational structure. In addition, faster approval of their modification was a vital request. This flexibility is satisfied by the introduction of the domain name hierarchy.

[1]The Advanced Research Projects Agency Network (ARPANET) was the leading implementation of the early Internet in the 1960s.

2.2.1 Domain name hierarchy

The hierarchical structure of the domain name space is a tree that consists of five distinct abstract levels. An illustration of the tree is given in Figure 2.1 and the abstract levels are listed in the right side of the same figure. At the top we can find the root-level domain that has only 13 root name servers, from A-root to M-root, deployed in approximately 500 sites across the world[2]. The root domain label is actually embedded in every DNS query which contains an invisible empty label at the end of the actual domain name. For instance, "*www.example.com.*" specifies the exact location and is referred to as absolute domain name or, more formally, fully qualified domain name (FQDN).

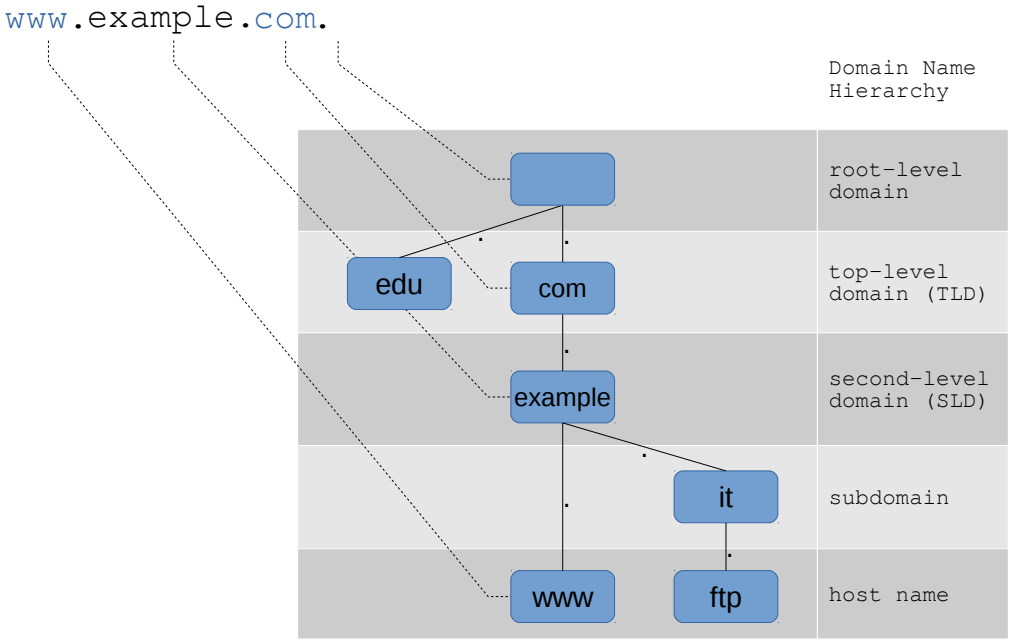


Figure 2.1: An overview of the domain name space. The tree in the middle is the hierarchical and logical structure of the domain namespace. The illustrated domain names are: "*edu*", "*www.example.com*" and "*ftp.it.example.com*". In the right side is the domain name hierarchy which separates the levels of the tree.

In any domain name, like the previously mentioned example, we can clearly distinguish the hierarchy of the different labels separated by the full stop character. By convention, it is written from right to left. The rightmost element is the root-level (although it is actually invisible/null) and the leftmost is the host name label. As we can see in Figure 2.1, lower (or left) of the root-level is the top-level domain (TLD) label. In the given FQDN ("*www.example.com.*"), the TLD domain label is the "*com*" field. It is followed by the second-level domain label (SLD), i.e. "*example*" in the our illustration. Additionally, there can be numerous subdomains before the actual label "*www*" of the host. The existence of a subdomain is not mandatory as it is for the TLD and SLD labels. Generally speaking, a domain name identifies the path that a client has to follow to locate a host, starting from the universally known location of the root.

2.2.2 DNS zone

The DNS allows decentralized administration and uses a distributed database because different authorities need to coexist in the same system. The whole name space is partitioned into various *DNS zones*. Specifically, a zone is a portion of the domain name space which is delegated to a

[2]http://www.root-servers.org, retrieved April 20, 2015

single administrator. The last one is authorized to maintain the DNS records of this contiguous portion. However, a DNS zone is not necessarily a complete branch of the domain name space. This means that a subdomain could belong to different DNS zones than the upper domain name and, therefore, under different administrations.

To illustrate the point further, let us consider the "*www.example.com*" domain name. A single entity is authoritative for the "*example.com*" zone. This entity manages DNS records that map domain names to resources. Specifically, these records contain the IP address of the web service hosted at "*www.example.com*" among others.

Furthermore, entities may have departments. For instance, the entity that is authoritative for the "*example.com*" domain could have an information technology (IT) department. In this case, the "*it.example.com*" domain could be managed by the department itself. This is an individual DNS zone under different authority. The IT department can register additional domain names under its own. For instance, it can register the "*ftp.it.example.com*" domain name for its ftp service. Any registration can happen without contacting the upper domain "*example.com*" administrator. The IT department is the only responsible for maintaining its own resource records.

2.3 Architecture

The Domain Name System uses a client-server model, where clients seek information located on servers throughout the Internet. The distributed model of the DNS involves various separate components and acts as a distributed database system. Moreover, it aims to equally distribute the network load in order to ensure availability and increased performance of the protocol. As a matter of fact, the DNS is continuously used by all the users of the Internet.

The DNS has three major fundamental elements [3]: first, the *domain name space and resources records* (RR) that specifies the hierarchical name space and data associated with domain names. Second, the *name servers* that are server programs and hold information about a subset of the domain space. Third, the *resolvers* which are the client programs of the protocol and extract the information from the name servers. Resolvers can optionally implement a cache.

An overview of the DNS components in a typical network is given in Figure 2.2. The left side of the figure shows the hosts (users). They belong to an ordinary network and want to access a domain name. It is not necessary that they are all in a single local area network. It can be a group of users sharing the same Internet service provider. Consequently, they share some common routing settings, such as the digital location of the default recursive name server. The latter is assigned by the network administrator. It can be seen in the figure, connected to the clients with the red curved lines. In the right side of the figure are the remote name servers, in the Internet, that contain the information that the hosts want to retrieve.

Now that we have defined the internal and external structure of the network, we will explain the role of a resolver, the utility of a name server, and finally, we will discuss how the DNS operates with a paradigm of a user case.

2.3.1 Stub resolver

Any host must have direct access to a local resolver which transforms requests from applications into standard queries. This local resolver is usually called a stub resolver. After processing the queries, it replies back to the application with a valid answer.

The norm is that the local resolver utilizes a network-wide recursive name server. The last one is responsible for answering the questions for the complete network. This is done to offload the network and prevent the deployment of a full resolver in every host. Moreover, it ensures the sharing of cache among the hosts. The use of centralized cache allows for a high hit ration, which eliminates the redundancy of unused information. This kind of network configuration is very common, where lighter stub resolvers are installed in the operating system. A stub resolver acts as the front end to the resolver located in the recursive caching name server [4].

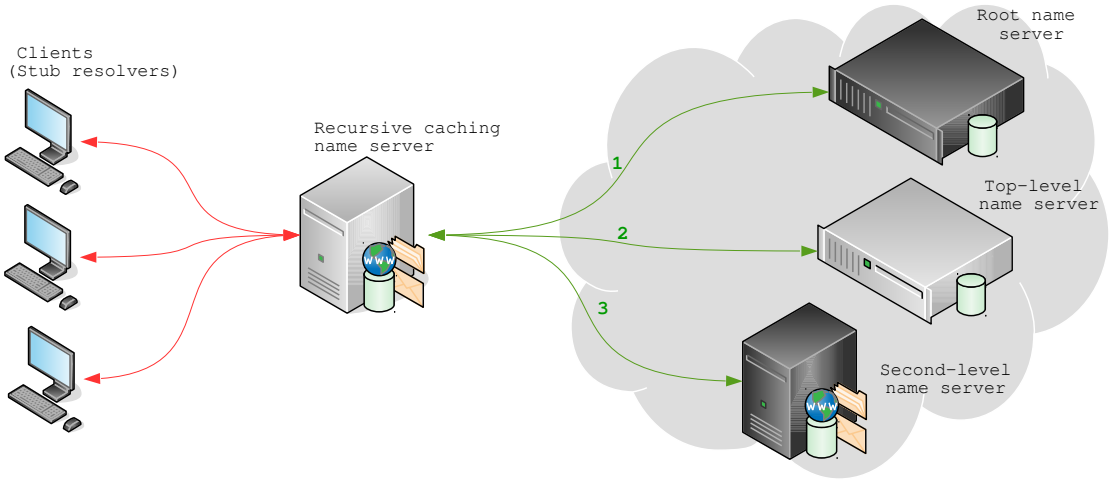


Figure 2.2: Overview of the DNS

2.3.2 Recursive caching name server

A recursive (caching) name server receives queries from clients and it is responsible to resolve them. By convention, it is simplistically referred to as resolver. Alternatively, the term recursive DNS server (RDNS) is used less frequently.

Clients initiate queries, which are simple questions for the IP address of a specific domain name. A resolver is responsible to conclusively answer with either an IP address or an error. An error means that the resolver could not find the requested information. Usually, this is caused because the domain name actually does not exist. While resolving a query, a resolver contacts as many servers as needed in order to obtain the information. We describe in detail the resolution process in Section 2.4.

A resolver has a very important goal to eliminate network delay and the load at the remote name servers by utilizing a cache [3]. It answers requests from its cache of prior responses. Consequently, a cache that is shared by multiple nodes is more efficient than a non-shared cache. A resolver that utilizes a cache is properly named as recursive caching name server.

2.3.3 Authoritative name servers

An authoritative name server (AuthNS) is the authority for a specific zone of the domain name space. Authoritative information is organized and maintained by domain administrators. The information is the complete database for a specific "pruned" subtree of the domain space [4]. This database contains the resource records of the zone. Records are, for example, the mapping of domain names to either IP addresses or references to another zone. Authoritative name servers also contain information about other applications of the DNS, which are described in Section 2.5.4.

In the right side of Figure 2.2, we can distinguish the separate authorities for three different abstract levels of the domain hierarchy. At top is the root name server that is authoritative for the root zone. It contains a list with every top-level domain name of the Internet. Although the distributed nature of the DNS, the root zone is coordinated by a single organization[3]. Currently, there are 1011 TLDs[4] worldwide, which are either generic (e.g., com, net) or country-code (e.g., nl). Below the root name server (Figure 2.2) there is a top-level name server, which is delegated administrative responsibility for a specific TLD. In turn, it hosts a list of lower-level authoritative name servers in their respective zone. At the bottom right of the figure we can find an authoritative name server for a specific SLD (e.g., "*example.com*").

[3]https://www.iana.org/domains, retrieved July 2015.
[4]http://data.iana.org/TLD/tlds-alpha-by-domain.txt, last updated Mon Jul 13 07:07:01 2015 UTC.

2.4 Query resolution

Clients send queries to the resolver (the recursive caching name server in Figure 2.2) whenever they want to access a domain name. The resolution of a user query is the process of translating a domain name to an IP address. It may involve several network accesses and an arbitrary amount of time [4]. The resolver is trusted to perform all the necessary steps to reply with the requested information.

Normally, the resolution process requires iteration of multiple steps. As an example consider the case where a client requests the IP address of "*www.example.com*" from the resolver. In turn, the resolver asks the root name server. This step is the first (1) green curved line in Figure 2.2. It is highly possible that the root name server does not know the exact information. However, it knows every existing TLD domain name (see Section 2.3.3). Then, it suggests the resolver to contact the name server for the "*com*" domain name. The resolver obtains this information and repeats the question to the top-level name server. That is the second (2) connection in Figure 2.2. It is possible that neither this server knows the final digital location. Therefore, it replies with the location of the name server for "*example.com*". The resolver retransmits for a third (3) time the query to the last name server. Eventually, the name server replies with the IP address for "*www.example.com*". The client receives this information from the resolver and connects to the destination.

The resolution process can be reduced when a cache is used. A resolver can use cache memory to temporarily store every information. As a result, whenever a domain names has been priorly requested, the resolver replies instantly with the cached information. Alternatively, a domain name with the same TLD may has been queried. In the last case, the resolver will contact directly the top-level name server, skipping the root.

The time that the domain name is temporarily stored is define by the authority of that domain. Every DNS record has a Time To Live (TTL) field. A resolver caches a domain name for as long as its TTL value indicates.

During the query resolution process, a resolver may return an error message. The error means that the resolver failed to correctly retrieve the information. One of the reasons could be that the requested domain name actually does not exist. In this case, the resolver replies with a non-existent domain (NXDomain) response. Although, NXDomain is not the only possible error message, a discussion of other messages is out of the scope of this report.

2.5 Advanced technologies

The ubiquitous integration of the DNS caused the development of technologies outside the standard scope. This has been done for many purposes, ranging from performance boosting to its utilization for other applications. In this section, we discuss the main extensions to the DNS.

2.5.1 DynDNS

Dynamic DNS (DDNS or DynDNS) is a mechanism that allows real time updating of resource records in a name server. DynDNS can assign a changeless reference to machines that change often IP address. A DynDNS client automatically updates the DNS record whenever the IP address changes. There are over fifty[5] DynDNS providers that offer free and fee-based services. It is widely used by users who want to easily host a personal website or connect to their home Virtual Private Network (VPN). Users can choose a subdomain name out of a list of existing domain names owned by the DynDNS provider.

[5] http://blogs.cisco.com/security/dynamic-detection-of-malicious-ddns, retrieved July 2015.

2.5.2 Reverse DNS resolution

Until now, we have discuss the process of resolving a domain name to an IP address. This is also known as forward DNS resolution. Similarly, reverse DNS resolution (rDNS) can get an IP address and determine the associated domain name. This process is also known as reverse DNS lookup.

Reverse resolution is primary used for network troubleshooting. It is feasible because a host name is assigned to every Internet-reachable host. This information is routed in the "*arpa*" TLD that stands for Address and Routing Parameter Area. A query for reverse resolving contains a different record type than a "forward" query.

2.5.3 Prefetching

In Section 2.4 we mentioned that the resolution of a user query may involve an arbitrary amount of time. This means that users may perceive latency. Prefetching is a mechanism used to pre-resolve domain names that are likely to be requested in the near future.

A resolver can prefetch domain names that are stored in the cache and expire shortly. As a result, it keeps the cache up to date and popular domain names never expire. Due to the fact that this functionality gives about 10 percent more traffic and load on the resolver, it is disabled by default[6].

Furthermore, prefetching is deployed in web browsers. The idea is that websites tend to have multiple links that may be interesting for visitors. Therefore, whenever a web page is fetched the browser automatically pre-resolves all the domain names contained in it. For instance, it is extremely likely that a user will select a result of a search engine. Prefetching attempts to save approximately 200 milliseconds[7] in users' navigation. Although this might be trivial, the main goal of prefetching is to prevent the worst case scenario. A severe delay during the query resolution is regularly over 1 second[7] which is at least noticeable.

2.5.4 Other applications

The DNS is not only about mapping domain names to IP addresses. The DNS has also been used in other applications for distributing information. Mail transfer agents are such an example. They leverage the DNS to deliver e-mails faultlessly. The mail exchanger mapping has a specific resource record that provides an extra layer of load distribution and fault tolerance.

Furthermore, an important nonstandard application is the Domain Name System Block List (DNSBL). A DNSBL is an on-line database of blacklisted elements that can be queried in realtime. We will discuss blacklists, in detail, in Section 3.4.1. A DNSBL is commonly used to filter bulk incoming emails. It can either support blacklisted domain names or IP addresses. DNSBLs utilize the light DNS format to offer conventional queries and transportation of the information. The small DNS answers can prevent a bandwidth overhead.

[6]Unbound 1.5.4 documentation, https://www.unbound.net/documentation/unbound.conf.html, retrieved Aug 2015.

[7]https://www.chromium.org/developers/design-documents/dns-prefetching, retrieved July 2015.

Chapter 3

Challenges

DNS traffic is generated by the majority of a computer's networking usage. Since the Domain Name System became widespread it has been frequently used by attackers. Generic malicious activities may use domain names similarly to legitimate services. Furthermore, other malicious activities may exploit properties of the DNS in order to achieve specific goals. In this chapter, we introduce the attacker model and misuses of the DNS. Then, we focus on detection methods and their privacy implications.

3.1 Attacker model

The DNS has properties that can increase the success of specific malicious activities. According to Cisco Security Intelligence Operations[1]: a malicious software (a.k.a. malware) is designed to "damage, disrupt, steal, or in general inflict some other bad or illegitimate action on data, hosts, or networks". Attackers use the DNS whenever they want to establish a channel of communication with the spread malware. In this report, we focus on malicious activities that transfer information through the Internet. They, for instance, exfiltrate stolen private information or attack legitimate on-line services.

Malicious software that communicate with the attacker can either be remotely controlled or not. The first kind is more flexible and adjustable. The second kind is usually more static and cannot evolve. In either case, attackers want to keep the communication undetected. Both kinds of malware can collect sensitive information to a dropzone. The remotely controlled malware can additionally receive updates or commands for further attacks.

Finally, malicious domain names are used by various types of attacks. In addition to using domain names for establishing a connection, malicious domain names may direct users to phishing, spamming or generic malicious websites.

3.1.1 Spyware

A spying software (a.k.a. spyware) is a type of malware that secretly gathers information about a person or an organization. A spyware is one of the most dangerous forms of malware because it does not damage or disturb services but it looks for and discloses personal information and real assets[2]. For instance, criminal organizations use this tool to collects financial information from infected hosts. Other attackers may use it to track on-line activities and deliver targeted advertisements. The last is also called an adware.

In particular, this kind of malware is designed to transmit the collected information to a dropzone, where an unauthorized party can collect them. Finally, some specific kinds may allow remote control over the device by disabling security settings.

[1] http://www.cisco.com/web/about/security/intelligence/virus-worm-diffs.html, retrieved July 2015.
[2] http://www.avg.com/a/us-en/what-is-spyware, retrieved Aug 2015.

3.1.2 Botnets

Botnets are a significant threat for the security and operation of legitimate networks. A botnet is a group of bots remotely controlled by an attacker, called botmaster. Bots are infected computers without the users noticing or consenting. This concentration of machines gives the attacker a powerful tool to perform many illegitimate actions such as distributed denial-of-service (DDoS) attacks, spam campaigns and hosting of phishing websites.

