University of Twente

Master Thesis

Proactively detecting crafted domains using active DNS measurements

Calvin Hendriks

Date:	June 30, 2020
Committee:	dr. A. Sperotto O.I. van der Toorn MSc dr.ir. M. Jonker dr. D. Bucur
Faculty:	Electrical Engineering, Mathematics and Computer Science (EEMCS)
Chair:	Design and Analysis of Communication Systems (DACS)

Abstract

Distributed Denial-of-Service (DDoS) attacks are a major threat in today's Internet landscape, affecting all kinds of services, from banks and telecommunication providers to social media platforms and gaming servers. In a Domain Name System (DNS) amplification attack, a popular type of DDoS attack, adversaries exploit the DNS infrastructure in order to amplify minor queries into becoming large responses towards their victim. In a process called crafting, attackers register domains and inflate them with different kinds of Resource Records (RRs) in order to guarantee a large response. Existing mitigation methods and previous studies often rely on passive DNS data or network packets from historic attacks in order to mitigate the effects of DNS amplification attacks. This prevents those methods from detecting newly crafted domains at an early stage. By combining *active* DNS measurements that cover over 60% of the DNS namespace with machine learning, we are able to detect crafted domains proactively. Our results show that the proposed method can detect over 92% of the crafted domains without miss-classifying any benign domain. Furthermore, this approach allows for a detection of crafted domains up to 540 days before they are misused in an attack.

Contents

List of Abbreviations

AS Autonomous System.

BGP Border Gateway Protocol.

CCC Cambridge Cybercrime Centre.

CDF Cumulative Density Function.

DDoS Distributed Denial-of-Service.

DNS Domain Name System.

DNSSEC Domain Name System Security Extensions.

DoS Denial-of-Service.

DPS DDoS Protection Service.

FN False Negative.

FP False Positive.

IP Internet Protocol.

ISP Internet Service Provider.

KCV K-fold Cross-Validation.

RR Resource Record.

RRL Response Rate Limiting.

TCP Transmission Control Protocol.

TLD Top-Level Domain.

TP True Positive.

TSC Time Series Classification.

UDP User Datagram Protocol.

1 Introduction

Distributed Denial-of-Service (DDoS) attacks are evolving both in capacity and frequency. In 2007, the largest DDoS attack was 24 Gbps [1] while more modern attacks, such as the 2018 attack on software development platform GitHub, have already crossed the 1 Tbps mark [2]. Successful DDoS attacks on public companies can cost millions in reduced stock value, damage to brand reputation and lost business revenue. DDoS attacks are not only used against public companies for financial motive, they can also be used against critical infrastructures in order to disrupt society. While this may seem like a scene from a futuristic movie, real-life examples include attacks on public transport [3], healthcare, and the telecommunications sectors [4]. Even though these attacks can often be mitigated by the use of DDoS Protection Service (DPS) providers such as Cloudflare or Akamai - which at the time of writing have a network capacity of 30 and 70 Tbps respectively [1][2] - it is estimated that the Internet Protocol version 4 (IPv4) can support DDoS attacks up to at least 108 Tbps [5]. A specific variant of DDoS attacks, the Domain Name System (DNS) amplification attack, has been used in record breaking DDoS Attacks, such as the 2013 attack on anti-spam list provider Spamhaus. Although the highest capacity attack of 1.7 Tbps was executed using another type of amplification factor [6], DNS amplification is still a widely used form of DDoS attacks among DDoS-for-hire platforms (also called booters). In their Q1 2019 DDoS Threat report, Nexusguard claims that DNS Amplification accounts for 42.94% of sighted DDoS attacks, showing that even though DNS amplification is not the record breaking form of attack it once was, it certainly is still the most popular one [7].

In a DNS Amplification attack, an adversary uses a combination of Internet Protocol (IP) address spoofing and open DNS servers (either an open DNS resolver or an authoritative name server) to generate a large stream of packets towards the intended victim. It does so by altering the source IP address in the header of the requests to the IP address of the victim and sending it to the DNS server. The server will now send the response back to the victim. Since this response can be up to 104 times larger (hence amplification) than the query, the adversary can generate a lot of attack traffic with a relatively small bandwidth requirement [8]. As previously noted, adversaries performing a DNS amplification attack exploit two concepts; IP address spoofing and the availability of open DNS resolvers. To prevent DNS amplification attacks from happening, these two enablers need to be targeted. IP address spoofing can easily be prevented if network operators implement network ingress filtering (described in BCP 38 [9]) and the open DNS resolvers can be altered to respond only to trusted sources. The problem with solving both of these concepts is the incentive structure of the internet. A network deploying ingress filtering protects its peers and does not receive any benefits itself. The same is true for blocking untrusted sources at a recursive DNS server.

Another solution to the DDoS problem that was already mentioned are DPS providers. When such a service is deployed, all incoming traffic is diverted to the (much more capable) infrastructure of the DPS provider were it is filtered before being returned to the client. A 2016 study on the adoption of these DPS providers among the domains in the generic Top-Level Domains (TLDs) .com, .net and .org - which at the time contained almost 50% of the global domain name space - found a growth in their adoption of 1.24x over 1.5 years [10]. Compared to the overall expansion of the domain space of just 1.09x during that time frame, this shows that the adoption of DPS providers is on the rise. However, this solution is of a reactive nature, as it only becomes effective once the attack is already in progress. While highly effective at combating the symptoms of a DDoS attack, it does not solve the problem.

Fortunately, adversaries who are preparing a sophisticated attack are often forced to expose some of their infrastructure to the outside world, which gives us a chance to detect it early in the attack's life cycle. In our case, this is due to the fact that the adversary wants a guaranteed large response to his DNS query, resulting in a maximum amplification. In order to achieve this in DNS, adversaries often make use of a so-called "crafted domain"; By registering a domain and inflating it by adding different kinds of records, such as A, AAA, MX or TXT records, the attacker gets the needed amplification factor. [8] [11]. This could provide us with an opportunity to identify the threat before it takes place.

[1] https://www.cloudflare.com/performance/

[2] https://www.akamai.com/us/en/resources/ddos-attacks.jsp

In line with the TIDE project [11], using a data set of active DNS measurements [12], we research the detection of malicious domains that are specifically crafted with the intention to be used in DNS amplification attacks. Furthermore, the goal of this study is to design a method to proactively detect these domains before they can be misused in an attack. To help achieve this goal, the following research questions have been defined:

- **RQ1:** What are the defining characteristics of domains misused in DNS amplification attacks?
- **RQ2:** What is the definition of a crafted domain?
- **RQ3:** How can the behaviour of crafted domains be automatically detected?
- **RQ4:** Does the presented method detect crafted domains before they are misused? If yes, what is the time advantage?

2 Background

2.1 Domain Name System

Whereas humans connect to an internet service using domain names (e.g. facebook.com), network devices make use of IP addresses. The DNS translates domain names to IP addresses, making it unnecessary for humans to remember these long addresses. A domain name generally consists of multiple parts. The domain name space is hierarchically divided into sub domains, with the root domain at the top. Beneath this root domain are the TLDs followed by second-level domains and finally lower-level domains (or subdomains)(Fig. 2.1).

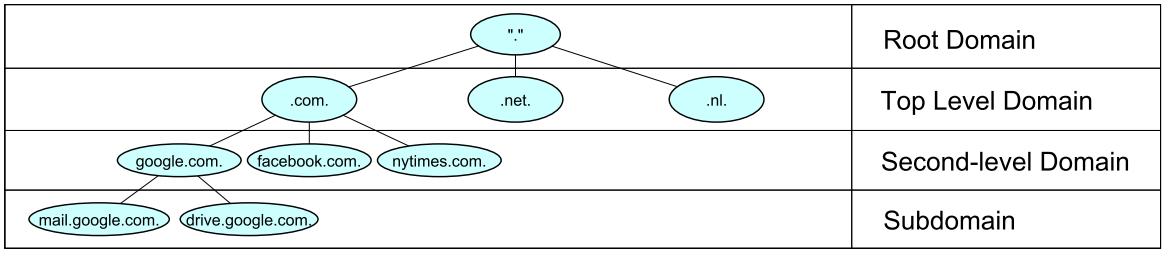


Figure 2.1: DNS Hierarchy

2.1.1 Recursive DNS resolver

When a user types the domain name "google.com." in their browser, the computer will contact a preconfigured recursive DNS resolver (most often the one assigned by an Internet Service Provider (ISP)). As the name implies, this server will recursively contact the correct nameservers to find the IP address of "google.com.". If the domain is not yet in the the resolver's cache, the resolver will first contact one of the DNS root nameservers, which will reply with the IP address of the appropriate TLD nameserver (in this case .com). The resolver will then contact the TLD server for the IP address of google.com.'s authoritative nameserver. Next, the resolver queries the authoritative nameserver for the IP address of "google.com". The response is then sent back to the user, who can now visit the webpage.

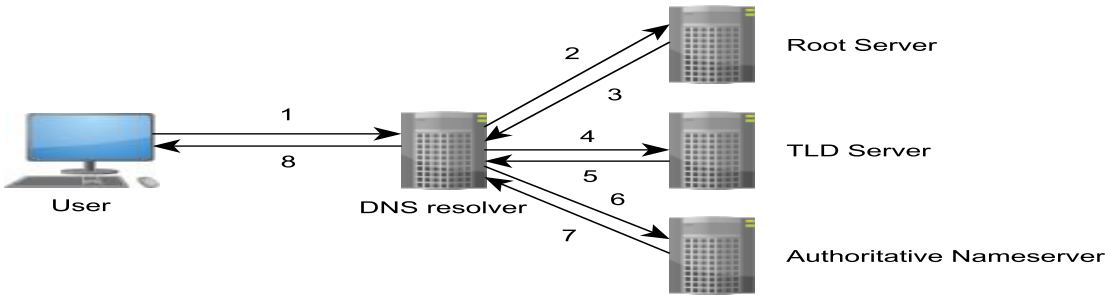


Figure 2.2: recursive DNS resolver

2.1.2 Zone file

A DNS zone is a portion of the DNS namespace that is managed by a company or administrator [13]. All of the information about a zone is stored in a DNS zone file and hosted on a nameserver. This

information is mostly structured into Resource Records (RRs). Each line represents a RR and can map different attributes. For example, an A record maps a domain name to an IPv4 address, an AAAA record maps a domain name to an Internet Protocol version 6 (IPv6) address, and an MX record maps a mail server to a domain name. To reuse the example from above, the user's web browser wants to know the value of the A record of the zone file that the domain "google.com." belongs to.

2.2 Denial of Service

A Denial-of-Service (DoS) Attack tries to prevent legitimate users from accessing a network resource, such as a website or e-mail service, by overloading the network, memory or processor of said resource. While an adversary can launch such an attack from their computer, a more common approach is a DDoS attack. In a DDoS attack, the adversary orchestrates a network of compromised computing devices to launch such an attack simultaneously. The advantage of this approach is twofold: first, the usage of compromised devices makes it more difficult for law enforcement to track down the attacker; secondly, it enables a much larger and more disruptive attack [14].

2.2.1 Semantic vs. Volumetric

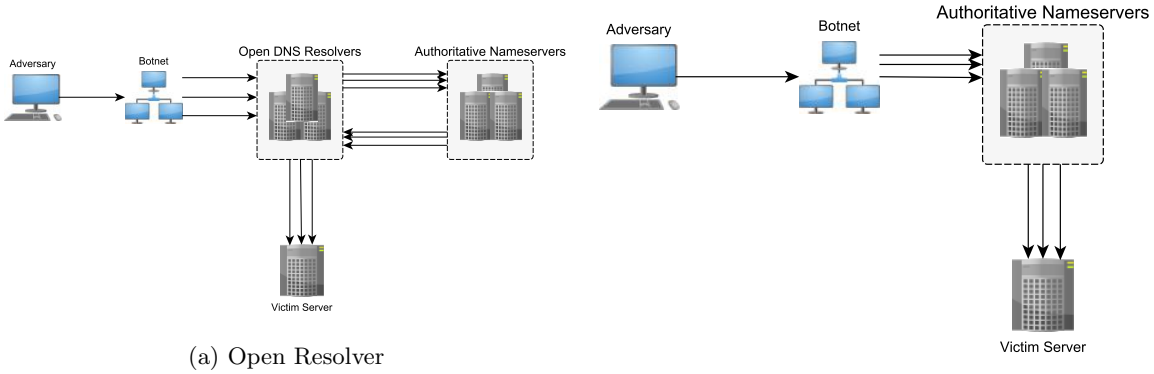
Although there are many variations of DDoS attacks, Mirkovic and Reiher [15] and Specht and Lee [14] both made a clear distinction between two categories. First, there are the so called so called semantic (or resource depleting) DDoS attacks in which the adversary tries to consume excessive amounts of resources by exploiting certain features or bugs of protocols or applications running on the victims servers. A well known example is a SYN flood where an adversary abuses the three-way-handshake in the Transmission Control Protocol (TCP) protocol to claim resources by sending a flood of SYN request to the server which he will never answer with an ACK message. Due to the nature of these attacks, they can be mitigated relatively easily by changing the exploited protocol or through filtering. The second category, however, is harder to mitigate. The so called volumetric (or brute-force / bandwidth depleting) DDoS attacks simply send large volumes of (often seemingly legitimate) traffic to the victim in order to saturate the network bandwidth, preventing access to legitimate users.

2.2.2 DNS Amplification attack

A DNS amplification attack is a specific kind of volumetric DDoS attack that abuses DNS in order to send large amounts of traffic towards a victim. The attack is performed by a using a technique called IP address spoofing, and is only possible when a DNS query is performed using the User Datagram Protocol (UDP). Although the DNS protocol runs on top of both UDP and TCP, most servers prefer UDP since DNS servers need to handle a lot of requests from different clients and a connection oriented protocol such as TCP would take up too many resources. This is because TCP performs a three-way handshake for each connection and retransmissions for unacknowledged packets to provide a reliable transport of data. UDP on the other hand, is a non-reliable and connection less protocol with minimum protocol mechanisms. It has no connection setup which results in a faster protocol, less network traffic and no resource consumption on the receiving machine.

In case of a DNS amplification attack, the adversary forges the source IP address header of the IP packet that contains his DNS query. A botnet under the adversary's control sends the query to a list of open DNS resolvers or authoritative name servers, for which the packet looks like any other packet. In case of the open DNS resolvers, the request is recursively processed until the domain name (or other resource) is found. The result, many times larger than the request, is then send to the forged IP address that was found in the IP packet (Fig. 2.3a). In case the adversary makes use of the Authoritative nameservers directly, the DNS resolver is skipped and the nameserver that hosts the correct zone file is contacted by the botnet directly (Fig. 2.3b). An increasing amount of resolver operators are following the RFC5358 access restriction guidelines, which aims at preventing the use of recursive DNS servers in a DDoS attack. It suggests that operators of these servers serve only the intended clients by validating the source IP address, discriminating between incoming interfaces or using signed queries. This has

caused attackers to change their approach and switch towards the use of authoritative name servers, which cannot follow these guidelines by design [16]. Traditionally, DNS limited the size of UDP replies to 512 bytes and switched to the TCP protocol for larger answers. This way, a high amplification factor could not be reached by adversaries since the random Initial Sequence Number (ISN) in TCP's three-way handshake is sent to the victim, preventing an attacker from establishing a connection using a spoofed address. However, many DNS servers have implemented EDNS0, a DNS extension that allows UDP responses up to 4096 bytes as this is needed for the implementation of Domain Name System Security Extensions (DNSSEC) [17].



(a) Open Resolver

(b) Auth. Nameserver

Figure 2.3: DNS Amplification attacks

2.3 Time Series classification

As mentioned in the introduction, attackers are known to register and inflate domains with records in order to guarantee a large response to a query. Using the OpenINTEL measurements on domains, classification algorithms could help us detect specific these domains that are specifically created to be used in an amplification attack. However, since the measurements are performed on a daily basis, the attributes in the data set are ordered chronologically. As a result, the classification problem becomes a Time Series Classification (TSC) problem, since discriminatory features in the data set (which are used to classify instances) may be dependent on this ordering [18]. TSC techniques can be divided in two categories: feature-based and distance-based (also called instance-based) [19]. Feature-based TSC methods extract features from the time series in the training set before the actual classification. Examples include mean, variance and entropy. This featured vector describes the properties of the time series and is used to train a classifier which can be used to classify new unseen time series. Since this feature vector is much smaller than the time series it represents, training times are reduced. However, this approach comes with some drawbacks as well. The feature extraction could be time consuming and require resources in the form of experts if done manually. Furthermore, the feature vector may not correctly describe the temporal structure of the original time series and therefore lose discriminatory information needed for classification. Distance-based methods define a distance measure that takes into consideration the temporal nature of time series. The distance between new time series and the labeled time series in the training set is calculated and fed to a distance-based classifier [20]. A popular method is to use a Nearest Neighbor classifier that bases its similarity on the Euclidean Distance or the Dynamic Time Warping (DTW) measure. However, the downside to this approach is that Nearest Neighbor is a lazy-learning algorithm, meaning that is has to calculate the distance to each instance in the training set for every new instance, resulting in high computational complexity.

