UNIVERSITÉ LIBRE DE BRUXELLES

MASTER THESIS

Bonet Detection Through Passive DNS analysis

Author:
Gérard TIO NOGUERAS

Supervisor:
Prof. Jean-Noël COLIN

A thesis submitted in fulfillment of the requirements for the degree of Masters in Cyber Security

Declaration of Authorship

I, Gérard Tio Nogueras, declare that this thesis titled, “Bonet Detection Through Passive DNS analysis” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master’s degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Gérard Tio Nogueras

Date: 19/08/2019

"The Domain Name Server is the Achilles heel of the Web. The important thing is that it's managed responsibly."

Tim Berners-Lee

UNIVERSITÉ LIBRE DE BRUXELLES

Abstract

Faculty of Science
Cyber Security

Masters in Cyber Security

Bonet Detection Through Passive DNS analysis

by Gérard TIO NOGUERAS

In the recent years, botnets have become the biggest cyber threat. For this reason a large part of the security scientific community has focused on their behavior and finding ways to detect them. Furthermore, in the recent years, bots have developed evasion techniques to hide their activity and make them more robust. This pushed another tendency focused in detecting botnets based on their evasion systems. In this paper, we aim to provide environments with smaller capabilities, a classifier, using a supervised approach with a very specific set of features, capable of detecting botnets activity through monitoring the DNS traffic.

keywords: botnet detection; botnet detection model; machine learning-based botnet detection; domain generation algorithm; botnet detection; fast flux botnet detection

Acknowledgements

Thanks to my project supervisor for staying positive and bearing with me for 2 years with a very low presence from my side. He provided me very valuable and critical information to make sure I would have the best chances to succeed.

Thank you to the large scientific community interested in botnet detection which has been cited in this paper. You have provided us with incredible information, the papers I have read during the whole process have inspired me with ideas for the thesis and helped me become more skilled at my current position as a Threat Hunter.

Thanks to my family and friends for the support during this time.

And finally thanks to my partner, Celine which has endured a lot during the writing of this paper.

Contents

List of Abbreviations

CnC	**C**ommand and **C**ontrol
DoS	**D**enial **o**f **S**ervice
DDoS	**D**istributed **D**enial **o**f **S**ervice
IRC	**I**nternet **R**elay **C**hat
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
IDS	**I**ntrusion **D**etection **S**ystem
RR	**R**esource **R**ecord
DNS	**D**omain **N**ame **S**ystem
TLD	**T**op **L**evel **D**omain
2LD	**2**nd **L**evel **D**omain
3LD	**3**rd **L**evel **D**omain
TTL	**T**ime-**t**o-**L**ive
CDN	**C**ontent **D**elivery **N**etwork
FFSN	**F**ast **F**lux
DGA	**D**omain **G**eneration **A**lgorithm
FQDN	**F**ully **Q**ualified **D**omain **N**ame
ML	**M**achine **L**earning
SVM	**S**upport **V**ector **M**achine
PCA	**P**rincipal **C**omponent **A**nalysis
LDA	**L**inear **D**isciminant **A**nalysis
kNN	**K**-**N**earest **N**eighbors
DT	**D**ecision **T**ree
LR	**L**ogistic **R**egression
XGBoost	e**X**treme **G**radient **B**oosting
AdaBoost	**A**daptive **B**oosting
ANN	**A**rtificial **N**eural **N**etworks
AV	**A**nti**V**irus

Chapter 1

Introduction

1.1 Research context

Vulnerabilities keep growing making botnets even easier to spread, which means cheaper and more powerful [1].

If you follow the news in cyber security, there are 3 hot topics which seem to prevail: Data breaches, new vulnerabilities and botnets. Botnets have become a presence that can't be denied and which security firms and SOCs have started to give more focus to. This is a compilation of Botnet news by Trend Micro[2] and a list of the variants that come up [3].

1.2 Research question

My research tries to answer the following question: "**How can we detect Botnets through passive DNS traffic analysis?**" With the following sub-questions:

- what are the best features to detect botnets ?
- what are the most effective machine learning algorithms for the features used?
- What are the current trends of Evasion and detection?
- Can we create a solution that detects effectively all botnets ?
- Can I find features not yet exploited to improve detection?
- Can I create an all-in model fitted for purposes not yet covered by other solutions.
- What effective model can help organisations detect botnets faster and more reliably?

1.3 Thesis objective

The objective of this master thesis in Cyber Security is to try to improve the all-in solutions for botnet detection through DNS traffic analysis with a machine learning approach and acquire along the way correctly conduct a research process and learn machine learning techniques to solve interesting problems. Different papers haven't always provided a deep study of the choice of features or provided a model adapted to certain environments which we will try to do and see if this can improve the existing models.

The objective of this thesis is to answer the research questions above. This will be done by reviewing the current (scientific) literature on botnets, in particular with relation to DNS. It is hoped that others can benefit from the knowledge that this thesis provides to implement better security measures within their own organisations. This thesis will provide the reader with in depth knowledge on the subject of botnets, and how they the threat of them within an organization can be mitigated.

Why did we decide to focus on the DNS protocol to detect botnets? The objective of this thesis is to improve the detection of botnets. Botnets use a lot of different protocols but papers on botnets have shown that most modern botnets use the DNS protocol to evade detection. Therefore, we think the most efficient way to detect botnets is to actually analyze their evasion system. This is why we have decided to focus on studying the DNS traffic for botnet detection.

Chapter 2

State of the art

In this chapter, we will define all the concepts that are needed to understand and follow our experiment. We will start with an extensive review of botnets, the DNS protocol, the abuses of the DNS protocol, this will be followed with explanation of the different concepts in machine learning present during our experiment, and finally we will end the state of the art by presenting the solutions we have used as baseline to our project and what papers we have used to improve them.

2.1 Botnets

As stated in the introduction botnets are an important problem for anyone involved somehow with the internet. They can result in great economic damage [4]. Especially with their continuous improvement to become more resilient and powerful which makes them an even more important threat.

Botnets can become very lucrative and can infect very large amount of devices resulting in scary tool [**pheonix**]. Here are some examples of the magnitude they can reach: **Flashback** with 600k compromised targets, **Grum** with 840k compromised devices and sending 40 Billion spam emails per month, **TDL-4** with 4.5 Million victims in first the 3 months and **Gameover ZeuS** with 1 Million infections, because of its resilience mechanisms this botnet was one of the hardest to take down.

The reason botnets are still an ongoing research topic is that there isn't a complete solution for their detection and mitigation. Researchers and organisations have to keep working to keep updated with all the new flavors criminals bring to the market.

2.1.1 Definition

What is a botnet? A botnet is a network of infected machines with programs called bots, these bots owned and controlled by a remote attacker called the botmaster. Users get infected via the same vector attacks used by malware, email attachment, malicious website, unaware download, etc. When bots usually infect these machines in stealthy manner, staying as unnoticeable as possible[5]. The control of such bots is done through the Command and Control (CnC) server. The CnC server allows the master to issue commands to and receives responses from individual bots or aggregations of bots. These exchanges are done to update the software of the malware, execute attacks, ex-filtrate data and more actions explained down below[6][7].

What is a bot? Bots are small programs allowing to remotely control and perform commands on computers. They are the foundation of botnets. The paper presented by the SANS institute considers two types of bots. Bots used to perpetuate attacks,

bots used for their content and both.
A clear distinction between a bot agent and a common piece of malware lies within a bot's ability to communicate with a Command-and-Control (CnC) infrastructure. CnC allows a bot agent to receive new instructions and malicious capabilities, as dictated by a remote criminal entity. This compromised host then can be used as an unwilling participant in Internet crime as soon as it is linked into a botnet via that same CnC.
These programs are embedded with port scanning, vulnerability scanning, exploitation kits and payloads that allow them to spread the botnet and infect their victims[8].
There are many different families of bots, some very modular such as the Agobot others less complete but easier to use such as the SDBot family. Bots families are also classified depending on the channel type and attack type, for example GT-Bots are a IRC bots but there are a lot of different protocols exploited as botnets channels. These 3 families are the most often found. Lesser usual ones have specific functions or plugins to fill in the gaps left by developers to customize the bots, a good examples would be the Dataspy Network X bots. There are very small bots such as the Q8 Bots and Perl-based bots that still allow for a large range of commands and attacks. Finally some bots are composed of a single file like Kaiten bot which makes it very easy to upload to compromised machines.

Why do botnets provide more powerful attacks? Botnets give the control to the botmaster of two critical resources: CPU (processing power) and IP addresses (anonymity). Even if the use of CPU stays low on the infected machines, the aggregate of bots can provide power equivalent to supercomputers with the additional perk of executing traffic from different addresses instead of a single IP [9].

What is the purpose of botnets? All these resources make botnets very powerful to execute network attacks [10]. Cybercriminals use botnets to execute a long list of malicious activities and structure related actions, we have listed some of these but any type of cyber attack can be uploaded to these bots and executed [11].

What are the advances in botnets? Another reason botnets are a big threat is that criminals have started to provide botnets as a Service (BaaS) which are considered a big part of the botnet economy. This popularized botnets are sold to anyone, this has made them an even bigger threat that they already were [12]. This BaaS is possible with decentralized architectures that can subdivided into smaller botnets to sold and then reintegrated to the parent botnet after use.

What types of actions are performed by botnets? A victim host could be infected by targeting known vulnerability or by infected programs. When the victim is infected, the botnet will try to stay stealthy and with the exploit kit installed, it can do an extensive amount of damage. Here are some of the methods to control the infected hosts.
This first list presents general use of botnets:

- **Distributed Denial-of-Service Attacks** These attacks provoke a loss of service or connectivity. Used by hacktivist, criminals and companies to disturb targets for recognition, financial gain or advantage over competition respectively. (The services could be email servers, production servers, web servers but also any device reachable) [11]

- **Spam email campaigns** Bots are set as proxy nodes and then used to send large amounts of spam and phishing emails.[11]
- **Sniffing Traffic** Bots can start watching packets going through the compromised machine and start retrieving all valuable data passed in clear-text. Botnets have been even found to analyze others bots data and take over them if they belong to another botnet.
- **Spying through Keylogging and file monitoring** Sniffing packets effectiveness is reduced by encrypted traffic. The solution is then to log the key strokes made by the users and retrieve sensitive information. This is done with a keylogger and filtering mechanism that targets specific use-cases (logins, password, ...) [11]
- **Spreading new malware** Botnets growth depends on their ability to expand. Compromised targets have an important task to keep spreading the botnet. They can download malware or send viruses via email, there are various methods depending of the botnet type and environment they want to spread the malware.[11]
- **Installing Advertisement Addons, Browser Helper Objects (BHOs) and Google AdSense abuse** These techniques are used for financial gain instead of disruption. These are based on websites using clicking based ads. The criminals set fake sites using advertising programs and then automate the bots to click on them to create revenue. Since the bots have different IPs it is very hard to detect the fraud.
- **Manipulating online polls and games** These are known to exist to influence decisions and are expected to be further used in the future. This is also very effective since each bot uses different IP addresses.
- **Mass identity theft** Stealing personal data such as mail accounts, intellectual property, military secrets, embarrassing information or bank credential. This is a combination of all of the above that allow to create campaigns based on that data collected. These campaigns make this stolen data effective through fake websites and spear phishing email attacks. [**tracking**] Regarding the personal information that can be found by bots on the home desktops, this paper explains that it must not be overlooked. The amount of data that can be contained in home applications can be very important(taxes, browsers, email contacts. They warn of the sensitive information that is often stored on home computers related to their company. This could expose intellectual property that can be then sold by the criminals.
- **Host illegal sites** Child pornography and black market sites are some examples.
- **Computation for cryptanalysis** This is a less expected use of botnets, but these distributed supercomputers can be used for cryptanalysis purposes. Computing rainbow tables, cracking passwords, bruteforcing keys or mining crypto currencies. With the success of cryptocurrencies in the last 2 years, this type of use for botnets has increased. The latest example is the Smominru Monero mining botnet, mining around 9000 moneros worth at the time 3 million$[13] [**tracking**].