Computers are often compromised through outdated software that makes it vulnerable to exploits. Then, a botmaster can set up a command and control (C&C) communication channel with the infected hosts. This communication has been analyzed by security researchers since its early development [5]. It seems that building a botnet is not as difficult as maintaining it and keeping it undetected -from security analysts or law enforcements- for a long period of time.

Botnets adopted the same architecture as legal networks of bots used for administrating an Internet Relay Chat (IRC) channel. Although the original term was benign, henceforth we consider only its malicious connotation.

Furthermore, this centralized communication of botnets has a single point of failure, the C&C server. If the infected hosts cannot access the C&C server, the botnet instantly become inoperable. Attackers were powerless to resist a take down operation. As a result, they started developing even more sophisticated techniques. There is a variety of obfuscated communication technologies (e.g., P2P-based C&C structures) [5], but from now on we focus on technologies based on the DNS.

Attackers want to hide the C&C server by repeatedly changing its digital location (IP address). They also want a reference in order for the bots to locate and reconnect to the C&C server. An easy way to achieve this goal is by taking advantage of the Domain Name System. However, a naive use of the DNS by a botmaster would still be unprotected from a take down operation. If a single domain name is used and it is forcibly unregistered, there still is a single point of failure.

Attackers' main goal is to prevent loss of control of the entire botnet. A milestone has been the development of automatically generated pseudo-random domains names [5]. Moreover, recent intelligent botnets operate even if their C&C server is blocked [31]. This survivability is due to the encoding of redundant domain names. These domains are used only when the control is lost in order to update the binary of the malware.

3.1.3 Phishing

Phishing is a high-tech scam that attempts to fraudulently acquire sensitive information (such as usernames, passwords, credit card details and bank account information) by masquerading as a trustworthy entity in an electronic communication[3]. Attackers usually reach the target by email or pop-up messages that direct users to disclose information at a fake website.

A phishing attack commonly directs users to official-looking website through similar-looking domain names. Therefore, an unaware and incautious user could be easily deceived. Detecting such malicious domain names is useful and could be used to protect users. Domain names related to phishing are malicious, but they do not indicate infection. In contrast to the implication of infection when connecting to a botnets' C&C domain name, visiting a phishing website does not prove that the user has disclosed any information.

3.2 Exploitation techniques

The DNS is being used by attackers to hide their activities and servers. Naive use of domain names could have no advantage or benefit. On the contrary, advanced techniques are exploited by attackers to become a powerful tool against detection.

[3] http://www.cisco.com/c/en/us/products/security/email-security-appliance/phishing_index.html, retrieved July 2015.

3.2.1 Fast flux

Fast flux is a technique that rapidly modifies DNS records in order to renew the IP address of a domain name. It has been used by legitimate operators to increase performance and availability of their services. The main objective is to equally distribute the network traffic among various servers of the same domain entity. Large server farms [30] and content distribution networks (CDN) have similar redundancy in their behavior.

Technically, fast flux is characterized by short-living TTL value, which usually is less than five minutes[4]. As a result, every temporary storage (cache) will refresh the malicious DNS entry after the time is passed. Meanwhile, the botmaster has already changed the IP address of the original entry. Consequently, any further query for the malicious domain -after this period of time- will get a different IP. This frequent renewal of IP addresses could evade most of the IP address oriented security countermeasures [6].

Fast flux has been intensively used by botnets to evade IP-based blacklist and resist take down operations against them [22]. Interestingly, no legitimate use of this technology had been noticed before the malicious use. Attackers deployed it for the first time in the Storm/Peacomm botnets in 2007 [5]. The malicious objective is that by the time an authority issues a warrant to intervene to take down a bot, its location will have already been changed. Likewise, the addition of a malicious IP into a blacklist is usually slower than the change happening with fast flux.

However, the simple version of fast flux can be neutralized by taking down the registered domain name. This is called simple-flux and its weaknesses led to the emerge of the improved double-flux architecture. In addition to the rapid change of IP addresses in DNS records, double-flux rapidly changes the authoritative name server of an entire DNS zone. In particular, attacker can set up a rogue authoritative name server which is actually part of the fast flux scheme. The server, instead of answering a query like an ordinary name server, it forwards the request to a hidden back-end to obtain the concealed information.

3.2.2 Domain flux

Domain flux is characterized by the abnormal use of multiple domain names. It eliminates the disadvantage of single point of failure, which is noticed in fast flux. It was firstly observed in 2009 by Stone-Gross et al. [7], who identified that bots were using a domain generation algorithm (DGA). The use of a DGA leads to the domain flux behavior. That is, the utilization of multiple domain names by the malware.

A DGA is shared between the botmaster and the bots. Its responsibility is to generate a list of pseudo-random domain names. A shared DGA approximately generates the same list of domain names, regardless the environment of the host machine. A pseudo-random DGA uses a seed, for example the current date, and roughly generates the same list as long as the same seed is used. The communication between the botmaster and the bots is possible because, the former registers a few of the generated domain names beforehand. Then, the bots send consecutive DNS queries, requesting the domain names in the list until one of them is registered. After retrieving the current IP address of the C&C server, the bots communicate with it to receive signals for attacks or deliver sensitive information.

One important part of a DGA is the given seed. Considering only the date could be easily analyzed to predict the malicious domain names of the following days. This is exactly what Stone-Gross et al. [7] did to take over the Torpig botnet. Firstly, they recognized the pattern of the generator. Secondly, they registered first -before the botmaster- the domain names that the botnet was going to use later. Interestingly, the botmaster changed the DGA, after retaking control over his botnet by distributing a new binary. Nowadays, botnets use as seed not only the date but also public information in order to make the domain names unpredictable. For instance, one of Torpig's DGA variations was using a character of the most popular Twitter search[5]. This variable character was used as an additional seed byte.

[4] http://www.spamhaus.org/faq/section/ISP%2520Spam%2520Issues#164, retrieved June 2015.
[5] https://seclab.cs.ucsb.edu/academic/projects/projects/your-botnet-my-botnet/, retrieved July 2015.

The main advantage of domain flux is that the domain name itself is not important and blocking it subsequently is fruitless. This matches the attackers' goal to expose the C&C server for the minimum amount of time. It has been recorded [8] that the whole process (i.e. the time window that the bots can contact the C&C server) takes less than an hour.

DGAs are continuously examined by the academic community [9, 32]. The domain names generated by a DGA are not totally indistinguishable because the generator is not a random oracle. Malware analysts intensively use reverse engineering to interpret these pseudo-random domain names [10]. The goal is to identify a pattern of the domain name generation that could lead to a botnet take down. However, manual investigation and especially reverse engineering is tedious and the results are uncertain. Therefore, researchers have started focusing on dynamical detection.

Finally, it is worth mentioning that domain flux is not only used by botnets. Antonakakis et al. [27] mention that DNS agility, as they refer to domain flux, is used by adware and spywares, among others.

3.3 Monitoring

Detection of malicious activities is crucial for the security, integrity and availability of any network. From a network administrator's point of view, mitigating malicious activity focuses on detecting malicious communications. That is, the communication between infected clients and hidden servers maintained by attackers. This detection varies on the detection method and the activity that can be detected. In the previous section we discussed about the emerging threat of malicious activities that take advantage of the DNS. Below, we focus on such activity and we discuss whether monitoring DNS traffic can be an effective detection strategy.

To understand deeply the network topology we must consider not only the DNS but also the forwarding devices (e.g., routers, switches or firewalls) and the network flow. An overview of a typical network can be seen in Figure 3.1. Figure 3.1 has the components that we described in Section 2.3 for Figure 2.2. In addition to the described components, we have included a centralized point where all the traffic of the network passes through. Forwarding devices are centralized and forward the traffic of a network. Additionally, they can perform basic filtering or mirror traffic to another machine that can process the packets in detail. For example, intrusion detection systems (IDS) can perform deep packet inspection to detect hostile actions that match specific rules. In Figure 3.1 we have included a flow collector that processes network traffic in order to extract network flows. There exist detection methods that are based on network flows [11, 12]. Nevertheless, in this report, we consider any technology beyond the DNS as supplementary to DNS-based detection.

The network topology is introduced because the centralized nodes can record every established connection. The reason for checking this is that a DNS query does not necessarily convert into an actual flow. In other words, a device may not always visit a queried domain name. Methods that combine DNS traffic with flows in order to verify this connection have been proposed [13]. In this report, we consider the network flows as a valid mean to increase the confidence of a DNS alert.

3.3.1 Types of monitored information

This report is based on two types of information that can be monitored: network flows and DNS data. Network flows can be observed so as to reveal suspicious connections. DNS information can be either generated or observed. The former is called active probing and the DNS data are extracted through multiple queries. The latter is called passive DNS (pDNS) and we discuss below the reasons that it is preferred.

3.3.1.1 Network flow

A network flow (also known as traffic flow, packet flow or just *flow*) is defined [16] as "a set of packets passing an observation point in the network during a certain time interval, such that all

packets belonging to a particular flow have a set of common properties". These common properties are meta-data, such as who talked with whom, for how long and with which protocol. A flow probe is responsible to process the traffic of a network and output flow traces.

Network flows satisfy two basic requirements for monitoring large and high-speed networks [13]. These requirements are the near real-time data analysis and the low storage space. Moreover, flow export protocols are widely integrated into high-end packet forwarding devices. Steinberger et al. [14] performed a survey and stated that 70% of the examined forwarding devices supported a flow export protocol. Additionally, Hofstede et al. [11] state that flow export protocols are well understood and are widely used for security analysis, capacity planning, accounting, and profiling, among others. They also show that network flow can significantly reduce data, in the order of 1/2000 of the original volume.

3.3.1.2 Active probing and pDNS

The DNS is an ubiquitous Internet protocol, which can carry both benign and malicious information. Therefore, it is a valuable target for security analysis. Monitoring and analyzing DNS traffic has been proposed as one of the most promising options to detect malicious activities [15]. There are two ways to take advantage of the DNS in order to extract interesting information. First, active probing that generates DNS queries regarding suspicious domain names. Then, it resolves these names to a number of name servers and analyzes the responses. The second way is passive DNS (pDNS) that records DNS traffic generated by others.

Passive monitoring has many advantages over active monitoring, such as comprehensiveness and easier implementation [17]. Moreover, it allows the observation of realistic behavior that is unpredicted and cannot be generated. It increases the network load on neither the name server nor the network. Finally, and more importantly, the observation remains stealthy because of its non-intrusiveness characteristics [15]. On the contrary, active DNS probing methods can be detected by the attacker. In this report, we examine methods that analyze pDNS traffic to detect malicious activities.

3.3.2 Monitoring positions

In general, monitoring network traffic is carried out for performance analysis, troubleshooting, researching and detection of security risks. Monitoring transmitted data to detect malicious activity can happen at several locations of a network. The different monitoring places lead to different kind of captured data with more or less information. In this report, we consider five different locations.

The first sensor (1) in Figure 3.1 can perform regular packet capture. It can be a forwarding device (e.g., central router) that can filter all the traffic of a network. This spot represents the elementary monitoring of raw packet traces that involves every transmitted packet and its payload. It is not necessary that the analysis is performed on the forwarding device. For instance, an IDS system might get mirrored traffic from the router, while the latter just forwards the packets without examining them.

The second observation point (2) analyzes network flows, gathered by a flow collector. Generating flows requires extra resources. However, a flow is aggregated information of each connection that may have transmitted several packets.

The third observation point (3) in Figure 3.1 can monitor the communication between the clients and the resolver. This location reveals all the DNS traffic generated by the users. The third link contains the greatest amount of information, regarding the DNS traffic of a single network. Although the resolution process is not visible (i.e. recursive queries), this information is public and can be reconstructed. Interestingly, even the total DNS traffic is only a small fraction of the total network traffic passing through the centralized forwarding device.

Another observation point can monitor traffic above the resolver, which is represented by the fourth sensor (4) in Figure 3.1. This link contains the communication between the resolver and remote authoritative name servers. The identity of the clients is masked because the resolver is

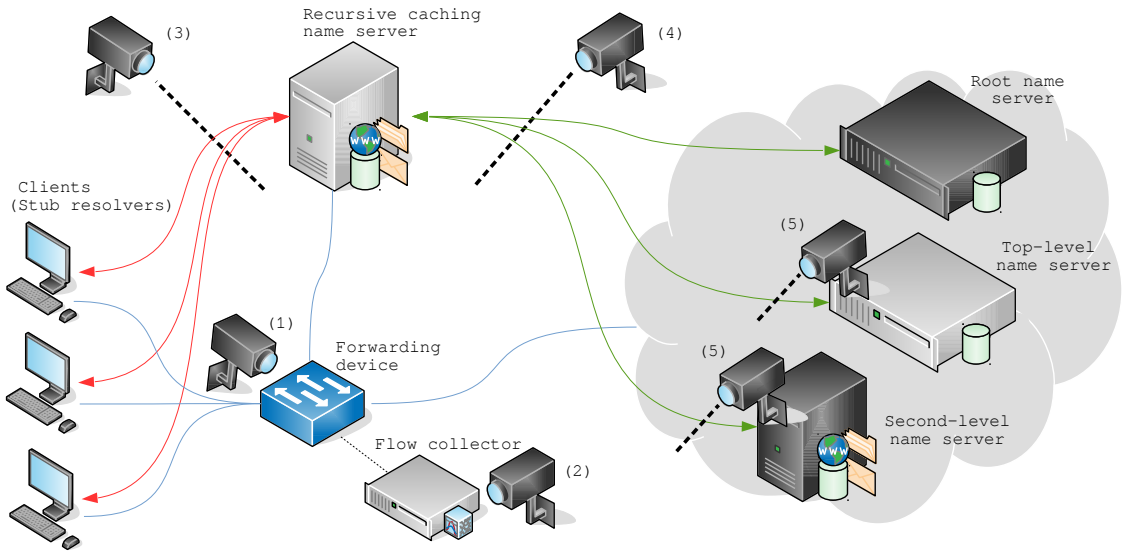


Figure 3.1: Monitoring places

responsible for obtaining the requested information. The resolver acts as a traffic mixer [18]. As a result, this is the only observation point within the network that does not allow detection of individual infected machines. Another property is that the resolver's cache replies for domain names whose TTL has not expired. Therefore, although users may constantly request a popular domain name, the domain name is visible on this link once in a while.

Outside of the local network, monitoring can happen at the upper DNS hierarchy. These are the fifth observation points (5) in Figure 3.1. Sensors are places at one or more authoritative name servers. These servers are delegated authorities for a complete zone (see Section 2.2.2). Therefore, monitoring at this level offers *global visibility*. That is, the visibility on all DNS messages related to the delegated zone. Positioning the monitoring at the upper DNS hierarchy reveals every resolver around the world that requested information for that zone.

3.3.3 Advantages and effectiveness of DNS monitoring

The main advantage of monitoring DNS traffic is that it is scalable. DNS traffic is only a small percentage of the overall network traffic [19]. Thus, it is feasible for a detection mechanism to process huge amount of data in real-time. On the other hand, monitoring the whole traffic of a network could be impractical in a large-scale network [24].

In addition, due to the nature of the DNS, a malicious activity can be detected before it actually happens. For instance, a bot may know the domain name of the C&C server. Then, in order to connect, it has to resolve the name to an IP address. At this point the activity could be detected, before the bot being able to connect to the C&C server. However, taking actions in such short notice is challenging. Such real-time processing could be used to block the malicious connection. In any case, it gives a slight time advantage.

Another advantage of monitoring DNS traffic is the robustness against encrypted or obfuscated communication [31]. Attackers have been using these techniques for years to evade traditional detection systems. Nevertheless, this does not apply to the DNS because it is not encrypted.

Furthermore, there are methods that can be applied only on DNS traffic. These are specifically designed for and depend on specific DNS behavior. For instance, DGA-based botnets query a lot of non-existing domain names. Monitoring such DNS traffic could quickly detect bot-infected hosts [19], before they actually act.

Remotely communicating malware have evolved during the last years. Similarly, dynamic detection methods that monitor DNS traffic have been proposed. A drawback of any DNS-based

detection system is that it misses other types of communications. For example, Choi et al. [23] argue that a malware evades detection if it uses the DNS only in initialization and never again. Moreover, observing a DNS query does not necessarily mean that the connection was established. Although that the two functions are related, it is always possible that a requested domain names was never accessed by the initiator.

Regardless the limitations, the use of the DNS has become essential for the operation of malicious network infrastructures. As shown by Lee and Lee [31], analysis of DNS traffic has become the most effective method to restrain them.

3.4 Detection

The detection of malicious activities goes beyond capturing data. The places where a sensor can be deployed in order to monitor traffic were discussed previously. In addition, the captured data can be examined in order to detect malicious activities, which, eventually, can lead to their mitigation. Detection of malicious activities can involve techniques such as blacklisting or dynamic detection methods.

3.4.1 Blacklist

It is very common that a node running a malicious software repeats the action regularly. Therefore, network administrators want to isolate these nodes. Blacklists have been the most reliable solution for a basic access control mechanism. The main idea is that whenever an element is involved in a recorded malicious activity then it is inserted into the blacklist as a hostile entity. Consequently, any repetition of the recorded action can be blocked immediately, before having any effect.

The most popular area that blacklists are deployed is spamming. They are used as a basic filter to block unwanted e-mails, part of a spamming or phishing campaign. Blacklists are not limited in this field and can be found in a diversity of environments. For instance, chatting applications have been using them to block unwanted users from accessing a chat room. A disadvantage of using blacklists is that an entity is blocked as long as it preserves the same identifier. The moment that the identifier is changed it is not blocked any longer.

3.4.1.1 Common type

There can be different types of blacklists with various entries. In this report, we consider IP-based and DNS-based blacklists, where malicious IP addresses and malicious domain names are listed, respectively.

Although there is a clear correlation between IP addresses and domain names, it is risky to modify a blacklist arbitrarily. Blacklists should be applied unaltered because every unverified addition has a significant risk of being a false positive. The safest option is to deploy a high-quality blacklist maintained by a reliable organization.

DNS-based blacklists tend to have less false positives than an IP-based blacklist. We can argue that nobody would choose to use a domain with a questionable history. On the other hand, users cannot freely choose their IP address. It is possible that an IP address related to a very heavy attack is released. Then, this blacklisted IP could be assigned to a benign user. This is one of the reasons that every IP-based blacklist should be updated often. There are blacklist that are updated every 10 or 15 minutes[6,7].

3.4.1.2 Extended type

In addition to the IP-based and DNS-based, there can be two expanded kinds of blacklist. One is a blacklist as an on-line service that gets IP addresses and performs reverse DNS lookups (see

[6] http://www.spamhaus.org/faq/section/Spamhaus%20SBL#9, retrieved July 2015.
[7] http://www.spamhaus.org/faq/section/Spamhaus%20XBL#98, retrieved July 2015.