3 Related Work

3.1 DDoS Mitigation

Since DDoS is such a big threat to the Internet infrastructure, a lot of research has been done on defending against DNS amplification attacks. In this section, we present related work that focusses on reducing the negative effects of a volumetric DDoS attack once it has been initiated. Geva, Herzberg and Gev [21] categorized the type of defenses against volumetric DDoS attacks into four groups.

First, rate limiting can be implemented. If a flow of packets is suspected of being part of an attack, its rate is limited at the router level instead of filtered. Existing solutions are Stateless Internet Flow Filter (SIFF), Traffic Validation Architecture (TVA), Proactive Surge Protection (PSP) and Backward Traffic Throttling (BTT). While some of these solutions can be quite effective, they either require changes to routers or cooperation between Autonomous Systems (ASes), both of which are a problem as these solutions protect external parties and therefore exclude the incentive for ASes.

Next, detouring and absorbing is a technique often used to counter DDoS attacks. Detouring overlays such as the Resilient Overlay Networks (RON) introduced by Andersen et al. [22] work by bypassing the Border Gateway Protocol (BGP) preferred route. However, this only works when there are different routes available, meaning that if a high percentage of ASes are congested, the chances of reaching the server are diminished. Absorption clouds such as Cloudflare reroute their customer's traffic to their high-capacity network by configuring the BGP or DNS protocol and scrub it of any malicious packages [10]. Netscout's 14th Annual Worldwide Infrastructure Security Report shows that 43% of the respondents make use of a cloud based DDoS mitigation service [23], reflecting the popularity of this mitigation technique.

Finally, there is a mechanism called breakthrough, which is mainly aimed at avoiding self-created volumetric DDoS attacks. By switching from TCP to a connectionless protocol such as UDP whenever there is a timeout, solutions such as QoSoDoS try to break through the congested network [24].

MacFarland et al. [25] look into the amplification ratio that can be accomplished using authoritative name servers and propose a mitigation strategy that organizations can deploy. The proposed strategy involves the outsourcing of the authoritative name server to an off-site DNS hosting service such as Cloudflare and blocking incoming DNS packets at the ISP level. To continue the use of a DNS resolver, an off-premises resolver should be created and made accessible through a tunnel. While this strategy would protect the deploying organization's network, it does not remove the root cause of the issue. In a report by the ICANN Security and Stability Advisory Committee (SSAC), a number of recommendations are offered to prevent DDoS Attacks leveraging the DNS infrastructure (i.e. DNS amplification attacks) [26]. One of the recommendations is that organizations running an authoritative name server employ a response rate limit. US Cert advises limiting the responses sent to the same client within one second to a maximum of 5 [27]. Rozekrans et al. [16] list some defense mechanisms and decide to focus on Response Rate Limiting (RRL). They conclude that RRL is an effective method against simple DNS amplification attacks. However, it does not protect against more sophisticated amplification attacks (i.e. using more name servers or using a zone file with many resolvable domain names).

While all of the mentioned mitigation techniques may reduce the problem of amplification DDoS attacks, they share a common flaw; none of them do so before the attack has been initiated. Instead, they only become effective once the attack is ongoing. In our research, we try to proactively identify the crafted domains used by adversaries, so that appropriate measures can be taken before they are exploited for an attack.

3.2 (Amplification) DDoS prevention

Although the aforementioned solutions can be effective, they are either reactive or require incentive to implement. Even though the solutions in this section still have an incentive problem, they are of a pro-active nature and prevent instead of mitigate reflection and/or amplification DDoS attacks.

While three out of four categories listed by Geva et al. mitigate the symptoms of a DDoS attack, the last category actually prevents a volumetric attack from taking place. This category of solutions, called filtering, can take place at different locations but is most effective if it occurs before the the congested link, since the victim often does not have the capacity to absorb the attack. An example of filtering is BCP38 or RFC2827 which recommend that ISPs employ ingress filtering, a technique that filters packets with source IP address headers external to that network. While many ISPs and ASes already implemented BCP38, results from the Spoofer Project [28] show that approximately 15% of IPv4 addresses can send spoofed packets. Although this might not seem like a large amount, the first and second most frequently occurring type of DDoS attack in the Nexusguard report are DNS and NTP amplification attacks respectively, showing that this small percentage of addresses is still large enough to cause major problems. In addition to BGP38, Geva et al. talk about Access Control Lists (ACLs), another filtering technique. While effective, it consumes significant amounts of memory and CPU usage. Finally, Remote-Triggered Blackhole is a filtering mechanism that uses fewer resources. This mechanism however is more strict and might filter non-attack traffic as well. This could make the attacked service inaccessible to innocent third parties, which was the the initial goal of the attacker.

The SSAC report mentioned earlier also recommends that network operators help prevent IP source address forgery and points to the implementation of BCP38. Next, they recommend that recursive DNS servers (i.e. resolvers) should restrict access to external sources and should only accept DNS queries from trusted sources in order to reduce the amplification vectors on the internet. Both of these recommendations have been discussed before and even though they would indeed solve the problem completely, the parties responsible lack an incentive to do so.

3.3 Malicious domain detection

A number of studies have been conducted on finding malicious domains using DNS data. This DNS data can either come from passively listening to queries send to DNS servers, or more active DNS data, such as a TLD zone file snapshot, registration information or resource records from a set of domain names resolved on a regular basis.

3.3.1 Passive DNS

The concept of passive DNS (pDNS) was first introduced by Weimer in 2005 [29]. His implementation, called dnslogger, monitors DNS queries and responses between a recursive resolver and authoritative names servers. The concept of pDNS data has been used in the past to detect different kind of malicious domains. Bilge et al. [30] present EXPOSURE, a system that uses pDNS data to detect non-specific malicious domains. From extensive research on DNS data, the authors defined a set of features that are indicative of malicious behaviour. These features are based on domain name, TTL, DNS answers and temporal properties. For the last set of features, the authors use Change Point Detection to detect if the domain is queried during a short time period or the queries show a regular repeating pattern. Furthermore, they use Euclidean Distance between daily time-series of the same domain to detect daily similarity of queries. With these features they train a J48 Decision Tree classifier to classify newly seen domains. Shi et al. [31] analyzed the performance of using Extreme Learning Machine (ELM), a rather new neural network type, on pDNS data to detect domains used in Command & Control (C&C) communications. Frosch et al. [32] use a different supervised learning algorithm, k-Nearest Neighbor, to detect C&C servers. Both Frosh and Shi et al. use features comparable to Bilge et al., except for the time-based features. For all three of these approaches, the benignly labeled domains come from an Alexa list of most visited websites [3], while the malicious domains come from a number of publicly available blacklists. Perdisci et al. [33] use a clustering algorithm to cluster candidate flux domains according to IP similarities. The clusters are classified using a C4.5 Decision Tree classifier trained on hand-labeled clusters.

[3] https://www.alexa.com/topsites/countries

The usage of pDNS to detect malicious domains presents a problem; a malicious domain can only be detected if it is queried on the monitored network. Therefore, detection mechanisms based on pDNS data can never be proactive, since a domain can only be detected once it is already in use by the adversary. Furthermore, domains that are not queried on the monitored network will never be detected.

3.3.2 Active DNS

Besides passive DNS data, researchers have also used active DNS measurements to identify malicious domains. Félegyházi et al. [34] predict clusters of domains related to a blacklisted domain. This clustering is based on nameserver information from a snapshot of the .com zonefile and registration information from WHOIS. While they can derive between 3.5 to 15 new domains (of which 93% is malicous or likely malicious), these domains will only be detected if a related domain (related nameserver or registration info) is in the blacklist. Fresh malicious domains registered by a completely new actor will not be clustered with any of the existing domains in the blacklist. He at al. [35] use second-order Markov models to express the likelihood of textual sequences in domain names falling into different categories. The difference of these Markov values are used as features, together with other textual features, and features based on nameservers that hosted the domain. Although they demonstrate that their approach is able to detect many malicious domains with a low False Positive (FP) rate, the classification is partly based on the fact that there is a discriminatory difference between benign domain names and malicious domain names, something that may not be true for Amplification domains. Hao et al. [36] present PREDATOR, a system that classifies new domain registrations using time-of-registration and registration history features available at registries, registrars and third-party services such as DomainTools and Who.is. It achieves a 70% detection rate with a FP rate of just 0.35% making it a very effective first line of defense. However, their methodology is based on the fact that adversaries show abnormal registration history. Therefore, deceiving domains that have benign registration characteristics and change their associated resource records during their lifetime will go undetected. Van der Toorn et al. [37] present an approach to proactively detect domains used to send snowshoe spam. Domains in the long tail of the OpenINTEL data set are classified using a set of classifiers. However, features of the domains that the set of classifiers are trained on are calculated on a daily snapshot, ignoring the time-series structure of the OpenINTEL project.

3.4 Machine Learning & DDoS Detection

Several studies have used a form of machine learning to detect DDoS attacks. Nguyen et al. [38] use a K-nearest neighbor (K-NN) classifier to classify the network status into each phase of a DDoS attack in order to detect it in the early stage. Suresh et al. [39] use chi-square and Information gain statistics to select the most important features from two sets of internet traffic. Using these features, machine learning methods from previously discussed work are applied to differentiate attack behaviour from normal behaviour and the results are analyzed. The used machine learning methods include Naive Bayes, C4.5, Support Vector Machine (SVM), K-NN, K-means and Fuzzy c-means clustering. Wu et al. [40] build a Decision Tree classifier built on the C4.5 algorithm in order to classify network traffic into benign or malicious. Sahi et al. [41] compare the performance of multiple classifiers when classifying network packets into benign or malicious. Next to some algorithms already discussed, they also test the performance of a Multilayer Perceptron classifier. Furthermore, they validate their results using K-fold cross-validation and the performance is evaluated using the classification performance measurements accuracy, sensitivity, specificity, and the Kappa coefficient.

These authors tackled the DDoS problem using machine learning on data that was available only after the attack had started. This makes their methods reactive and therefore incapable of preventing the attack from launching. While our approach also involves machine learning, we train our classifier on active DNS data instead of passive DNS data or network packets. This enables us to proactively detect malicious domains and prevent them from being used in a DNS amplification attack.

4 Methodology

The goal of this research is to proactively detect domains which are purposely configured to be used in DNS amplification attacks. Taken from a high-level perspective, the approach is as follows: first, data is gathered on domains that have been used for amplification attacks in the past (Section 4.1) and a set of features is defined that help discriminate domains used in amplification attacks from regular domains (Section 4.2). Second, we present our definition of a crafted domain and use a clustering machine learning technique to create a labeled data set (Section 4.3). Next, a set of classifiers is trained on this labeled data set in order to see which one is suited best to detect domains that are seemingly created for amplification attacks (Section 4.4). Finally, we evaluate how early our detection of these domains is through simulations.

4.1 Data sets

In order to find domains that are purposely configured for an amplification attack, we acquired two data sets. The first data set contains the domain names and dates of domains that are abused in DNS amplification attacks. Second, we gathered historical data on the RRs that are associated with these domains. With both these data sets combined, we are able to see how the RRs of these domains behave up until they are abused.

4.1.1 Cambridge Cybercrime Center

To create a labeled data set, we made use of data from a project by the Cambridge Cybercrime Centre (CCC) that deploys honeypot UDP reflectors in order to study the life cycle of reflection attacks among a number of UDP protocols. Measurements for this project have been ongoing since July 2014 and it is estimated that between 85.1% and 96.6% of the UDP reflection attacks are seen by the honeypots [42]. While the projects stores much information about the packets received by the honeypots (such as source and destination IP), we are only interested in packets using the DNS protocol, and more specifically the domain that was queried.

4.1.2 OpenINTEL

The OpenINTEL project sends a list of queries (see Table 4.1) once every 24 hours for every second-level domain in a TLD [4], creating a time series [12]. The measurements cover a list of generic TLDs, including .com, .net and .org, 13 country-code TLDs as well as the Alexa top 1 million and the Cisco Umbrella top 1 million [5]. The results are stored in a Hadoop cluster and is accessible through Apache Impala. This allows for massively parallel SQL queries. The results of these measurements are used to calculate the features described in Section 4.2

[4] https://openintel.nl/background/
[5] https://openintel.nl/coverage/

Table 4.1: Query types

Resource Record	Description
SOA	Information about the zone (e.g. email address of admin)
NS	Nameserver for the domain
A	IPv4 address for a name.
AAAA	IPv6 address for a name
MX	Mail server for the domain
TXT	Arbitrary text strings
DNSKEY	Public key for validating DNSSEC signatures
DS	Hash of DNSKEY. Used in zone delegation.
NSEC3	Allows proof of non-existence.
CAA	CA Authorizized to sign certificates for this domain
CDS	Child copy of DS record (to signal parent)
CDNSKEY	Child copy of DNSKEY record (to signal parent)

4.2 Features

In this section we will present our approach towards answering RQ1: *What are the defining characteristics of domains misused in DNS amplification attacks?* Since the primary goal of a DNS amplification attack is to create as much traffic as possible, we assume that the domains that are being abused to perform such an attack have a relatively large response to a specific RR query. With this assumption in mind, we developed a method that helps us estimate the response size for a number of query types. This enabled us to analyse the defining characteristics of abused domains (answering RQ1) and turn these into discriminating features that we can use to train our classifier.

4.2.1 Chosen domains

First, we have to combine the data from the OpenINTEL and CCC projects to create a data set on which we can calculate and analyse our features. To have a good representation of abused domains, we take a random subset of domains from the CCC data and lookup these domains in the OpenINTEL database. To be able to put the future results in perspective, we also have to find a group of domains that represent the average domains in the entire domain namespace. For this purpose, we selected a random subset of domains from the .org, .net and .com TLDs that do not occur in the CCC dataset. The amount of domains from each TLD in this sample data set is chosen proportionally to the amount of domains in the whole TLD.

4.2.2 Estimating response size

For each RR type that is queried, the DNS response contains one or more DNS answer sections, depending on the amount of RR's of that specific type the queried domain has. Each answer section is structured as Table 4.2. NAME is the domain name that was queried, and its length depends on the usage of data compression, which we will explain further in this section. TYPE and CLASS are fields which contain a 2-byte RR type or class code respectively, TTL is a 4-byte field that specifies the number of seconds the results may be cached and RDLENGTH is a 2-byte field that specifies the length of the RDATA field. Finally, the length of the RDATA field depends on the RR type that is queried.

NAME
TYPE
CLASS
TTL
RDLENGTH
RDATA

Table 4.2: DNS Answer structure

To estimate the total size of the RDATA field(s) for each RR type, for each domain in both categories, we group by day and count the number of occurrences for each RR measured and sum the amount of bytes in each field corresponding to that record. For example, we count the total amount of bytes in the `TXT` record text field and the amount of bytes of all the signatures in `RRSIG` records. We end up with a giant table that looks like Table 4.3. Next, using the record count and sum of bytes, we look at the respective RFC to see how many bytes the RDATA field(s) of that RR would take up in the DNS response. Table 4.4 shows the size of the RDATA field for each record type.

domain_hash	date	count_A	count_AAAA	count_CNAME	count_MX	count_TXT	sum_TXT_char
00041d5a90...	2017-11-01	1	1	0	0	1	39
	2017-11-02	1	1	0	0	1	39
	2017-11-03	1	1	0	0	1	39
	2017-11-04	1	1	0	0	1	39
	2017-11-05	2	2	0	0	4	456

Table 4.3: Example of grouped record count and sum of bytes in RDATA fields

Record Type	**Size (bytes)**	**RFC**
A	4	1035
AAAA	16	3596
CNAME	2	1035
MX	2	1035
NS	2	1035
TXT	nr. of characters	1035
SOA	24	1035
SPF	5 + nr. of characters	4408
CAA	2 + len(tag) + len(value)	6844
TLSA	3 + len(certificate data)	6698
RRSIG	18 + len(signer name) + len(signature)	4034
(C)DNSKEY	4 + len(pub key)	4034 & 7344
(C)DS	4 + len(hash)	4034 & 7344
NSEC3PARAM	5 + len(salt)	5155
NSEC3	6 + len(salt) + len(next domain hash)	5155
NSEC	len(next domain)	4034

Table 4.4: Record types and their estimated size of the rdata field

While many parts of the response are of fixed size (depending on the queried RR type), the inclusion of a domain name in the response can result in different sizes. In order to reduce the size of messages,

DNS tries to avoid the repetition of domain names by pointing towards the domain name somewhere else in the message. Such a pointer takes up only two bytes and can therefore significantly reduce the size of a response message. For example, a pointer could be used in the NAME field of the DNS answer or in the RDATA field of a requested NS record. However, the use of pointers is optional and it is therefore not possible to accurately predict the size of a DNS response that contains references to other domains names. Instead, we choose for the best case scenario of two bytes whenever we have to estimate the size for the RR fields that mention other domain names.

Therefore, for each DNS answer section, the sum of the NAME,TYPE,CLASS,TTL and RDLENGTH is 12 bytes. Finally, to calculate the response size to a query for a specific RR type, the amount of records for that RR type is multiplied by 12 and added to the size of all RDATA fields of that record type. This results in data formatted as Table 4.5.

domain_hash	date	size_A	size_AAAA	size_CNAME	size_MX	size_TXT
00041d5a90...	2017-11-01	16.0	28.0	0.0	0.0	51.0
	2017-11-02	16.0	28.0	0.0	0.0	51.0
	2017-11-03	16.0	28.0	0.0	0.0	51.0
	2017-11-04	16.0	28.0	0.0	0.0	51.0
	2017-11-05	32.0	56.0	0.0	0.0	504.0

Table 4.5: Example of estimated sizes

4.2.3 Size to feature

Using the estimated response size to each query performed by OpenINTEL and the `ANY` query, we used distribution plots to look at statistical features that are often used in feature-based TSC, such as mean, minimum, and maximum. Furthermore, we tried to describe the temporal structure of our data using features that consider the growth of each of the response sizes. The results of this analysis and the final features can be found in Section 5.1.