This second list targets activities of bots on the compromised machines to take full control[14]:

- **Secure the system(close NetBIOS shares, RPCDCOM) to avoid infection by other criminals** This could also mean remove existing bots on the machine. Bots will make sure their host is properly hardened to avoid overtake.

- **Redirect traffic for the botnet** Depending on the topology used, bots might be used as proxies to send commands, updates, data to the rest of the bots.
- **Kill unwanted process running on the system** This joins the hardening objective. Bots want to take full control and make sure no process is limiting their actions (usually trying to stay stealthy while doing it).
- **Test for virtual machines and/or debugger software** Part of the resilience of botnets resides in the obscurity of the mechanism they use. Honeypots will try to capture bots and do malware analysis to understand how they work. This tests will try to prevent this analysis to happen. (By deleting themselves or not executing in these environments)
- **Add or delete auto-start applications** To stay resilient after reboot or even fresh installs, bots have mechanisms to stay persistent on the machine after those events.
- **Run or terminate programs** This one is obvious but after exploiting the victim, their goal is to execute actions on the machine.
- **Download and execute files** This allows them to update their software, download exploits and payloads. It can also be used to upload normal programs used for some of the tasks they want to execute.
- **Perform address and port scan** Another important one to pivot inside networks and expand the botnet surface.
- **Communicates with a handler or controller via public servers or other compromised systems** This is the main channel communication with the botmaster.
- **Data storage** One of the tactics used by botmasters to keep their anonymity is to use their botnet as a distributed database and saving the data obtained on the bots, this gives more distance with the stolen data.[**tracking**]

2.1.2 Life cycle

What are the steps that make up a botnet life cycle? Life cycle execution might differ from one bot to another but they have generally a common structure. Here is the common structure [15] [16] [17] presented in these surveys:

1. **Exploitation** The first phase is the infection of the host. The bots gets access to the victim host through different possible vectors (email attachment, vulnerability scanning and exploit, obtained credentials, malicious site, ...). The next step of this phase consists on uploading to the host the binary of the bot. The bot connects to a server of the botmaster and downloads it. This step is very important for this thesis because it is the first DNS lookup the bot will perform and it has been noticed as the most consistent behavior of bots. This is where the bots are going to start to hide their DNS activity [18] [19] [20] [21].

2. **Rallying** This is the phase where the bots will establish the link with the botmaster command and control servers (CnC), join the botnet and wait for instructions. When establishing the connection with the botmaster bots have to use stealth techniques to avoid getting discovered and more importantly revealing the CnC. These techniques will be discussed in the Misuses and abuses of DNS [22] [23].

3. **Attack/execution** From this point on the bot is ready to get the orders from the botmaster and start executing actions. This is where the different actions defined above are executed by the botnet [24].

4. **Update and maintenance** This phase allows the botmaster to update periodically the software of the bots, new exploits, new attacks, the same way administrators patch their software. This final phase is a loop that goes back to the second phase where the bot contacts the CnC server to proceed to this binary upload and commands fetching. This is the second phase where the DNS request will use evasion techniques to stay hidden. [25]

2.1.3 The Channel

What does it mean to join the botnet? The bots will rarely directly connect to the CnC servers, they will go through proxies or peers bots, (structural nodes of the botnet) to obtain commands and updates. The knowledge of these nodes and the techniques used to communicate with them are the channel of a botnet.

What are the channels used by botnets? The channel's resilience of a botnet is critical to ensure good communication with the CnC server. It is also a critical component because its failure is usually the end of the botnet's life. There are multiple ways of securing and hiding their communication channel: tunneling through protocols, encryption, DNS evasion techniques. [10] The typical protocols that are used by bots to reach their CnC are these: IRC, HTTP, HTTPS, DNS, MAIL, SSH, etc. The use of different protocols implies there are a multiple botnet's communication topologies. The different topologies provide trade-offs in terms of bandwidth, rallying, stealth, ...

What are these different protocols ?

What are the goals of these different channels? The main goal of the botnet channel is to provide a vector for bots to reach their CnC servers and maintain a connection with it. This is the only way the botmaster is able to keep control his botnet. This is why the means of communication are built around these main needs. If bots aren't able to reach the botnet or their CnC, they won't be able to update their software and receive commands [8].
Reaching and locating the CnC servers is the first challenge the channel needs to handle. Failing to do so will leave the bot unusable for the botmaster or left in a sleeping mode. In this state, the bot keeps on with the harvesting of the victim host and retry the missed CnC regularly.
The second challenge being its ability to maintain the channel. This is where resilient techniques have evolved to achieve this goal. These technologies will be detailed in the abuses of DNS section. CnC servers and their channels are really what differentiates botnets from other malware [26].

How do botnets achieve such channels? To stay invisible and persistent channels have gone through different methods, here are the main ones researches have found throughout the analysis of a large number of botnets [22] [27]:

- **Hardcoded IP**: The bot software has the IP address of the CnC server hardcoded somewhere in its binary. The server can be found through reverse engineering and the botnet could be stopped or suspended for a certain period.
- **Dynamic DNS**: This is a solution to the hardcoded IPs. In this case the botnet will have multiple CnC servers migrating frequently on its will. In addition to

using a dynamic list of servers, it uses dynamic DNS in order to avoid detection or suspension and keep the botnet portable. This allows the queries to be redirected if they were to fail. This behavior is known as herding, it provides mobility and stealth.
- **Distributed DNS**: To avoid law, botmaster locate their DNS servers outside of the law's jurisdiction. Bots have the addresses of the DNS servers and contact them to resolve the IP address of the CnC servers.

2.1.4 Topology

How are the channels used ? Now that we know the purpose of channels and what they provide to botnets we are going to explore the different topologies used by botmasters using different channels and architectures.

What are the different topologies used by botnets? The differences between topologies are related to protocols of communication and positioning of the CnC. Their structure will result as mentioned above in trade-offs for its different specifications [14]. As explained in this paper[10], there are two main structures which all the other topologies are built on:

- **Centralized**: This is the simplest structure. The CnC is the center of the architecture, responsible directly of the data and command exchanges with the bots. This central unit operates the whole botnet. The main advantages is speed and simplicity, this makes it easier to plan attacks and arrange the botnet. The big problem is that the CnC is the single point of failure of the architecture. If it goes down, the whole botnet is rendered ineffective. The main protocols used are IRC(Internet Relay Chat) and HTTP(Hyper Text Transfer Protocol, the protocol used to communicate between browsers and web servers). IRC is a client-server application for text messaging. The reason it is used as CnC servers is because it can set communications anonymously, between one and multiple users and is very easy to setup. Using the HTTP started because IRC channels were becoming to popular and IRC detection systems were being put in place. But this isn't the only reason: HTTP allows to hide CnC servers behind normal web traffic. This is perfect to be invisible to firewalls and IDS(Intrusion Detection Systems). One of the differences between both lies in how the information is passed: with IRC CnCs bots receive flows of commands from their botmaster, HTTP CnC wait for bots action to send them the commands.[8]
- **Decentralized**: To avoid the single point of failure, botnet designers decided for a peer-to-peer (P2P) communication channel. This structure is much more resilient to detection and avoids the single point of failure. All bots are interconnected with each other and each one acts as client and server. New bots only need the addresses of some bots in the botnet to start communicating with the rest of the botnet. If parts of the botnet are suddenly offline or captured by authorities, the rest can still function normally and adapts rapidly to the situation.[28]

What are the metrics used to assess these architectures? The important metrics for a botnet are a combination of the below sections:

- Resiliency: The ability to resist different events such as the loss of nodes in the botnet, loss of a CnC, blacklisting of domain names, federal investigations, etc.

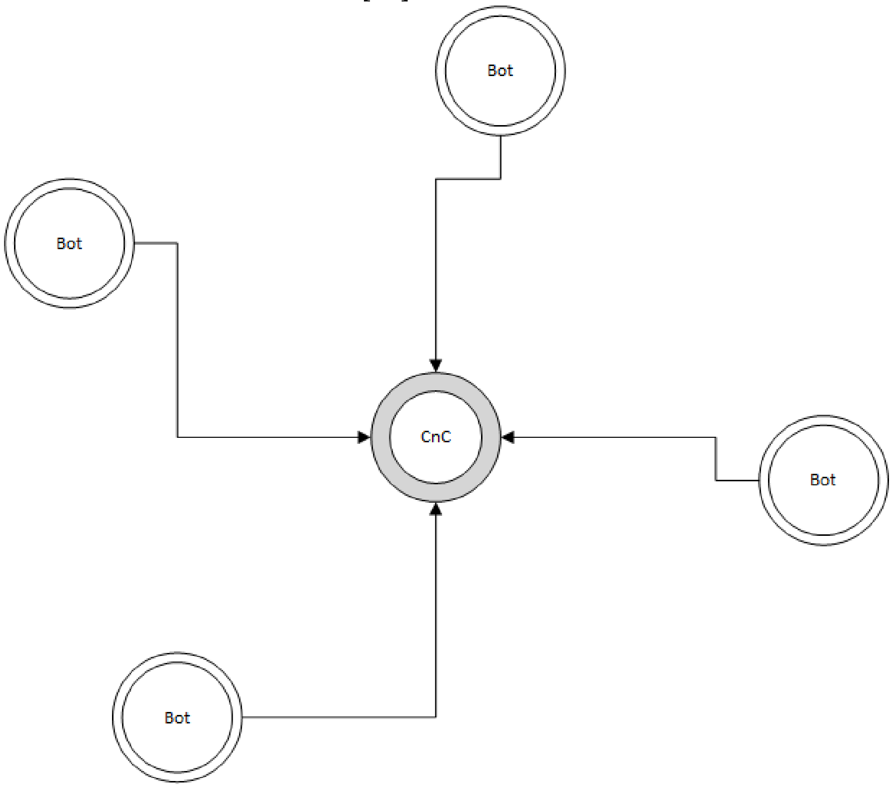
- Latency: Reliability on the transmission of messages. The botnet provides the bots a protocol to ensure the transmission of messages without.
- Enumeration: Accurately predict the botnet's size.
- Defense: protection mechanisms against reverse engineering, static and dynamic analysis, virtual environment execution.
- Financially: Its potential to be partitioned and sold into sub-botnets.