Section 2.5.2). Then, these retrieved domain names are matched against a DNS-based blacklist. Another option can be to fully resolve a DNS-based blacklist and use the answers to create an IP-based blacklist. Both options are not as accurate as applying directly the original blacklist. Especially the latter can be easily polluted. An attacker that performs an attack can subsequently change the DNS record. For instance, he could assign an IP address of a legitimate service that he does not own. Then, resolving a DNS-based blacklist would include this benign service. This example can illustrate the risk of arbitrary modification of a blacklist.

3.4.1.3 Indication

There is a variety of blacklists that indicate different malicious activities. For instance a blacklists may contain nodes that regularly send spam e-mails and, therefore, the acceptance of electronic mail is not recommended. Another kind of blacklist (or blocklist) is advisory "drop all traffic" list, where the nodes are part of identified cyber-crime operations. The latter is very strict and only a subset of the previous kind.

In this report, we are especially interested in blacklists that indicate infected users. Any node communicating with a blacklisted element should be considered as infected. For instance, communicating with a C&C server is a strong indication that the machine is part of a botnet. On the contrary, visiting a blacklisted domain name that hosts a phishing website, does not mean that the node is infected. In the last case, a user could be a victim of scamming and his identity could be stolen. However, the machine is not definitely infected and the user may not have disclosed any information at all. Consequently, only specific blacklists, such as a list of C&C servers, can indicate infected nodes with high confidence.

3.4.2 Whitelist

A whitelist is a list of entities that are being provided a specific access. Sometimes a whitelist is complete, in a way that anything absent is blocked. In this report, we consider whitelists of legitimate Internet locations, which cannot be complete.

The most interesting type of whitelist contains legitimate domain names. The identification of benign domain names is usually based on the popularity. Thus, a whitelist is normally a list of a number of the most visited domain names.

3.4.3 Dynamic detection

Although blacklists have been an important weapon for fighting cyber threats, they are not satisfactory. There are two obvious limitations. First, the malicious activity will succeed at least once before it can be detected. Second, the human resources that have to investigate this activity are time-consuming and costly. The fact that a malware can operate for a long period of time before it can be blocked makes the dynamic detection imperative. Disposable domain names make the situation even worse. Short-lived domain names are utilized to evasively move the C&C infrastructure [27]. In many cases, by the time the manual investigation is completed the domain is already abandoned. Dynamic detection could be used to automatically identify new malicious domain names while they are still active.

3.5 Discussion

Dynamic detection of malicious activity is important in order to automatically monitor intruders in networks. However, there are a number of challenges that should be addressed to enable this operation. In the remainder of this section we discuss these challenges.

3.5.1 Limitations of dynamic detection

We distinguish three important properties of every dynamic detection system. If these properties are not fulfilled, the system is limited. First, it must be able to run in real-time. A real-time system can process data virtually immediately after they appear. This mean that it will not be too long before threats are detected. Real-time systems [15, 25] are important in order to detect threats while they are still active. It is particularly important when suppressing the detected threats with countermeasures.

The second critical property is the ability to run automatically. The system must require as little human interference as possible. Manual investigation is a bottleneck for an autonomous system, which eliminates the advantages of dynamic detection.

Third, the false positive rate of an autonomous system should be low. A false positive is a false alert regarding a user's activity. Although sophisticated algorithms exist, they always produce a percentage of false alerts. Eventually, it may lead to inconvenience if the user is notified or blocked. False positives are a pitfall because the system loses its reliability. We consider any false positive rate above 1% as inadequate. Assuming that for a large user population the system generates 1000s of alerts, this would affect 10s of users misleadingly.

3.5.2 Privacy issues

In accordance with the EU Data Protection Directive[8], the fundamental rights and freedoms of natural persons should be protected, and in particular their right to privacy. Among the given concepts of privacy [20] we consider the following: control over personal information, freedom from surveillance and protection of one's reputation.

Monitoring any kind of traffic generated by users may remove, challenge or lessen privacy. In general, either capturing data that may contain personally identifiable information or identifying a single user's activities could be a violation of privacy. Furthermore, depending on the level of intrusion (infringe on the principle of data minimisation), it could be incompatible with EU data protection law[9]. Next, we discuss the privacy implications regarding the monitoring positions, described in Section 3.3.2. A summary of the effects on users' privacy can be seen in Table 3.1.

position	privacy
1	violated
2	violated
3	violated
4	preserved
5	preserved

Table 3.1: Effect on users' privacy when monitoring at each position in Figure 3.1.

To begin with, monitoring the forwarding device, position (1) in Figure 3.1, definitely violates privacy and data protection laws. Capturing directly the packets of the network traffic allows access to the data payload, which may contain personal information[9]. The fact that the payload is observable means that monitoring can identify not only the visited address but also what was typed or viewed.

Network flow exports, that can be seen at the second observation point (2), traditionally inspect only the header of a packet which is less privacy-sensitive than inspecting the entire packet. However, analyzing flow traces can still identify individuals and track individual activity [21]. Therefore, it violates the privacy of the users.

The remaining three positions observe DNS traffic, which may violate the privacy of users depending on the exact location of the sensor. Formally, the only personally identifiable information that queries and replies contain is a client's source IP address. Any personal information is not

[8] http://ec.europa.eu/justice/data-protection/, retrieved October 2015.
[9] https://ec.europa.eu/digital-agenda/en/net-neutrality-challenges, retrieved July 2015.

directly revealed, apart from accidental disclosure (e.g., typing a user's name instead of a domain name). However, queries could conceivable reveal the end-user's activities, which would be considered as personally identifiable information under some international definitions [17]. Whenever DNS traffic can be linked back to the initiator, it is considered as "sensitive information" (i.e. personal information).

Specifically, monitoring between the clients and the resolver, which is the third observation point (3) in Figure 3.1, reveals the source IP address of each query. This means that every query can be linked back to an end-user. Thus, this link is highly privacy-sensitive and should be treated with respect.

A resolver receives queries from users and initiates the recursive mode (see Section 2.4), without revealing who originated the query in the first place. Consequently, the fourth position (4), which is located between the resolver and the name servers, has the significant advantage that the IP addresses of the individuals are masked. As long as there are not just a handful of users, their privacy is preserved. Spring et al. [17] demonstrate that a larger number of users increases the uncertainty in identifying a user session. Although confidentiality restrictions may apply, the fourth monitoring position (4) does not violate the privacy of the users. Interestingly, Perdisci et al. [18] state that systems that monitor DNS traffic at this position are easier to be adopted, due to being respectful of privacy. Finally, for the fifth observation point (5) applies the same privacy preserving property as for the fourth.

3.5.3 Summary

To summarize, the following are the challenges that we distinguished regarding a dynamic detection system.

real-time: A system should be able to process all the traffic in real-time. A real-time system processes data virtually immediately after they appear.

autonomous: A system should operate without manual intervention.

low false positive rate: The rate of false alert should be lower than 1%. In case that the threshold is exceeded the system is considered to be ineffective.

respectful of privacy: A system should respect the privacy of the users. In other words, it should be able to neither read a packet's payload nor link the captured data back to an end-user.

Chapter 4

Literature review

There is a large body of work related to the detection of malicious activity by analyzing DNS traffic. The methods vary on several fundamental attributes, which makes every proposed scheme quite unique. In this chapter, we summarize and categorize the related literature. Moreover, we characterize a set of criteria for each method.

We have picked methods that excelled and set the ground truth for detection using DNS traffic. A visual and comprehensive representation of the characterization is given in two tables. Firstly, we present the fundamental properties of each method (Table 4.1). Then, we discuss the criteria that are summarized in Table 4.2. Finally, we classify the examined literature according to the detection approach used by researchers.

4.1 Properties

For each method we determine a set of properties, summarized in Table 4.1. The table provides an overall picture of the proposed methods. Moreover, it intends to introduce to the reader their fundamental differences. The determination of such simple characterizations is challenging, especially for the first three abstract properties. It symbolizes a bold attempt to create a generic view of the prominent existing literature. In total, Table 4.1 includes six properties. The *system* property is a very simplistic characterization of an overall method (e.g., reputation system). Normally, we reproduce the authors' choice, but in case that they have not named it, we use an intuitive name.

The *approach* property attempts to identify what the authors have chosen as their own criteria to perform detection. In other words, it is mainly an indication of what a method examines in order to achieve its goals. For instance, a method may examine features of domain names (e.g., the name's lexicology or TTL values) or patents of specific behavior (e.g., group activity) that could indicate maliciousness.

The *methodology* represents the families of the applied algorithms. For instance, if a method uses the "X-means unsupervised machine learning" algorithm to correlate domain names, we note machine learning in the methodology. Methods usually classify actions, which is an ambiguous term. Therefore, mentioning "classification" in the methodology field is only happening for methods with a rigorous classifier module.

The *detection* property is the detection target of the method. That is, the type of malicious activity that a method attempts to discover. There are three options: generic, botnet and domain flux detection. The detection target can give an insight of how the system performs and how it affects other properties. For instance, a correlation between detection target and approach could been noticed.

The availability (*available*) of the actual implementations of a method is listed for anyone interested in future work. The available attribute denotes whether the implementation is available for download and execution. There are three different notations: "no", "yes" and the conditionally available "R". The verdict "no" means that -to the best of our knowledge- the implementation is

not publicly available. "yes" means that the implementation is a fully developed free and open-source software (FOSS). "R" means that a closed-source version is available by request. We should note that we have not requested every method. Therefore, it does not necessarily mean that a unavailable implementations is not available by request.

Finally, *external source* states whether a method leverages historical information. Such information can be an external blacklist (BL) or whitelist (WL). Generating a blacklist beforehand (e.g., by executing malware samples in a control environment) is considered an external source as well. The motivation for choosing this attribute is that external source can influence the detection rate. For instance, a method could mark as malicious a domain name that has similar features with a blacklisted one. Such a method relies heavily on the blacklist's quality. For instance, using a private blacklist, instead of a publicly available, could increase the accuracy of a method [27].

Method	System	Approach	Methodology	Detection	Available	External source
Choi et al. [23] (pre BotGAD)	anomaly-based	group activity	similarity	botnet	no	WL
Choi et al. [24] (first BotGAD)	metric model	group activity	similarity, stochastic	botnet	no	WL
Choi et al. [25] (BotGAD)	machine learning	group activity	correction, similarity, clustering, hypothesis	botnet	R	WL, BL
Sato et al. [26]	scoring	co-occurrence	classification, similarity	generic	no	BL
Antonakakis et al. [27] (Notos)	reputation	statistical features	classification, clustering, characterization	generic	no	WL, BL
Antonakakis et al. [28] (Kopis)	reputation	statistical features	machine learning, classification	generic	no	WL, BL
Bilge et al. [29] (EXPOSURE)	reputation	features	machine learning, decision tree	generic	no	WL, BL
Marchal et al. [30] (DNSSM)	data mining	features	machine learning, clustering	tunneling, fast flux	yes	-
Lee et al. [31] (GMAD)	graph structure	sequential correlation	correlation, graph-construction, filtering, clustering	generic	no	BL
Antonakakis et al. [32] (Pleiades)	DGA-bot	NXDomain	clustering, correlation, classification	domain flux	no	WL, BL*

Table 4.1: Properties of each system.

4.2 Criteria

In addition to the properties, we introduce a set of criteria that highlight the details of applying each method. We discuss the reasons for choosing the criteria, the limitations and challenges. The final verdicts are a result of our understanding, conclusions derived from the papers and our overall experience.

Table 4.2 contains our judgments for six specific criteria. First, we examine whether a method can dynamically detect new malicious domain names. Second, the method's capability of dynamically detecting infected users. The first two criteria determine the overall detection target of a

method. It is worth noting the difference from the *detection* property which detects the type. For instance, a method's goal could be to detect new domain names of malicious domain flux activity. Alternatively, it could aim to detect hosts infected by a botnet.

Third, we examine the privacy implication of applying a method in real-life. Satisfying this criterion depends on whether the privacy of users is respected. Normally, monitoring users without taking measures to ensure privacy is considered a violation. The significance of protecting privacy is discussed in Section 3.5.2.

Fourth, the autonomous criterion is strongly related with the first two criteria. It is satisfied if the dynamic detection of either new domains or infected users is completely automated. A system is autonomous if it does not require supervision or manual intervention at any step apart from training. Formally, the output of an autonomous system must be a list with either new malicious domains or infected hosts. If the output is a group of domains that require manual inspection, the system is not considered autonomous even though it may run automatically.

Fifth, we report the performance ability of a method to run in real-time. A real-time system processes data virtually immediately after they appear.

Sixth, we record the maximum false positive (FP) rate of a method. This is a reliable indication of how well the system performs. For example, marking a legitimate domain name as malicious is considered as false positive. We have selected the FP rate, rather than for instance false negatives, because it could seriously damage a system's reliability. False positives cause complains from legitimate entities that consider themselves as treated unfairly.

At this point we must emphasize the impossibility of independently verifying the last three criteria. It is extremely challenging to justify if a method is autonomous, able to run in real-time and its max FP rate by theory. Unfortunately, most of the implementation are not available. Therefore, the reported values are approximate estimations based on the description and claimed results by the authors. Even this way is sometimes difficult because not all researchers consider the same criteria. For instance, some tend to report the maximum FP rate, while others report the average rate. Therefore, the comparison is slightly unbalanced. The ideal scenario for the FP rate would be to run each method given the same dataset, which was not possible.

4.3 Classification

Below, we classify existing proposals in the area of DNS-based detection of malicious activity. The main classification is based on the detection approach. Moreover, we summarize the methods and justify the verdicts for the criteria set out in Section 4.2.

4.3.1 Group activity

In 2007, the time that researching botnet mitigation was in an early stage, Choi et al. [23] identified group activity as an inherent property of botnets. As we have defined in Section 3.1.2, a botnet consist of a group of bots. The authors demonstrated how monitoring group activities can be used as an anomaly-based mechanism to dynamically detect botnets.

The basic definition of group activity states that it has fixed size of bots which have intermittent activities. To elaborate, they assume that a group has fixed size because only botnet members query the malicious domain name. In addition, they state that these queries are temporary and simultaneously. The last assumption is that botnets usually use DynDNS (see Section 2.5.1) to register malicious domain names. Interestingly, the method leverages historical information. They use a whitelist to exclude queries for legitimate domain names. Although, the method is obsoleted, it is noteworthy due to the original introduction of group activity.

One interesting limitation is the focus on DynDNS traffic. It has been noticed that most professional criminal botnets have moved away from DynDNS services[1]. The reason is the provider's

[1]Operation Aurora - The Command Structure,
https://www.damballa.com/downloads/r_pubs/Aurora_Botnet_Command_Structure.pdf, retrieved July 2015.

Method	New domain detection	Infection detection	Respectful of privacy	Autonomous	Real-time	Max FP(%)
Choi et al. [23] (pre BotGAD)	✓	?	✗	✓	✗	?
Choi et al. [24] (first BotGAD)	✓	✓	✗	✓	✓	?
Choi et al. [25] (BotGAD)	✓	✓	✗	✓	✓	0.31
Sato et al. [26]	✓	✓	✗	?	?	4
Antonakakis et al. [27] (Notos)	✓	✗	?	✓	?	0.38
Antonakakis et al. [28] (Kopis)	✓	✗	✓	✓	✗	0.5
Bilge et al. [29] (EXPOSURE)	✓	?	✓	✓	✓	1
Marchal et al. [30] (DNSSM)	✗	✗	✓	✗	✗	-
Lee et al. [31] (GMAD)	✓	✓	✗	✗	?	0.5
Antonakakis et al. [32] (Pleiades)	✓	✓	✗ *	✗	?	1

Table 4.2: Judgments about the criteria. '✓' means that the criterion is satisfied, '✗' that it is unsatisfied and '?' that it is uncertain.

aggressive stance against abuse. Although there are still incidents[2], this focus excludes, for example, botnets with rogue name servers.

Two years later, Choi et al. [24] presented a primitive version of BotGAD, which stands for Botnet Group Activity Detector. Similarly to the prior work, it is also based on group activities. They claim that group activity had remained a common property of botnets at that time. The method does not depend on the content of the transferred packets nor the signature, but only on behavior properties. It follows the same concept as before and it utilizes a whitelist. However, the new method can run in real-time and uses a different generic metric model. The definition of group activity has slightly more flexible size, because trivial changes in groups may occur. In addition to the intermittent property, they recognize that it has intensive occurrence.

The new proposed scheme has two extra steps: group classifier and botnet reporter. The first keeps separate storage for each group, that makes it easier to process. The latter takes the final decision whether the group is malicious, suspicious or false positive, according to the combination of the similarity, periodicity and intensity of the group's activities. Moreover, they have used stochastic methods to optimize the parameters of the system, such as the time window and the group size threshold. Finally, they described six well-defined steps to manually verify the results.

In 2012, Choi et al. [25], acknowledged three limitations of the previous systems. These are: (i) the flexibility of the group size is not sufficient enough, (ii) a slight change in a parameter could have huge impact in the results and (iii) DGA-based botnets are not detected. Consequently, BotGAD was extended with error correction, cluster analysis and hypothesis test. The most advanced and complicated enhancement is the cluster analysis method that uses the "X-means"

[2]Microsoft takes on global cybercrime epidemic in tenth malware disruption, https://blogs.microsoft.com/blog/2014/06/30/microsoft-takes-on-global-cybercrime-epidemic-in-tenth-malware-disruption/, retrieved July 2015.

unsupervised machine learning algorithm to correlate domain names. Domain correlation is a significant improvement that allows detection of botnets that use evasion techniques, such as domain flux. Specifically, it correlates domain names according to a set of features related to DNS lexicology, query and answer information. As a result, all domain names of a cluster are labeled together depending on the output of the hypothesis test. Eventually, BotGAD achieves less than 0.4% false positive rate with good performance.

To conclude, Choi et al. [23, 24, 25] keep sub-optimizing the method which has achieved good results. However, the modules have been increased that could lead to needlessly complexity. Scrutinizing traffic to identify group activity has been proven a useful tool to dynamically detect new malicious domains. Nonetheless, it remains a tough challenge to successfully apply it in real-life. One fundamental limitation is that investigating group activity can only detect infected groups but not single nodes. Furthermore, depending exclusively on the behavior property makes the system vulnerable to simple evasion techniques that bypass specific rules. Notably, in addition to a whitelist, the authors examine the presence of blacklisted SLD names to influence the results of the latest BotGAD.

Finally, the only criterion in Table 4.2 that has not been convert in the discussion so far is privacy. None of the methods based on group activity is respectful of privacy. This is true due to the fact that these methods examine DNS traffic generated directly from users without taking privacy into consideration.