4.3 Generating the labeled data set

The aim of this research is to automatically detect domains which are specifically made for DDoS attack. In order detect these domains by using a classifier, a labeled data set is required. However, from our CCC data set we do not know for certain if the abused domains were created specifically for this purpose. Other authors called the creation of domains for a DDoS attack 'crafting' [8] [43]. Unfortunately, they do not define the characteristics of such 'crafted' domains. Therefore, in order to generate a labeled data set we first need to answer RQ2: *What is the definition of a crafted domain?*

4.3.1 Formal definition of crafted domain

The primary goal of a DNS amplification attack is to generate and send as much traffic towards the victim as possible using the DNS protocol to amplify the bandwidth. A crafted domain is created by the attacker to help achieve such an attack. Therefore, the goal of a crafted domain is to generate a large response to a specific query type (e.g. `ANY` or `TXT`). While an adversary could craft and sign a domain with a cryptographic algorithm that uses large keys and results in large signatures, we assume that this is unlikely. This is due to the fact that the use of existing legitimate domains has a number of advantages for attackers. First, the authoritative name servers of these domains often have more bandwidth available than open resolvers. Second, the attacker is less likely to be discovered and prosecuted since they did not register these domains. Finally, the DNS responses to these domains are harder to be filtered out as this would also impact legitimate queries [8]. There is a large set of legitimate signed domains to choose from that result in a large response. This can be seen from

Fig. 4.1, as only about 5% of the signed domains in the Alexa top 1M result in a response that is smaller than 1000 bytes. This leads us to Definition 4.1:

Definition 4.1 *Crafted domain: A non-DNSSEC-signed domain in the Domain Name System that is purposely configured to give an unusually large response to a certain query type.*

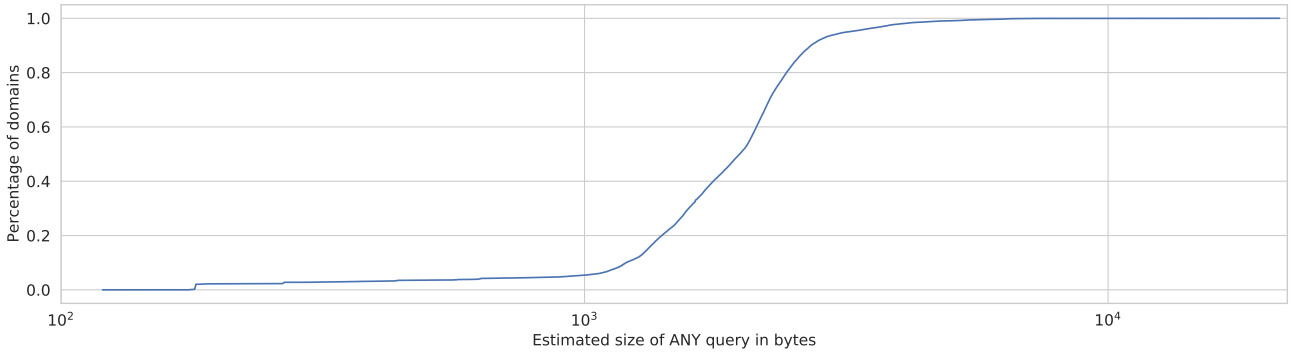


Figure 4.1: CDF for signed domains Alexa top 1M

4.3.2 Quantifying and extending our definition

While a basic definition of a crafted domains was established, it needed to be quantified because the term "unusually large response" is rather subjective and therefore does not yet allow for the labeling of domains. To determine a specific size that is considered a large response, a Cumulative Density Function (CDF) was made of all the non-DNSSEC signed domains in the Alexa top 1M on one day. We assume domains in the Alexa top 1M to be non-crafted as the method that Amazon uses to rank domains makes it hard for crafted domains to acquire a high rank. This method ranks domains on traffic data provided by a combination of browser extensions and Alexa scripts on websites. Furthermore, data of the past three months is used in calculating the rank, in order to prevent any daily spikes. Therefore, in order for a crafted domain to be included in the Alexa top 1 million websites, it should be popular among real life users for at least three months [6]. Since crafted domains are mainly created with an attack goal in mind, we believe it is highly unlikely that such a domain will gain popularity among actual humans and enter the Alexa top 1 million.

In order for our definition to accurately encompass domains that are specifically created to perform DNS amplification attacks, it needed to be extended. Therefore, we looked for recurring patterns in the estimated responses to an `ANY` query over time. More specifically, we were interested in patterns that occur in our CCC dataset, but do not occur in abused Alexa domains. Since Alexa domains are assumed to be benign, we think that such patterns belong to crafted domains and can thus be used to help make our definition more detailed. Finding patterns in data can be done automatically without any prior knowledge from statistical regularities and is called unsupervised learning [44]. More specifically, the machine learning technique of clustering is defined as the unsupervised classification of patterns in groups (called clusters) [45], such that similar instances (with similar patterns) are grouped together, while different instances belong to different groups [46]. Although many clustering techniques exist, we settled on the very popular K-means clustering due to its linear complexity, making it effective to cluster large data sets, and it's availability in the scikit-learn library [7]. Using the patterns we discovered, we updated our definition and labeled all the CCC domains that match this definition as crafted while labeling the abused Alexa domains as benign.

[6] https://www.alexa.com/about
[7] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

4.4 Training classifiers

To answer RQ3 (*how can the behaviour of DDoS crafted domains be automatically detected?*) we chose to take a machine learning approach. The driving force behind this decision is the fact that if new data on crafted domains would become available, we could simply retrain the classifier in order to be up-to-date on the newest domain crafting techniques. In total we tested 12 classifiers. These are:

- Naive Bayes
  - 'BernoulliNB'
  - 'GaussianNB'
  - 'MultinomialNB'
- Decision Tree
  - 'DecisionTreeClassifier'
  - 'RandomForestClassifier'
- Nearest Neighbor
  - 'KNeighborsClassifier'
  - 'RadiusNeighborsClassifier'
- Gradient Descend
  - 'GradientBoostingClassifier'
  - 'SGDClassifier'
- 'Support Vector Classifier (SVC)'
- 'MLPClassifier'
- 'AdaBoostClassifier'

The motivation behind these classifiers is based on their use by other researchers in the field of feature based TSC [47–50] and their availability in the scikit-learn Python library [51].

After the features are calculated over our labeled data set, we obtained our data set where each row represents a domain, and each of the 12 columns contains a feature. This data set was then split up into two sets: a training set and a test set. The training data was, as its name suggests, used to train and tune the parameters of the classifiers (Section 4.4.1). The test set was kept hidden from the classifiers so that we could analyze the performance of the tuned classifiers on unseen data to select the best one. (Section 4.4.2).

4.4.1 Hyperparameter tuning

With different parameters, the performance of each classifier can differ greatly. We must therefore know what the optimal parameter values are for each classifier in order to have a comparison that is based on the best possible performance. Using K-fold Cross-Validation (KCV) [52], we were able to compare each combination of parameters for a given classifier, without having to set apart yet another part of our data. The KCV method splits the training data into K parts , called folds, and trains the classifier K times on K-1 folds. Each time a different fold is held out to calculate the performance score of a certain parameter value combination. The average of these K scores is the final performance of the classifier with those parameters. This is done for all parameter value combinations. As we are aiming at detecting crafted domains in an early stage, we intend our solution to be used in combination with other DNS amplification attack preventing measures. For this reason, we prefer the minimization of False Negatives (FN) over the minimization of FP, as FP might still be removable from our detection results while FN will lead to missed domains. We consider a crafted domain as positive and a benign domain as negative. The recall metric was chosen to measure the performance when tuning the hyperparameters of our classifiers, as it minimizes FN. It is defined as the number of

True Positive (TP) divided by the actual amount of positives, which is the sum of correctly (TP) and incorrectly (FN) labeled positive domains (Eq. 4.1).

$$Recall = \frac{TP}{TP + FN} \tag{4.1}$$

4.4.2 Selection of best classifier

Since the goal of our research is to develop a method that can detect crafted domains proactively, we analyzed the performance of each tuned and trained classifier when it is given features that are calculated with data from before the earliest date of abuse. In steps, we recalculated the features over the domains in the test set, each step removing more data as we moved back in time. A visual representation of this process can be found in Fig. 4.2.

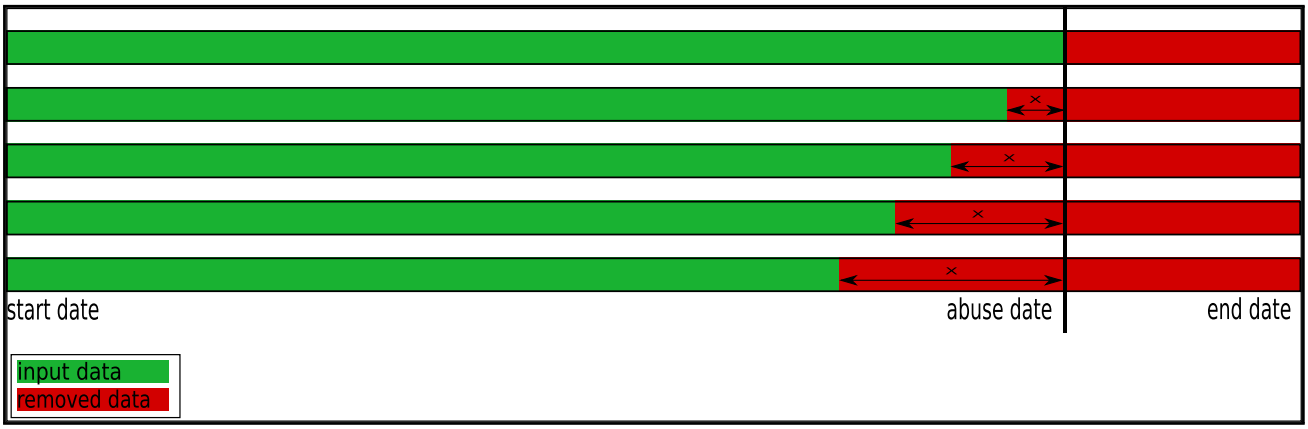


Figure 4.2: Simulation of pro-active detection

Although we selected the hyperparameters of our classifiers based on the highest recall score, this metric is far from perfect. If a classifier had labeled all domains as crafted, the recall score would be equal to 1, as there are no FN. However, this is probably misleading, as the number of FP would be quite high. The precision metric could help us detect these kind of situations, as it decreases if the number of FP goes up (Eq. 4.2).

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

However, for the selection of the best classifier, we still prefer FP over FN. Therefore, we settled on evaluating the performance by means of the F-measure metric, defined in Eq. 4.3. This metric expresses the performance of a classifier, weighing recall β times as much as precision. A value of 2 was chosen for β, meaning that the metric favors the reduction of FN over the reduction of FP while preventing a score of 1 if a classifier would label all domains as crafted.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{4.3}$$

5 Results

5.1 Feature Selection

In this section we will present the results of our feature analysis. Due to the high availability of legitimate signed domains and the advantages that they bring (see Section 4.3.1), it was decided that we remove signed domains from both the CCC and sample data sets. While doing this, a few interesting details about the signed domains in our CCC data set were discovered. Since these findings are outside the scope of this thesis, they can be found in Appendix A.

5.1.1 Statistical Features

To get an idea of what the domains in each group look like, we averaged each type of RR and summed them to see how large the average domain is in each group, and what RR takes up the most space on average (see Fig. 5.1). The first thing that came to our attention was the much bigger presence of `TXT` records in domains from the CCC dataset. In the group of sample domains, the `TXT` records take up roughly 17% of the response to an `ANY` query, while this over 24% in the group of abused domains.

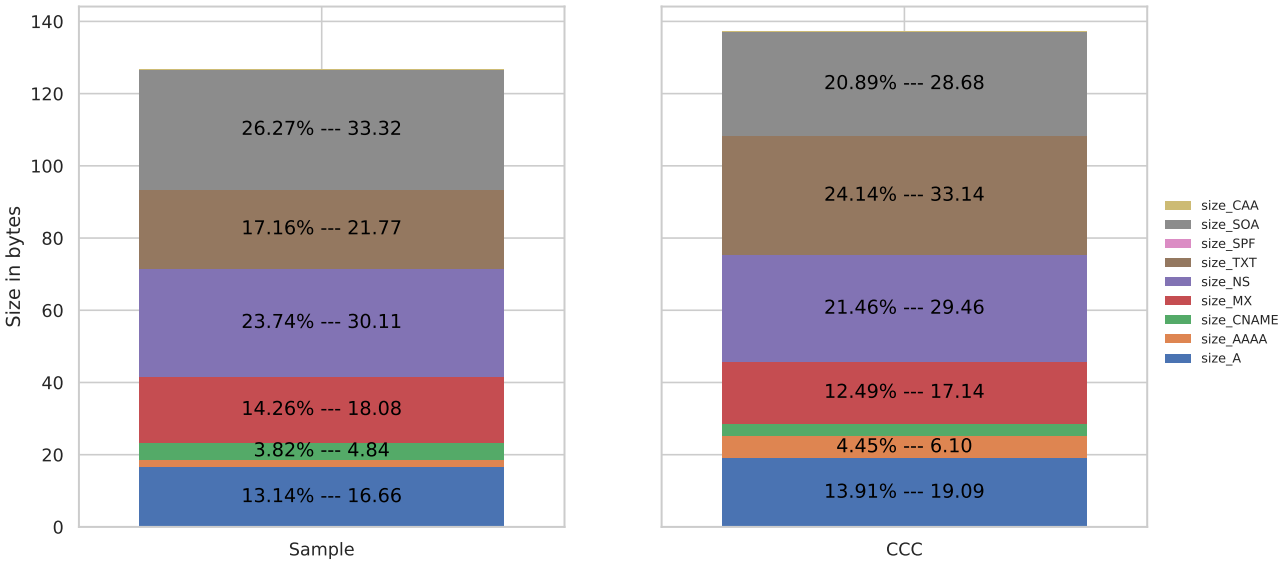


Figure 5.1: Average size of unsigned domains in sample and CCC dataset

Average sizes of domains do not give the whole picture however. Since the goal of an amplification attack is to generate to as much traffic as possible, we have plotted the distribution of the maximum response to an `ANY` query (over the whole time series) for the domains in each group. Since `TXT` records had a significantly larger impact on the average CCC domain, a distribution of the maximum size of combined `TXT` records (per domain) was also plotted. In Fig. 5.2 we notice a few things. First, we see that the extreme outliers for both query types mainly belong to the abused domains. Second, there is a peak in the `ANY` query plot in the bin for values between 1025 and 1050 bytes. These domains almost exclusively belong to a domain parking service that has the same `TXT` records, accounting for 935 bytes, in a large number of their domains. This can be seen in the plot for the `TXT` query. Finally, the CCC domains have ,on average, a slightly larger maximum response to an `ANY` query (~157 bytes) compared to their sample counterparts (~145 bytes) and a much large maximum response to a `TXT` query (~30 bytes vs ~46 bytes).

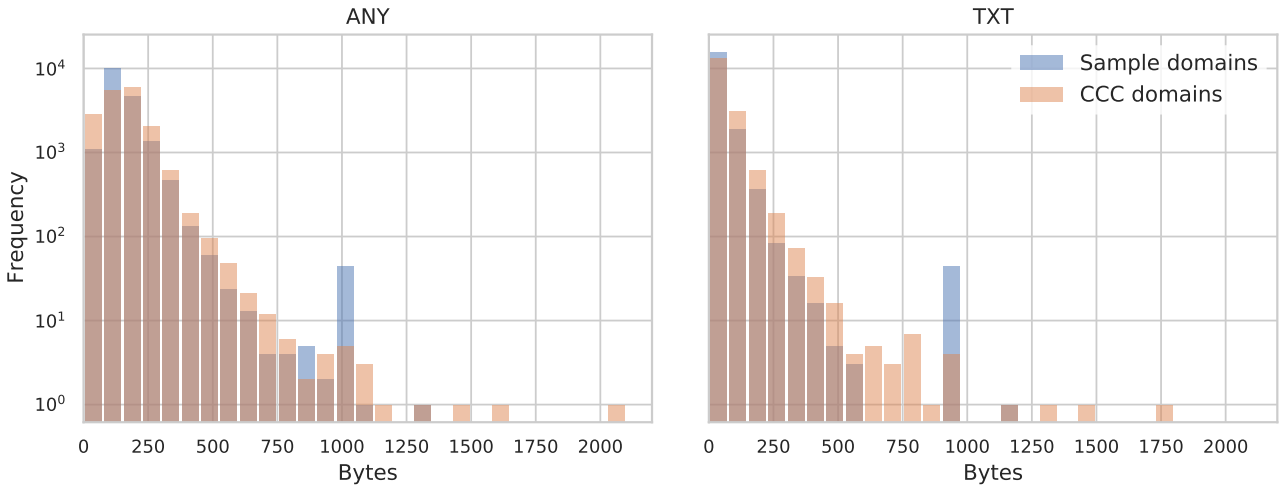


Figure 5.2: Distribution of maximum `ANY` and `TXT` query response size for non-signed domains

The same pattern is found in the distribution plots of yet another statistical feature often used in feature-based TSC: the minimum function. Fig. 5.3 shows that, with the exception of the `TXT` records from the domain parking service, the tail of the distribution is dominated by domains from the CCC group. Furthermore, the average of these minimum response sizes to an `ANY` query is again slightly larger for the CCC domains compared to the average domains (~95 vs. ~83 bytes respectively) while the relative difference is again larger for the response to a `TXT` query (~16 vs ~11 bytes).