Botnets have followed the evolution of the defenses they were up against, for this is reason botnet operators have now a large choice of architectures when it comes to create one. Botnets topologies have been optimized to sustain most defenses and allow for large remote oversee. The choice of topology will be largely influenced by the business model the botnet operator has in mind.

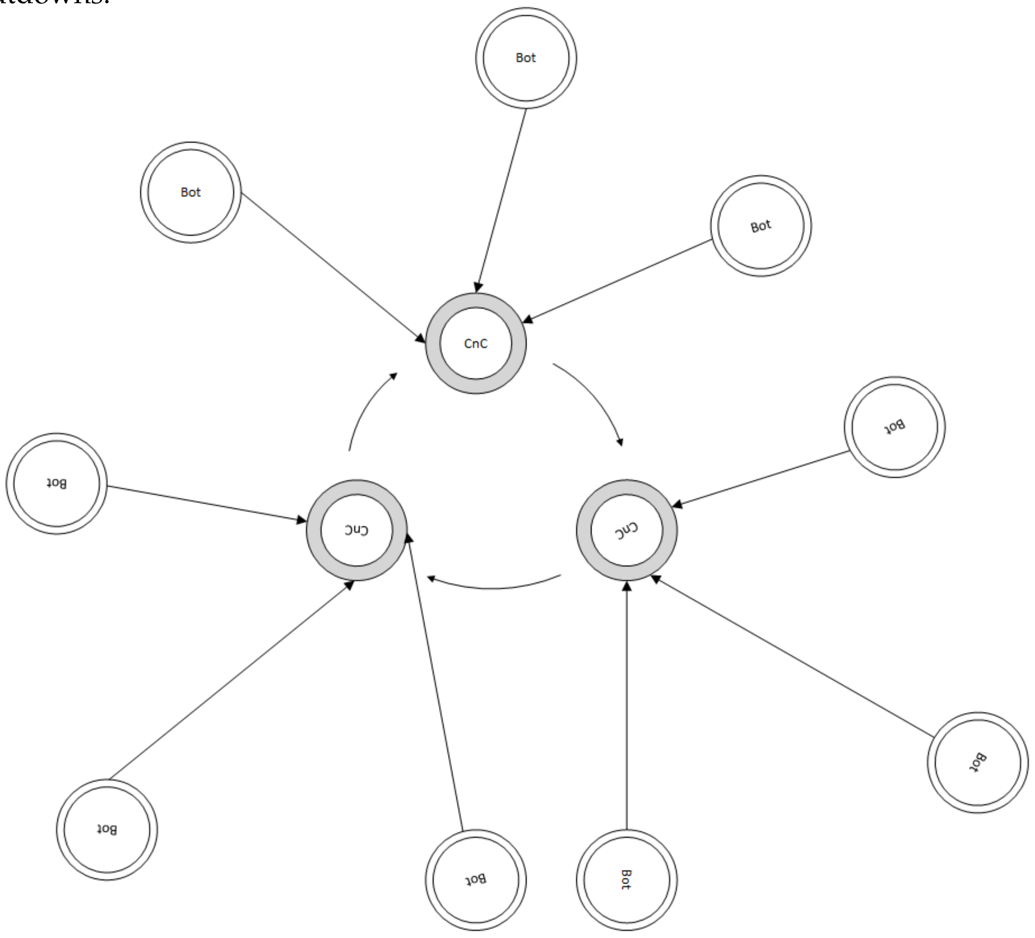
What are the different topologies? CnC topologies encountered in the wild typically match one of the following types:

- Star
- Multi-server
- Hierarchical
- Random

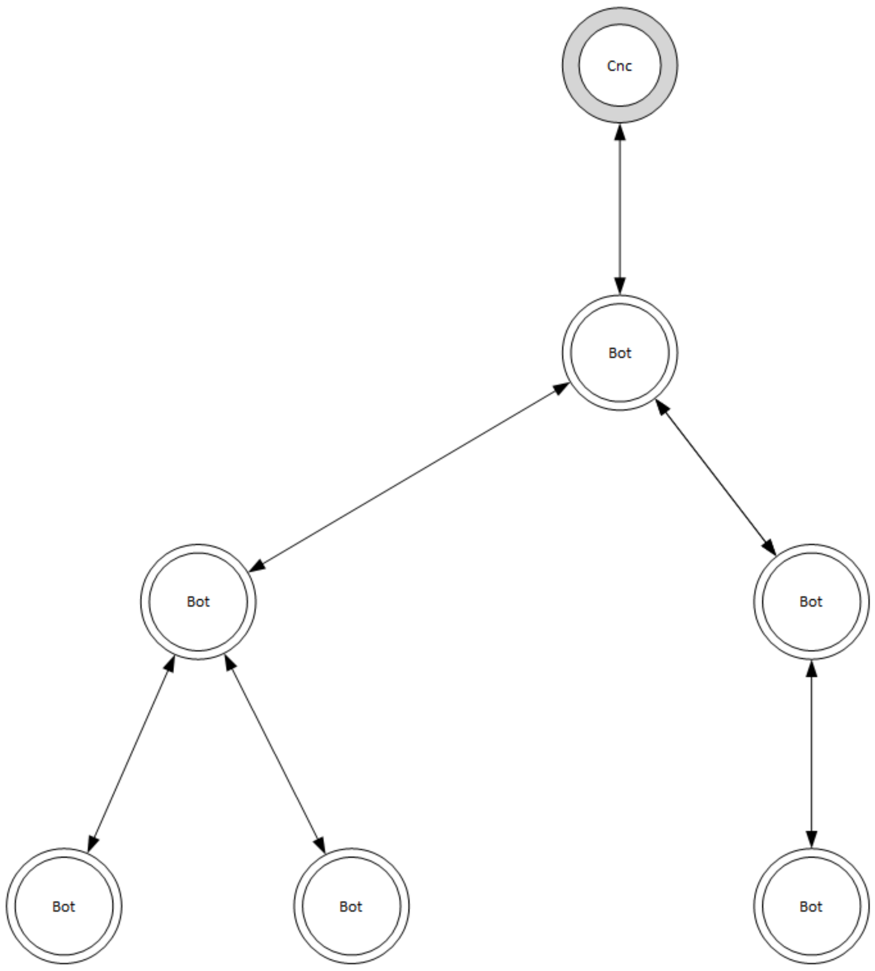
Star The Star topology relies upon a single centralized CnC server to communicate with the rest of the botnet. Each bot agent is issued new instructions directly from the central CnC point[8]. When a bot agent compromises a new victim, it is configured to reach its central CnC, where it will register itself as a botnet member and await for new instructions. The main problem with this topology is the single point of failure that constitutes the CnC server [10].



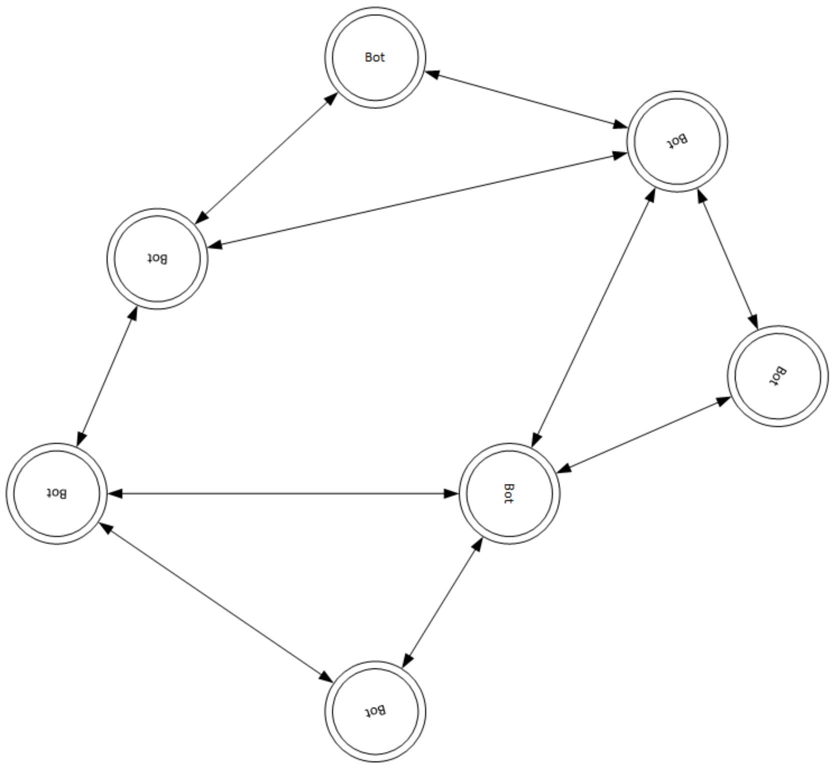
Multi-server Multi-server is the logical follow up of the star topology, it is the combination of star CnC botnets joined together with the CnC servers connected to eachother. This is close to what is done in cluster database management with multiple servers deployed for load balancing and data replication. This ensures that if a CnC server is removed from the botnet, the other CnC servers will take its load and manage the bots that were connected to it. This topology is more complicated to setup, botmasters can even add a geographical component by having these CnC servers in the countries with bots deployed to improve speed and improve resistance to legal shutdowns.



Hierarchical Hierarchical topology is a tree based structure where any part of the tree can be used as a botnet on its own. In this topology, bots can proxy the CnC commands and instructions to the rest of the tree. Another interesting aspect of this architecture is that bots do not know the location of the rest of the botnet. They are aware of parts of it. This allows makes it harder to take down the botnet and allows to segment it for selling or leasing. The downside of it is the latency of the botnet introduced by its branching rendering certain attacks difficult.



Random - P2P This structure is decentralized and is composed of dynamic master-slaves or P2P relationships. Any bot can be used as CnC by the botmaster and relay them to the rest of the bots. To be told apart from the other traffic going through the botnet, traffic with commands will have a specific identification as a signature. This topology is very hard to take down because any node can be used as CnC, it also hard to hijack because there isn't a central structure and communications between nodes don't always use the same paths. The weakness of this topology is that it can reveal a lot of information about the botnet by simply monitoring a infected node and its communications with external hosts.



What design to pick ? Here is a summary [29] of the features taken into account when creating a botnet.

Pros	Cons
Star	
Speed of Control The direct communication between the CnC and the bots allows data to be transferred rapidly	**Single point of failure** CnC blocked or otherwise disabled results in the botnet rendered ineffective.
Multi-server	
No single point of failure Load balancing and replication prevents it from happening and maintains control of the botnet. **Geographical optimization** Geographical location of severs speeds up communications between bots where the CnC servers are situated and help with law take downs.	**Requires advance planning** To achieve an infrastructure that is resilient and balanced such as multi-servers demands further preparation.
Hierarchical	
Re-sale The botnet's owner can segment the sections of their botnet for lease or resale to other criminals. **Hidden topology** Compromised bots don't know the structure of the botnet therefore they are unable to leak much information.	**Command latency** Because commands must traverse multiple communication branches within the botnet, there can be a high degree of latency with updated instructions being received by bot agents. This delay makes some forms of botnet attack and malicious operation difficult.
Random	
Highly resilient The decentralized infrastructure and the many-to-many communication links between bot agents make it very resilient to shutdown.	**Command latency** The random nature of communication links between bots adds unpredictability to the system which can result in high levels of latency for some clusters of bot agents. **Enumeration** The analysis of a bot and its exchanges reveals a lot about the botnet structure and components.