4.3.2 Co-occurrence

Sato et al. [26] propose a scoring system to expand blacklists by monitoring co-occurrence relation. The authors examine pairs of domain names that are frequently queried by multiple infected hosts. The assumption is that if one is black (where they mean malicious) the other one is black as well.

The method consists of three steps. First, the identification of infected hosts using a blacklist. Second, the expansion of the blacklist. This step distinguishes suspicious domain names requested by infected hosts. Then, checks if a suspicious domain name has hight co-occurrences relation with a blacklisted name. Third, the identification of unknown infected hosts that request a domain name of the expanded blacklist.

The degree of co-occurrence relation between two domain names is measured with a similarity coefficient[3]. The similarity is determined by: the total number of hosts that requested both domain names, divided by the total number of hosts that requested at least one. For each suspicious domain name the similarity coefficient between it and every malicious domain is calculated. The aggregation of these similarity coefficients is the final score. If the score is above a given threshold, the domain name is considered as malicious. The relation of a malicious and a suspicious domain name strongly relies on defined thresholds.

The authors have proposed an improvement to reduce the negative score of popular domain names. The assumption is that if a domain name is popular it will be requested multiple times by both infected and non-infected hosts. Therefore, they define the weight ratio -for each domain name- of the total number of infected hosts over the total number of all the hosts that requested it.

The authors also examine a variation where heavily infected users influence less the score. In general, they examined four different variations. However, the system fails to achieve less than 4% false positive rate. This could be considered as a drawback of not taking into account the positive reputation (e.g., domain's relation with a legitimate service). Moreover, the authors recognize that it could easily be polluted if bots were querying legitimate domain names.

Finally, we cannot conclude about the performance or the implementation's autonomy because they are not clearly documented.

[3]The authors do not use the term similarity. However, they refer to Jaccard index which is also known as Jaccard similarity coefficient. We prefer the second term to ensure consistency in this report.

4.3.3 DNS features

A different class of methods identifies and measures DNS features to detect malicious activities. This category is mainly composed by reputation systems, that compute reputation scores for sets of domain names. In addition, we examine a data mining system that is based on DNS features.

A reputation system considers various features that are fed as input to the reputation engine along with the actual domain names. Often, the process intensively leverage historical information, such as blacklists or whitelists, to evaluate unknown domain names. This external source of information is used to model profiles of legitimate and malicious activity. Then, the assignment of proper reputation to newly emerging domain names is usually based on the similarity to these modeled profiles. Normally, the assigned reputation score is in range, instead of being either good or bad.

Additionally, next to the reputation systems we describe a data mining system. This system examines features of domain names too. However, it does not consider historical information. As a result, the suspicious features of domain names are not flexible.

4.3.3.1 Reputation systems

Notos [27] was the first comprehensive dynamic reputation system for evaluating unknown domain names. It was introduced in 2010. Notos has extended preexisting spam detection systems by exploiting network-based features. At that time, the handful of reputation systems were using -and could evaluate- only IP addresses. The authors propose a more general system that can identify a variety of previously unknown malicious domain names, not limited to spam.

The key idea of Notos is the creation of a benign and a malicious profile, according to a set of domain features. Then, it matches unknown domain names to one of the two profiles. Antonakakis et al. [27] assume that hostile registrations of domains have unique characteristics, distinguishable from benign ones. The reason behind this assumption is that fraudulent activities act stealthily in order to avoid triggering alerts. Therefore, they are not ordinary.

Building the model of the benign profile requires a large amount of data that was gathered from multiple resolvers across the world. These datasets contain, as they call it, historical DNS information. Additionally, for the model of the malicious profile the authors gathered data from spam-traps, honeynets and malware analysis services. A common tactic is to run malware samples in a controlled environment in order to capture the other end of the communication. These logs and publicly available blacklists constitute the malicious knowledge base.

Notos extracts features from historical information to categorize the knowledge base of legitimate or malicious domains. Then, it is fed to the reputation engine. The output of Notos is a reputation score for each examined domain name. Specifically, if there is evidence of relation with a known source of malicious activity, the reputation of the unknown domain name is decreased. Likewise, if there is evidence that connects it with a professionally run service, the score is increased.

Notos identifies three categories of statistical features that should be different for each profile. Firstly, the network-based features are used to detect any use of fast-flux and allow through domain names that have a noticeable static profile. In total they distinguish eighteen statistical features. For example, Notos examines the total number of IP addresses historically assigned to the domain name as well as the diversity of the geographical locations. The network-based features are extracted from a set of IP addresses. This set includes IP addresses associated with the unknown domain or any of its subdomain names.

Secondly, the zone-based features extract statistical information from all the domain names historically associated with the unknown domain. This association is determined based on IP addresses. This is, any domain name that has used the same IP address as the domain name under inspection. The knowledge base is used to extract statistical features of two groups of characteristics: string features of the actual domain names and TLD features related to the number of distinct top-level domain labels among the set. The assumption, in this case, is that domain names with common IP addresses will have strong similarities. For instance, “www.example.com”

could have the same IP address as "ftp.example.com" because the same server hosts both a web server and an FTP server. On the other hand, malicious names pointing to the same IP address will not have such similarities. This is because attackers want their domain names to look random. In total, the zone-based features are seventeen. For example three of those features are the average length of their names, the total number of different TLD values and the deviation of 2-grams distribution. For more details on this set of features see Section 3.2.2 of [27].

Thirdly, evidence-based features are simple alerts of how many of the IP addresses exist in the malicious knowledge base. Their noteworthy assumption is that malicious domain names often point to the same IP addresses, as a consequence of the more expensive renewal of domain names than IP addresses. However, a domain name queried by a malware is not instantly assigned with low reputation. Evidence-based features are not an ultimate proof without considering the other two categories of features.

The described features are investigated by the reputation engine of Notos in order to evaluate unknown domain names. The engine has to complete an off-line training phase before being able to operate in on-line mode. This preprocessing creates (i) network profiles models, using the network-based features, for classes with well-known network behaviors (e.g., popular domain zones or DynDNS). Moreover, it uses, in addition to network-based, zone-based features to execute the domain clustering module. (ii) The clusters are marked as high or low risk depending on their percentage of malicious entries. Finally, (iii) it calculates the evidence-based feature for every domain in the knowledge base and stores the output. The three output vectors of the training phase can be occasionally reconstructed off-line.

During the on-line mode the statistical features of the unknown domain are calculated and fed to the reputation engine. The engine determines whether it belongs to a specific well-known class (i), which is the risk of the closest cluster (ii) and, finally, if there is any direct evidence of relation with a blacklisted IP address (iii). The output is the final reputation score.

Regarding the privacy criterion, Notos does not utilize the source information of users that initiate DNS queries. Therefore the system -as it is- cannot detect infected users and may respect privacy. However, the data that they obtain could contain private information and they do not explicitly state of considering privacy. The final verdict of this criterion is uncertainty.

There are not sufficient statistics of the implementation's performance and the ability to run real-time is difficult to determine. Although there is preprocessing (off-line) phase, it is unclear whether the on-line mode can run in real-time or not, hence the uncertainty in the corresponding cell in Table 4.1.

Antonakakis et al. [28] proposed a more abstract reputation system, called Kopis, in 2011. At that time the most sophisticated approaches were the aforementioned Notos and a primitive version of EXPOSURE, which we will discuss further down. The authors state that the existing solutions are limited due to their local deployment. They, instead, propose a system that uses data collected at the upper DNS hierarchy (see Section 3.3.2).

Monitoring at the upper DNS hierarchy collects data from multiple resolvers. Therefore, it has global visibility. On the other hand, local visibility involves monitoring a few resolvers. Antonakakis et al. [28] define that global visibility should be a property of every dynamic detection -of malicious domains- system. The reason is that an early warning will detect malware at another network before they actually penetrate into the local network. From an administrator point of view this could have significant importance if the administrated network is a secondary target.

They emphasize that to achieve a "meaningful level of visibility" it is not sufficient to obtain data below the resolver (see Section 2.3.2). It is extremely challenging to combine various sources in order to increase the partial visibility due to the operational cost, communication issues and, undoubtedly, privacy concerns. Therefore, monitoring anonymized data at the upper level can overcome these issues. However, this convenience comes with two limitations. First, being unable to detect infected clients. Second, miss any request for cached information.

The approach of Kopis is similar to Notos and involves the extraction and comparison of statistical features of domain names. Similarly, a knowledge base and public blacklists are used to train the system, before switching on the operation mode. However, monitoring at the upper DNS hierarchy, naturally, requires the extraction of alternative features. For instance, the diverse

location of the resolvers that resolved a specific domain and their cardinality of users. Additionally, they examine IP reputation information for each domain. They, also, claim that the system can succeed in detecting new malicious domains even if this information is missing.

Initially, the training phase utilizes a knowledge base of well-known legitimate and malicious domains to build a statistical classifier. This is done using supervised machine learning that may last several days. Eventually, during the operation mode, Kopis collects queries and replies for one day, which they consider as optimal, and computes several statistical features of every requested name. Then, the classifier evaluates the domain name and output the final verdict.

We should note that we have marked Kopis, in Table 4.2, as not being able to run in real-time. If we consider that a blacklist may change in less than one hour (see Section 3.4.1, 3.4.3) detecting new malicious domain every 24 hours is not considered as real-time.

Kopis is a neat system with an essential classifier, in contrast to Notos that involved three different components. However, selecting the right classifier and the best-suited machine learning algorithm is far from trivial. In a long-term experiment they used three different classifiers: random forest, random committee and k-nearest neighbors, which had been selected among much more options during a short-term experiment.

Bilge et al. [29] propose EXPOSURE, a successful detection method of new generic malicious domain names. To the best of our knowledge, it is the only method -of the examined literature- that was applied in real life. EXPOSURE was offered as an on-line[4] free service for a long period of time, over one and a half years.

The authors distinguish fifteen features of domain names that can be used to distinguish malicious domain names from benign ones. Six of them had been priorly used to detect fast flux networks, but the others form a novel approach. The features are grouped in four categories: time-based features, dns-answer-based features, TTL-based features, and domain-name-based features. These features can be extracted from anonymized DNS records.

Technically, EXPOSURE gets DNS traffic and extract the proposed features. Then, the system uses a blacklist as well as a whitelist to distinguish between known and unknown traffic. The labeled domain names are sent to the learning module, while the unlabeled ones are sent to the classifier. The learning module processes the historical information to produce two detection models. Specifically, after an optimal seven-day machine learning process it creates a benign and a malicious model. Eventually, the classifier, which is build as a decision tree algorithm, decides whether a domain names is malicious or benign. Thereafter, the training stage can be repeated daily. Notably, Bilge et al. [29] applied a genetic algorithm, a search heuristic, to refine the list of examined DNS features.

4.3.3.2 Data mining

Marchal et al. [30] propose a data mining approach to dynamically detect new malicious domains. The authors focus on the practical aspects of monitoring DNS traffic. They designed and implemented the DNSSM framework, which is not standalone, but can be used by an analyst or a software to detect previously unknown malicious domain names. The authors have distinguished a set of relative features and proposed a machine learning methodology.

Most of the related literature, that we discuss in this chapter, uses pDNS (see Section 3.3.1.2) as the source of detection. However, it is not explicitly documented how data are processed and the majority of the implementations are unavailable. Marchal et al. [30] actually implemented a passive DNS sensor that extracts DNS related traffic from captured traces. Next, it inserts every DNS answer to a relational database system (MySQL). The centralized storage allows adjustable display of the data for an inquisitive analyst. Each view can extract and display specific features. For instance, given a threshold, the domain flux view prints domain names which have at least this number of NS records. Interestingly, the authors published DNSSM with a functional user interface as a FOSS, which is the only FOSS that we came along during this report. DNSSM is not publicly available because nobody hosts it on-line at this moment. Nevertheless, we have

[4] http://exposure.iseclab.org, retrieved July 2015.

marked it as available, in Table 4.1, because the license allow redistribution under the terms of the GNU General Public License version 2.

DNSSM focuses on behavior property and does not require any external knowledge, such as blacklists. This means that it can only detect specific kinds of malicious activities. DNSSM targets two major exploited techniques. First, the tunneling of IP traffic over DNS traffic, which is a known abuse of camouflaging regular traffic to look like DNS traffic and remain undetected. The second target is to detect fast-flux techniques (see Section 3.2.1).

The relative features are related to the aforementioned detection targets. The simplest of the 10 features are: the mean TTL value and the subdomain count for a given domain. Additionally, they calculate an entropy-based index of IP address scattering through different subnets on the Internet, among other features. The authors claim that these attributes are adequate to distinguish DNS tunneling and fast flux activity from regular DNS traffic.

DNSSM detects group of domains that share common behaviors related to the proposed features. They propose the use of machine learning that can perform a classical unsupervised clustering algorithm to achieve essential mining on the extracted features. In their experiments they used the "weka" machine learning tool and the "k-means" algorithm for clustering.

The final output of the experiments was seven clusters of domains that were presented through manual investigation. Due to the fact that it does not characterize individual domains, we have marked it as not been autonomous as well as unable to detect new malicious domains. Although the clusters are created automatically, we have defined an autonomous system as a system that outputs new malicious domains without requiring manual intervention. The cluster could significantly reduce the search area for new malicious domains but marking all domain as malicious would be irrational. Furthermore, DNSSM monitors traffic in the upper DNS hierarchy and, therefore, cannot detect infected users and respects privacy.

4.3.4 Sequential correlation

Recently, it has been observed that sophisticated malware uses various evasion techniques to remain undetected, in real life [31]. Most of them have been discussed in the limitations of the current literature [25] but have not been very well confronted. For instance, sub-grouping is used to evade detection by monitoring group activities. Additionally, interleave queries of legitimate (or invalid) domains with malicious queries could corrupt clustering.

Lee et al. [31] have proposed a Graph based Malware Activity Detection (GMAD) mechanism, that constructs a graph expression of the overall DNS queries. The goal is to detect new malicious domains, infected clients and specific malicious activities. These activities involve blacklist checking and fake querying.

The approach examines the sequential correlation of domain names, that is the correlation between two domains that are queried after or before each other. For instance, using a web browser to visit "*www.example.com*" will automatically resolve "*www.iana.org*" as well. These DNS query patterns have high sequential dependency, due to pre-resolving (see Section 2.5.3) or redirections.

Sequential correlation is a spatial property caused by the query patterns of infected clients. The authors distinguish spatial properties from temporal properties, such as timing synchronicity of group activities. They claim that sequential correlation is much less influenced by the amount of queries, the infection rate and legitimate activities. As a result, the mechanism can detect even malicious domains with a low rate of queries.

Technically, the domain-nodes are connected with a correlation value, depending on the percentage of common query source IP addresses. Then, a clustering is performed with a simple threshold of the correlation value, the number of queries and the number of clients. Finally, if a cluster contains a blacklisted domain, it is marked as malicious.

It is worthy mentioning that this mechanism is robust against intentionally injection of legitimate domains by a malware. An attacker may inject a legitimate domain name (for instance, "*www.example.com*") between malicious ones. By this way a detection system may categorize it as malicious. However, the natural sequential correlation of legitimate domains prevails and these

names cluster separately. For instance, "*www.example.com*" and "*www.iana.org*" cluster together, although the malicious injections. Consequently, GMAD utilizes a blacklist but not a whitelist. The assumption is that a cluster has either malicious or legitimate domain names. If it is not malicious, it is legitimate.

Moreover, this mechanism can increase the chances of a botnet's successful take down, that requires the blocking of its malicious domains, altogether. In contrast to the preceding systems, GMAD is not limited to inconstant patterns or fixed features. The mechanism categorizes the botnet's domains in a single cluster and the only requirement is that one of them is blacklisted.

The authors compared GMAD with BotGAD, described in Section 4.3.1, using the same data set. They claim that GMAD detected twenty-eight times more malicious domains. Their interpretation of the superior performance is that GMAD can detects sparse and low-rate malware activities. BotGAD has been expanded to detect DGA-based botnets, in addition to group activity. Nevertheless, Lee et al. [31] indicate that BotGAD cannot detect multi-domain malware[5] that do not use a DGA or cause little lexical similarity.

Finally, the authors claim that GMAD achieves scalable detection in huge network environments. However, they also mention that "the mechanism is not yet fully automated and does not necessarily work in real time". We have marked GMAD as being neither autonomous nor able to run in real-time, due to the fact that any evidence to support the opposite is not provided.

4.3.5 NXDomains

The systems in this category target malicious domains of botnets that use the domain flux technique. Malicious use of domain flux is strongly aligned with the use of domain generation algorithms (DGAs), a fact which is described in Section 3.2.2. Researches have used non-existing domain signals (NXDomain), that are mentioned in Section 2.4, as a tool to detect such algorithmically generated domain names. This category mainly involves various methods to analyze clusters of NXDomain and dynamically flag those generated by a DGA-bot.

Pleiades [32] is a DGA-bot detection system, placed below the local resolver. Initially, it monitors and stores all the NXDomain replies within the network. Pleiades aims to automatically identify DGA domain names. To this end, it searches for relatively large clusters of NXDomains that have common properties. Specifically, domain names clustered together have (i) similar syntactic features, which is based on the assumption that a single DGA does not create completely indistinguishable domains. For instance, two of the syntactic features examine if domains have similar length or similar character frequency distribution. In addition, domains in the same cluster (ii) must have been queried by overlapping suspicious machines during a time window. The intuition behind the second property is that multiple nodes will be infected by the same DGA-based malware within a network.

Technically, all the NXDomain packets are fed to the DGA Discovery process. The process preforms clustering according to the statistical features and the "overlapping" property, individually. Then, the two distinct views of similarities are reconciled in a single cluster during the correlation phase. The objective of the overall stage is that each cluster represents non-existing domain names generated by the same DGA binary. After the unsupervised clustering is completed, the clusters are filtered, where legitimate, known malicious DGAs and -clusters grouped due to- common typos are discarded.

Next, the DGA Classification and C&C Detection module uses supervised learning techniques to process every new DGA cluster discovered during the previous phase. An important component of this module is the DGA Modeling, which already possesses a whitelist and a blacklist. The last ones are generated only by executing known DGA-based malware in a controlled environment. The DGA Modeling expands the unlabeled clusters to include every NXDomain triggered by the involved hosts. This is needed to extend the data in order to build better statistical models. The authors claim that the statistical learning algorithms eliminates the limited unrelated domains. Moreover, the supervised DGA Classifier is used to assign scores of confidence of the relation

[5] A malware that uses multiple domain names is characterized as multi-domain.

between a set of NXDomains and a modeled DGA. This is the classifier that is actually used during the cluster filtering, that we mentioned before, to discard already known and modeled DGAs.