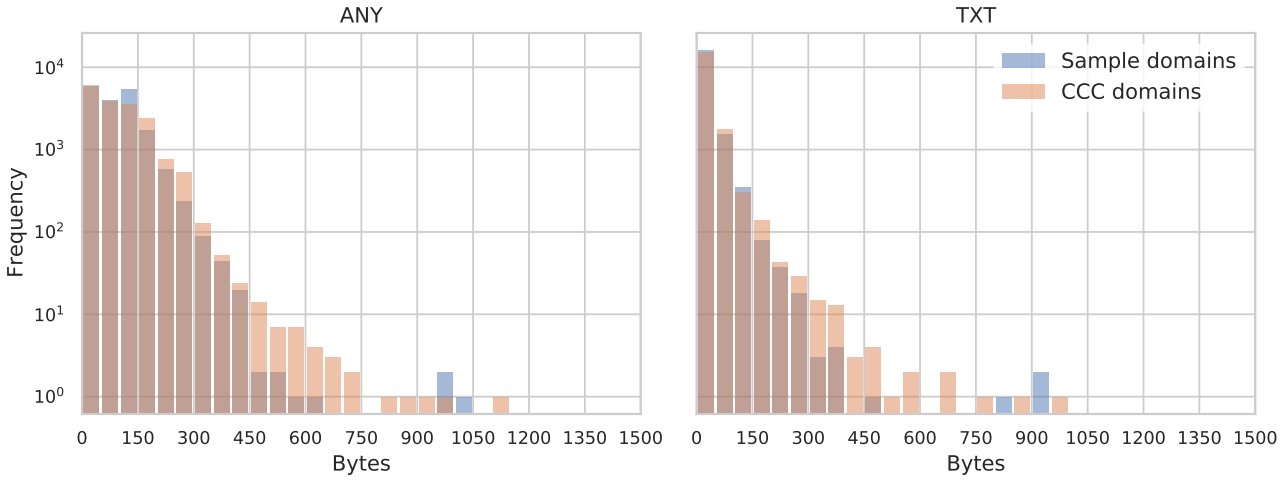
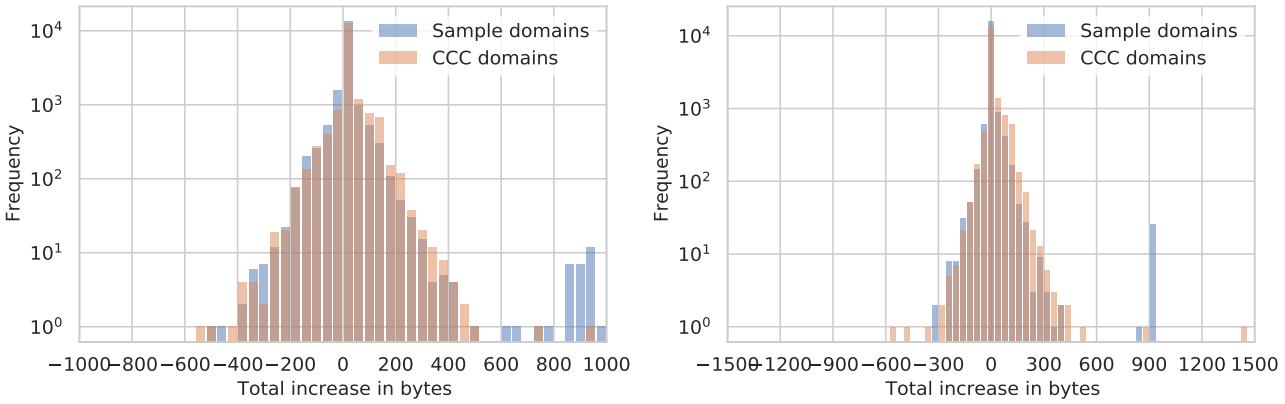


Figure 5.3: Distribution of minimum `ANY` and `TXT` query response size for non-signed domains

5.1.2 Temporal Features

After analyzing the statistical features for both groups, the focus shifted towards features that capture how the response size behaves over time. The first feature that was analyzed is the growth of the response size for both the `ANY` and `TXT` query types. Fig. 5.4a shows that the domains in the CCC group grow more in `ANY` query response size over time on average (13 bytes), compared to the sample domains (6 bytes) and that their standard deviation is roughly the same at 59 at 57 bytes respectively. Fig. 5.4b gave us the suspicion that most of the growth in `ANY` query response is coming from a growth

in TXT records. Furthermore, it shows yet again the TXT records for the domain parking service as a peak. Next, to see if our suspicions are correct, the average increase (or decrease) in response to each query type was calculated. Fig. 5.5 shows that, indeed most of the growth is in TXT records size. While increase in TXT record size is responsible for a significant part of the overall increase in ANY query response over time in both data sets, it is noticeably higher for abused domains (86% of total increase) compared to sample domains (73% of total increase).



(a) ANY (b) TXT

Figure 5.4: Distribution of growth in ANY and TXT query response size

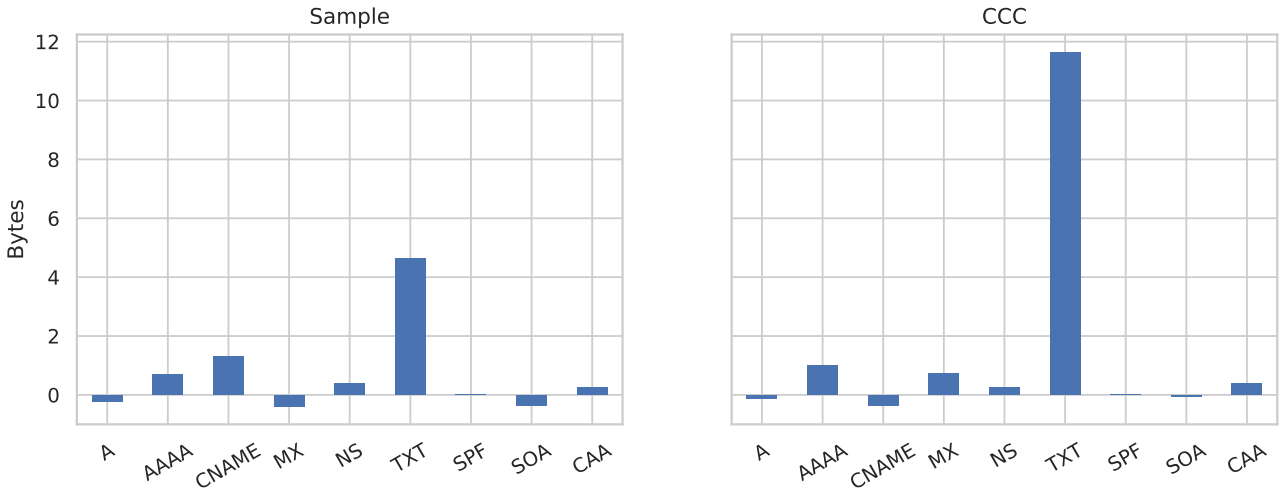


Figure 5.5: Average total increase per record type

However, at this point it is unclear if this growth is caused by a few major or many minor changes. Therefore we plotted the amount of changes in ANY query response size against the amount of changes in the TXT query response size and fitted a linear regression model. Fig. 5.6 shows the fitted regression line with a 95% confidence interval for both the sample and CCC data set. We notice two things: First, that there are more changes in the abused domains compared to the sample domains. On average, an abused domain has 7.5 changes during the measured time period, while a sample domain has 5.3 changes. Second, the fitted linear regression on the CCC data set is steeper and has a narrower confidence interval compared to the sample data set. This means that for any amount of changes in

ANY query response size, the corresponding amount of changes in TXT query response size will be higher for CCC domains.

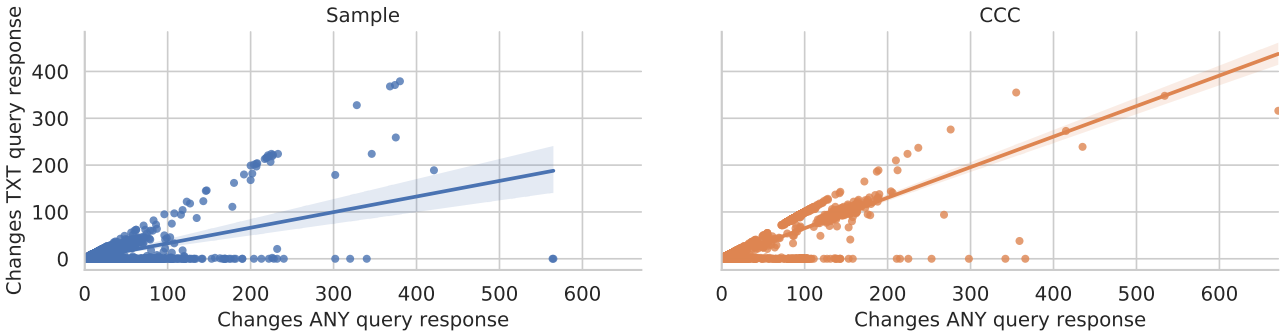
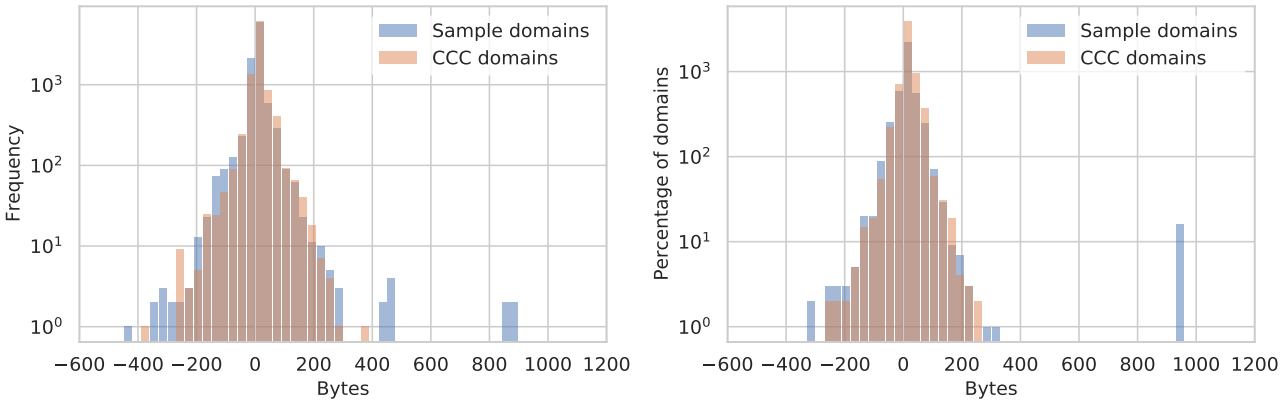


Figure 5.6: Changes in TXT query response vs changes in ANY query response

Finally, we analyzed the average step size of each domain. We defined the average step size as the amount of bytes the a query response increased in total divided by the amount of changes it has received. The results for the ANY query, shown in Fig. 5.7a, tell us abused domain have a bigger step size with a mean of 9.5 bytes and a standard deviation of 36.6 bytes, compared to a mean of 4.8 bytes and a standard deviation of 42.5 bytes for the sample domains. The difference in step size for the TXT query are smaller; fhe average step size for CCC domains is 12.8 bytes, while the average step size for the sample domains is 12.9 bytes (see Fig. 5.7b). However, this feature can still be quite discriminating as the standard deviation is 34.3 bytes for CCC domains compared to 71.2 bytes for the sample domains.



(a) ANY

(b) TXT

Figure 5.7: Distribution of stepsize for ANY and TXT records

5.1.3 Final Features

In the section above, we have analyzed domains for which we know that they have been abused in the past and compared them to a random sample of domains. During this process, we uncovered multiple insights. First, on average, non-signed domains are slightly bigger over the whole time period and also have a higher maximum and minimum size. Furthermore, a bigger part of their size is dedicated to TXT records and when we look at the distribution of maximum size and minimum TXT query response size, almost all the large outliers belong to the CCC dataset. Next, we looked at how these domains

behaved over time. Here we see a similar pattern. The abused domains grow slightly more over time, and much of this can be explained by increase in `TXT` record size. Finally, we looked at the amount of changes that a domain receives and the size of these changes (step size). We found that abused domains have more changes over time; a high percentage of these changes are caused by alterations of `TXT` records and—on average—these changes are slightly bigger than the sample domains. Therefore, our list of features becomes as follows:

- minimum `TXT` query response size
- minimum `ANY` query response size
- maximum `TXT` query response size
- maximum `ANY` query response size
- mean `TXT` query response size
- mean `ANY` query response size
- growth in `TXT` query response size
- growth in `ANY` query response size
- amount of changes in `TXT` query response size
- amount of changes in `ANY` query response size
- step size of `TXT` query response size
- step size of `ANY` query response size

5.2 Data set generation

In this section, we will describe the results of creating a feature-based representation of domains from our CCC and OpenINTEL data sets, which are either labeled benign or crafted. It is split into two sections: the quantification of the term "unusually large response" (Section 5.2.1) and the use of clustering to find patterns that are unique to crafted domains (Section 5.2.2).

5.2.1 Large domain threshold

Fig. 5.8 shows us that 99 Percent of the unsigned domains in the Alexa snapshot had an estimated response to an `ANY` query of 640 bytes or less. Since we assume that these domains are of a benign nature, we believe that any unsigned domain that has an estimated response above 640 bytes at any given time is reason for suspicion. However, this suspicion does by no means *guarantee* that these domains are crafted. For example, when we inspect our CCC data set and look at domains that have also been in the Alexa top 1M list, we see that of those matches (roughly 3000 domains) over 50 percent have a maximum size above the 640 bytes threshold somewhere in the past 2 years of data we analyzed. While it may not come as a surprise, it shows that attackers will also misuse benign domains which happen to respond with a large result. Therefore, we will need more characteristics of these crafted domains if we want to correctly label (and ultimately detect) them, as size alone will lead to many false positives.

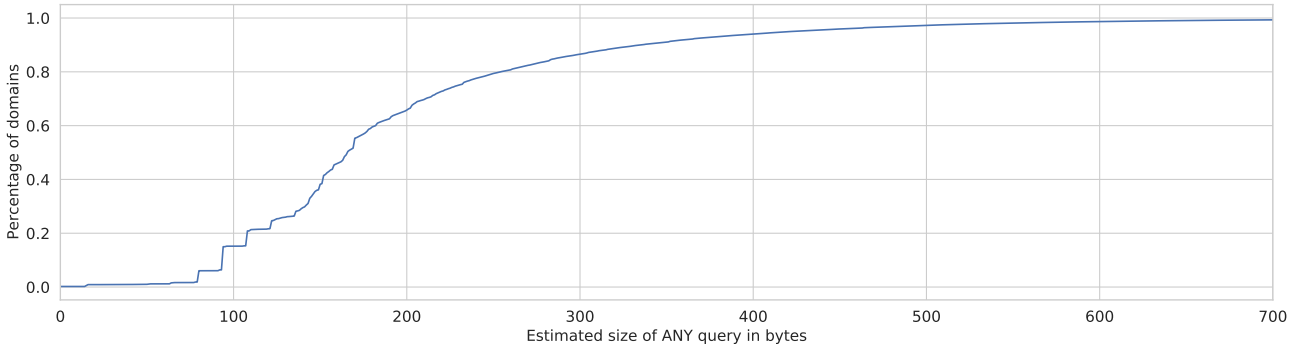


Figure 5.8: CDF for non-signed domains in Alexa top 1M

5.2.2 Clustering

Now that we have a threshold with which we can select large domains, we want to find which of those domains are purposefully crafted with an amplification attack in mind and which ones are not. In this section we present the results of the K-means clustering algorithm that helped in finding a unique pattern in crafted domains that we used to update our definition and create a labeled data set.

5.2.2.1 Temporary data set To start off, we create a data set that contains potentially crafted domains by taking all the .com domains in our CCC data set that are *not* in our openINTEL Alexa data set and have an estimated response size to an `ANY` query above 640 bytes somewhere in the time series. Next, a data set containing benign domains was created by taking domains that cross the same threshold of 640 bytes and *are* in our OpenINTEL Alexa data. Finally, it was decided that we cluster on the whole time series of estimated response sizes to the `ANY` query. This was done for two reasons; First, by using the whole time series as one sample instead of a feature representation of said time series, we might detect a pattern that is only visible over time. Second, by using the response to the `ANY` query (instead of any other query type, such as `TXT`), patterns that occur in any of the other query type resonses will still be visisble, as they directly influence the the response size to an `ANY` query.