2.2 Uses and abuses of DNS protocol

The latest trend of botnet hide their channel through the DNS protocol. They use it to hinder their identification and rallying process [**survey2**].

2.2.1 The DNS protocol

What is the DNS protocol ? DNS stands for Domain Network System which main purpose is to "resolve" the IP address of a domain name (i.e. google.com).

How does the DNS work ? When an application tries to reach a certain domain, it send a DNS requests for the resolution of the domain name to the DNS server.
The server replies with a DNS response that contains the "answer" requested or additional information on how to obtain it such as another DNS server.

What is the original purpose of DNS? The idea behind the protocol was to provide a human readable domain name to servers. That way humans could identify these domains and associate them with something concrete. The protocol simply looks for the server with the lookup table transforming them into machine readable addresses [30].

Example of the DNS protocol This will later help us understand the abuses of the protocol [31][32].

When a client (user or device) tries to reach mail.google.com it sends a DNS request for that domain name from a **DNS client**.
The **DNS server** defined by the client receives the request and searches for it in its records. If it finds it then it sends the IP address to the DNS client. Otherwise, it will contact DNS name servers that could have the domain in their records following a specific logic in its search.
It will start by querying the **Root DNS servers** that will give it directions through the branches of the DNS tree hierarchy to the **Top Level Domains** DNS servers(TLD). It will first look for the TLDs resolving .com, then the name server that resolves google.com, down to the name server that will resolve mail.google.com to its IP address. The DNS server that received the first request will receive the response and send it to the client finally.
Finally, the client can now connect to the server using the resolved IP address.

Structure of the DNS packets? To understand how the protocol is exploited we need to dive into the specifics of the packet structure.

```
DNS packet
    +---------------------+
    |        Header       |
    +---------------------+
|       Question       | the question for the name server
+----------------------+
|        Answer        | Ressource Records (RRs) answering the question
+----------------------+
|      Authority       | RRs pointing toward an authority
+----------------------+
|      Additional      | RRs holding additional information
+----------------------+
```

The format of a DNS message is the same for a request or a response but parts of the message will be filled differently. In the request, the client will fill the **Question section** with the information that needs to be resolved [33] :

```
                               1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

value	explanation
q_name	Domain name requested (domain to be asked)
q_type	Type of RR record requested
q_class	Class of the request (often IN for internet)

.

The server will respond by filling the **Answer section** [34]:

```
                               1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                                               /
/                      NAME                     /
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
/                     RDATA                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

The important parts are the TTL (Time To Live) and the RDATA. TTL contains the span of time for which the Answer is valid, RDATA contains the answer to the RR requested.
Here is a list of the main RR types[35] and what they query.

- **A or AAAA**: translation of a hostname into an IP address (IPv4 or IPv6).
- **MX**: information regarding the mail servers of the domain queried (ex: DNS request for google.com with RR=MX could return mail.google.com)
- **NS**: information about the DNS server used by that domain.
- **TXT**: Text description of the domain queried.

Are there different methods to obtain the requested RR There are 2 methods of searching through the different DNS server from the **authoritative DNS** (DNS server assigned to or by the client DNS that will be queried first). It can be recursive or iterative. **Iterative mode** is an interaction between the authoritative DNS and all the other DNS servers where all request are initialized by it and all responses come back to it. **Recursive mode** is an interaction between DNS servers relaying the request and then coming back with the final answer.
The Root DNS servers are always iterative, this has been set to avoid a DoS of those servers which could be caused if the were recursive. These servers are the backbone of the internet.

2.2.2 Botnets abusing DNS

In this survey [36], they present in dept the different techniques abused in the DNS protocol, here is a summary.

Why would botnets abuse the DNS protocol? DNS is a very attractive protocol for its versatility. Because DNS is used by all machines to locate other machines, DNS traffic is very normal in any network. Furthermore, the DNS protocol offers a lot of flexibility regarding its uses and this is where attackers have started using DNS for other purposes. Looking up through DNS requests the CnC servers is an essential part in the lifecycle of a botnet. This has also made aware malicious actors that DNS traffic would be inspected to track them or detect them. To make the CnC lookup more resilient, botmasters searched multiple ways to lookup hostnames or play around them. This is where certain features of the DNS protocol became handy to fulfill that goal.

What features did they exploit? Because of the growth of the internet, content providers have built complex infrastructures. Their goal is to sustain the load of the traffic and provide the best services. Some of the things implemented are load balancers, data fail-over, high availability through replication , security with end-to-end encryption, etc. To do so, they can use DNS features that allow to manage this type of architectures effectively. These features have inspired malicious actors with the following misuses: fast-flux, domain flux and DNS tunneling.

Domain flux

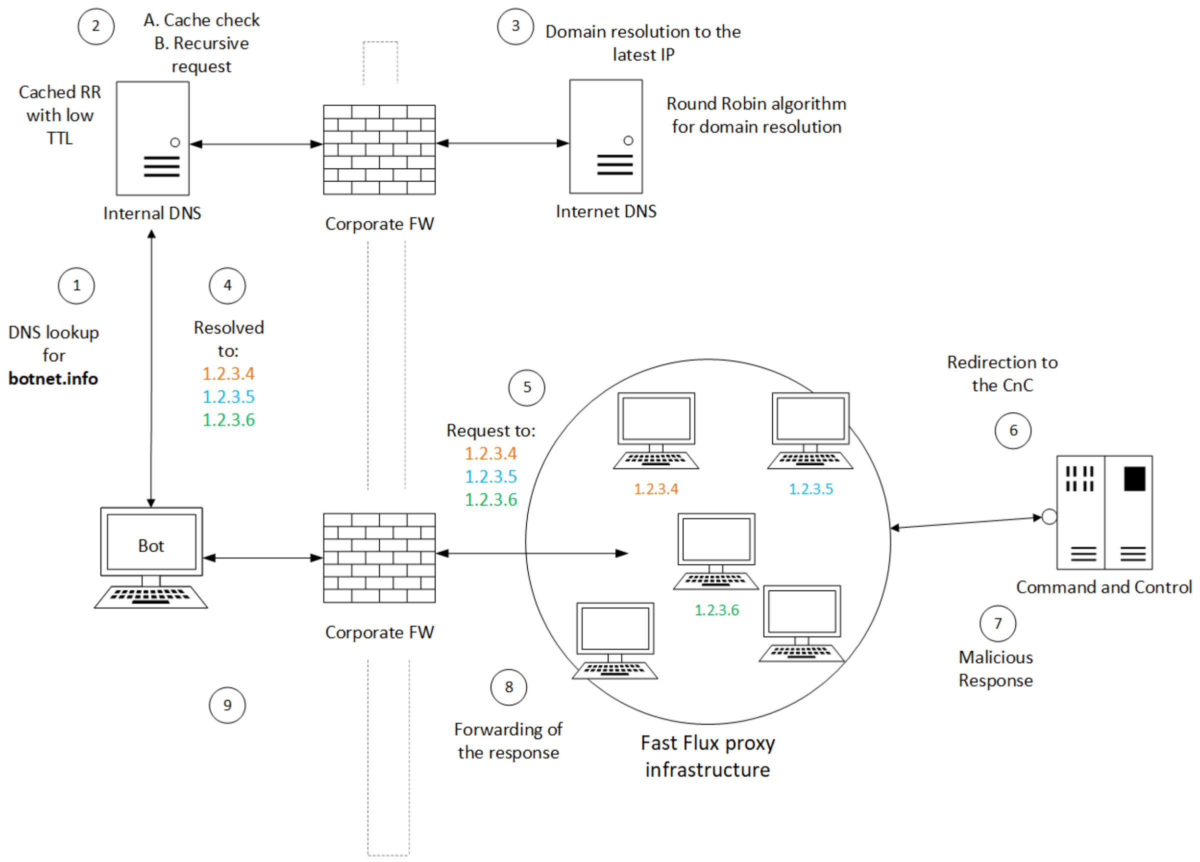
Domain-flux is a type of DNS feature that allows multiple domains to point towards the same IP address, making it hard to blacklist the domains related to the botnet. Bots are equipped with a special Domain Generation Algorithm (DGA). This is used to generate an ensemble of domains from which it will try to contact to received the next update. The idea behind generating such a big amount of domains is to register the domains that will come at a particular time and only register them exactly when they want to update the botnet and do it for a certain amount of time. The botmaster knows which domains are generated at a particular point in time since that is the seed of the algorithm and can register them before they are queried, that way they control when botnets can reach their CnC. This also improves the resilience of botnets infrastructure [8].

There are the 2 techniques used to achieve domain fluxing

- **Domain Wildcarding** abuses the wilcarding capabilities of the DNS protocol. This can create rules that make all Fully Qualified Domain Names (FQDN) point to the same IP address. A rule defined by "*.example.com" would group all FQDN under that scope (mail.example.com, service.example.com, 123.example.com). DNS wildcarding is typical for phishing and spamming botnets. It allows to bypass some of the anti-spam defenses and even to use the wilcard argument as information to identify the different nodes (china01.example.com, china02.example.com) [37].
- **Domain Generation Algorithms(DGA)** is the latest technology used by botnets for domain flux. It consists of algorithms that generate pseudo-random domain names based on a seed (this is the changing factor in the algorithm, ex: current time) chosen by the botmaster. This creates a list of FQDN that change constantly. Bots will try to reach all of the FQDN generated and when the botmaster wants to communicate with them, he will simply compute a couple of future FQDN and register them for his CnC servers to make them reachable by the bots and send the next instructions or updates. Since these domains only last a short amount of time, it becomes very complicated to block all of the possible generated domains through blacklisting or find the C2C servers.
  In this github repository [38], they have compiled some examples of DGA used by famous botnets. As expected, the algorithms have 2 main functions, one that gets the seed (from input to the algorithm or using dynamic values such as date and time), the second is the domain generation, which will usually choose a TLD and then appends the result obtained from the random function with the dynamic seed.

IP flux

What is the concept of IP/Fast-flux networks? IP-flux does the following: associate a certain number of IP addresses to a single FQDN. When sending a request to for the FQDN, one of these addresses is picked using the round-robin algorithm. This technique is normally used for load distribution, load balancing or fault-tolerance. All these addresses potentially host the identical servers, round-robin simply decides on the order it will present them when a request is made for the FQDN. The purpose was to enable IP-fluxing for Content Delivery Networks(CDN) to be able to point customers towards other nodes in the network to obtain the content sought. Round-robin [39] divides time into small periods and presents equal blocks of these addresses in a circular order, it doesn't provide priority for any of the blocks of addresses or specific addresses [40].