Pleades can also detect infected hosts. This is achieved by extracting the statistical features -that are used for the clustering- from subsets of NXDomains produced by a single host. Then, the DGA Classifier evaluates them. If there is strong correlation with a specific DGA the host is reported as compromised.

Eventually, the C&C Detection part of the module monitors the valid DNS queries initiated by the recorded infected hosts. These active queries are compared with non-existing domain names generated by a DGA, using Hidden Markov Models. The output is the likelihood of a domain name, as a sequence of characters, being generated by a specific DGA. If the probability is high, the domain name is marked as malicious.

Although the clustering is unsupervised, the classification and, especially, the C&C detection rely on supervised learning and require manual inspection. Therefore, we consider that the system does not satisfy the autonomous criterion in Table 4.2.

The system clearly monitors DNS traffic directly from the clients and could be strongly argued that it violates privacy. We have, indeed, marked the criterion as unsatisfied but we have also added a remark in Table 4.2. It is worthy mentioning that Pleiades mainly processes NXDomains and it expands the monitoring only if there is evidence of infection. It could be argued that privacy is respected at some level but this would be a debate out of the scope of this report.

Chapter 5

Approach

In the previous chapter we examined methods that perform DNS-based detection of malicious activity. It would be interesting to compare these methods to determine their efficiency and accuracy. Unfortunately, the majority of them are unavailable; hence, the comparison is not feasible. Another idea has been to create a reputation-based system that evaluates the outputs of the examined methods. If multiple implementations mark the same activity as malicious, we increase the confidence that it is indeed malicious. However, the methods do no adequately satisfy the defined criteria. There is only one method that is autonomous, able to run in real-time and with (claimed) false positive rate lower than 1%.

Nevertheless, this report attempts to observe the generic behavior of DNS-based detection. The remainder of this report examines whether monitoring DNS traffic can increase the effectiveness of detecting infected computers. Initially, we investigate the concept of combining DNS-based and flow-based monitoring into a single detection system. To this end, we define **Scenario 1** and **Scenario 2**, which are evaluated regarding performance and accuracy. Finally, we investigate whether DNS-based monitoring can be a standalone detection method, by defining **Scenario 3**.

5.1 Overview

Monitoring DNS traffic is an efficient way to detect attacks. Furthermore, observing a network flow can increase the confidence of a DNS alert. This is true because a DNS query indicates that a node requested the IP address of a digital location, while a flow demonstrates that the node actually connected to that digital location.

DNS-based and flow-based monitoring clearly have both advantages and disadvantages. Performance is the strong advantage of the former because the volume of DNS traffic is just a fraction of the network flows. On the other hand, false alerts could be triggered by DNS queries whose answers were never used to establish a connection. These properties are reversed when monitoring network flows. Flow-based detection that processes all the flows (non-sampled data) is 100% accurate about the establishment of a connection. However, network flows consist a larger volume of information than DNS traffic.

An ideal detection system would have the performance of DNS-based monitoring and the accuracy of flow-based monitoring. In this report, we investigate three scenarios that monitor DNS traffic in order to effectively detect infected computers. Before describing them we define the common ground and a set of common characteristics.

The target of each scenario is to detect infected users within a network, based on historical knowledge. A common way to detect infections is by utilizing a blacklist with reported malicious end-nodes. Any connection to a blacklisted node should be considered as malicious and the initiator as infected. More information about the challenge of choosing the appropriate blacklist that indicates infection with high probability can be found in Section 3.4.1.

Scenario 1 and **Scenario 2** combine DNS-based and flow-based monitoring into a single

Figure 5.1: Approach for **Scenario 1** and **Scenario 2** using DNS-based prefiltering before performing flow-based filtering. The rectangles are processes. The rectangles with rounded corners are data and, specifically, the dotted are input or output of the system.

detection system. An overview can be seen in Figure 5.1. Both scenarios include two processes: the DNS-based prefiltering and the flow-based filtering. The first process receives a blacklist and the DNS packets (pcap). It is not necessary that the datasets are static but they could be inputs from live traffic. The DNS-based prefiltering process matches the blacklist against the feed of DNS responses. Every match is stored into an intermediate blacklist (minimum blacklist), which would be much smaller than the original. Normally, there are only a few hits of a given blacklist. This minimum blacklist contains only threats that are "active" in the network at that moment. Next, a traditional flow-based filtering matches the minimum blacklist against the feed of network flows (nfcapd). The last process produces the alerts of this detection system.

The motivation is that observing an established connection increases the confidence of a DNS-based alert. Moreover, reducing the blacklist fed to the flow-based monitoring can increase its performance. In Section 5.2, we discuss in detail the relation between blacklist size and performance of flow-based filtering.

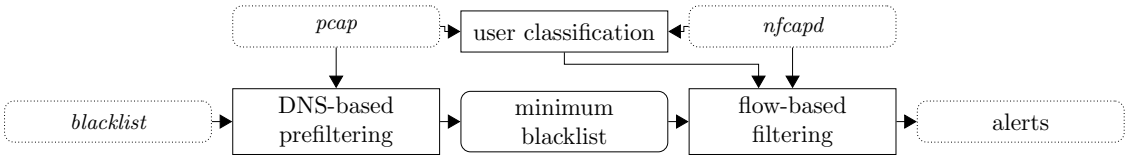


Figure 5.2: Approach for **Scenario 1** and **Scenario 2** including the user classification module.

The proposed system strongly relies on the DNS information. In order to precisely evaluate it, it is essential to estimate the percentage of users that utilize monitored or unmonitored resolvers. The initial step of the measurements is to distinguish the users using the user classifier module. This module can be seen in Figure 5.2 among the other processes of the approach (i.e. DNS-based prefiltering and flow-based filtering). The user classification module takes into consideration both the DNS and flow data in order to distinguish the user groups. Eventually, evaluating each group individually can specify separate potential for detection regarding the group that the user belongs.

Another possible advantage of the approach for **Scenario 1** and **Scenario 2**, besides performance, could be noticed in networks with not many infected user. Whenever the DNS-based detection does not output an alert, the flow-based monitoring could be halted (if it is used for only this task). Consequently, the machine could be maintained or just perform another task. Moreover, we could argue that privacy of the users could be respected, which can lead to halt monitoring whenever there does not exist a security threat.

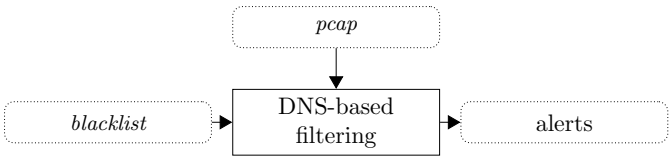


Figure 5.3: Approach for Scenario 3 using only DNS-based filtering.

Finally, **Scenario 3** examines whether DNS-based monitoring can be a valid standalone detection method. Consequently, we completely remove the flow-based filtering process, as it can be seen in Figure 5.3. The single DNS-based filtering process does not precede another filtering process and outputs the alerts directly. The motivation for this scenario is to determine the accuracy rate of a DNS-based detection system.

5.2 Performance of flow-based filtering

Applying a large blacklist on flow-based monitoring could be highly demanding. Exporting network flows and even storing them consumes resources. Expanding this functionality to filter a big blacklist may not always be possible and, especially, for large institutes, such as SURFnet, with billions of flows per hour.

Scenario 1 and **Scenario 2** examine whether we can effectively reduce a blacklist before it is fed to flow-based filtering. However, it must be priorly examined whether this reduction makes the filtering of network flows feasible or at least faster. In order to examine the last point, consecutive filters on network flows were executed for various sizes of blacklist. Moreover, the experiment was repeated over different datasets, provided by SURFnet, in order to verify that the results are typical. Each experiment was executed three times and the median value has been recorded so as to discard extreme values. The results are displayed in Figure 5.4, where we can see the relation between size of blacklist and number of flows per second that can be processed.

The experiments were run on a modern server that runs GNU/Linux (Debian squeeze-lts x86_64), has two INTEL XEON L5640 processors (2.26GHZ, 6 Cores, 12 threads) and 32GB memory, using nfdump version 1.6.1.

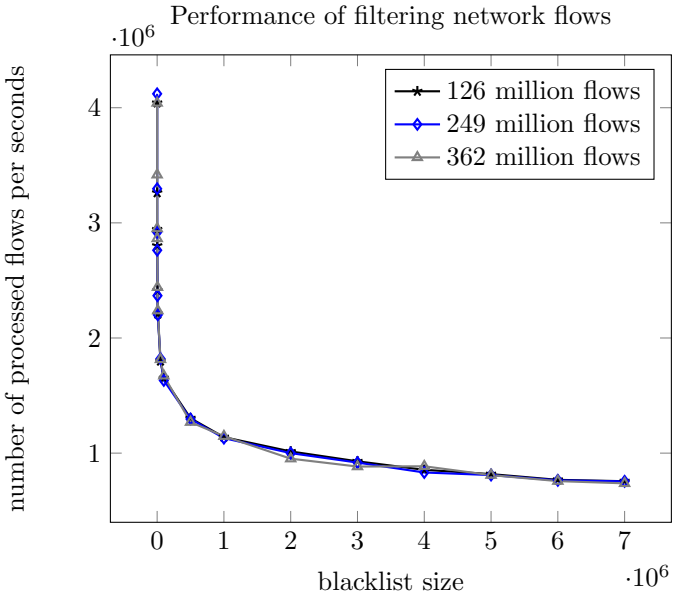


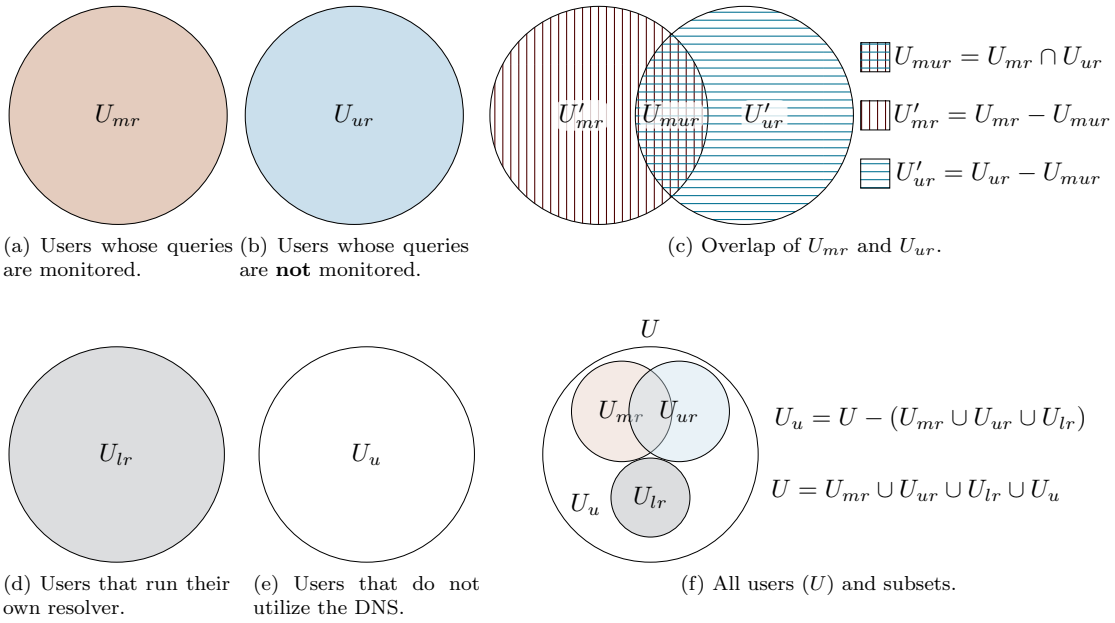
Figure 5.4: Filtering network flows for variable size of blacklist. The range of the size is between 10 elements and 7.6 millions elements.

Figure 5.4 displays three separate lines, one for each dataset, which largely overlap. The figure indicates that a reduction of the blacklist can increase the performance. Specifically, filtering a blacklist with 500 thousands elements can process approximately 1.29 million flows per second. If the blacklist is reduced to one tenth of its size (50 thousands elements), the system can process more than 1.81 million flows per second. This means that the performance of network flow filtering is increased by 40%. Further reduction to one hundredth of the original blacklist can increase the performance by 86% and process roughly 2.41 million flows per second.

The determination of a more accurate metric for the ideal reduction and the most efficient processing is out of the scope of this report. Considering the variation of flows per seconds that depends on the day time, future work can examine the optimal reduction to ensure that the processing does not exceed the hardware capabilities.

5.3 User classification

Monitoring DNS traffic at the resolver does not necessarily contain all the queries and replies that were transferred within the network. Although we can observe the queries to and the responses from the monitored resolver, any user can choose to utilize an external unmonitored resolver. It is also possible for smaller networks, such as a laboratory, to run their local unmonitored resolver. DNS-based detection is expected to fail to detect infected users, if they utilize unmonitored resolvers, due to lack of information. For further analysis, it is essential to separate those users. The user classification module is used to evaluate the system, although it is not necessary for its operation.



(a) Users whose queries are monitored.

(b) Users whose queries are **not** monitored.

(c) Overlap of U_{mr} and U_{ur}.

(d) Users that run their own resolver.

(e) Users that do not utilize the DNS.

(f) All users (U) and subsets.

Figure 5.5: Definition of user groups

In order to understand the behavior of the users, they must be categorized according to the resolver that they utilize. Figure 5.5 helps illustrate the distinction of the user groups and their overlap. We consider four generic groups of users. First, the set U_{mr}, which stands for users (U) that send queries to and receive responses from the monitored (m) resolvers (r). This set of users can be seen in Figure 5.5a. The second generic group is defined as U_{ur} and consists of users that have communicated with an unmonitored (u) resolver (see Figure 5.5b). However, the first two are not exclusive groups because a node may belong in both. Any host can be configured to utilize multiple resolvers.

In Figure 5.5c we can see the overlap of the two generic groups that reveals the exclusive groups of users. The subset U'_{mr} (relative complement of U_{ur} in U_{mr}) represent the users that exclusively utilize internal monitored resolvers. The subset U'_{ur} (relative complement of U_{mr} in U_{ur}) is the group of users that exclusively utilize external unmonitored resolvers. Eventually, the subset U_{mur} is the intersection of U_{mr} and U_{ur}. It represents users that utilize both monitored and unmonitored resolvers.

Another generic group (U_{lr}) contains users that run their own local (l) resolvers (r) (see Figure 5.5d). These nodes are resolvers that independently communicate with authoritative name servers to request information about zones. They are distinguished based on specific DNS labels that appear only when a query is directly answered by an authoritative name server. Their purpose might be to serve a single node or a small group of users in an office or a laboratory.

Finally, the users in group U_u utilize an unspecified (u) number of resolvers because no such communication can be detected. It is possible that a user has not requested DNS information at all within the examined time-window. Alternatively, the node might utilize one of the unmonitored resolvers in group U_{lr}. Another possibility is that the node contacts an external resolver in a non-standard way that could not be identified (e.g., different port that the standardized for DNS traffic).

To summarize, we consider five distinct groups of users. The first group (U'_{mr}) consist of nodes that exclusively use monitored resolvers. The second group (U'_{ur}) consist of nodes that exclusively use unmonitored resolvers. The third group (U_{mur}) contains nodes that utilize both monitored and unmonitored resolvers. The nodes that run their own local resolver compose the fourth group (U_{lr}) and finally, the users that do not communicate with any resolver compose the fifth group (U_u).

5.4 Scenarios

Below we define the scenarios that are investigated.

Scenario 1 The first scenario examines whether DNS-based prefiltering can effectively increase the performance of flow-based filtering. The goal is to perform flow-based filtering using a much smaller blacklist than the original. The decrease of the size of blacklist can significantly increase the performance of flow-based filtering, as it is shown in Section 5.2. In addition, this reduction should not have a considerable effect on the results. It should not lead to inaccurate results, beyond the accepted false tolerance.

Technically, **Scenario 1** operates at two observation points at the same time. The described processes: DNS-based prefiltering and flow-based filtering (see Section 5 and Figure 5.1) are placed one at each observation point. Firstly, the DNS-based prefiltering process is placed at position (1) in Figure 5.6 (this figure has the components that were described for Figure 3.1, without the extra sensors and name servers). The monitored link is above the resolver or, in other words, between the recursive caching name server and the remote name servers. Monitoring at this link is respectful of privacy as it is discussed in Section 3.5.2. DNS-based prefiltering depends on an IP-based blacklist (IPBL) in order to detect malicious IP addresses included in DNS responses. The output of the first sensor is an intermediate blacklist (Minimum IPBL) that indicates active threats at that moment. Then, flow-based filtering is performed at the second observation point (2), using the intermediate blacklist. Eventually, the last process reports the detected infections to the administrator.

Furthermore, we can examine whether this design can detect infected users whose DNS traffic is not monitored. DNS-based prefiltering produces an intermediate blacklist by observing the users that contact the monitored resolver. The intermediate blacklist can then be applied to another population (i.e. another group of users). In case that most of the infected users are detected we can support the universality of this method. That is, applying the intermediate blacklist to another group of users will have the same accuracy as if their DNS traffic was monitored. This might work for two reasons: the homogeneity of the user population (if they all exist in the same network) and -most importantly- the "popularity" of a malware. A "popular" (or active) malware acts intensely at a specific period of time.

Scenario 2 This scenario examines whether DNS-based prefiltering can be optimized by slightly altering the design. For this scenario we move the DNS sensor from the link above the resolver (between the recursive caching name server and the remote name servers as it is shown in Figure

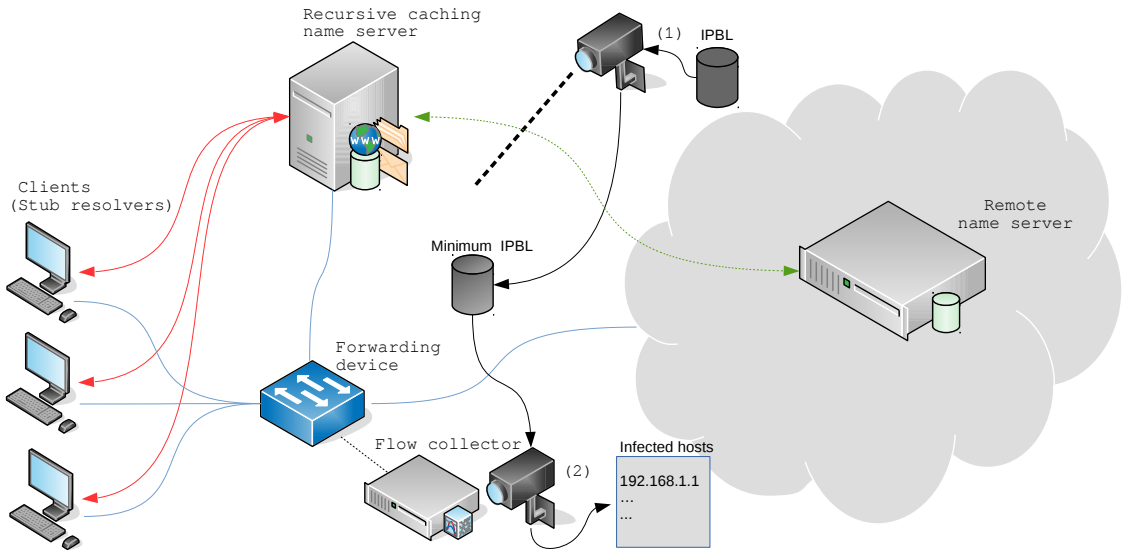


Figure 5.6: Design of **Scenario 1**, where the DNS-based prefiltering process is placed between the resolver and the remote name servers.