5.2.2.2 K-value While K-means clustering has many advantages, such as its linear time complexity, one of the main disadvantages is the manual determination of K. To determine this value, we used the so-called elbow method in combination with the Silhouette score. The elbow method states that the best value for K is when there is no more significant decrease in the Sum of Squared Errors (SSE), also called Inertia, when K is increased. The name of the method is derived from the typical shape of the graph when plotting Inertia vs K-value. The best K value is where this elbow occurs (and the Inertia does not decrease significantly anymore). However, such a clear elbow is not always present in the graph, which can make it difficult to settle on a good K value. Another way of determining the best value for K is by using the Silhouette score. The Silhouette score is a value between -1 and 1 that indicates how close a data point is to another cluster. A value of 1 indicates that it is very far away from other clusters, 0 indicates that it is near another cluster, and -1 that is in the wrong cluster. The optimal K-value is when the average Silhouette score is the highest.

5.2.2.3 Discovered patterns The potentially crafted data set was clustered first. In Fig. 5.9 the Inertia and Silhouette score for clustering with multiple K-values can be found. According to the silhouette score, we should take K=3. However, the inertia drops significantly when K is increased from three to four, while the Silhouette score does not drop all that much. This means that with four cluster instead of three, the data points within each cluster are closer together, while the distance to other cluster does not decrease by much. Therefore we settled on a value of four for K and the

clusters this produces are seen in the left plot of Fig. 5.10. Each line represents the center of a cluster. Domains that belong to a certain cluster have a similar pattern of `ANY` query response size over time as the illustrated center of that cluster. The legend of the plot shows the label that is given to each cluster (in case K=4 these are values 0 up until 3) and the amount of domains in each cluster.

As expected, the two smaller clusters in the graph (where smaller means having a smaller response size) contain more domains than the two clusters with a high response size. Furthermore, we see that the smaller clusters also grow slowly but steadily over time. To see if this growth can be explained by underlying patterns, a higher value of K is chosen and some interesting clusters appear. Two out of the three new clusters display a significant increase in estimated response over time. Upon closer examination, it appears that the clusters labeled 0 and 2 (with lowest response size) are now split into four clusters (labeled 0,2,4, and 6). Cluster 2 is more or less a straight line, while clusters 0 and 4 show a clear pattern of growth, which explains the steady growth in the previous cluster labeled 2. Furthermore, the cluster previously labeled 0 (now labeled 6) remains steady for a longer period of time before increasing in response size. Over two years, clusters 0, 4, and 6 increase 194, 494, and 154 percent in size respectively.

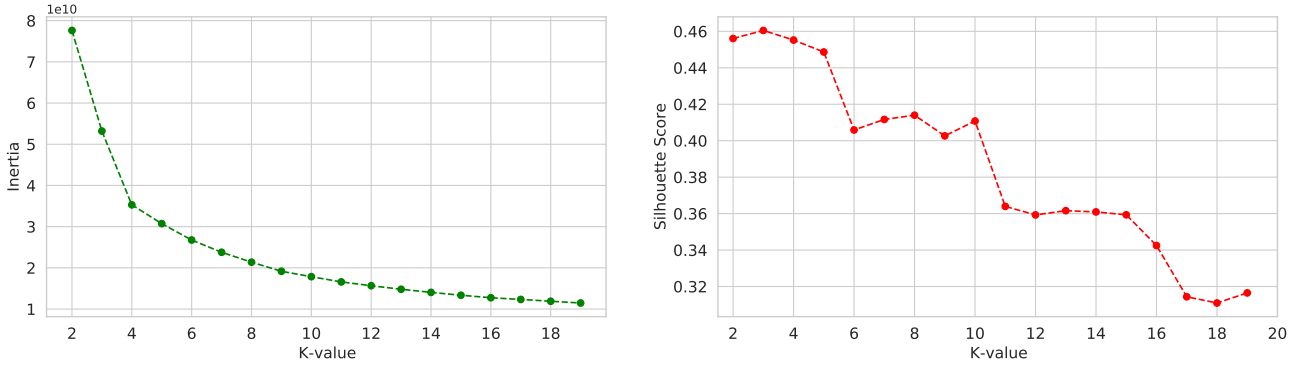


Figure 5.9: Potentially crafted set: Inertia and Silhouette score for different k-values

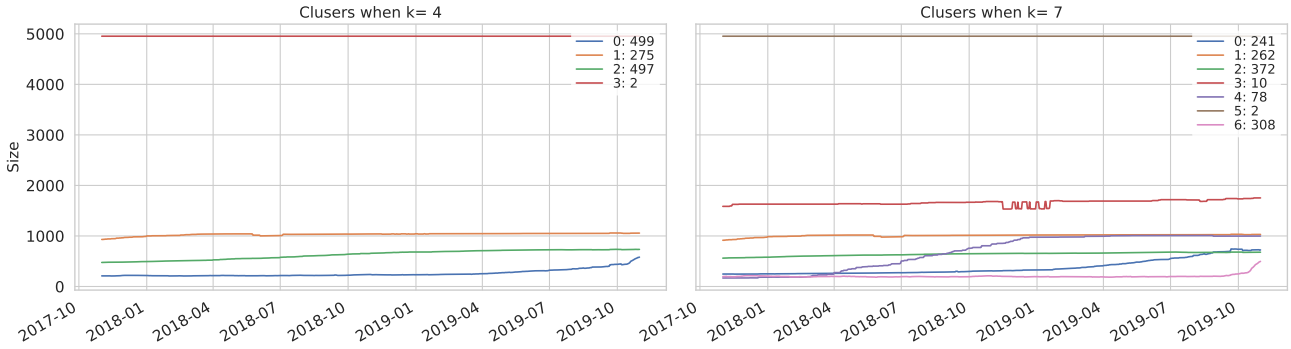


Figure 5.10: Potentially crafted set: Clusters for different k-values.

Next, we clustered domains in the benign data set as well. This way, we can see if the pattern of extreme relative growth also appears in benign (but large) domains. Fig. 5.11 shows the Inertia and Silhouette score for these domains at different values for K. With the same reasoning as before, a K-value of three is chosen although the highest silhouette score is at K = 2. Fig. 5.12 shows the clusters that are created with K = 3. Unlike the domains in the potentially crafted set, when the value for K is increased to seven, no new extreme patterns of growth appear. Where the potentially crafted

domains had three clusters (with k = 7) with a relative growth between 154 and 494 percent, the highest amount of relative growth in the abused Alexa domains is in cluster 6, which shows a growth of 50 percent. The fact that these clusters are so close together also explains the very quick decrease in Silhouette score once K is more than two.

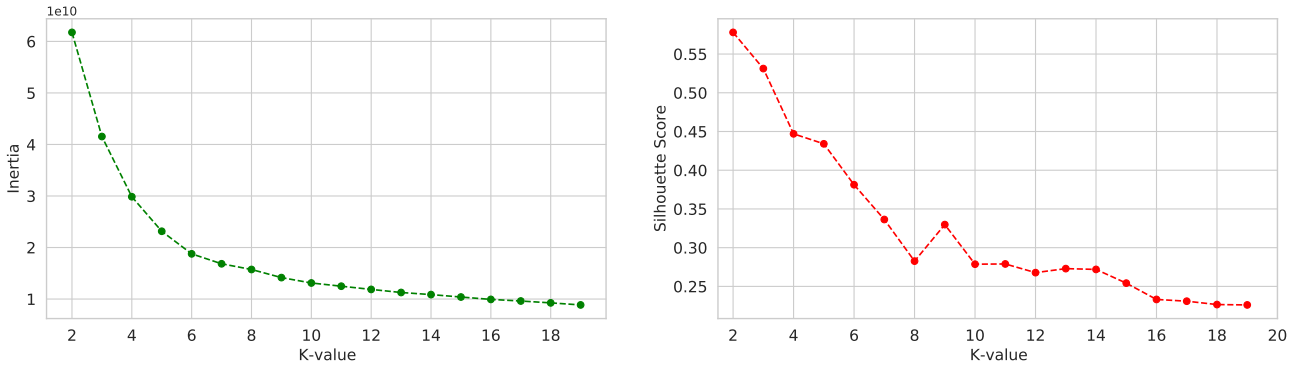


Figure 5.11: Benign set (Alexa): Inertia and Silhouette score for different k-values

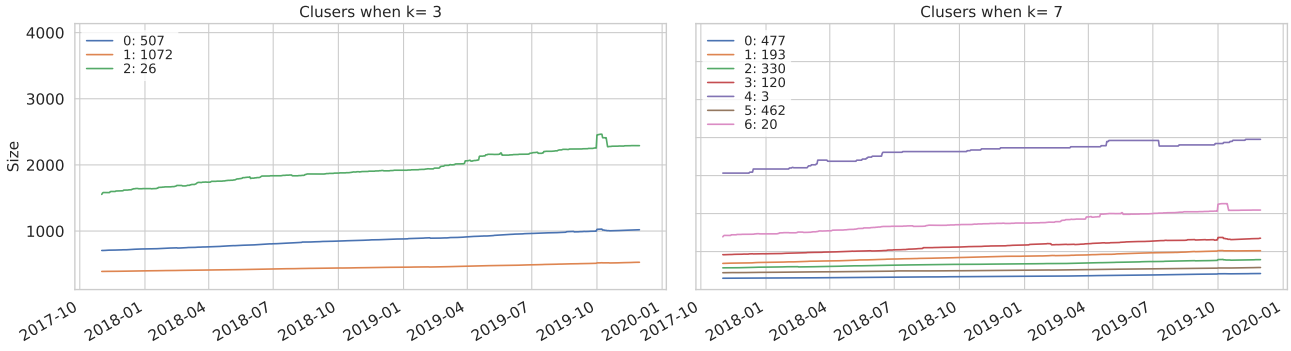


Figure 5.12: Benign set (Alexa): Clusters for different k-values.

Now that it had been established that the pattern of large growth over time ,which was present in the potentially crafted set, is not (or barely) present in benign domains, we inspected the domains that were in these clusters of growth. While the center of the clusters show a steady but large growth over a period of months, many domains in these clusters show a very sudden increase in response size. An example of such a spike for each cluster can be seen in Fig. 5.14. However, these sudden spikes in response size do not all occur on the same date within each cluster, and therefore the center of the cluster does not show such a spike but rather an average of these spikes. While such spikes in response size make up the pattern for many of the domains in these growing clusters, the pattern of a large steady growth rate is also present. Fig. 5.15 shows examples of domains in the growing clusters that grow steadily over time.

5.2.3 Final data set

The fact that these patterns of large relative growth, either steadily over time or in a sudden peak, almost only occurred in the data set of potentially crafted domains and not in the abused (but benign) Alexa domains, is reason for us to believe that this is a characteristic of crafted domains. Since high relative growth is rather subjective, the amount of relative growth in `ANY` query response size was calculated for all domains in the potentially crafted and benign sets. The CDF for this growth is

shown in Fig. 5.13. This figure shows that at the 75^{th} percentile, the domains in the potentially crafted set have 442.9 more percent points of growth compared to the benign domains. At the 95^{th} percentile, this difference is 653.8 percentage points. For the domains in the benign set, only 1% have grown 430% or more in response size, giving us a number to the term "large relative growth" . The new definition of a crafted domain is updated and can be found in Definition 5.1. To generate the final data set of crafted and benign domains, all domains in the CCC data set that fit Definition 5.1 are labeled as crafted and added to the the data set of abused Alexa domains, which were labeled benign.

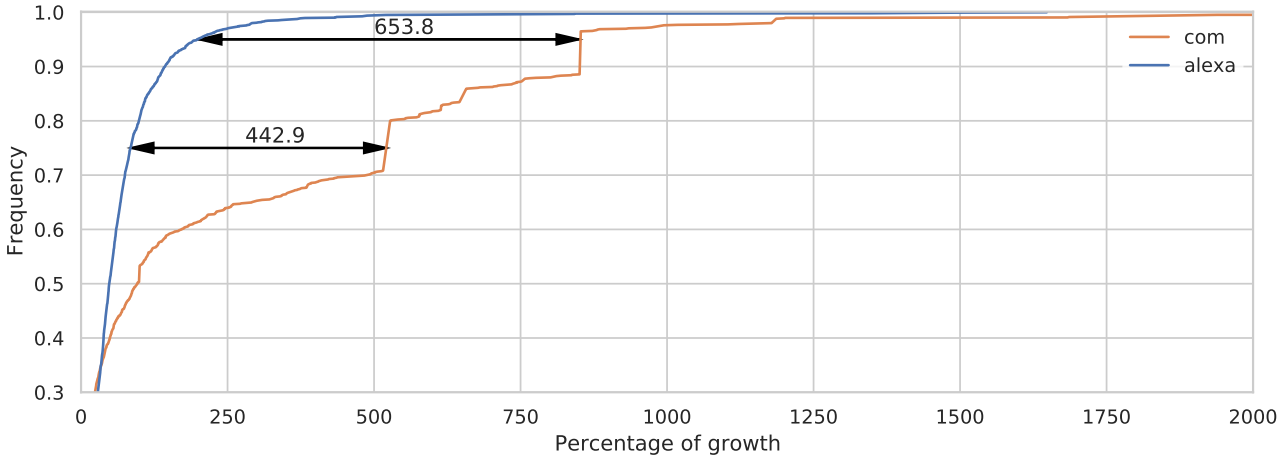


Figure 5.13: CDF of growth in benign and potentially crafted data sets.

Definition 5.1 *Crafted domain: A non-DNSSEC-signed domain in the Domain Name System that is purposely configured to give an unusually large response to a certain query type (> 640 bytes). Furthermore, the response size shows an extreme amount of relative growth (> 430%), whether slowly over time or in sudden spikes.*

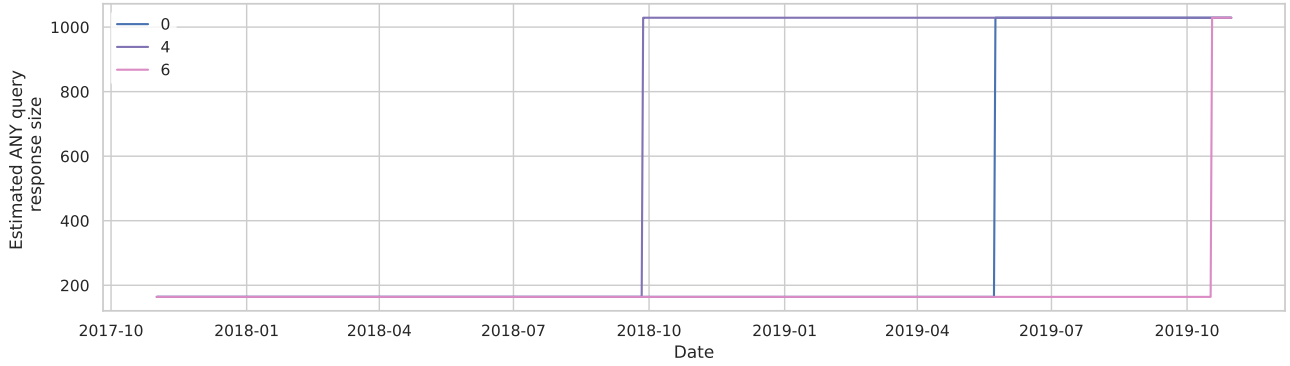


Figure 5.14: Example of spikes in clusters

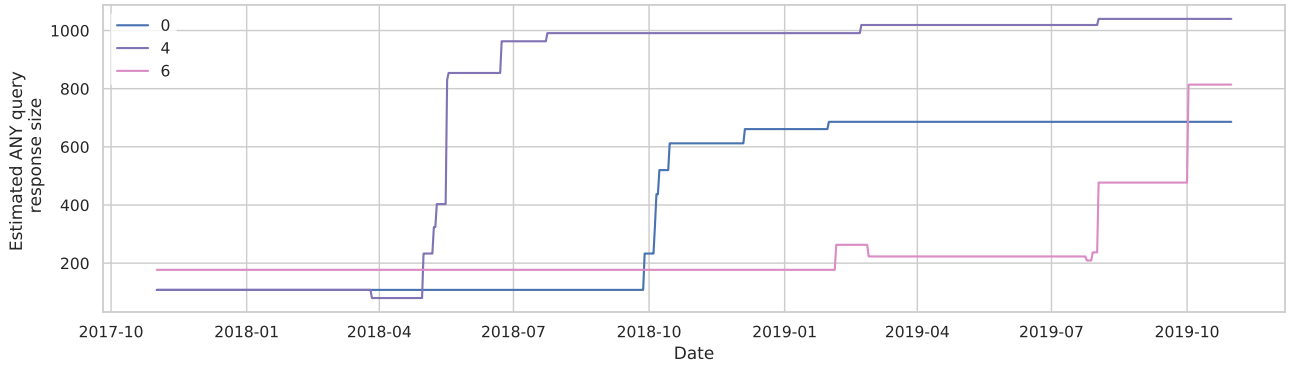


Figure 5.15: Example of steady growth in clusters

5.3 Classifier Performance

5.3.1 Initial performance

After the tuning of our hyperparameters (see Appendix B), the performance of each classifiers on the unseen test data was analyzed. We chose to start with the performance on features for each domain in the test set calculated over all the data up until the first day of abuse. The reasoning behind this is that for the classifier to see features based on data from days after this point in time, it would need to wait until after the initial attack has already taken place. This is entirely against our objective of having a proactive approach. The results of the different classifiers on these features can be seen in Table 5.1.