How do botnets exploit this feature ? Fast-Flux is mostly used by botmasters to hide a malicious network behind a large amount of dynamic proxies(flux-agents).[41] When a bot tries to connect with the CnC his request to the FQDN goes through a DNS server that returns one of these proxies which is picked from an immense list of rotating addresses. After that, the flux-agent relays the client's request to the mothership.[42] Behind the curtain of redirections created by the network of proxies, botmasters use it to distribute updates or host malicious content. They key elements for FFSN strength are very short TTLs and the round-robin answer from a large list of agents[43][44]. The Fast-flux Service Network (FFSN) motherships are the controlling parts of the networks. They are very similar to the command and a control (CnC) system found in conventional botnets but provide more features. It is observed that these nodes are managed as CDN servers with the same traits (high availability, load-balancing,...). To manage this complex network they collect all the information on the IP addresses assigned to the domain name and how those IP addresses (A and NS records) change over time.[8]

why is IP flux so effective? Unfortunately, botnets use the DNS traffic as any other legitimate host, which makes differentiating the legitimate DNS traffic from the illegitimate, a very challenging problem. Moreover, they use techniques to hide their communication with the bots to evade any deployed botnet detection processes. The botmasters use the DNS services to hide their command and control (CnC) IP address to make the botnet reliable and easy to migrate from server to another without being noticed.[19]

The power of FFSN is allowing one domain name to have an unlimited number of IP addresses. The IP addresses belonging to such a domain act as a proxy for any device attempting a connection with their respective CnC server. This process helps botnet controllers avoid detection and blacklisting. Attackers have developed better techniques utilizing IP-flux over time, here are the different categories:

- **Single-flux**: Multiple IP addresses are assigned to the same domain (either CNAME or A records). The IP addresses of the bots are constantly registered and unregistered to the domain record. They have low TTL and most are proxies for master servers.[42].
- **NS flux**: Multiple NS records assigned to the same domain. This an additional layer of redirection, making the request go through multiples DNS servers before it reaches one that actually resolves the domain.)
- **Double-flux**: Multiple name servers are assigned to the same domain and then use single-flux for the multiple IP addresses of the master. This provides a second layer of redundancy. This also means that the TTLs are short for the A records and the NS records too.

DNS tunneling

DNS tunneling is a technique used to bypass restriction on a protocol or hide certain activity by embedding it in the DNS protocol. For example using the TXT as q_type or the subdomain itself to actually carry encoded data (i.e bWFsd2FyZQ.maliciousdomain.com, where bWFsd2FyZQ is encoded text). Usually, the payload will be fragmented to avoid unusual long domains or packets. This has been done to avoid restrictions but botnets also use it to hide malicious traffic or payloads. They can also use DNS tunneling to remain undetected [45] while ex-filtrating data. The only positive aspect about this abuse is that there are almost no legitimate reasons for this application as we saw with fast-flux, this makes obvious malicious behavior stands behind if this type of traffic is discovered [46].
Paloalto's research unit provided a nice overview of the DNS tunneling practices[47]. They explain how the protocol can be used either to ex-filtrate or infiltrate data with infected machines. They show how different parts of the protocol can be exploited. They divided the botnet traffic using dns tunneling into 3 types: heartbeat, ex-filtration and infiltration. A heartbeat will be a simple query for an A record met with a NXDOMAIN response. An ex-filtration can only use the query to send data therefore, it is usually fragmented through multiple A queries with the data embedded in the query as mentioned above. A big problem remains that because DNS uses UDP it can't rely on the same assurances provided by TCP. And finally infiltration, this use case has a lot limitations then the ex-filtration because it can hide the encoded data in the values of the records such as the TXT record. It has the same problem as the ex-filtration which it can't rely on the protocol to ensure the data is correctly arrived. An interesting question they ask is how does the bot know when to ask for a TXT record instead of a A record. An they use a lot of different options but the most common one is the following: the malicious DNS server will respond with a NOERROR response when it is ready to provide the next payload and that is how the bot will change its next query for a TXT record.

Domain shadowing

The last abuse of DNS which has been recently detected, used by botnets as a communication channel is DNS shadowing. DNS shadowing is an abuse obtain by stealing account credentials from legitimate domains for the purpose of creating subdomains aimed at malicious servers. This provides criminals with a great amount of subdomains that inherit their parent's domain reputation. This allows to bypass a lot of features based on reputation and 2LD.
Malicious actors cycle through the creation of subdomains which they delete shortly after, similar to the fast-flux rotation behavior.
The reason this type of activity spawned is due to IDS using detection based on reputation scores for the domain names flagging domain fluxing and fast-fluxing making it complicated for botmasters to use those evasion systems as much. Malicious actors realized they could bypassed most detection systems by using the reputation of legitimate domains and that is when they started campaigns to harvest this specific type of credentials[48][49].

In particular, the bad actors harvested a large amount of credentials of domain owners[50][51] (e.g., through phishing emails or brute-force guessing) and logged into their accounts to create subdomains. This technique is called domain shadowing, and such subdomains are called shadowed domains. The malicious subdomains inherit the reputation of legitimate apex domain[52][53].

2.3 Machine learning approach

We use machine learning in our thesis to create models around certain behaviors botnets show to help us distinguish them from the legitimate traffic. The next sections will detail what machine learning is, how it works and its implementation in the thesis.

2.3.1 Machine learning

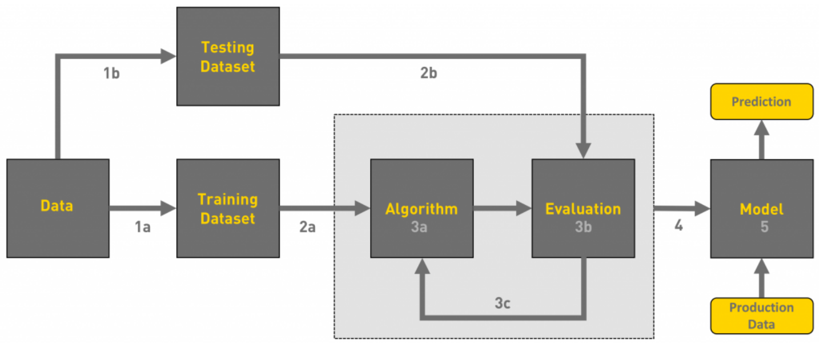
Machine learning (ML) is a mathematical study of algorithms and statistics that allow to learn and improve a certain task without being directly programmed. The goal is to improve the performance of specific tasks by building models of the problems. [54]

2.3.2 Machine learning for botnet detection

The goal of botnet detection is to distinguish between 2 classes of traffic: normal and malicious. We are interested in one of the tasks that machine learning algorithms excel at which is classification. After training the models, called classifiers, they are able to assign classes to any new data.

2.3.3 Machine Learning workflow

Machine learning has a specific pipeline that needs to be followed to obtain the best results [55]: Gathering the data, selecting the features, extracting the features, choosing the right classifier, training and testing the model, and finally, assessing the prediction capabilities of the model. Each step is explained below and its implementation will be detailed in the experiment chapter.



Data gathering

The machine learning algorithms need to be fed data that has information about the problem. We need data that will be able to shape and create the model. In our case, we searched for captures of DNS traffic available publicly to study botnets, preferably labeled data as malicious or botnet. This labeling is essential for the learning stage. Another step which is important when creating the dataset is to balance it out correctly between the 2 labels. If there is an imbalance, the label with the greater presence in the dataset will influence the model excessively and poorly train it. The dataset research will be detailed in .

Data pre-processing

Data pre-processing involves a number of processes, it aims at improving the quality of the data and depends on the type of data in the dataset. Its 2 main processes are feature selection and feature extraction. The feature selection consists on deciding what elements of the data are relevant to create the model and feature extraction consists in cleaning the data and formatting it correctly to be ingested by the machine learning algorithms optimally.

Feature extraction It is important to realize that algorithms are programs that only understand numerical data. The raw traffic captures can't directly be ingested by the algorithms, these can only digest 3 types of formats: numerical (TTL value for a RR), categorical(DNS record types such as A, AAAA, TXT, CNAME, ...) and ordinal (short, medium, long). These formats can always be converted into numerical values which is the only thing algorithms can use. It is important to realize that only features from the data convertible into one of these formats will be able to be used. Format isn't the only thing that is important, with large datasets, we need to get rid of all the noise that they might contain as well as find missing data or inconsistency in parts of the traffic. This section will be detailed in

Feature extraction techniques To clean and extract the features from the raw data, there are common techniques used.
The first is **conversion of data**, this consists in transforming the categorical and ordinal features into numerical ones (i.e [high, medium,low] would become [2,1,0]). This can even be improved by using hot encoding which creates a feature for each value of the category. Hot encoding is nevertheless mostly used for large categorical features.

The second consists in dealing with **missing data** by either removing it or using an average for that feature not to have it influence the other data for that feature.
Thirdly, **anomalies** due to human errors for example need to be discovered and corrected manually.
Another aspect to consider are the scales of the features. Certain algorithms will give more weight to features that have a higher scale which would create a bias in the data. To deal with scale issues between the features there are two used techniques: **Normalization** and **Standardization**.
We normalize the features[56] by dividing them by the maximum for that feature. The goal of the normalization is to avoid features unbalanced analysis by the algorithms. Features such as TTL with a range of values in [300-86400] compared to features such as the number of resolved IPs with values in this range [0-15], the difference in scale could have more weight in the algorithm due to the scale difference of the features. The normalization will project the features into ranges between 0 and 1. This makes the models less affected by scales and improve their learning.
Standardization aims to achieve similar results as normalization but through another method. It works by re-scaling the data so it has a mean of 0 and a standard deviation of 1. Standardization is shown to improve comparison between features with different units and scales as well as make the training process better for the classifiers.
As explained in this article[57] normalization and standardization seem to always improve quality of the results but the choice of scaling function can have important different results. Therefore, this is why will will test our results using the different scalers available for each classifier.

Feature selection is an important step of the workflow. The choice of the data that will be ingested is crucial and directly connected to the results we will obtain. We can select features based on pattern or data type.
Pattern based selection means through the statistical analysis of the data and using unsupervised learning algorithms to extract tendencies from the results. This aims at finding patterns for each label and understand the underlying reason.
Data type based means that we focus on the specific features related to the traffic from the labeled data. Most of these features have already been analyzed by other researchers and will be based on the nature of the DNS traffic coming from malicious or normal labels.

After you have chosen the features from your dataset, there are a couple of things to look for: The number of features used can be a major impact of your research in regards to redundancy of the features, complexity related to number of dimensions, overfitting the model. Our next step is to decide which features we want to keep for our experiment.
Here are some questions to help us in that selection.