5.6) to the link below the resolver (between the clients and the resolver as it is shown in Figure 5.7). The advantage of monitoring the link below the resolver is that it indicates DNS records that have already been cached. This leads to rise the alert frequency in case of domain names that stay in the cache for a long period of time (high TTL values). What we want to examine is whether there are more DNS-based alerts and the detection rate is increased.

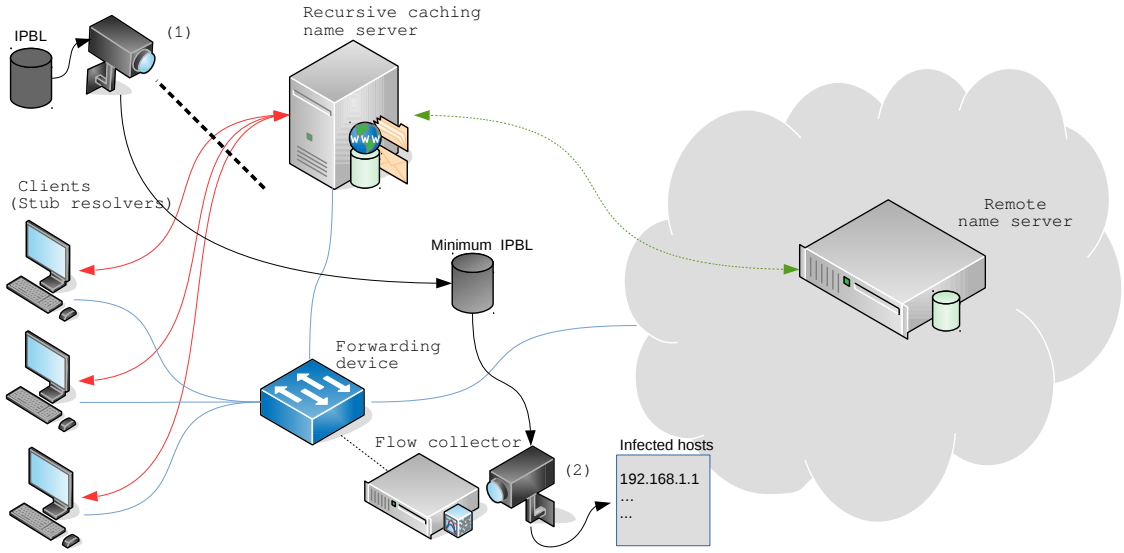
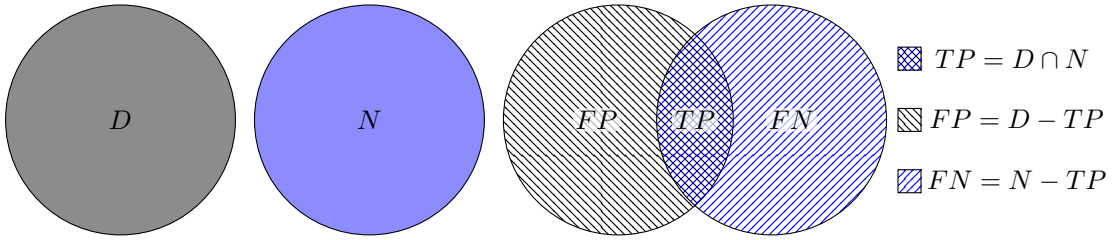


Figure 5.7: Design of **Scenario 2**, where the DNS-based prefiltering process is placed between the clients and the resolver.

An important drawback of this possible optimization is that the new position is more privacy-sensitive (see Section 3.5.2). In fact, the IP addresses of the clients are visible, despite the fact that the method neither records nor examines them. Under specific circumstances there could be a trade-off between privacy preservation and detection accuracy, which should be balanced.

Scenario 3 In this scenario we examine whether DNS-based filtering can be used as a standalone detection system. Until now we have examined DNS-based filtering only as prefiltering that reduces a blacklist fed to flow-based filtering. However, the DNS sensor at position (1) in Figure 5.7 can report not only malicious destinations but also the probable infected clients that requested them. What we want to investigate is whether these reports are satisfactory so as to completely skip the monitoring of network flows.



(a) DNS-based alerts (b) flow-based alerts (c) Overlap of (a) and (b)

Figure 5.8: Comparison of DNS-based and flow-based filtering.

In order to measure the detection rate, given a blacklist, we separately filter DNS traffic and flows. Since we consider the flow-based results as the correct (most accurate), we want to examine how close to those are the DNS-based results. What we take into account is the number of infected nodes that are detected in either case. Consequently, there are two overlapping sets of results, as it is shown in Figure 5.8. The infected nodes, detected by both detection methods, are in the intersection of the two sets. They compose the true positive (TP) alerts of DNS-based detection (see Figure 5.8c). Moreover, the nodes that appear in a flow-based alert but not in a DNS-based are false negatives (FN). Likewise, those that appear in a DNS-based alert but not in a flow-based alert are false positives (FP). Everything outside both sets is a true negative (TN) of DNS-based filtering.

In order to evaluate the system we examine precision (i.e. the proportion of the true positives among all the positive results), recall (i.e. the proportion of the true positives among the flow-based alerts, which are the true positives and the false negatives) and accuracy (i.e. the proportion of true results -both true positives and true negatives- among the results). Accuracy is the strongest measurement because it includes precision and considers trueness. In addition to the precision rate, it examines the proportion of true negatives against all the negative results.

Chapter 6

Experiment setup

The setup of our experiments requires a dataset, blacklists and tools to analyze them. In this chapter, we characterize the obtained data, we describe how they are analyzed and we define the hypothesis that we want to validate for each scenario.

6.1 Dataset

The dataset consists of DNS traffic and network flows for a specific period of time. We obtained this data from a large university in the Netherlands, which is connected to SURFnet. The original volume of information that we obtained can been seen in Table 6.1. There are two types of captures: pcap files that contain only DNS packets and nfcapd that contain exported flows. All the data were captured on September 4th between 12:00 and 14:00 CEST.

First of all, we must clarify that the flow data had been reduced before they were given out. We have been informed that the total number of flows, prior to the reduction, was 61782973. Roughly two third of them were removed in compliance with the following rules: the flows must originate within the network and they must egress the network. Next, the origin IP address in the flow data has been anonymized (interchanged) with a prefix-preserving symmetric cryptographic algorithm. The DNS data have not been reduced but they have also been anonymized using the same -unknown to us- secret key. The anonymized DNS field are: the source address in queries and the destination address in responses. These rules have been applied by the provider in order to reduce the privacy impact of this analysis.

	pcap (below)	pcap (above)	nfcapd
size	3487MB	705MB	520MB
total (packets/flows)	16712328	3569186	20766404*
IPv4 (packets/flows)	16594995	3032016	20029309
IPv6 (packets/flows)	117333	531152	737095
unique source IP addresses	16589	36781	12623
unique destination IP addresses	16612	38275	3506477
unique connections (src,dst)	-	-	5719619
unique connections (src,dst,date)	-	-	6047220

Table 6.1: Information about the obtained data volume.

In Table 6.1, beyond the size and total number of packets there are statistics about the Internet Protocol (version 4 or 6). Moreover, the number of unique source and destination addresses are recorded. There are three columns in the table. The first two are for the DNS data below and above the resolver. That is, the links between the clients and the resolver and between the resolver and the authoritative name servers, respectively. Actually, there are five monitored resolvers, whose

packets are gathered together. Therefore, we consider them as one. Finally, the last column show information about the flow data.

In addition to the previous statistics, we record the unique connection for the flow data. A unique connection is when the tuple source and destination IP addresses is the same. Furthermore, we record the triplet that takes into consideration the timestamp. As a result, latter appearances of flows from the same node and to the same destination are not aggregated. The last does not distinguish flows that were consecutive and at exactly the same time (fraction of a second difference).

6.2 Blacklists

One important component of the experiments is the blacklist that indicates malicious activities and particularly infected users. We have obtained several blacklists from two distinct sources.

First, we have been granted access to a set of blacklists provided by the Spamhaus Project: the Spamhaus Botnet Controller List (BCL), which is an advisory "drop all traffic" blacklist[1] consisting of single IPv4 addresses, used by cybercriminals to control infected computers (bots). The DROP (Don't Route Or Peer) and EDROP blacklists[2], which are advisory "drop all traffic" too. They consist of netblocks (range of IP addresses) that are "hijacked" or leased by professional spam or cyber-crime operations (used for dissemination of malware and botnet controllers). Finally, the Spamhaus Exploits Block List (XBL)[3], which is a realtime database of IP addresses of hijacked computers infected by illegal third party exploits, including open proxies, worms/viruses with built-in spam engines, and other types of trojan-horse exploits.

Second, we obtained a publicly available blacklist[4] of known botnet command and control servers, by the Emerging Threats provider of threat intelligence. According to their website[4]: "the list should be considered *very* highly reliable indications that a host is communicating with a known and active Bot or Malware C&C server". This blacklist includes external sources from four additional providers (Shadow Server, Spyeye Tracker, Palevo Tracker and Zeus Tracker).

blacklist	IP addresses	netblocks	updated every
botnetcc	492	-	20 minutes
botnetcc_extensive	17	-	20 minutes
emergingcc	915	-	1 day
sbl_drop	-	712	1 hour
sbl_edrop	-	71	1 hour
xbl	10380840	-	15 minutes

Table 6.2: The obtained blacklists, including information about their entries and frequency of updating.

An overview of all the blacklists that were used in our measurements can be seen in Table 6.2. We are especially interested in the botnetcc and emergingcc lists because they both list only active C&C servers. These can indicate infected hosts with high confidence, as it is discussed in Section 3.4.1.3.

The botnetcc list is updated more frequently than two hours, that is the captured time window. For our complete analysis we selected the instance that existed at exactly one hour after the capture had started. However, we may have missed blacklisted elements which may have been added later or elements that may have been used and then removed within the first hour. This is possible because botnet activity may last less than an hour as it is mentioned in Section 3.2.2. Consequently, we created a separate botnetcc_extensive list that contains every extra element of

[1] https://www.spamhaus.org/bcl/, retrieved September 2015.
[2] https://www.spamhaus.org/drop/, retrieved September 2015.
[3] https://www.spamhaus.org/xbl/, retrieved September 2015.
[4] http://doc.emergingthreats.net/bin/view/Main/BotCC, retrieved September 2015.

the botnetcc list within a larger time window of four hours (two hours before and two hours after the original).

Finally, we want to examine whether the removed flows influence our measurements. As it is mentioned in Section 6.1, flows whose destination IP address remains inside the network have been removed. Specifically, we look for blacklisted elements that belong to those netblocks that were removed. We observed that 1 IP address in the emerging blacklist and 10 in the xbl belong to the removed netblocks. This means that we cannot see any possible flow with a destination one of those 11 blacklisted IP addresses. Nevertheless, we can examine if any host received one of those IP addresses in a DNS response. We are able to observe it because the DNS responses are plain, apart from the destination field which is anonymized. By filtering the DNS traffic we did not see any response with a discarded blacklisted destination. Consequently, we assume that the reduction does not considerably influence the overall measurements.

6.3 Groups of users

Before describing the evaluation methodology of each scenario, the possibility of users using external resolvers must be examined (see Section 5.3). It is expected that the majority of the users utilize the monitored resolvers. People commonly keep the default configuration of their connectivity. The characterization of the user groups can reveal the trend regarding the users' preferred utilized resolvers. If many users utilize unmonitored resolvers, it is clear that DNS-based monitoring misses a large amount of information.

In order to determine the groups we can examine the obtained network flows. The objective is to observe utilization of unmonitored resolvers. It is reasonably assumed that connections related to the DNS are established on destination (remote) port number 53, which is the standardized behavior [4]. However, it is also possible that other types of connection are established on port number 53. The kinds of the destinations are not instantly clear and need further examination.

6.3.1 Methodology to identify the destination

The initial step is to extract the destination IP address of every flow that was established on port 53. Each destination is either a public resolvers or an authoritative name server. A query for "*surfnet.nl*" is actually sent to each one of those. In case the reply is a refused response (REFUSED), the destination is an authoritative name server, which refuses to resolve queries. Otherwise, if it replies with a valid response (NOERROR), it is probably a resolver. Finally, if there is another type of answer or no answer at all, the node cannot be identified and it is ignored.

Normally, an authoritative name server answers only queries regarding its own zone and local information. However, misconfigured authoritative name servers may reply to any query, even for a completely different zone. An additional step can distinguish misconfigured authoritative name servers from the detected resolvers.

To illustrate the instructions let us consider the authoritative name server for "*surfnet.nl*", whose IP address is 192.87.106.101. What we know is the IP address and we want to examine whether it is authoritative for any domain name (although we actually know that it is authoritative for "*surfnet.nl*'). Initially, we perform a reverse DNS lookup (see Section 2.5.2) for 192.87.106.101, whose answer is[5] "*ns1.surfnet.nl.*". Next, we request the Start of Authority (soa) record for the second-level domain name of the previous answer. In our example[6] the requested soa is for "*surfnet.nl.*". The process is repeated for the third-level domain name in case that the second is a generic, such as "*co.uk*". Eventually, if the reply is an authoritative answer (flag aa in the response), the server is an authoritative name server. In our example it is an authoritative answer and we verify that there is an authoritative name server.

[5]The executed command is dig -x 192.87.106.101
[6]The executed command is dig soa surfnet.nl. @192.87.106.101

6.3.2 Methodology to determine the user groups

Previously, we retrieved the flows, established on port 53, and identified the type each destination. The identified types are either resolvers or authoritative name servers. In this step, we examine the source of the flows in order to determine the user groups.

Any user who contacted an identified authoritative name server belongs to the group U_{lr} that run their own local resolver. An IP address that appears in this group cannot appear in another group. The reason is that they obtain DNS information directly from the source and their behavior is dissimilar to a normal user's. Furthermore, the users who communicated with an external resolver compose the group U_{ur}. Likewise, the users who appear in the DNS capture compose the U_{mr} group that utilize the monitored resolver. However, they must have received a response that contains a queried domain name and at least one valid IP address in the answer section.

Eventually, the users that do not belong to any other group are part of the U_u group. Unfortunately, we cannot identify if, for instance, they utilize a local resolver (group U_{lr}) because the internal flows are missing (see Section 6.1).

The final step is to separate the exclusive groups from the U_{mr} and U_{ur} groups. The IP addresses that appear in both groups compose the U_{mur} group. The U'_{mr} and U'_{ur} groups consist of those that are unique in U_{mr} and U_{ur}, respectively.

6.4 Database

Every extracted information is inserted into a MySQL database. This can increase the processing time without maintaining custom lookup tables that are prone to human errors.

DNS data are stored in four tables, two for each dataset (see Section 6.1). Specifically, table *resp* contains every DNS response sent from the resolver to the clients. The response must have valid an A record field (i.e. IPv4 address) and a queried domain name. Each row of the table has four columns: the number of the frame (increasing unique identifier from the pcap file), the IP address of the client (destination of the response), the queried domain name and the timestamp. A second table *resp_dnsa* is used to record the IP addresses contained in the answer. The unique frame number is used to relate them back to the response without violating the first normal form (i.e. each attribute contains only atomic values). Likewise, tables *aaresp* and *aaresp_dnsa* contain the authoritative answers from the link between the resolver and authoritative name servers.

Table *flows* stores the obtained flow data and has three columns: the source IP address, the destination IP address and the timestamp. Moreover, there is one table for each user group, with their IP address in a single column.

Finally, the blacklists are also loaded. The tables for the blacklists that contain single IPv4 addresses have a single column. Meanwhile, the tables for the blacklists that contain netblocks have two columns. The reason is that in addition to an IP address they contain a network mask (CIDR notation). In order to increase the lookup process, the minimum and maximum hosts of that subnetwork are calculated and stored in the database. For instance, if the CIDR in the blacklist is "*192.168.1.0/24*", it is inserted the minimum host "*192.168.1.1*" and the maximum host "*192.168.1.254*" of that netblock. Consequently, we can easily evaluate if an IP address belongs to a subnetwork by comparing if it is greater than the minimum host and lower than the maximum host.

6.5 Scenarios

Below we examine how to evaluate each scenario and what do the results mean. Each investigation methodology corresponds to the relative scenario in Section 5.4.

Scenario 1 The first scenario examines whether DNS-based prefiltering can effectively increase the performance of flow-based monitoring. We want to validate the following hypothesis:

(a) the intermediate blacklist should be much smaller in order to increase the performance of flow-based filtering.

(b) this reduction should not decrease the detection rate, beyond some accepted false tolerance.

For the first hypothesis (a), we can record the size of the intermediate database at the end of the time window. Technically, DNS-based prefiltering requires a DNS sensor that filters responses using a given blacklist. Every detected malicious IP address is then inserted into the intermediate blacklist. Normally, the IP address remains there either until its TTL value expires or until it is removed from the original blacklist. For simplicity and due to the limited time window of our dataset, we do not consider removing elements from the intermediate blacklist. As a result, the intermediate blacklist in our experiments is only growing. Eventually, we record its maximum size in the results.

It is possible that the processing time would be further reduced if DNS-based prefiltering was running for a larger time window. However, due to the larger intermediate blacklist the detection rate could be increased. Our experiments were performed for a two-hour time window and we assume that two hours is the limit that an element can remain in the intermediate blacklist.

In order to evaluate the results we can consider the relation between blacklist size and the performance of flow-based filtering, that was discussed in Section 5.2. The smaller the intermediate blacklist the faster the processing of flows.