The first thing we notice is that none of the the classifiers are able to correctly classify all of the crafted domains, even though all of the data up until the first abuse date is translated into our features. In order to investigate why some crafted domains are miss-classified, we analyzed which features deliver the most information gain and what the feature values of the miss-classified domains were. Since the decision tree is one of the most comprehensible classifiers available, we used it to perform the information gain analysis.

A decision tree consists of a set of nodes that are split into sub nodes based on a threshold value check on certain features in the data set. The edges that connect the sub nodes to the parent node are based on the answer to this threshold check. An instance in the data that needs to be classified travels down this tree of nodes until it reaches a leaf node. This leaf node, a node with no more nodes beneath it, has a label attached to it and all the instances that reach it are labeled as such. For each new split of nodes, the features and their corresponding threshold value are chosen based on a certain metric, in our case Entropy. Entropy describes the randomness in our data set, and the feature and threshold combination that results in the highest decrease in Entropy, i.e. the highest information gain, is chosen for that split. A visualization of our trained Decision Tree can be found in Appendix C. The sci-kit learn library can give us the the information gain for each feature, normalized by the amount of nodes the feature appears in and the amount of instances that reach those node. The normalized information gain for our features is as follows:

- step size of `TXT` query response size: 0.8336
- growth in `ANY` query response size: 0.0869
- minimum `ANY` query response size: 0.0653
- maximum `TXT` query response size: 0.0142

From these results, we can see that the step size of the response size to a `TXT` query is by far the most important feature for the decision tree. Upon closer inspection of the crafted domains that are miss-classified by the 3 best classifiers, we notice that their sudden growth in `TXT` (and therefore also

ANY) query response size, appear completely or partly after the abuse date. Therefore, these domains do not fit the definition of a crafted domain at the time of abuse (only after) and the important features described above have more or less the same values as the benign domains. This makes it impossible for the classifiers to correctly classify them. To get an idea of the discriminating power of the other features we developed, we omitted the 4 features used by our first decision tree from the training set and trained a new decision tree. The normalized information gain for these features is as follows:

- step size of ANY query response size: 8.515e-01
- growth in TXT query response size: 1.276e-01
- maximum ANY query response size: 2.008e-02
- amount of changes in ANY query response size: 7.687e-04
- average ANY query response size: 9.347e-05

The features that describe the average, minimum, and amount of changes in TXT response size added no information gain. Considering the fact that the most important features were already left out when training this new decision tree, we assume that these features do not provide much discriminating power to any of the classifiers.

Table 5.1 shows why the F2-score was chosen over purely the recall metric. While both BernoulliNB and MultinomialNB classifiers have a slightly lower amount of FN (and thus a good recall score), they have 22 and 23 FP respectively, compared to 0 FP by other classifiers. If these classifiers were to classify a large number of domains, e.g. an entire TLD, the list of crafted domains that they would produce would contain many benign domains. This makes it very hard for a third party to utilize this list in a DNS amplification preventing method.

Classifier	TN	FP	FN	TP	Recall	Precision	F2
KNeighborsClassifier	836	0	3	39	0.928571	1.000000	0.942029
AdaBoostClassifier	836	0	3	39	0.928571	1.000000	0.942029
DecisionTreeClassifier	836	0	3	39	0.928571	1.000000	0.942029
RandomForestClassifier	836	0	4	38	0.904762	1.000000	0.922330
SVC	836	0	4	38	0.904762	1.000000	0.922330
GaussianNB	836	0	5	37	0.880952	1.000000	0.902439
RadiusNeighborsClassifier	836	0	5	37	0.880952	1.000000	0.902439
MLPClassifier	836	0	5	37	0.880952	1.000000	0.902439
SGDClassifier	834	2	5	37	0.880952	0.948718	0.893720
BernoulliNB	814	22	2	40	0.952381	0.645161	0.869565
MultinomialNB	813	23	2	40	0.952381	0.634921	0.865801
GradientBoostingClassifier	836	0	7	35	0.833333	1.000000	0.862069

Table 5.1: Classifier performance on the test data.

5.3.2 Early detection

In this section we will present the performance of the classifiers when detecting crafted domains in an early stage. In increasing steps, we delete data from before the initial abuse date after which we recalculate our features. This way we simulate a situation in which the amplification attack has yet to occur in order to analyze the capabilities of our classifier regarding the proactive detecting of crafted domains. Due to the computation complexity of predicting all the domains in the test set for each classifier, the steps in the simulation increase in size. First, the F2 score was calculated for each day in the week leading up to the abuse date after which we switched to steps of 7 days for 3 more steps (so 14, 21 and 28 days before the attack). After this, the steps were increased to 30 days up until the the maximum amount of days in our time series was reached. However, if a domain is abused early in the data range of our initial data set, the removal of days before this day of abuse will mean that

at one point there are no days left to remove and the features will all be 0. Therefore, these domains are removed from the data set, and the amount of domains that are left after each removal step are displayed in Fig. 5.16. The big drop in number of domains when we remove 210 days of data can be explained by the fact that 427 of the initial 878 domains in the test set are first abused on the 1st of May 2018, which is 181 days from the start of our data set. When we remove data up until 210 days before this date, these domains have no data left and are therefore removed.

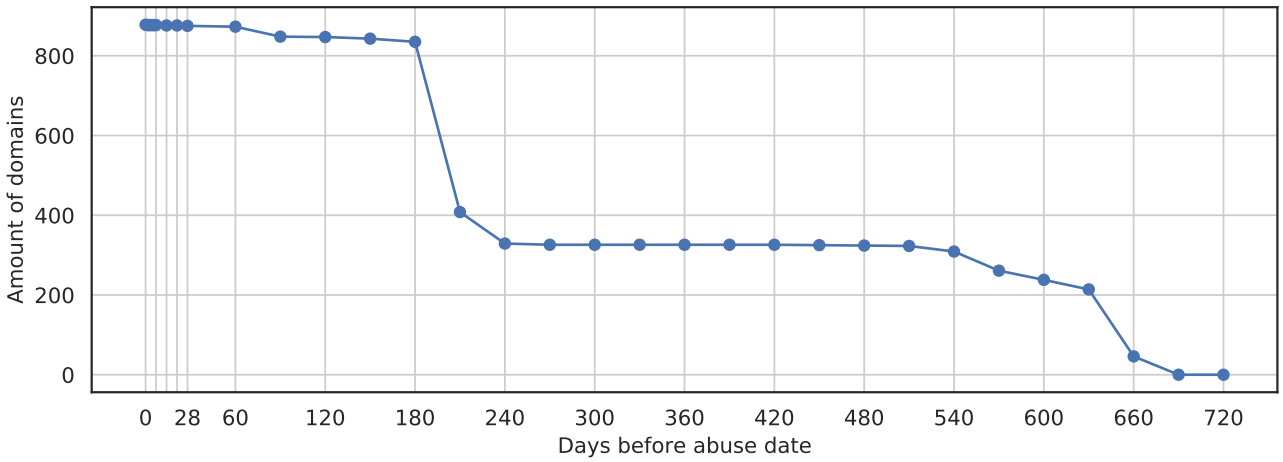


Figure 5.16: Amount of domains in test set after each data removal step

The initial F2-score at 0 removed days and the average F2-score over all the simulations can be seen in Table 5.2. On average, the AdaBoostClassifier scores the best among all classifiers, closely followed by the DecisionTreeClassifier.

Since a graph displaying all the F2-scores for all the classifiers would become too cluttered, we have plotted the F2-scores of the top 5 classifiers with the highest average F2-score. The classifiers in the top 5 of initial F2-scores also happen to have the highest average F2-score, meaning that the other classifiers do not perform significantly better once more data is removed. The resulting F2 score for these classifiers at every step of the simulation can be seen in Fig. 5.17. While at each step of the proactive detection simulation the performance of the classifiers differ, the AdaBoost and DecisionTree classifiers perform the best up until 180 days before the initial attack. Furthermore, the KNeighbors classifiers, which shared the top spot for best initial score with the AdaBoost and DecisionTree classifiers, drops much quicker in F2-score and therefore has a significantly lower average F2-score.

All the classifiers show a similar pattern in F2 score; For the first 14 days, all classifiers results stay the same. Then, when they need to classify domains 21 days before the initial attack, the F2 scores drop. At this step, the same domains that were miss-classified before are miss-classified again. However, since only data from 21 days before the abuse date is considered, a few more crafted domains have their discriminating characteristics (i.e. a sudden growth in query response size) missing from the features.

Classifier	Average F2-score	Initial F2-score
AdaBoostClassifier	0.647996	0.942029
DecisionTreeClassifier	0.646386	0.942029
KNeighborsClassifier	0.638064	0.942029
RandomForestClassifier	0.634846	0.922330
SVC	0.631624	0.922330
MLPClassifier	0.630273	0.902439
BernoulliNB	0.625707	0.869565
GaussianNB	0.624786	0.902439
SGDClassifier	0.619295	0.893720
MultinomialNB	0.615364	0.865801
RadiusNeighborsClassifier	0.611088	0.902439
GradientBoostingClassifier	0.597873	0.862069

Table 5.2: Average performance over simulations

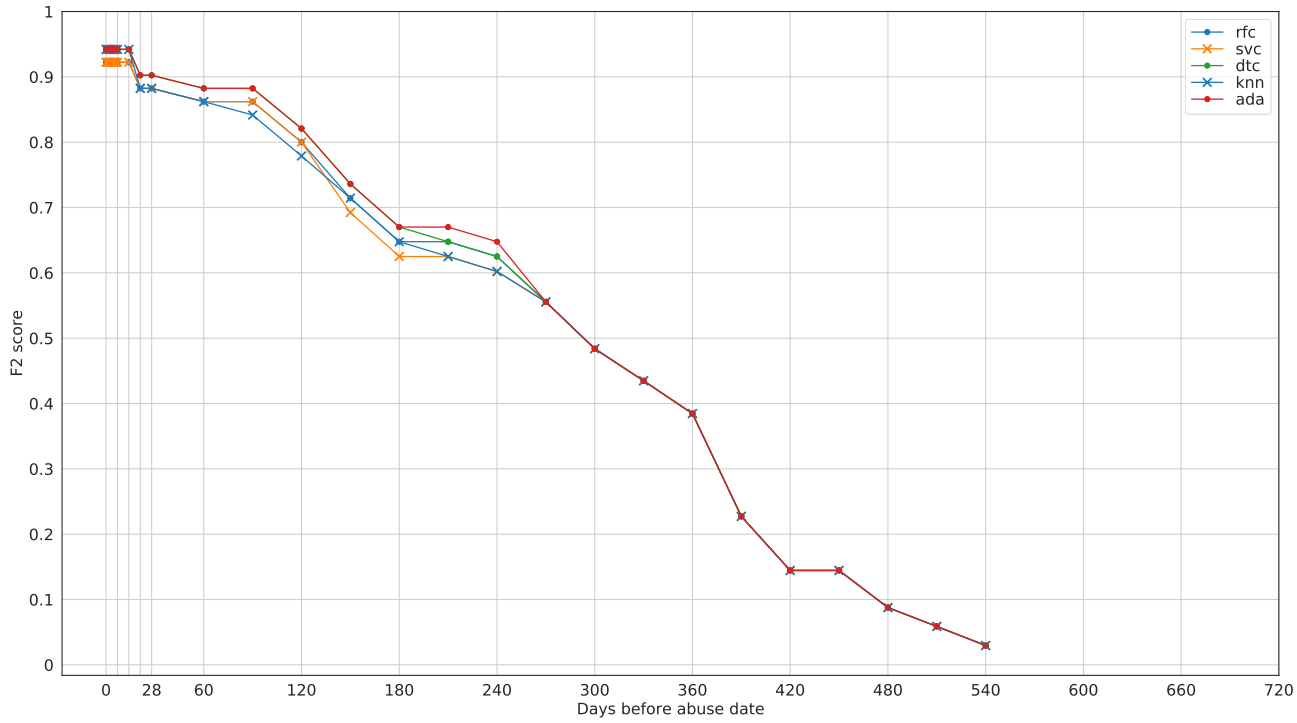


Figure 5.17: Classifier F2 score for each data removal step

In fact, almost all of the drops in F2 scores have the same cause. That is, at each step, more crafted domains have their typical characteristics fall outside of the time frame that is used to calculate the features (and do not fit the definition of a crafted domain anymore). This observation is supported by the fact that almost all classifiers make no mistakes in labeling the benign domains (i.e. they have no FP). This can be seen in Fig. 5.18, which shows that all classifiers except SGD, MultinomialNB, and BernoulliNB have a perfect precision score, regardless of the time between classification and time of abuse. At 570 days before the attack or more, all crafted domains are labeled as benign, and the precision drops to zero as there are no more TP. Still, we checked for FP at the last 4 steps, as precision can be 0 when there are FP but no more TP.

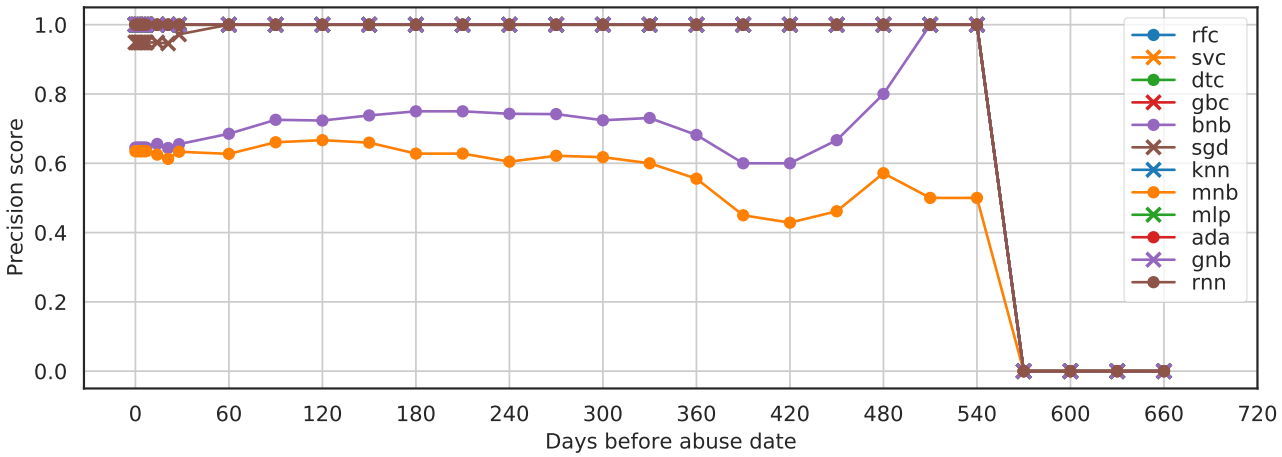


Figure 5.18: Classifier Precision score for each data removal step

Since our test set contains only 139 crafted domains, the F2-scores of each classifier could differ quite significantly if a different train/test split is used. At every new split, a different number of domains that have their discriminating characteristics after the abuse date could end up in the test. Therefore we measured the F2-score of the classifiers for 6 different splits at 0 days and 540 days before the abuse date. These 2 steps of the simulation were chosen as they highlight the best and worst possible performance of all the classifiers. Due to computational power, we could not do this for all the steps of the simulation. Fig. 5.19 shows the mean and standard deviation of the 6 runs. While some classifiers have an average score at 0 days prior to the attack that is slightly different than our initial run (see Table 5.1), the top 5 is still the same as the one we chose for our analysis in Fig. 5.17.

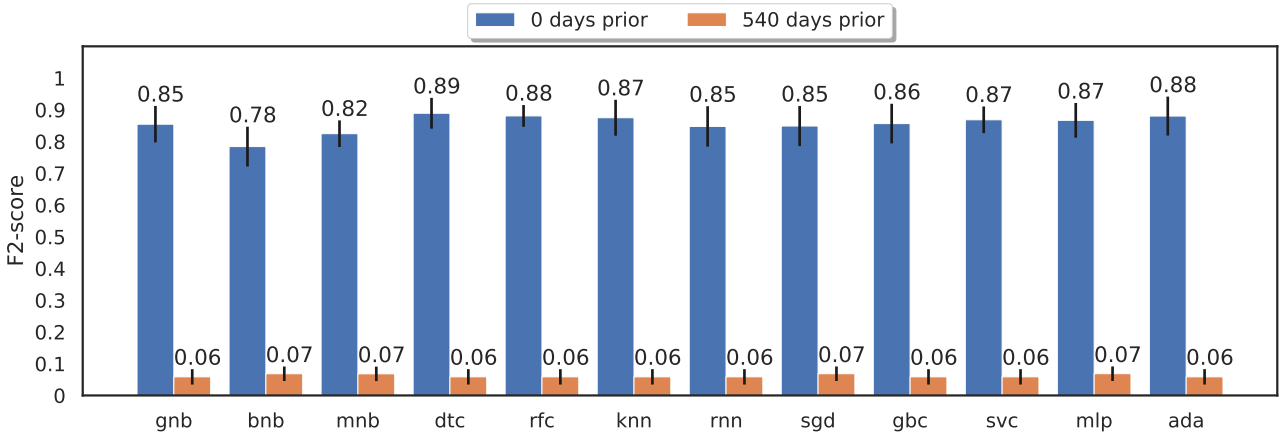


Figure 5.19: Mean and std. deviation of F2-scores at different train/test splits.