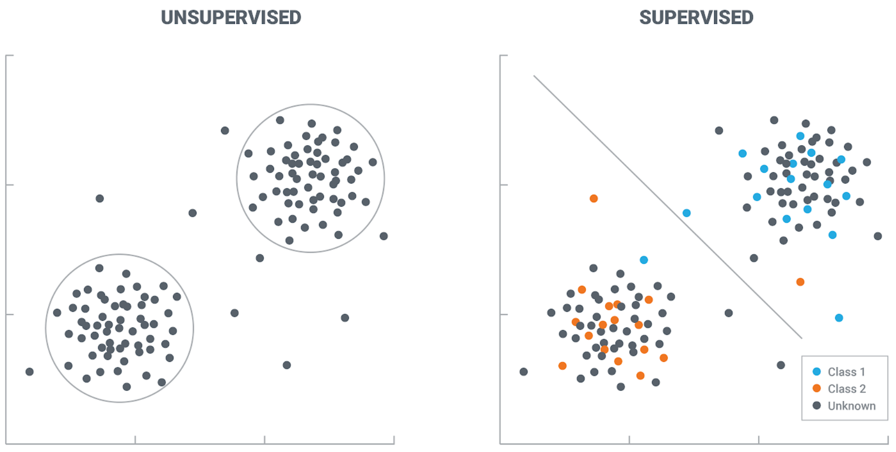
Are the features independent enough from each other to have a balanced weight between general features? The problem that often arises in ML is that different features are providing the same type of information and not adding value. This can push the balance of the models in their direction and result in a biased model. There are multiple solutions used in ML for this issue: analyzing feature correlation and removing strong ones, scatter mix plots to understand links between features and finally importance analysis through the use of the Random forests algorithm.

Are there issues with large sets of features? This is where we can see the curse of the dimensionality manifest itself. Where problems such as exponential computing and overfitting come in to play. Some machine learning algorithms handle large sets of features very well but others have an exponential complexity when it comes to the number of features and it can end up slowing your research process. The problem of overfitting is important to take into account when doing ML and keep in mind that our dataset might not represent the global view of the problem we are solving. This means that if we create a model that doesn't generalize enough it won't work on other data as well even if you obtain really good results.
To solve dimensionality there are a couple of solutions, some are part of reducing the amount of features by removing the less independent ones or important ones. There are other solutions such as Principal Component Analysis (PCA) and Linear Discriminant Analysis(LDA)[58].
PCA is a procedure that transforms a group of features that could be correlated into linearly uncorrelated features. The features provided are named principal components and provide valuable features for model creation but they are also hard to interpret. Its use introduces a part of shadow in the results which is a trade-off to take into account when using it.
LDA models distributions of the features given into classes then uses Bayes to estimate their probability, it is mostly used in ML for dimensionality reduction.
Both are similar in the way that they look for the best way to explain the data they are given.

How to chose the right algorithms?

To answer this question we first need to learn which are the differences between the available algorithms and what they look to achieve:

What are the different machine learning categories of algorithms? There are three mains categories of algorithms in machine learning based on their learning method: supervised learning, unsupervised learning and semi-supervised learning. These categories are based on the training data given to the algorithms. **Supervised algorithms** are provided labeled data, they analyze the features and learn what characteristics are proper to each label. This is how the classifiers build models to predict the labels for new data. **Unsupervised algorithms** use unlabeled data as input. This type of algorithm will group data based on similar values for different features. Semi-supervised algorithms is the combination of both, where the training data is partially labeled. This is very useful for datasets where there is only partial labelisation available.



[59]

Now we need to decide what algorithm makes more sense to model the data in accordance to the problem that is being solved. We are trying to solve a detection problem. This falls into a classification problem where supervised algorithms prevail[60]. To find the best algorithm to model the problem we will test out different algorithms and compare them under the same conditions.

Training the models

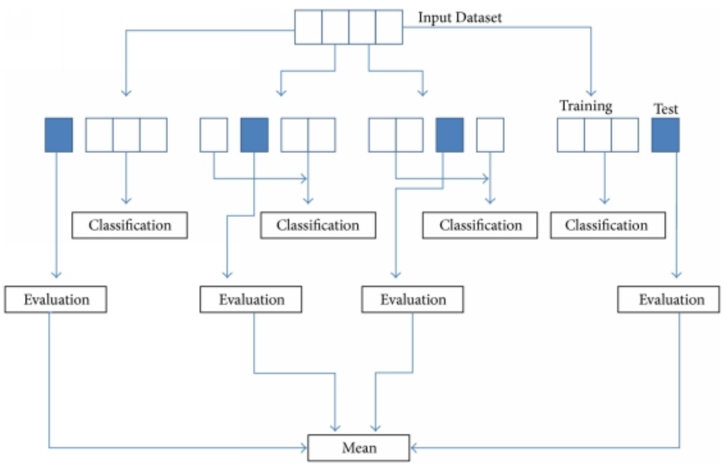
After the steps above, we have a balanced and clean dataset, we have the features that will be used to model the classes and we have a list of algorithms.

1. Now comes the part were we are going to feed this data into our algorithms and start creating a model. The way models are verified is by dividing the dataset into 2 sets, one for training and one for testing. The way datasets are divided into the 2 sets are up to us, the standard is 80% for the training set and 20% for the testing set. The choice is ours, as long as the testing set is large enough to provide accurate statistical results and is diverse enough to be a good representation of the rest of the dataset.

2. Each algorithm possesses parameters which are called hyper-parameters, that allow us to fine-tune the algorithms to the problem and the data. However, it would take a very experienced data scientist to be able to manually set those parameters correctly. For problems with small dimensions, the model can be visualized and it becomes easier. In our case, we have too many features to visualize the resulting model in a way that will help us understand what needs to change. For this reason, there are 2 main techniques that are used to help with this issue which are the **Grid Search** and **Random search**[61].
   The grid search selects a set of values for each parameter and then runs the training of the model each time with a different combination until they are all covered. The output is the combination of parameters with the best accuracy score. This disadvantage of the technique it is affected by the curse of dimensionality.
   The random search works similarly but instead of trying all the value in the

range of the parameter, it randomly picks a subset of combinations. For lower dimensional problems, random search has been proven to perform better.

3. To validate the performance of the trained models in the least biased manner the machine learning community uses a technique called the k-fold cross validation[62]. For each one of the algorithms, the training set is divided into k random folds(equal subsets). One of the folds is then chosen as the testing fold and the rest as training for the algorithms. The algorithm is trained and tested. We repeat this process k times until all the folds have been used as the testing fold. This provides the best classifier able to generalize the data in the training dataset. It is important to point that for classification problems, we use a variation which is the stratified k-fold cross validation, this technique ensures that through the folds the balance of classes keeps the same proportions. The common value for k is 5 or 10. The results obtained are a realistic and unbiased representation of the model's performance.



Testing the model

We now have obtained a classifier, we need to see how to performs with our testing set. The process is pretty simple, we take the classifier and give it our testing set as input, it is important to know that the set provided to the classifier are the features only, the testing set given to the classifier isn't labeled. The result is the labeling of the testing set. From this point, we enter the analysis of the results where we compare the resulting labels from the real class they belong to.

Analysis of the model's results

In this section, we will use the metrics obtained from the results of the model's predictions to provide us information on the experiment and to enable the feedback loop. For classification problems, the most common technique is the confusion matrix which can derive a large number of useful metrics including accuracy, precision, recall, F1-Score. There are also other advanced metrics [63] that can be introduced such as: Logarithmic Loss, Area Under ROC Curve and more. All will be described in the result chapter. These metrics will help us understand what can be improved and if the direction we are taking is improving or not the model. From there we will adapt the features and the parameters of experiment to improve the results. The goal is to reach the best results possible.

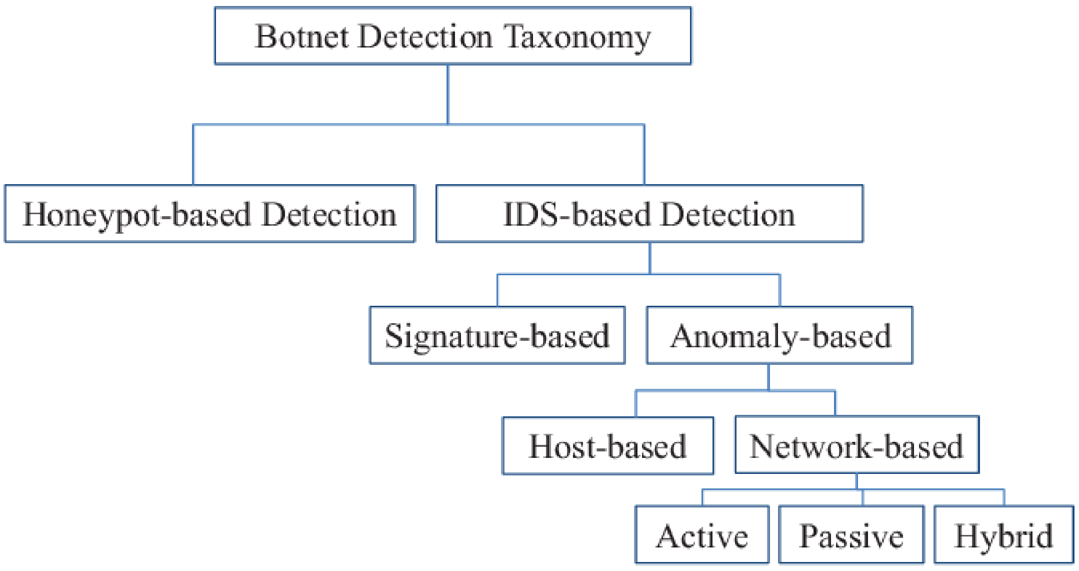
Result interpretation

The last part of the experiment is the interpretation of the results. We want to understand how our classifier works. To do so, we have at our disposal tools such LIME, SHAP, RFs and more that help us understand which features are important and what are the model's behaviors[64]. Those tools will be defined more in depth in the result's chapter, under the interpretation section.

2.4 Classification of botnet research and detection

In this section, we are going to present the current state of the botnet detection taxonomy and show the weaknesses and strengths of each method. This will be followed by a discussion in the next chapter of the current state of the art when it comes to passive traffic analysis for botnet detection and what we are going to bring to it.

2.4.1 Taxonomy of Botnets



Detection systems

In a recent survey, Alieyan et al. [4] presented the taxonomy of botnet detection techniques. We will follow down the path of the detection taxonomy until reaching our center of interest. As we can see on the figure above, there are 2 type of systems used for detection: **Honeypots** and **Intrusion Detection System** (IDS). Honeypots aim at creating an environment specially forged to attract malicious traffic and extract information from the behavior of the malicious actor on the host. IDS aim at analyzing the network and security logs, and alert the security analysts of any malicious or suspicious traffic. The IDS is divided into 2 techniques. An older one **based on signatures** and newer one **based on abnormal behavior**.

A legacy problem

A big trend among companies for a long time and even today was solely use of a signature-based detection in their IDS [51]. However, these signatures have shown to be ineffective against bots that are constantly getting updated with new code and new evasion techniques. Furthermore, they are ineffective against any new type of emerging botnet. Signature-based detection is great for known botnets[65] and should always be implemented in IDS solutions but is insufficient and should always be coupled with other techniques. The recent introduction of other techniques such as abnormal behavior has been mostly motivated by the goal to solve this issue.

Where are botnets detected?

In the anomaly detection technique, researchers have found relevant data in different locations: directly on the host affected and throughout the network the host belongs to. Location wise, here are 3 types of behaviors observed from bots[66] :

- **Network based behavior**: this is the observable network traffic between botmaster and bots. The goal is to uncover the communication channels and any traffic related to attacks.
- **Host based behavior**: observable activity on the host infected by the botnet. This activity is mostly what Antivirus (AV) software cover.
- **Global correlated behavior**: This focuses on the global characteristics of botnet behavior, they focus on fundamentals structures of the botnets that could emerge and be used for detection.