Regarding the second hypothesis (b), we can test the accuracy by comparing the results of **Scenario 1** with the original. The original results occur if only flow-based detection was applied (i.e. without the DNS-based prefiltering) given the same blacklist. The alerts that appear in both results are the true positives of **Scenario 1**. Those that appear in the original results but not in the results of **Scenario 1** are false negatives. Finally, false positives do not exist because the DNS sensor is not used for detection but only reduces the blacklist fed to flow-based monitoring.

```
1  SELECT flows.*
2  FROM (
3      flows
4      INNER JOIN (
5          umur
6      ) ON flows.src=umur.src
7  )
8  INNER JOIN (
9      botnetcc
10 ) ON flows.dst=botnetcc.ip;
```

Listing 6.1: MySQL query to obtain the flow-based results for the U_{mur} user group, given the botnetcc blacklist.

Since all the information exist in the database (see Section 6.4) we can obtain the original results by executing a single MySQL query that can be seen in Listing 6.1. The sample MySQL query returns the flow-based alerts for all the flows of the user group U_{mur} (see lines 3-6), given the botnetcc blacklist (see lines 2-10).

The results of **Scenario 1** can be produced by executing a MySQL query with more joins. There are five tables that need to be processed: the flows, the DNS traffic between the resolver and the authoritative name servers (2 tables), the selected user group and the blacklist. Listing 6.2 illustrates how the tables are combined. We will begin explaining from the most nested join between the lines 12 and 16. All the IP addresses that appear in the answer section of a DNS response are examined whether they exist into the given blacklist. Next, code in lines 10-18 retrieves the timestamps when those IP addresses were requested, using the unique number (i.e. frame) of the packet. The latest result is grouped by the blacklisted IP address that was detected and its earliest timestamp (lines 9-19). The reason is that after a malicious IP address is inserted

```
1  SELECT flows.*
2  FROM (
3      flows
4      INNER JOIN (
5          umur
6      ) ON flows.src=umur.src
7  )
8  INNER JOIN (
9      SELECT aaresp_dnsa.dnsa as dnsa, MIN(aaresp.timestamp) as timestamp
10     aaresp
11     INNER JOIN (
12         aaresp_dnsa
13         INNER JOIN (
14             botnetcc
15         )
16         ON aaresp_dnsa.dnsa=botnetcc.ip
17     )
18     ON aaresp.frame=aaresp_dnsa.frame
19     GROUP BY dnsa
20 ) as alerts
21 ON flows.dst=alerts.dnsa AND flows.timestamp+INTERVAL 5 MINUTE>=alerts.timestamp
```

Listing 6.2: MySQL query to obtain the results of **Scenario 1** for the U_{mur} user group, given the botnetcc blacklist.

in the intermediate blacklist it is redundant to insert it again. Eventually, the IP addresses and their earliest timestamps compose the table alerts (line 20) that DNS-based prefiltering produces.

The remaining lines in the query process the flow data. Similarly to the previous query (see Listing 6.1), users that do not belong to the examined user group are discarded (lines 3-6). Finally, line 21 determines the flows that have destination an entry of the intermediate blacklist, but only after the entry was inserted. The timing condition verifies that the timestamp of a DNS-based alert is earlier than the timestamp of a flow. Notably, we delay the timestamp of the flows by five minutes because separate machines captured the two datasets, which may not be synchronized. In any case, it usually takes a few minutes for the flows to be exported.

The results produced by executing the second query are the true positives. Meanwhile, the false negatives can be identified by subtracting the true positives from the original results. This can be easily done if the original results (Listing 6.1) are inserted into a temporary table n and the results of **Scenario 1** (Listing 6.2) are inserted into another temporary table p. Then, executing the mysql query is Listing 6.3 outputs the false negatives.

```
1 /* n contains the original results        */
2 /* p contains the results of Scenario 1   */
3 SELECT *
4 FROM n
5 WHERE n.frame NOT IN (SELECT frame FROM p)
```

Listing 6.3: MySQL query to extract the false negatives of **Scenario 1**.

What we specifically want to observe in this comparison is the number of infected users that were detected. This is related to the alerts but not absolute. A single user may have produced multiple alerts by contacting multiple malicious destination or multiple times a single malicious destination. Therefore, in addition to the number of alerts we distinguish the number of unique source and destination IP addresses. The sources reveal the number of detected infected users, while the destinations reveal how many entries of the intermediate blacklist were observed.

Scenario 2 The investigation of **Scenario 2** is identical to the investigation of **Scenario 1**. We actually repeat the exact steps and analysis with the only alternation the different DNS capture

given as input. For instance, the query to obtain the results of **Scenario 2** for the U_{mur} user group, given the botnetcc blacklist, is the same as Listing 6.2, where aaresp (lines 9, 10 and 18) and aaresp_dnsa (lines 9, 12, 16 and 18) are replaced by resp and resp_dnsa, respectively. As it is described in Section 6.4, tables aaresp and aaresp_dnsa contain the DNS responses from the link between the resolver and the authoritative name server, while tables resp and resp_dns contain the responses from the link between the clients and the resolver.

For **Scenario 2** we want to validate the same hypothesis as for **Scenario 1**, including one more:

(a) the intermediate blacklist should be much smaller in order to increase the performance of flow-based filtering.

(b) this reduction should not decrease the detection rate, beyond some accepted false tolerance.

(c) the detection rate is increased compared to **Scenario 1**.

We would expect this scenario to increase the detection rate as a consequence of eliminating the cache effect. The link between the clients and the resolver can see queries whenever they are requested. On the contrary, in the design of the previous scenario the queries are visible only when they are initially requested or when a TTL value expires and the resolver re-requests the information. However, if the accuracy of this scenario is the same as the accuracy of **Scenario 1**, the latter is preferred due to the privacy benefits.

Scenario 3 In the previous scenarios, DNS-based prefiltering was performed prior to flow-based filtering. In this scenario, the standalone DNS-based detection reports directly any detected alert. In order to evaluate **Scenario 3** we separately filter DNS traffic and flows, given the same blacklist. The experiments are repeated only for the groups whose queries are monitored at least partially (i.e. U'_{mr}, U_{mur} and U_{lr}). The rest of the groups are excluded because we have not obtained DNS traffic generated from those at all.

For **Scenario 3** we want to validate the following hypothesis:

(a) the precision rate should be high, which means that the nodes that are detected by DNS-based filtering should be detected by flow-based filtering as well.

(b) the recall rate should be high, which means that the nodes detected by flow-based filtering should be detected by DNS-based filtering as well.

(c) the accuracy should be high, which means that DNS-based and flow-based filtering should have approximately the same results.

In order to calculate the precision, recall and accuracy rates, we priorly need to count the true positives, false positives, true negatives and false negatives. In order to count the true or false alerts, we firstly need to produce the results of **Scenario 3** and the original results. The original results occur if only flow-based detection was applied given the same blacklist. They are produced using Listing 6.1 as it was described before. Similarly to the measurements for **Scenario 1**, we will compare the results of **Scenario 3** with the original results.

The results of **Scenario 3** can be produced using the query in Listing 6.4. The query is similar to the one in Listing 6.2 (lines 10-18), without considering the flow information. In addition, the user distinction is performed on the client that received the DNS response, instead of the source of the flow. In detail, all the IP addresses that appear in the answer section of a DNS response are examined whether they exist into the given blacklist (lines 5-9). Next, code in lines 3-11 retrieves the client that requested those IP addresses, using the unique number (i.e. frame) of the packet. Finally, in lines 2-15, users that do not belong to the examined user group are discarded. The results are the alerts produced by DNS-based filtering.

The next step is to compare the results of **Scenario 3** with the original. The original flow-based results, produced by Listing 6.1, are stored into a temporary table n. The DNS-based results, produced by Listing 6.4, are stored into a temporary table d. From the two temporary tables we can now extract the true positives. A DNS-based alert is a true positive if and only if there

```
SELECT resp.*, resp_dnsa.dnsa
FROM (
    resp
    INNER JOIN (
        resp_dnsa
        INNER JOIN (
            botnetcc
        )
        ON resp_dnsa.dnsa=botnetcc.ip
    )
    ON resp.frame=resp_dnsa.frame
)
INNER JOIN (
    umur
) ON resp.client=umur.src
```

Listing 6.4: MySQL query to obtain the results of **Scenario 3** for the U_{mur} user group, given the botnetcc blacklist.

```
/* n contains the original results        */
/* d contains the results of Scenario 3   */
SELECT n.frame as nframe, n.src, n.dst, d.dframe
FROM n
INNER JOIN (
    d
) ON d.client=n.src AND d.dnsa=n.dst AND d.timestamp<=n.timestamp+INTERVAL 5
    MINUTE AND d.timestamp+INTERVAL 9 MINUTE>=n.timestamp
```

Listing 6.5: MySQL query to obtain the true positives of **Scenario 3**.

is a flow to the detected malicious destination within nine minutes. The true positives can be extracted by executing the query in Listing 6.5. The query joins the two temporary tables and for every DNS-based alert verifies the following: there is a flow (n.dst) to the requested malicious IP address (d.dnsa), from the same user (d.client=n.src) and within nine minutes. Notably, we again delay the timestamp of the flows by five minutes but only when it is checked that the flow started after the response was sent. The results are the true positive DNS-based alerts.

Since the goal is to detect infected users, we can count the distinct source IP addresses in order to extract the actual true positives. Since the true positives are known we can easily count the false positives, which are the unique source IP addresses that exist in the temporary table d without the true positives. Likewise, the false negatives are the unique source IP addresses that exist in the temporary table n without the true positives. Eventually, the true negatives are all the users of the group that do not belong to any of the true positive, false positive or false negative. Eventually, we can use the formulas to compute the precision, recall and accuracy rates.

Chapter 7

Evaluation

In this chapter, we present the results of our experiments. We initially describe the user groups and, next, we evaluate each scenario.

7.1 User classification

In order to determine the user groups we firstly need to examine the destinations. After they are identified, we present the user groups along with their cardinality.

7.1.1 Identification of destinations

The initial step is to export network flows that were established on port 53. In total, there are 172141 such flows. The unique flows, regarding the source and destination IP addresses, are 8708. Specifically, there are 1779 unique users and 3758 unique possible resolvers.

Further analysis on the possible resolvers showed that 2501 of the 3758 are authoritative name servers. In fact, these are the name servers that refused to answer a query. 700 nodes cannot be identified and 557 are functional publicly available resolvers. Out of the 557 public resolvers, 333 have replied with authoritative answer. Consequently, we consider them as misconfigured authoritative name servers that reply to queries. Although they actually have the role of the resolver and name server at the same time, we classify them only in the group of authoritative name servers.

7.1.2 Determination of user groups

Each group of users has a different size, which is determined next. An overview is shown in Table 7.1. There are 271 users who contacted one of the 2834 $(2501 + 333)$ discovered authoritative name servers. These users belong to the group U_{lr}. In total, 224 $(557 - 333)$ public resolvers -that are not misconfigured authoritative name servers- have been detected. The generic group of user U_{ur} that communicated with an external resolver can be identified using this information. Eventually, filtering the flows reveals 1179 nodes within the network that connected to one of those destinations.

Next, we want to identify the second generic group U_{mr} of users that resolve their queries to the monitored resolver. From the DNS capture we extracted 16382 users that belong to this group. Eventually, from the 17509 users that appear in either the flow or the DNS dataset, 740 do not belong to any group and are categorized in the U_u group. The calculation of the remaining groups can be seen in Table 7.1.

group	utilized resolvers	cardinality	extracted from
U	any	17509	flow & DNS data
U_{lr}	(l)ocal (r)esolver	271	flow data
U_{mr}	(m)onitored (r)esolvers	16382	DNS data
U_{ur}	external (u)nmonitored (r)esolvers	1179	flow data
U_{mur}	monitored & unmonitored resolvers	1063	$U_{mr} \cap U_{ur}$
U'_{mr}	exclusively monitored resolvers	15319	$U_{mr} - U_{mur}$
U'_{ur}	exclusively unmonitored resolvers	116	$U_{ur} - U_{mur}$
U_u	(u)nspecified number of resolvers	740	$U - (U_{mr} \cup U_{ur} \cup U_{lr})$

Table 7.1: User classification, including description, size and source.

7.2 Processed data

The primary assumption, stated in Section 5.1, is that the DNS traffic is just a fraction of the network flows. The number of DNS packets and the number of flows that each scenario processes can be seen in Table 7.2. The first two scenarios rely on the fact that filtering DNS responses is trivial compared to filtering network flows. As we can see from the table, the DNS responses examined by **Scenario 1** are just 1.6% of the total number of flows. Assuming that processing 1 flow requires the same time as processing 1 DNS packet, the primary assumption holds. As a result, DNS-based prefiltering is time-efficient. Possible reduction of the blacklist, that is fed to flow-based filtering, results to increased performance of the overall system.

	Scenario 1	**Scenario 2**	**Scenario 3**
DNS packets to inspect	959808	5908639	5908639
flows to inspect	61782973	61782973	-

Table 7.2: Data that each scenario processes.

For **Scenario 2** there are more DNS packets because it monitors the link between the clients and the resolver. This reveals queries for domain names that are cached in the resolver. In this case, the DNS packets are 9.6% of the flows. Finally, **Scenario 3** is the fastest alternative because the network flows are not monitored at all.

7.3 Results of Scenario 1

The measurements for **Scenario 1** can be seen in Table 7.1. The figure is divided into four parts, one for each blacklist. The botnetcc_extensive and sbl_edrop blacklists have been excluded from all the results because there were no hits.

The only metric for the results is the recall rate. Recall is the proportion of the detected alerts among all the results that should have been detected. Each table firstly presents the number (N) of infected users that can be detected in flows by filtering the original blacklist. These are the correct result that should be detected.

The next column of the table records the number (P) of infected users, detected using the proposed approach. The users are detected in flows by filtering the intermediate blacklist. The intermediate blacklist has been produced by the DNS-based prefiltering, as it is described in Section 5.1, given the original blacklist. At the caption we can see the size of the intermediate blacklist out of all the elements of the original blacklist. More detailed results of the experiments can be found in Appendix A.

Moreover, we divide the groups -in each table- between those whose queries are monitored at least partially (i.e. U'_{mr}, U_{mur} and U_{lr}) and those whose queries are not monitored at all (i.e. U'_{ur} and U_u). The first three groups can reveal the performance of the method. Meanwhile, applying

group	N	P	Recall
monitored queries			
U'_{mr}	4	1	25%
U_{mur}	1	1	100%
U_{lr}	0	0	-
subtotal	5	2	40%
unobserved queries			
U'_{ur}	0	0	-
U_u	0	0	-
subtotal	0	0	-
total	5	2	40%

(a) botnetcc (1/490)

group	N	P	Recall
monitored queries			
U'_{mr}	12	3	25%
U_{mur}	2	1	50%
U_{lr}	9	5	56%
subtotal	23	9	39%
unobserved queries			
U'_{ur}	2	1	50%
U_u	2	0	0%
subtotal	4	1	25%
total	27	10	37%

(b) emerging (13/915)

group	N	P	Recall
monitored queries			
U'_{mr}	568	24	4%
U_{mur}	121	6	5%
U_{lr}	57	1	2%
subtotal	746	31	4%
unobserved queries			
U'_{ur}	2	0	0%
U_u	35	0	0%
subtotal	37	0	0%
total	783	31	4%

(c) sbl_drop (32/712)

group	N	P	Recall
monitored queries			
U'_{mr}	6183	677	11%
U_{mur}	828	102	12%
U_{lr}	196	47	24%
subtotal	7207	826	11%
unobserved queries			
U'_{ur}	65	9	14%
U_u	369	44	12%
subtotal	434	53	12%
total	7641	879	12%

(d) xbl (633/10380840)

Figure 7.1: Measurements for Scenario 1, where we divide the groups between those whose queries are monitored at least partially and those whose queries are not monitored at all.

the intermediate blacklist on the groups whose queries are not monitored can reveal the universality of the method. That is, whether the intermediate blacklist generated by the "monitored" groups can detect infected users of other groups.

In Table 7.1a we can see that DNS-based prefiltering reduces the original botnetcc blacklist from 490 elements to only 1. By filtering the flows given this single malicious destination, 40% of the infected users are detected. Moreover, in Table 7.1b, DNS-based prefiltering reduces the original emerging blacklist from 915 elements to 13. These 13 blacklisted elements reveal roughly less than 40% of all the infected users. Since for this blacklist there are hits in the "unmonitored" groups, we can observe that 25% of those are detected, although their queries are not monitored.

The first two blacklists contain IP addresses related to botnet activity, which utilize frequently the DNS. Therefore, as it is expected, the recall rate is higher for those. However, we can clearly see that 40% is the peak that can be achieved. The remaining experiments show a much less recall rate which can be lower than 5%.

The largest experiment is for the xbl blacklist which has more than 10 millions entries. We can see that DNS-based prefiltering reduces the blacklist to only 633 elements. According to Section 5.2, this reduction could lead to a performance increase greater than 300% (from 0.7 to 2.9 million flows processed per second). However, the recall rate is still significantly low, between 11% and 24%, regarding the "monitored" group.

Eventually, we can argue that the hypothesis **Scenario 1a** is strongly validated, while the hypothesis **Scenario 1b** is definitely not validated.

7.4 Results of Scenario 2

The results of the measurements for **Scenario 2** are identical to those of **Scenario 1**; hence, we do not present them. There is a slight increase in the intermediate blacklist (from 633 to 646) for the xbl experiments. However, the percentages of the results remain the same. More details regarding the measurements can be found in Appendix A.

This experiment did not result to the expected increase in the detection rate. The main reason is that malicious IP addresses commonly have short TTL value. As a result, the cache effect is eliminated and the queries appear in the link between the resolver and the authoritative name servers. Eventually, we can argue that, similarly to **Scenario 1**, the hypothesis **Scenario 2**a is validated, while the hypothesis **Scenario 2**b is not. In addition, the hypothesis **Scenario 2**c is not validated, since the detection rate has not been increased.

7.5 Results of Scenario 3

The measurements for **Scenario 3** can be seen in Table 7.2, where we repeat the analysis for each blacklist and each "monitored" group. For this scenario, we compare the flow-based filtering (N) to individual DNS-based filtering (D). We present the detected infected users of each and the true positives (TP), which are common in both. Technically, a DNS-based alert is a true positive only if it initiates a flow within nine minutes. Anything in the flow-based results that was not detected by the standalone DNS-based filtering ($N-TP$) is considered as false negative. Likewise, anything detected by DNS-based filtering and not by flow-based filtering ($D-TP$) is considered as false positive.