6 Discussion

In this section we will discuss our interpretation of the results. We start with the selection of the best classifier, followed by a discussion on the time advantage of our approach compared to the more common reactive solutions. Next we present the limitations of our work and discuss possible solution for future work.

6.1 Best classifier

As shown in the Section 5.3, the DecisionTree and AdaBoost classifiers perform the best amongst all classifiers up until 180 days before the initial attack. While some classifiers perform better at detecting crafted domains at an even earlier stage, their best possible performance in detecting these domains at any stage is much worse. Thus, the maximum number of crafted domains that these classifiers can detect before the attack is lower. We chose to value a reduction in missed crafted domains over slightly earlier detection. A crafted domain detection seven months prior to the attack instead of six months brings only a slight advantage, as six months is probably enough time to implement the information in a follow-up mitigation strategy. On the other hand, only a few missed crafted domains can fuel another devastating DNS amplification attack. Therefore the decision for best classifier was made between the DecisionTree and AdaBoost classifiers. This decision is primarily based on the time complexity of classifying an unseen instance (i.e. classification complexity). For a DecisionTree classifier, the time complexity for classifying an unseen instance is dependant on the depth of the tree [53]. The depth of a tree is defined as the longest path from the root node to a leaf node. In our case, the depth of the trained DecisionTree was 3. The AdaBoost classifier is a so-called ensemble classifier, which combines the output of multiple base classifiers in order to make a final classification. Therefore, its classification complexity is dependent on the amount of base classifiers and their time complexity for classification. [54]. Since our trained AdaBoost classifier is tuned to use 100 base classifiers (or estimators), its final time complexity will be significantly higher than the DecisionTree. The lower classification complexity of the DecisionTree classifier makes it more suited for detecting crafted domains among a large set of domains, such as a TLD or even the entire DNS name space.

6.2 Time advantage

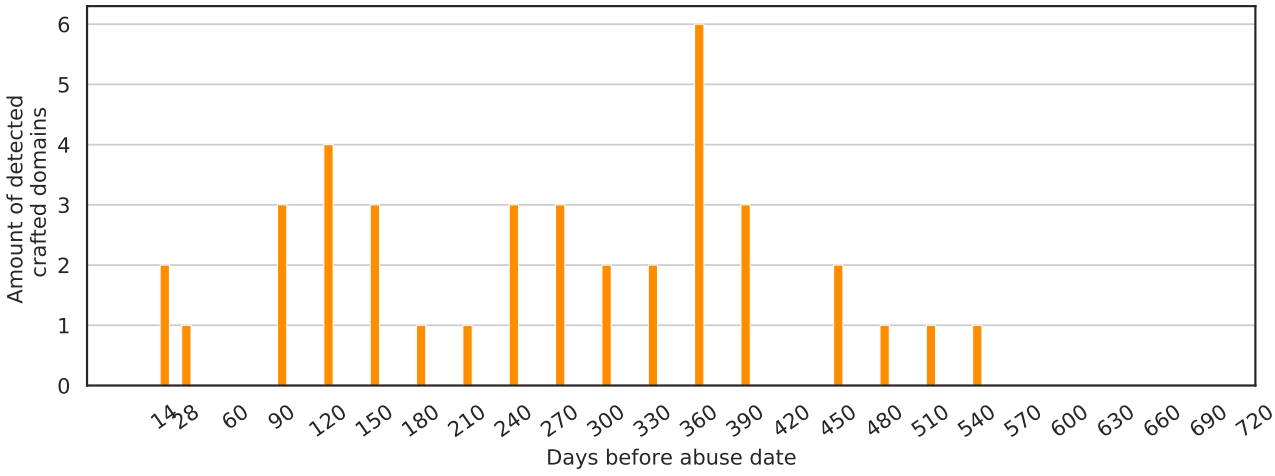


Figure 6.1: Early detection of crafted domains

In this section we will analyze the time advantage that our approach has compared to reactive DNS amplification solutions, such as RRL or absorption clouds. Since these solutions are only effective once an attack has started, we define our time advantage as the window of time between the detection of a crafted domain by our method and the initial abuse date. This time advantage was analyzed for the trained DecisionTree classifier, as we have concluded it to be the best. Fig. 6.1 shows the amount of unique crafted domains that were detected against the amount of days this detection occurred before the initial attack. In total, 39 out of the 42 crafted domains could be detected at least 14 days before the initial attack, with the largest possible time advantage being 540 days. The 3 undetected crafted domains were already analyzed in Section 5.3.1, were we saw that a sudden peak in query response size came after their initial abuse date. This means that these domains only fitted our definition of a crafted domain *after* the initial attack took place. The impact of these undetected domains in a DNS amplification attack (performed at the date of abuse in our data set) would therefore be minimal. With this in mind we can answer RQ4: *does the presented method detect crafted domains before they are misused? If yes, what is the time advantage?* Our method is able to detect crafted domains—which fit our definition of a crafted domain at the time of classification—with a time advantage between 14 and 540 days.

6.3 Limitations

While the results of our research are promising, it also comes with some limitations. First, only a time frame of two years was taken from the domains in the CCC data set due to computational, storage, and network limitations. In order to filter out domains that had a large response size, all the OpenINTEL data that was available on the domains in the CCC data set had to be transformed into query response estimations. Although this step was performed on the OpenINTEL cluster, it still took a considerable amount of time. Once these estimations had been calculated, they had to be queried from the cluster to a local machine in order to calculate our features. We are aware that this could also be performed on the cluster using a framework for distributed computing. However, learning to use this framework was out of the scope for this thesis. Second, we have measured the performance of the classifiers on a data set that we labeled ourselves instead of using a data set of known crafted and benign domains. While we do think that our method for generating this labeled set is accurate, we were unable to verify if the domains labeled as crafted were actually crafted. To our knowledge, no such list of crafted domains has been made public. However, a number of booters (DDoS-for-hire services) have been tracked down all over the world in the last couple of years. Evidence that was gathered by law enforcement, such as the domain names that were owned by these services, could help us verify if our method of generating a labeled data set is indeed correct.

6.4 Future work

To see if classifier results can be improved by training on a larger time frame, query response size estimations could be calculated for all the data on our labeled domains that is available in the OpenINTEL project. To enable large scale detection of crafted domains, our presented methodology has to be altered slightly. First, the response size estimations need to be calculated over all the data in the OpenINTEL project. While this could take up to a few days to calculate, it has to be performed only once. Subsequent response size estimations could be calculated after the daily measurements. Second, the feature calculation has to be rewritten in a framework for distributed computing in order to convert the large amount of data into features. Finally, the DecisionTree classifier should also be implemented in such a framework, as the classification of multiple million domains will require large amounts of processing power. Fortunately, Apache Spark, an open-source distributed cluster-computer framework, has a machine learning library that contains multiple types of classifiers, including a DecisionTree implementation [8].

[8] https://spark.apache.org/docs/2.2.0/mllib-decision-tree.html

7 Conclusion

The goal of this study was to design a method to proactively detect so-called crafted domains before they can be misused in a DNS amplification attack. In order to achieve these goals the following research questions were defined:

- **RQ1:** What are the defining characteristics of domains misused in DNS amplification attacks?
- **RQ2:** What is the definition of a crafted domain?
- **RQ3:** How can the behaviour of crafted domains be automatically detected?
- **RQ4:** Does the presented method detect crafted domains before they are misused? If yes, what is the time advantage?

We answered RQ1 in multiple steps. First, we developed a method to estimate the response size to different query types from the measurements of the OpenINTEL project. Next, we created a data set consisting of a random selection of abused domains from the CCC data set and a random selection of domains from the three major generic TLDs. Finally, we used our query response size estimation on this data set in order to perform an analysis of the size characteristics of the two domain groups. From this analysis, a number of characteristics of abused domains were found, such as a higher minimum, maximum and mean response size to both a `TXT` and `ANY` query.

To answer RQ2, we generated a data set that contained two types of domains; Alexa top 1M domains that *were* in the CCC data set and domains from the CCC dataset that were *never* in the Alexa top 1M (and could potentially be crafted). By analyzing the CDF of `ANY` query response sizes of the abused Alexa domains, we concluded that a crafted domain should have a response size to some query of at least 640 bytes. After gathering domains from our data set that passed this requirement, we levered the power of K-means clustering to discover that there were a number of clusters with a pattern of extreme growth in the group of potentially crafted domains. After a more in-depth inspection of this growth pattern, we updated our definition of a crafted domain to include this growth and came to our final definition:

Definition 7.1 *Crafted domain: A non-DNSSEC-signed domain in the Domain Name System that is purposely configured to give an unusually large response to a certain query type (> 640 bytes). Furthermore, the response size shows an extreme amount of relative growth (> 430%), whether slowly over time or in sudden spikes.*

A machine learning approach was taken and performed on a labeled data set of crafted and benign domains to answer RQ3. We were able to generate this final data set of crafted and benign domains by gathering domains from the CCC data set that fitted the definition which we presented to answer RQ2. The characteristics from our answer to RQ1 were used to generate a set of features that enabled us to perform feature-based TSC. By calculating these features over data that is available *before* the domains are misused, we simulated a pro-active approach. We trained and tuned a set of 12 classifiers on these features and analyzed their performance at periods in time before an attack. While the AdaBoost and DecisionTree classifiers showed similar performance, it was argued that the DecisionTree classifier is more suited for the classification of the entire DNS namespace due to its low time complexity. This classifier is able to identify crafted domains with an F2-score of 0.94 between 0 and 14 days before the attack, while creating no FP over the whole time frame of the test set. Using this method, we were able to gain a time advantage of 540 days compared to reactive DNS amplifcation solutions, giving us an answer to RQ4.

These results are promising for the prevention of DNS amplification attacks. The time advantage gives enough time to take further action against domains that are identified as crafted, while the lack of FP means that acting on this information will not affect benign domains.

References

[1] S. Waterman, "Arbor: DDoS attacks growing faster in size, complexity," 2017. [Online]. Available: https://www.cyberscoop.com/ddos-attacks-growing-arbor-networks/

[2] P. Paganini, "Github hit by the biggest-ever DDoS attack that peaked 1.35 Tbs," March 2018. [Online]. Available: https://securityaffairs.co/wordpress/69762/hacking/github-largest-ddos-attack.html

[3] ——, "Swedish transport agencies targeted in DDoS cyber attacks," 2017. [Online]. Available: https://securityaffairs.co/wordpress/64317/cyber-warfare-2/swedish-transport-agencies-ddos.html

[4] Z. Zorz, "Hackers who DDoSed African telecom and US hospital get long prison sentences - Help Net Security," 2019. [Online]. Available: https://www.helpnetsecurity.com/2019/01/14/ddos-attacks-prison-sentences/

[5] E. Leverett and A. Kaplan, "Towards estimating the untapped potential: a global malicious DDoS mean capacity estimate," *Journal of Cyber Policy*, vol. 2, no. 2, pp. 195–208, 2017.

[6] B. Viglarolo, "World record DDoS attack hits 1.7 Tbps, thanks to Memcached flaw," 2018. [Online]. Available: https://www.techrepublic.com/article/world-record-ddos-attack-hits-1-7-tbps-thanks-to-memcached-flaw/

[7] Nexusguard, "DDoS Threats Report 2019 Q1," Tech. Rep., 2019. [Online]. Available: https://www.nexusguard.com/threat-report-q1-2019

[8] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC and its potential for DDoS attacks," 2014, pp. 449–460. [Online]. Available: http://dx.doi.org/10.1145/2663716.2663731.

[9] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing," 2000. [Online]. Available: https://tools.ietf.org/html/bcp38

[10] M. Jonker, A. Sperotto, R. Van Rijswijk-Deij, R. Sadre, and A. Pras, "Measuring the adoption of DDoS protection services," *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, vol. 14-16-Nove, pp. 279–285, 2016.

[11] A. Sperotto, O. Van Der Toorn, and R. Van Rijswijk-Deij, "TIDE - Threat identification using active DNS measurements," in *SIGCOMM Posters and Demos 2017 - Proceedings of the 2017 SIGCOMM Posters and Demos, Part of SIGCOMM 2017*, 2017, pp. 65–67. [Online]. Available: https://doi.org/10.1145/3123878.3131988

[12] R. Van Rijswijk-Deij, M. Jonker, A. Sperotto, and A. Pras, "A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 34, no. 6, 2016. [Online]. Available: http://www.ieee.org/publications{_}standards/publications/rights/index.html

[13] CloudFlare, "DNS Zone." [Online]. Available: https://www.cloudflare.com/learning/dns/glossary/dns-zone/

[14] S. M. Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures," *International Workshop on Security in Parallel and Distributed Systems*, no. 9, pp. 543–550, 2004.

[15] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, p. 39, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=997150.997156

[16] T. Rozekrans, M. Mekking, and J. de Koning, "Defending against DNS reflection amplification attacks," 2013. [Online]. Available: https://www.nlnetlabs.nl/downloads/publications/report-rrl-dekoning-rozekrans.pdf

[17] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," 2014. [Online]. Available: https://christian-rossow.de/publications/amplification-ndss2014.pdf

[18] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.

[19] G. A. Susto, A. Cenedese, and M. Terzi, *Time-Series Classification Methods: Review and Applications to Power Systems Data*, 2017, no. March 2019.

[20] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 12, pp. 3026–3037, 2014.

[21] M. Geva, A. Herzberg, and Y. Gev, "Bandwidth distributed denial of service: Attacks and defenses," *IEEE Security and Privacy*, vol. 12, no. 1, pp. 54–61, 2014.

[22] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *Operating Systems Review (ACM)*, vol. 35, no. 5, pp. 131–145, 2001.

[23] Netscout, "NETSCOUT's 14th Annual Worldwide Infrastructure Security Report," *Netscout*, 2019. [Online]. Available: https://www.netscout.com/press-releases/netscout-releases-14th-annual-worldwide-infrastructure

[24] M. Geva, "Ensuring qos during bandwidth ddos attacks," Ph.D. dissertation, Bar Ilan University. Department of Mathematics and Computer Science, 2013.

[25] D. C. Macfarland, C. A. Shue, and A. J. Kalafut, "Characterizing optimal DNS amplification attacks and effective mitigation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8995, pp. 15–27, 2015.

[26] ICANN Security and Stability Advisory Committee, "SSAC Advisory on DDoS Attacks Leveraging DNS Infrastructure," Tech. Rep. February, 2014. [Online]. Available: https://www.icann.org/en/system/files/files/sac-065-en.pdf

[27] US-CERT, "DNS Amplification Attacks," 2013. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA13-088A

[28] Centre for Applied Internet Data Analysis, "The Spoofer Project," 2015. [Online]. Available: https://spoofer.caida.org/summary.php

[29] F. Weimer, "Passive dns replication," in *FIRST conference on computer security incident*, 2005, p. 98.

[30] L. Bilge, E. Kirda, C. Kruegel, M. Balduzzi, and S. Antipolis, "EXPOSURE : Finding Malicious Domains Using Passive DNS Analysis," *Ndss*, pp. 1–17, 2011. [Online]. Available: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:EXPOSURE+:+Finding+Malicious+Domains+Using+Passive+DNS+Analysis{#}0

[31] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," *Neural Processing Letters*, vol. 48, no. 3, pp. 1347–1357, Dec 2018. [Online]. Available: https://doi.org/10.1007/s11063-017-9666-7

[32] T. Frosch, M. Kührer, and T. Holz, "Predentifier: Detecting botnet c&c domains from passive dns data," in *Advances in IT Early Warning*. Fraunhofer Verlag, 2013.

[33] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive DNS traces," *Proceedings - Annual Computer Security Applications Conference, ACSAC*, pp. 311–320, 2009.

[34] M. Félegyházi, C. Kreibich, and V. Paxson, "On the potential of proactive domain blacklisting," *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more (LEET 2010)*, pp. 6–6, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1855692

[35] Y. He, Z. Zhong, S. Krasser, and Y. Tang, "Mining DNS for malicious domain registrations," *Proceedings of the 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2010*, 2010.

[36] S. Hao, A. Kantchelian, B. Miller, V. Paxson, and N. Feamster, "PREDATOR: Proactive recognition and elimination of domain abuse at time-of-registration," *Proceedings of the ACM Conference on Computer and Communications Security*, vol. 24-28-Octo, pp. 1568–1579, 2016.

[37] O. Van Der Toorn, R. Van Rijswijk-Deij, B. Geesink, and A. Sperotto, "Melting the snow: Using active DNS measurements to detect snowshoe spam domains," in *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 2018, pp. 1–9. [Online]. Available: https://openintel.nl/

[38] H.-V. Nguyen and Y. Choi, "Proactive detection of ddos attacks utilizing k-nn classifier in an anti-ddos framework," 2010.

[39] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting ddos attacks," in *Advances in Network Security and Applications*, D. C. Wyld, M. Wozniak, N. Chaki, N. Meghanathan, and D. Nagamalai, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 441–452.

[40] Y.-C. Wu, H.-R. Tseng, W. Yang, and R.-H. Jan, "Ddos detection and traceback with decision tree and grey relational analysis," *IJAHUC*, vol. 7, pp. 121–136, 03 2011.

[41] A. Sahi, D. Lai, Y. Li, and M. Diykh, "An efficient ddos tcp flood attack detection and prevention system in a cloud environment," *IEEE Access*, vol. 5, pp. 6036–6048, 2017.