Since AVs already provide coverage of the host-based behavior, we have decided to focus on the locations where IDS still have a lot of room for improvement, we focused mainly on the network aspect because obtaining data to analyze global network correlation isn't easily obtained.

What are the types of approaches for network behavior detection?

They are classified into 2 categories: **passive detection** and **active detection**[67].
Passive detection consists in gathering data through monitoring logs. Activity on the network is tracked without interfering with it, also making it harder for botmasters to notice it. This method is limited in the amount of data it can gather. Some examples of this approach: deep packet inspection through IDS, flow records analysis for traffic flow pattern identification, DNS monitoring, spam records analysis for botnet correlation, application log files analysis.
Active detection differs from the passive approach by interacting directly with the information it observes. Because of the changes it may introduce, this approaches can be detected by the botmaster that might change the behavior of the botnet or add elements of evasion. There are 2 main examples: Sinkholing, this consists in redirecting the traffic of the botnet to a controlled machine to cut off the CnC, and Infiltration, which tries to wiretap or take control over the botnet from the inside by reverse engineering the malicious code and traffic. Other examples of active detection: FFSN tracking, IRC traffic analysis, peer-to-peer networks enumeration.

Everything presented above applies to all botnet using different protocols. The interest to study this part of the taxonomy, particularly the DNS passive traffic approach comes from the progress it has shown and the large amount of possibilities that still can be studied.
Let us now analyze the particularities of the DNS passive traffic analysis.

2.4.2 Passive DNS detection Techniques

What are the different passive detection methods? With the DNS logs captured, researchers have proposed different approaches to utilize them, most of the methods involve a statistical analysis or the use of machine learning algorithms to create classifiers. Others have opted for detection through visual representations to detect anomalies.
The reason we have decided to focus on the machine learning approach as mentioned in the introduction is two-fold: a will to develop our machine learning skills and methodology but also because botnet detection state of the art has shown it to be the most effective method, achieving better scores then the other techniques in the taxonomy.
As explained in the "abuse" section of the DNS chapter, a new type of detection techniques have been studied that aim at finding bots using the DNS protocol to evade the more usual detection techniques which is what we aim to further study and hopefully improve the current solutions.

2.5 Related work

Our end goal is to find the best features and techniques possible to improve the current solutions that aim at detecting all the evasion techniques, that we have named **Combined Solution**. The reason we focused in combined approaches is because we believe that most of these features alone can be part of legitimate use-cases and combining multiple evasion techniques allow to improve the detection on malicious activity. To do so, we have structured our research as follows:
First, we will present the combined experiments. These experiments do not focus on a single type of detection mechanism but aggregate the different detection techniques used by botnets. We will discuss the features they have extracted, what models they have created and how they have implemented their approach.
Secondly, to achieve our goal towards improving these solutions, we will discuss the weaknesses of their approach and how we plan to improve it.
Thirdly, we will present the current state of research, the studies that we have used to improve the solutions. To structure this part we have divided the related papers by evasion technique.
The objective is to learn the different approaches from different papers, find better features and better models, see which we think could improve the current solutions and then test them out in our experiment.

The first step in the research was to find the models that would be our baseline to compare our experiment and improve upon. We found a couple of solutions which are referenced in most botnet detection papers as the most extensive . We have Bilge et al.[68] with Exposure and Heuer et al.[69] with "Recognizing Time-Efficiently Local Botnet Infections".

2.5.1 Exposure Experiment

Exposure is a botnet detection solution that analyzes passive DNS traffic and uses a J48 DT algorithm to predict new malicious domains. The particularity of this study was its continuous improving architecture as well as the ability of Exposure to detect new malicious domains before they are added to public blacklists.

Dataset

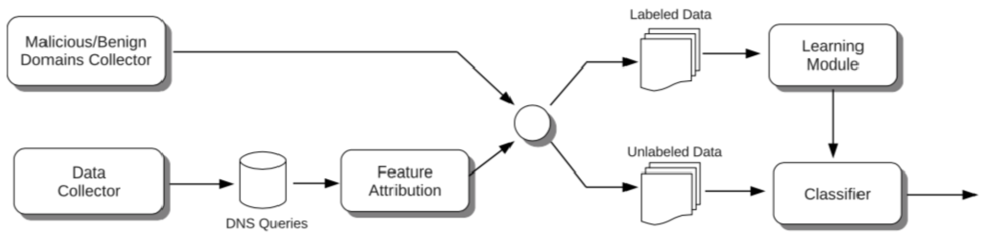
The training set is a capture of passive DNS traffic from the Security Information Exchange(SIE) for 2.5 months and then almost 1.5 years for the complete assessment of the tool. Since the original dataset was around 100 billion DNS queries, to reduce the amount of volume they used 2 assumptions: whitelisting of Alexa top 1000 removed 20 billion queries and whitelisting domains older than 1 year (from which they could only find 0.09% suspicious confirming their assumptions) removed 40 billion queries.

Architecture

The architecture is based on 5 Components.

1. Dataset through passive DNS traffic collection.
2. Collection of legit and malicious domains.
3. Domains used to label the output of the Feature Attribution component.
4. The learning module uses the labeled data to train the classifier.
5. The classifier trained with the learning module predicts the unlabeled traffic.

It is important to point that everyday during the 2.5 months training period the collection of legit and malicious domains was improved with the predictions made by Exposure and reviewed by the writers and by updating the list with any new additions to blacklists or whitelists available.



Features

The set of 26 atomic features proposed for the training of their classifier is divided into four categories: time-based, answer-based, TTL-based and domain name-based.

Time-based They proposed time-based features because they realized that analyzing many requests to a particular domain over time, patterns indicative of malicious behaviour would emerge. They found that there was a increase in activity followed by a decrease. Also that there was a burst of requests in a short time.

1. **Short Life**
2. **Daily similarities**
3. **Repeating patterns**
4. **Access ratio**

DNS answer based Here is where they focused on the domain-fluxing technique. In such cases, the DNS server cycles through the different IP addresses linked to the domain in a round robin fashion and returns a different IP mapping each time. In addition, malicious domains typically resolve to compromised computers that reside in different locations. The attackers typically use domains that map to multiple IP addresses, and IPs might be shared across different domains. This information resulted in this list of features:

5. **Number of distinct IP addresses** for a given domain during the whole capture.

6. **Number of distinct countries** of the IP addresses for given a domain

7. **Number of domains share the IP with**

8. **Reverse DNS query results**. The outputted list of IPs from the reverse DNS query is used to extract 5 atomic features
   - Ratio of IP addresses that cannot be matched with a domain name (NX domains)
   - Used for DSL lines
   - Belong to hosting services
   - Belong to known ISPs
   - Can be matched with a valid domain name

TTL value based Malicious traffic usually has low TTL when doing fluxing but they are not the only ones, CDNs do too. So these feature are not to be considered alone but in complement of other features. FFSN are usually detectable because of low TTL and growing list of distinct IP addresses for a domain.

9. **Average TTL**

10. **Standard Deviation of TTL**

11. **Number of distinct TTL values**is the simple TTL count for the entry.

12. **Number of TTL change** is the different TTLs values for the same IP/domain.

13. **Percentage usage of specific TTL ranges** This feature is splitted into 5 ranges: 0-1,1-100,100-300, 300-900 and 990-inf. The value computed for each of the ranges is the percentage of TTLs in each range compared to the total TTLs.

Domain name based Finally, a couple of simple features to expect detection of DGA: there is a big difference between legit domain names and domains generated by DGAs(Domain Generation Algorithms(DGAs).

14. **Ratio of numerical characters** compared to length of domain name.

15. **Ratio of the length of the LMS**, where LMS stands for longest meaningful string. This consists in finding in the domain name the longest substring which is a meaningful word.

Feature Set	#	Feature Name	# of Atomic Features
Time-Based Features	1	Short life	2
	2	Daily similarity	1
	3	Repeating patterns	2
	4	Access ratio	2
DNS Answer-Based Features	5	Number of distinct IP addresses	1
	6	Number of distinct countries	1
	7	Reverse DNS query results	5
	8	Number of domains share the IP with	1
TTL Value-Based Features	9	Average TTL	1
	10	Standard Deviation of TTL	1
	11	Number of distinct TTL values	1
	12	Number of TTL change	1
	13	Percentage usage of specific TTL ranges	5
Domain Name-Based Features	14	% of numerical characters	1
	15	% of the length of the LMS	1

Feature selection

The feature selection phase highlighted the following: the combination of all the features provided the least error rate, the last 2 categories (TTL and Domain-name based) had high error rates when tested by themselves and the timing-based category had the most weight but also had the most complications in regards to extracting the features because of the non sufficient time points available.
Finally they used a genetic algorithm ECJ20 evaluated with a j48 DT to reduce the number of features into a 14 feature subset selected from the original 26.

Type	Feature
Time-based features	Feature 1a, Feature 2, Feature 3a,b
DNS answer-based features	Feature 5, Feature 6, Feature 7b,c,e
TTL value-based features	Feature 11, Feature 12, Feature 13a
Domain name-based features	Feature 14, Feature 15

Results

With the selected features and using a DT classifier, they obtained a detection rate around 98% and a false positive of around 1%.
The detection rate wasn't the main purpose of the paper but to detect new malicious domains, detect them before their appearance into mainstream blacklists. To assess Exposure's capabilities, they checked the 569 additions during the capture period in the main blacklists, with 216 seen in capture for the training period and 211 fitting the requirements for the time-based features, all were detected as malicious previous to their apparition. This proved Exposure was a successful solution for detecting new botnets.

2.5.2 Recognizing Time-Efficiently Local Botnet Infections

Heuer et al.[69] did a case study on local infections related to botnets. They propose a very large set of 23 features meant to detect botnets in networks. These features aim at the different evasion techniques that are used by botnets and the purpose is to be able to detect any form of Botnet in the network analysed. Unfortunately, the full paper wasn't available to us so we had to satisfy ourselves with the abstract and figures available publicly.

Architecture

The architecture of their study was very typical for a machine learning approach. The interesting part of the experiment was related to the labeling. The used an extensive list of 22 blacklist and the Alexa 500k for the whitelisting. Another interesting part was the semi-manual labeling of the DGA domains which allowed to improve the training process by balancing the dataset and providing more data for the malicious traffic.

Features

We can see the list of features initially proposed but we don't know the feature selection process they did during the ML workflow.

Domain-name based	LMS numerical / character Ngram
DNS-based	DBForwardBackwardSimilarity DBDialup Nb distinct IP addresses Nb distinct Countries Nb distinct Domain Share IP NXDomain MXRecordPresent SOA (min, exp, refresh, retry) TXTRecordLength
AS-based	Nb distinct AS Reputation AS Size of AS
Time-based	Daily similarity Repeating patterns ShortLife
TTL-based	Average TTL Nb distinct TTL values Nb TTL changes % usage of TTL ranges Std deviation TTL

Results

They decided to test 6 classifiers to find the most optimal for the problem. The algorithms tested were ANN, Bayes, DT(J48), RF, KNN and SVM. The classifiers were trained on 2 different datasets.