By knowing the true positives, false negatives and false positives we can compute the recall (the proportion of the true positives among the true positives and the false negatives) and precision (the proportion of the true positives among all the positive results) rates. Furthermore, we consider the accuracy rate which takes into account the true negatives. True negatives are all the users of the group that do not appear in any of the true positives, false negatives or false positives. The total number of users in each group has been presented in Table 7.1. Specifically, accuracy is the proportion of true results -both true positives and true negatives- among all the users.

blacklist	group	N	D	TP	Recall	Precision	Accuracy
botnetcc	$U_{mr'}$	4	4	1	25%	25%	99.96%
	U_{mur}	1	2	1	100%	50%	99.91%
	U_{lr}	0	1	0	-	0%	99.63%
emerging	$U_{mr'}$	12	8	1	8%	12%	99.88%
	U_{mur}	2	2	1	50%	50%	99.81%
	U_{lr}	9	2	2	22%	100%	97.42%
sbl_drop	$U_{mr'}$	568	36	23	4%	64%	96.36%
	U_{mur}	121	6	6	5%	100%	89.18%
	U_{lr}	57	4	1	2%	25%	78.23%
xbl	$U_{mr'}$	6183	899	188	3%	21%	56.22%
	U_{mur}	828	77	22	3%	29%	19.00%
	U_{lr}	196	44	17	9%	39%	23.99%

Figure 7.2: Measurements for Scenario 3. The cardinalities of the U'_{mr}, U_{mur} and U_{lr} groups are 15319, 1063 and 271, respectively

The recall rates are low and close to those of the measurements for **Scenario 1**. The precision rates are generally higher but still not satisfactory. Finally, the accuracy rate is very high for the small blacklist but only because there are not a lot of hits (i.e. a huge percentage of true negatives). For the largest xbl blacklist, the accuracy can be lower than 20%. Eventually, we can argue that none of the hypothesis is validated.

Chapter 8

Conclusions

8.1 Threats to validity

We recognize two threats to validity regarding the dataset and one regarding the measurements. First of all, the experiments have been performed using a single dataset which does not prove the universality of the results. Ideally, the experiment could be repeated on a dataset obtained from another entity. Similarly, the limited two-hour time window may have influenced the results. The reason is that we cannot observe the behavior of malicious activity for a longer time window or at different time, like, for instance, at night.

Another threat to validity is that a blacklist is the exclusive indication of malicious activity. Although they are frequently updated, a blacklist can never be complete. There are always malicious entities that are not blacklisted. In addition, there is a slight possibility that a blacklisted element is not malicious. Further rigorous manual investigation is required in order to prove the results.

8.2 Summary

In this section, we present the conclusions which are linked to the research questions.

Can we dynamically detect new malicious activities by monitoring DNS traffic? Our investigations show that dynamically detecting new malicious activities by monitoring DNS traffic remains a difficult task. We have presented various methods that detect malicious activities and, especially, previously unknown malicious domain names. The examined methods show limited real-life applications and performance. Only a single system (EXPOSURE) has evolved from prototype, whose results were available in an on-line service for a long period of time, before it was shut down. Under private communication we were informed that the reason for the shut down was the complains about false positives.

Can we effectively detect infected computers within a network by taking advantage of DNS-based monitoring? **Scenario 1** and **Scenario 2** examined a single detection system that combines DNS-based and flow-based monitoring. This method performs DNS-based prefiltering in order to increase the processing time of flow-based filtering. Although performance-wise the results are good, the system fails to achieve a sufficient level of accuracy. Our measurements show that the infected computers that are detected are at most 40% of the actual infected.

Can DNS-based monitoring be a valid standalone detection method? The results of **Scenario 3** show that standalone DNS-based detection does not have adequate accuracy. The recall rate is often lower than 20% (between 4% and 19% for the experiments with at least two true

positives) which means that more than 80% of the infected users are not detected. The precision rate is higher (between 40% and 100% for the experiments with at least two true positives) but it often remains under 51% which cannot be considered sufficient.

8.3 Future work

In this section, we present some recommendations for future work.

More precise measurements for the standalone DNS-based detection (Scenario 3). The measurements in this report have been based on blacklists maintained by reliable organizations. Nevertheless, no dynamic blacklist can be complete. There always will be malicious elements that are not listed in any blacklist. The evaluation of **Scenario 3** assumes that the results of flow-based monitoring are the correct. If there is a DNS-based alert but not a flow-based, the former is marked as false positive. However, it could actually be a true positive that does not appear in the flow-based results.

This can happen if a single response contains multiple IP addresses from which one is blacklisted. The response triggers a DNS-based alert that this node is infected (requested a blacklisted element). Then if the node connects to an IP address from the response, other than the blacklisted, a flow-based alert is not triggered. However, it is highly likely that the other IP addresses in the same response are malicious too. Manual investigation is necessary to verify if the DNS-based alert is actually a false positive or a missed true positive.

Evaluation of standalone DNS-based detection based on domain names. In this report, we have focused on IP-based detection. In addition, filtering blacklisted domain names is used as a detection mechanism. Future work can examine the precision of DNS-based detection given a domain name blacklist.

The measurement of the false positive rate could be interesting. False positives are the DNS-based alert that have not initiated a flow. For instance, if a node sends a query and receives an answer for "*malicious.com*" then there is a DNS-based alert. If a flow has been established to one of the IP addresses in the answer, the DNS-based alert is a true positive. Otherwise, if no flow has been observed, the alert has been a false positive.

Interestingly, it is not possible to calculate the false negative rate. A domain name blacklist can be applied neither directly nor indirectly on flows. As it is mentioned in Section 3.4.1, converting by resolving a list of domain name to a list of IP addresses is not 100% accurate, especially if attackers maintain these domain names. The list of IP addresses could vary depending on the time of the resolution and could always contain deliberate false positives.

Blacklist expansion using flow information. It is common for a single DNS response to contain multiple IP addresses. Even if one is blacklisted, it is risky to instantly blacklist the rest of them. As we have previously mentioned an attacker could deliberately put legitimate IP addresses next to the malicious. This would result in blacklisting legitimate services if this process was autonomous.

A way to expand a blacklist could consider the DNS responses and flow information. Network flows record not only the source and destination of a connection but also the type, such as port number, number of packets and protocol (UTP or TCP). By observing the malicious destinations one could create their profile (e.g., 10-12 packets via TCP on port number 1025). Any IP address is matched against the malicious profile if both IP addresses appear in the same response. This expansion can be evaded if the attacker emulates the behavior of the legitimate IP address. However, this would at least reduce the flexibility of the attack.

Bibliography

[1] P.V. Mockapetris. Domain names: Concepts and facilities. RFC 882, November 1983. Obsoleted by RFCs 1034, 1035, updated by RFC 973. 3

[2] P.V. Mockapetris. Domain names: Implementation specification. RFC 883, November 1983. Obsoleted by RFCs 1034, 1035, updated by RFC 973. 3

[3] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (INTERNET STANDARD), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936. 5, 6

[4] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (INTERNET STANDARD), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604. 5, 6, 7, 41

[5] Security Whitepaper, The Role of DNS in Botnet Command & Control. Technical report, OpenDNS, 2012. http://info.opendns.com/rs/opendns/images/OpenDNS_SecurityWhitepaper-DNSRoleInBotnets.pdf. 10, 11

[6] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton. Real-time detection of fast flux service networks. In *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, pages 285–292, March 2009. 11

[7] Brett Stone-gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your Botnet is My Botnet : Analysis of a Botnet Takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, volume 97 of *UCSB Technical Report*, pages 635–647. Univesity of California, Santa Barbara, ACM, 2009. 11

[8] Martin Grill, Ivan Nikolaev, Veronica Valeros, and Martin Rehak. Detecting dga malware using netflow. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 1304–1309, May 2015. 12

[9] Hachem Guerid, Karel Mittig, and Ahmed Serhrouchni. Privacy-preserving domain-flux botnet detection in a large scale network. *2013 5th International Conference on Communication Systems and Networks, COMSNETS 2013*, 2013. 12

[10] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 635–647, New York, NY, USA, 2009. ACM. 12

[11] A. Sperotto and A. Pras. Flow-based intrusion detection. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 958–963, May 2011. 12, 13

[12] G. Androulidakis and S. Papavassiliou. Improving network anomaly detection via selective flow-based sampling. *Communications, IET*, 2(3):399–409, March 2008. 12

[13] Milan Čermák, Pavel Čeleda, and Jan Vykopal. Detection of DNS Traffic Anomalies in Large Networks. In *Advances in Communication Networking*, pages pp 215–226. Springer International Publishing, 2014. 12, 13

[14] Jessica Steinberger, Lisa Schehlmann, Sebastian Abt, and Harald Baier. Anomaly detection and mitigation at internet scale: A survey. In Guillaume Doyen, Martin Waldburger, Pavel eleda, Anna Sperotto, and Burkhard Stiller, editors, *Emerging Management Mechanisms for the Future Internet*, volume 7943 of *Lecture Notes in Computer Science*, pages 49–60. Springer Berlin Heidelberg, 2013. 13

[15] Leyla Bilge, Engin Kirda, Christopher Kruegel, Marco Balduzzi, and Sophia Antipolis. EXPOSURE : Finding Malicious Domains Using Passive DNS Analysis. *Ndss*, pages 1–17, 2011. 13, 17

[16] B. Claise, B. Trammell, and P. Aitken. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011 (INTERNET STANDARD), September 2013. 12

[17] Jonathan M Spring and Carly L Huth. The Impact of Passive DNS Collection on End-user Privacy. *Satin*, pages 1–11, 2012. 13, 18

[18] Roberto Perdisci, Igino Corona, and Giorgio Giacinto. Early detection of malicious Flux networks via large-scale passive DNS traffic analysis. *IEEE Transactions on Dependable and Secure Computing*, 9:714–726, 2012. 14, 18

[19] Reza Sharifnya and Mahdi Abadi. A novel reputation system to detect DGA-based botnets. In *Computer and Knowledge Engineering (ICCKE), 2013 3th International eConference on*, number Iccke, pages 417–423. 2013. 14

[20] Daniel J Solove. Understanding Privacy. *Harvard University Press, May 2008; GWU Legal Studies Research Paper No. 420; GWU Law School Public Law Research Paper No. 420*, (May), 2008. 17

[21] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *Communications Surveys Tutorials, IEEE*, 16(4):2037–2064, Fourthquarter 2014. 17

[22] Yacin Nadji, M Antonakakis, and R Perdisci. Beheading hydras: performing effective botnet takedowns. *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 121–132, 2013. 11

[23] Hyunsang Choi, Hanwoo Lee, Heejo Lee, and Hyogon Kim. Botnet detection by monitoring group activities in DNS traffic. In *CIT 2007: 7th IEEE International Conference on Computer and Information Technology*, pages 715–720, 2007. 15, 20, 21, 22, 23

[24] Hyunsang Choi, Heejo Lee, and Hyogon Kim. Botgad: Detecting botnets by capturing group activities in network traffic. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, pages 2:1–2:8, New York, NY, USA, 2009. ACM. 14, 20, 22, 23

[25] Hyunsang Choi and Heejo Lee. Identifying botnets by capturing group activities in DNS traffic. *Computer Networks*, 56:20–33, 2012. 17, 20, 22, 23, 27

[26] Kazumichi Sato, Keisuke Ishibashi, Tsuyoshi Toyono, and Nobuhisa Miyake. Extending black domain name list by using co-occurrence relation between dns queries. In *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, LEET'10, pages 8–8, Berkeley, CA, USA, 2010. USENIX Association. 20, 22, 23

[27] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a Dynamic Reputation System for DNS. *USENIX Security'10: Proceedings of the 19th USENIX conference on Security*, pages 1–17, 2010. 12, 16, 20, 22, 24, 25

[28] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou II, and David Dagon. Detecting Malware Domains at the Upper DNS Hierarchy. In *USENIX Security Symposium.*, volume 11, pages 1–16, 2011. 20, 22, 25

[29] Leyla Bilge, S Sen, and D Balzarotti. EXPOSURE: a passive DNS analysis service to detect and report malicious domains. *ACM Transactions on ...*, V(4), 2014. 1, 20, 22, 26

[30] Samuel Marchal, Jerome Francois, Cynthia Wagner, Radu State, Alexandre Dulaunoy, Thomas Engel, and Olivier Festor. DNSSM: A large scale passive DNS security monitoring framework. In *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, pages 988–993, 2012. 1, 11, 20, 22, 26

[31] Jehyun Lee and Heejo Lee. GMAD: Graph-based malware activity detection by DNS traffic analysis. *Computer Communications*, 49:33–47, 2014. 1, 10, 14, 15, 20, 22, 27, 28

[32] Manos Antonakakis and Roberto Perdisci. From throw-away traffic to bots: detecting the rise of DGA-based malware. *Proceedings of the 21st USENIX Security Symposium*, page 16, 2012. 1, 12, 20, 22, 28

Appendix A

Detailed experiment results

Table A.1 presents a more comprehensive view of the results of **Scenario 1**. A part of this information has been presented in Section 7.3. In detail, Table A.1 contains the results for each blacklist and for each user group. The column N contains the original results, that are produced by filtering the network flows given the full blacklist. This column is divided into three parts: the total number of hits, the unique source IP addresses and the unique destination IP addresses. The column blacklist shows the size of the intermediate blacklist out of the total blacklisted elements of each blacklist. The intermediate blacklist is produced by the DNS-based prefiltering process. Furthermore, the column P contains the results of **Scenario 1**, that are produced by filtering the network flows given the intermediate blacklist. This column is divided into three parts, similarly to the N column. Eventually, we present the recall rate.

Likewise, Table A.2 contains the full results of **Scenario 2**. The structure of this table is the same as of Table A.1.

In addition, more detailed results of **Scenario 3** can be found in Table A.3. This table shows the flow-based results (N), the individual DNS-based results (D) and the true positives of DNS-based detection (P). Eventually, we present the recall, precision and accuracy rates based on the detected infected users (i.e. unique source IP addresses).

	group	N total	N uniq src	N uniq dst	blacklist	P total	P uniq src	P uniq dst	Recall
botnetcc	U	245	5	4	1/490	3	2	1	40%
	U'_{mr}	244	4	4		2	1	1	25%
	U'_{ur}	0	0	0		0	0	0	-
	U_{mur}	1	1	1		1	1	1	100%
	U_{lr}	0	0	0		0	0	0	-
	U_u	0	0	0		0	0	0	-
emerging	U	1539	27	25	13/915	207	10	6	37%
	U'_{mr}	528	12	12		26	3	2	25%
	U'_{ur}	3	2	2		2	1	1	50%
	U_{mur}	47	2	2		31	1	1	50%
	U_{lr}	742	9	13		148	5	4	56%
	U_u	219	2	4		0	0	0	0%
emerging_ext	U	15	4	1	0/23	No DNS-based alert			-
	U'_{mr}	0	0	0					-
	U'_{ur}	0	0	0					-
	U_{mur}	0	0	0					-
	U_{lr}	15	4	1					-
	U_u	0	0	0					-
sbl_drop	U	2671	783	405	32/712	193	31	18	4%
	U'_{mr}	1348	568	122		124	24	14	4%
	U'_{ur}	13	2	3		0	0	0	0%
	U_{mur}	399	121	85		64	6	13	5%
	U_{lr}	794	57	265		5	1	2	2%
	U_u	117	35	11		0	0	0	0%
sbl_edrop	U	234	52	46	1/71	Empty set			0%
	U'_{mr}	63	21	19					0%
	U'_{ur}	0	0	0					-
	U_{mur}	28	10	10					0%
	U_{lr}	142	20	21					0%
	U_u	1	1	1					0%
xbl	U	1084289	7641	234295	633/10380840	3928	879	185	12%
	U'_{mr}	398200	6183	101168		2833	677	152	11%
	U'_{ur}	12978	65	4402		27	9	5	14%
	U_{mur}	157792	828	44835		361	102	41	12%
	U_{lr}	505712	196	145083		370	47	37	24%
	U_u	9607	369	1873		337	44	15	12%

Table A.1: Detailed results of **Scenario 1**.

	group	N total	N uniq src	N uniq dst	blacklist	P total	P uniq src	P uniq dst	TP
botnetcc	U	245	5	4	1/490	3	2	1	40%
	U'_{mr}	244	4	4		2	1	1	25%
	U'_{ur}	0	0	0		0	0	0	-
	U_{mur}	1	1	1		1	1	1	100%
	U_{lr}	0	0	0		0	0	0	-
	U_u	0	0	0		0	0	0	-
emerging	U	1539	27	25	13/915	207	10	6	37%
	U'_{mr}	528	12	12		26	3	2	25%
	U'_{ur}	3	2	2		2	1	1	50%
	U_{mur}	47	2	2		31	1	1	50%
	U_{lr}	742	9	13		148	5	4	55%
	U_u	219	2	4		0	0	0	0%
sbl_drop	U	2671	783	405	27/712	193	31	18	4%
	U'_{mr}	1348	568	122		124	24	14	4%
	U'_{ur}	13	2	3		0	0	0	0%
	U_{mur}	399	121	85		64	6	13	5%
	U_{lr}	794	57	265		5	1	2	2%
	U_u	117	35	11		0	0	0	0%
xbl	U	1084289	7641	234295	645/10380840	5417	875	192	11%
	U'_{mr}	398200	6183	101168		4302	670	159	11%
	U'_{ur}	12978	65	4402		27	9	5	14%
	U_{mur}	157792	828	44835		340	101	40	12%
	U_{lr}	505712	196	145083		385	48	37	24%
	U_u	9607	369	1873		363	47	18	13%

Table A.2: Detailed results of **Scenario 2**.

group		N			D			P			Recall	Precision	Accuracy
		total	uniq src	uniq dst	total	uniq src	uniq dst	total	uniq src	uniq dst			
botnetcc	U'_{mr}	244	4	4	12	4	1	6	1	1	25%	25%	99.96%
	U_{mur}	1	1	1	3	2	1	1	1	1	100%	50%	99.91%
	U_{lr}	0	0	0	2	1	1	0	0	0	-	0%	99.63%
emerging	U'_{mr}	528	12	12	11	8	7	2	1	1	8%	12%	99.88%
	U_{mur}	47	2	2	4	2	2	1	1	1	50%	50%	99.81%
	U_{lr}	742	9	13	6	2	6	7	2	2	22%	100%	97.42%
sbl_drop	U'_{mr}	1348	568	122	172	36	24	141	23	13	4%	64%	96.36%
	U_{mur}	399	121	85	67	6	21	73	6	13	5%	100%	89.18%
	U_{lr}	794	57	265	24	4	13	11	1	2	2%	25%	78.23%
xbl	U'_{mr}	398200	6183	101168	6672	899	588	304479	186	144	3%	21%	56.20%
	U_{mur}	157792	828	44835	239	77	67	645	22	28	3%	29%	19.00%
	U_{lr}	505712	196	145083	138	44	53	400	17	14	9%	39%	23.99%

Table A.3: Detailed results of **Scenario 3**.