[42] D. R. Thomas, R. Clayton, and A. R. Beresford, "1000 days of udp amplification ddos attacks," in *2017 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2017, pp. 79–84.

[43] D. Cornell, "DNS Amplification Attacks," 2014. [Online]. Available: https://umbrella.cisco.com/blog/dns-amplification-attacks/

[44] L. K. Saul and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *Journal of machine learning research*, vol. 4, no. Jun, pp. 119–155, 2003.

[45] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[46] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[47] S. Brahma, R. Kavasseri, H. Cao, N. R. Chaudhuri, T. Alexopoulos, and Y. Cui, "Real-time identification of dynamic events in power systems using pmu data, and potential applications—models, promises, and challenges," *IEEE transactions on Power Delivery*, vol. 32, no. 1, pp. 294–301, 2016.

[48] S. Alshahrani, M. Abbod, and B. Alamri, "Detection and classification of power quality events based on wavelet transform and artificial neural networks for smart grids," in *2015 Saudi Arabia Smart Grid (SASG)*, 2015, pp. 1–6.

[49] C. Fan, F. Xiao, and S. Wang, "Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques," *Applied Energy*, vol. 127, pp. 1 – 10, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261914003596

[50] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *2014 7th international symposium on resilient control systems (ISRCS)*. IEEE, 2014, pp. 1–8.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[52] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection."

[53] P.-N. Tan, M. Steinbach, and V. Kumar, "Classification: basic concepts, decision trees, and model evaluation," *Introduction to data mining*, vol. 1, pp. 145–205, 2006.

[54] A. L. Prodromidis and S. J. Stolfo, "Cost complexity-based pruning of ensemble classifiers," *Knowledge and Information Systems*, vol. 3, no. 4, pp. 449–469, 2001.

Appendices

A DNSSEC

To remove signed domains from our data set, we looked at the whole date range of each domain to see if at any time it contained either `DNSKEY` or `RRSIG` records. If so, the domain were flagged as signed and kept separate from domains that have never contained such record. However, on inspection of the average signed domain (see Fig. A.1) it became clear that our approach for deleting signed domains might not be correct. The figure shows that an average signed domain in the CCC data set has 96.39 bytes of signatures. However, the smallest signature size at the moment is 64 bytes, meaning that a signed domain on average has only 1,5 signature. Combined with the fact that each resource record group needs its own signature, this seemed very unlikely.

Fig. A.2 shows the amount of signed domains in each data set. While the amount of signed domains in the sample set grows only slightly, the amount of signed domains in the CCC data set grows very suddenly. A possible explanation could be that a DNS operator (or DNS Hosting Provider) added DNSSEC support and automatically signed the domains under its control. The domains could also be owned by the same company, which decided to sign their domains. Since these domains are signed at the end of the time frame, they do not contain keys and signatures at most days in the time span. This brings down the average size drastically.

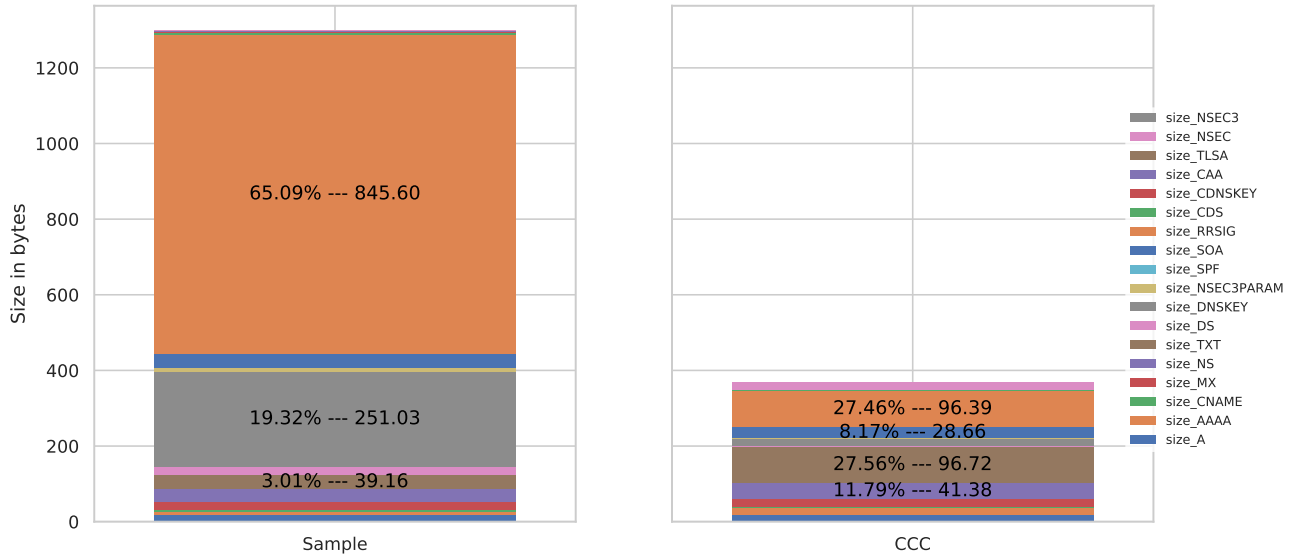


Figure A.1: Initial average size of signed domains in sample and CCC dataset

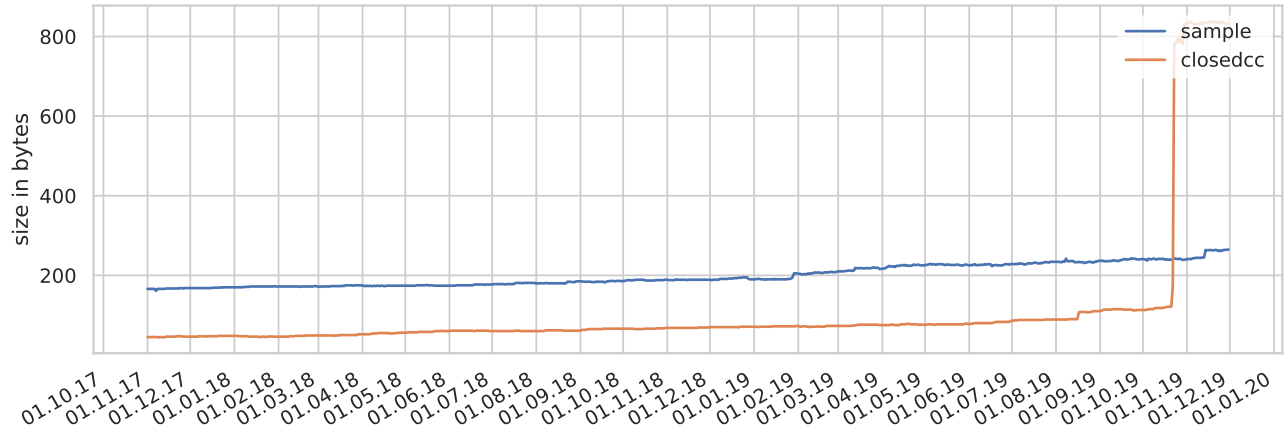


Figure A.2: Number of signed domains in sample and CCC dataset

Therefore, to get a better idea of what an average signed domain looks like, only the dates on which these domains contained `DNSKEY` or `RRSIG` records were considered for the average. The results can be seen in Fig. A.3. Still, the signed domains in the CCC data set are significantly smaller than the ones in our sample set. This is due to the fact that many of the `DNSKEY` records in CCC domains contain Elliptic Curve Crypography (ECC) keys, instead of the much larger RSA keys (see Fig. A.4. Furthermore, these keys also result in much smaller signatures which ultimately results in a much smaller response to an `ANY` query

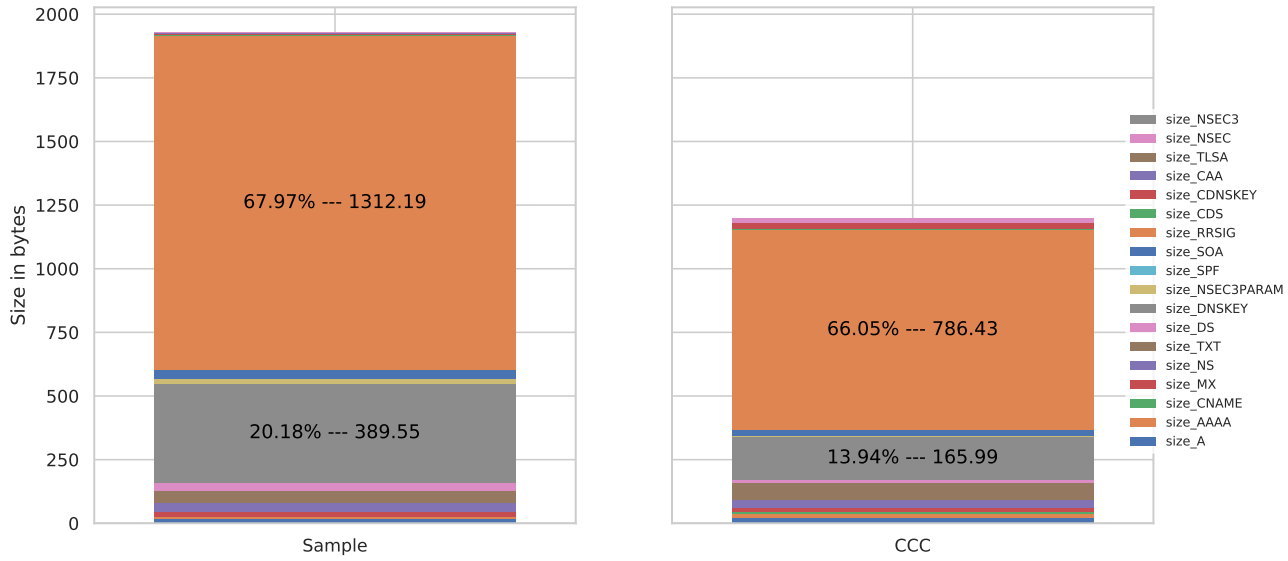


Figure A.3: Average size of signed domains on days that they contain signatures or `dnskey` records

Figure A.4: Amount of domains that contain `dnskey` records using the different algorithms

B Hyperparameter tuning

Classifier	Parameter	Values	Chosen Value
BernoulliNB	alpha	[0, 1e-06, 1e-05, 1e-04, 1e-03, 1e-02, 1e-01, 0.15, 0.2, 0.25, 0.3, 0.5, 1, 5]	0
GaussianNB	var_smoothing	[1e-11, 1e-10, 1e-09, 1e-08, 1e-07, 1e-06, 1e-05, 1e-04, 1e-03, 1e-02, 1e-01, 1e+00]	1e-04
MultinomialNB	alpha	[0, 1e-05, 1e-04, 1e-03, 1e-02, 1e-01, 0.15,0.2,0.25,0.3,0.5,1,5]	0
DecisionTreeClassifier	criterion	['gini', 'entropy']	'entropy'
	splitter	['best', 'random']	'best'
	max_depth	[None,1, 2, 3, 4, 5]	3
	max_features	['sqrt','log2',1, 2, 3, 4,5, 6,7,8,9,10,11,12]	8
RandomForestClassifier	n_estimators	[10,100,500]	10
	criterion	['gini', 'entropy']	'entropy'
	max_depth	[None,1, 2, 3, 4, 5]	3
	max_features	['sqrt','log2',1, 2, 3, 4,5, 6,7,8,9,10,11,12]	12
KNeighborsClassifier	n_neighbors	[1,2,3,4,5,6,7,8,9]	5
	weights	['uniform','distance']	'uniform'
	algorithm	['auto', 'brute', 'ball_tree', 'kd_tree']	auto
	metric	['manhattan','euclidean','chebyshev']	'manhattan'
	leaf_size	[10,20,30,40,50]	*only for algorithms ball_tree and kd_tree*
RadiusNeighborsClassifier	radius	[0.001,0.01,0.1,0.5,1,10,100, 500,1000,10000]	1000
	weights	['uniform','distance'],	'distance'
	algorithm	['auto','brute','ball_tree','kd_tree']	'auto'
	metric	['manhattan','euclidean','chebyshev']	'manhattan'
	leaf_size	[10,20,30,40,50]`	*only for algorithms ball_tree and kd_tree*
GradientBoostingClassifier	loss	['deviance','exponential']	'exponential'
	learning_rate	[0.0001,0.001,0.01,0.1,0.5,1]	0.5
	n_estimators	[1,10,100,1000]	100
	subsample	[0.001,0.001,0.1,0.5,1,5,10,100]	0.5
	criterion	['friedman_mse','mse','mae']	'mae'
SGDClassifier	loss	['hinge', 'log', 'modified_huber', 'squared_hinge', 'perceptron']	'squared_hinge'
	penalty	['l2', 'l1', 'elasticnet']	'elasticnet'
	alpha	[1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3]	1e-4
	l1_ratio	[0.0001,0.001,0.1,0.15,0.20,0.25, 0.30,0.35,0.4,0.5,0.6,0.7,0.8]	0.4
SVC	kernel	['linear','rbf','poly','sigmoid']	'poly'
	C	[1e-04, 1e-03, 1e-02, 1e-01, 1e+00, 1e+01, 1e+02, 1e+03]	10

Continued on next page

Classifier	**Parameter**	**Values**	**Chosen Value**
	gamma	[0.001 0.01 0.1 'scale' 'auto']	'scale' *(only for kernel = poly,rbf,sigmoid)*
	degree	[1,2,3,4,5]	5 *(only for kernel =poly)*
	coef0	[0,1,2]	*(only for kernel = sigmoid)*
MLPClassifier	solver	['lbfgs','sgd','adam']	'sgd'
	activation	['tanh', 'relu', 'logistic']	'relu'
	alpha	[0.0001,0.001,0.01,0.1]	0.001
	learning_rate	['constant','invscaling','adaptive']	'constant' (only for solver = sgd)
AdaBoostClassifier	base_estimator	GaussianNB(), DecisionTreeClassifier(), KNeighborsClassifier(), SGDClassifier(), SVC(), MLPClassifier()	GaussianNB()
	n_stimators	[1,5,10,25,50,100],	100
	learning_rate	[0.001,0.01,0.05,0.1,0.3,1]	0.01

Table B.1: Tuned hyperparameters

C Decision Tree

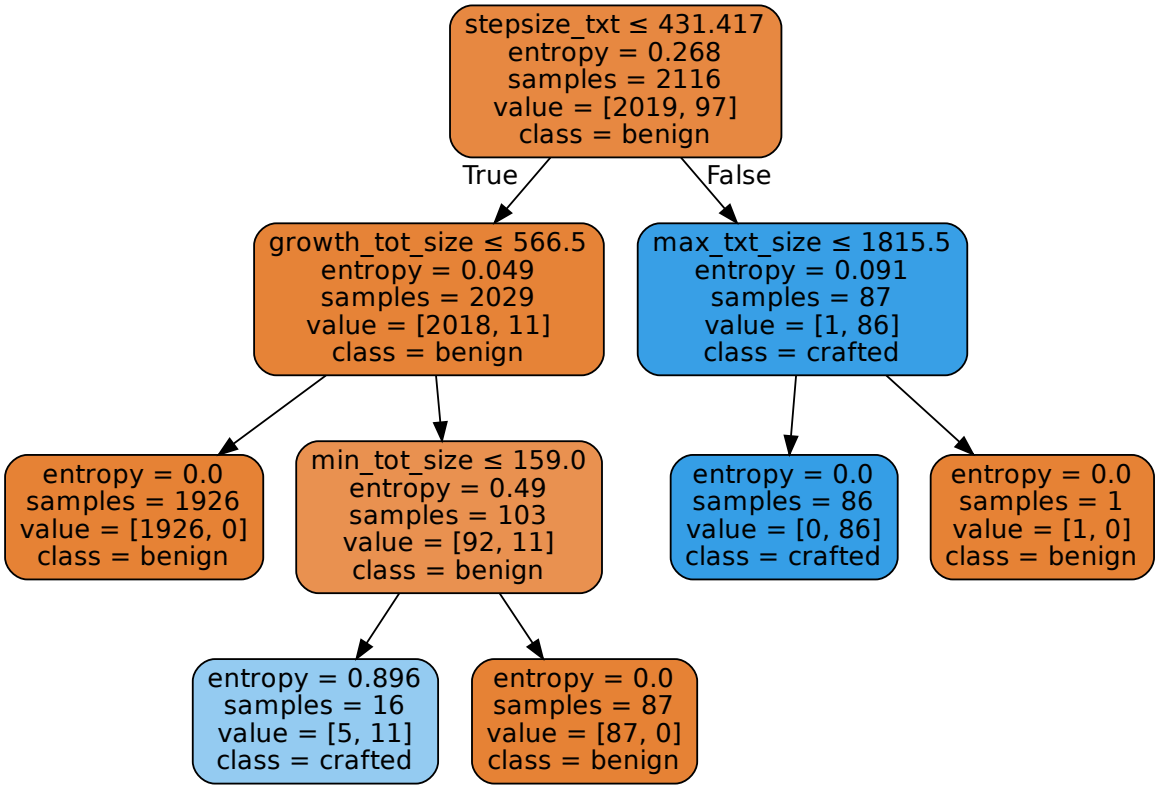


Figure C.1: Visualization of the trained Decision Tree