Automatic labeling This was the raw dataset simply using the blacklists and whitelists for labeling. The results showed that the accuracy(95.2%) and the false positive rate(around 1%)were the best for the ANN and the DT.

Semi-manual labeling For the dataset, where the DGA domains underwent an additional labeling process resulting in a batter dataset the results were the following: 95.7% - 95.9% accuracy for ANN, RF and DT (J48), and a false positive rate of 0.4% for the DT, followed closely by 0.6% for the ANN.

2.5.3 Weaknesses and possible improvements

The detection rates obtained by Exposure are really good but they are hard to implement in a real environment because the time-based features are hard to obtain. There is a need of the same domains/IPs appearance a certain number of times in a specific time range to extract the time-based features. Furthermore, in their study, they showed that these features were the strongest features out of the four sets, we conclude that without the access to these features the experiment would probably not show the same level of success.
[69] had great results and very promising features. On the other side, the training dataset used seemed a bit small compared to the amount of complexity around botnets. This looks like it could have introduced a bias in the training to the dataset.

What we aim to do in our experiment is to provide smaller environments with similar detection capabilities as Exposure. This requires features available at a lower scale. We want to explore other features proposed by the scientific community that could fill that gap and provide with similar results. We also notice no features targeting DNS tunneling or DNS shadowing which are 2 techniques that most recent botnets have picked up even more to do channel communication, ex-filtration and improving their evasion.
We think that by combining the best features from both papers with the addition of features aiming at other techniques, will improve the general detection capabilities of newer families of botnets but also provide smaller environments with such detection capabilities.

2.5.4 Focused experiments

Here are research papers that have proposed different techniques to detect botnets using these different evasion techniques presented in the abuse section of the DNS chapter. These are the papers where we found features and ideas to improve the above experiments. For clarity, we have divided them by evasion mechanisms.

Domain-flux

Domain-fluxing detection is mostly about analyzing domain names, here are the papers that attempt to do that with different metrics and features.

Truong et al. [70] came up with an experiment allowing them to detect domains generated by humans or algorithms. This is the base-ground study of DGA detection.

Yadav et al. [71] propose an unsupervised approach based on anomaly detection with a set of metrics analysing ngrams of the SLD. They use the Kullback-Liebler divergence measure with uni-grams and bi-grams, the Jaccard index between bi-grams and the last feature and the Edit distance. These 3 features are used widely in the DGA detection research because of their efficiency.

Schiavoni et al. [29] go a step further then Truong et al. with their Phoenix project by improving the initial experiment with additional features to cluster groups of DGAs under botnet families. The project works in 2 phases: DGA discovery and DGA detection. In the discovery phase, they apply the following filters that focus on linguistics: percentage of meaningful words in the domain name and the popularity of the n-grams of the domain. They construct a base generated with the top 100.000 domains from *alexa.com*. Then define the Mahalanobis distance and the thresholds, using known malicious domains, to determine when domains can be considered DGAs. Their approach is able to associate new DGAs to botnet families and follow their evolution.

Ahluwalia et al. [72] analyse the basic features that are common to most domain generated by DGA and provide advanced linguistic features to improve the results. The motivation for their work was due to recent botnets using shorter DGA lengths to blend with the other domains. They then propose 3 primitive features that capture linguistic and structural characteristics and 2 more advanced features that cover the shortcomings of the primitive ones. These features are simple but obtain really good results.

Antonakakis et al. [73] proposed a different system called 'Notos' based on dynamic reputation of odd domain names. They studied the different aspects around the historical DNS data of domains that could be relevant to group domains based on their legitimacy. They use 3 categories of features: network, zone and evidence. They opted for a unsupervised approach using the clustering algorithm X-means. The purpose of this paper is to utilize some of the historical DNS features to be used in our all-in solution. Specifically, using some of the classes of domains defined by the authors as categorical features (popular domains, common domains, Akamai domains, CDN domains and dynamic DNS domains).

Thomas et al. [74] realized that during the domain generation process, most of the domains will not be up. This should result in a lot of NXDomain responses. Furthermore, the caching of NXDomains is limited which means that they cannot hide this traffic. Their contribution consists of a clustering technique based on domain names and request patterns; and similarity metrics for malicious domains detection.

In another paper from Yadav et al. [75], they explore the detection possibilities in the DNS failed queries due to the fast fluxing behavior. Botnets are querying a large number of domains which are only up for specific amount of times, this results in NXDOMAIN responses from the DNS queries emitted, their research has tried to use this information to improve current detection systems.

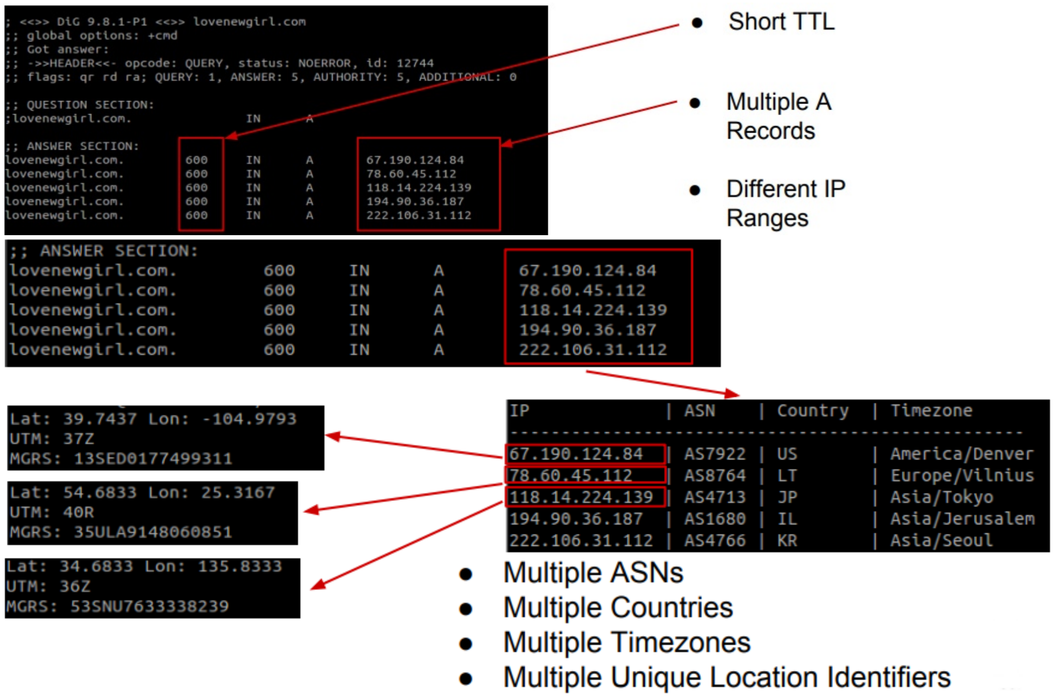
Antonakakis et al. [76] proposed Pleiades, their second big botnet project after the Notos experiment. First, following the same machine learning idea of Notos but applied to the DGA algorithms, meaning they applied the X-means clustering algorithm to DGA algorithms. Secondly, they used a boosted decision tree classification algorithm (Alternating Decision Tree) to test associations of an NXDomain's response to DGAs. Finally, they created Hiddem Markov models for each of the DGAs domains to be able to classify the responses to a DGA directly. We'll use their work to improve the DGA features already gathered with the other papers.

Fast-flux

The behavior of IP fluxing has been well analyzed by the community and most papers propose similar features to detect fast-flux. They often differ with different settings for their experiments. On the other hand, the big challenge for our thesis was to find the features that allowed us to find differences betweent malicious fast-flux networks(FFSN) and content delivery networks(CDNs). Both use fast-flux for different reasons making it hard to differentiate.

The common features presented by Salusky et al.[7], Nazario et al.[41], Perdisci et al.[77], Holz et al.[78] and Stalmans et al.[79] bring forward the features to detect fast-fluxing in general such as a large amount of unique A RRs for a domains, numerous unique NS for a domain and different Autonomous System (ASN) for the IPs linked to the same domain.
The next papers present additional features to distinguish detection specifically of the FFSN from the CDNs.

Nazario et al. [41] provided a really good visual support to how the FF traffic features present themselves.


```
; <<>> DiG 9.8.1-P1 <<>> lovenewgirl.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12744
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 5, ADDITIONAL: 0

;; QUESTION SECTION:
;lovenewgirl.com.               IN      A

;; ANSWER SECTION:
lovenewgirl.com.        600     IN      A       67.190.124.84
lovenewgirl.com.        600     IN      A       78.60.45.112
lovenewgirl.com.        600     IN      A       118.14.224.139
lovenewgirl.com.        600     IN      A       194.90.36.187
lovenewgirl.com.        600     IN      A       222.106.31.112
```

```
;; ANSWER SECTION:
lovenewgirl.com.        600     IN      A       67.190.124.84
lovenewgirl.com.        600     IN      A       78.60.45.112
lovenewgirl.com.        600     IN      A       118.14.224.139
lovenewgirl.com.        600     IN      A       194.90.36.187
lovenewgirl.com.        600     IN      A       222.106.31.112
```

```
IP              | ASN    | Country  | Timezone
----------------------------------------------------------
67.190.124.84   | AS7922 | US       | America/Denver
78.60.45.112    | AS8764 | LT       | Europe/Vilnius
118.14.224.139  | AS4713 | JP       | Asia/Tokyo
194.90.36.187   | AS1680 | IL       | Asia/Jerusalem
222.106.31.112  | AS4766 | KR       | Asia/Seoul
```

```
Lat: 39.7437 Lon: -104.9793
UTM: 37Z
MGRS: 13SED0177499311
```

```
Lat: 54.6833 Lon: 25.3167
UTM: 40R
MGRS: 35ULA9148060851
```

```
Lat: 34.6833 Lon: 135.8333
UTM: 36Z
MGRS: 53SNU7633338239
```

Perdisci et al. [77] aimed at improving the Honeynet's features [7]. Their method consisted in collecting recursive DNS logs for a period of time, filter most of the non-fast-fluxing traffic. Secondly, they grouped the domains names using different features such as similar ISPs, same CDN. And finally, they classify, using a Decision Tree algorithm, the clusters of domain names as malicious or legitimate. They used a base of features provided by [80] and added their own. We used some of the features they presented for our classifiers as well as the traffic volume reduction filters to improve some of the features already on-boarded.

In the following paper, Holz et al. [78] propose some novel features compared to the other papers. They presented the restrictions FFSNs face compared to CDNs: problems such as location and uptime that FFSN can't garantee. From all the features they captured they introduced functions to classify FFSN and CDNs:fluxiness and flux-score. They even considered the possibility of botnets mimicking CDNs but the metrics used already take into account the restrictions FFSN have that can't be avoided. The rest of the study approaches the detection of FFSN using the HTML content returned by the spam websites. We focused our interest in the 2 functions presented in the paper as new features for our classifiers.

Celik et al. [81] have regrouped the large majority of features encountered in the other papers accompanied with some novel additions. They have 5 categories of features: answer-based, domain name-based, spatial-based, network-based and timing-based. We added to our list of features the ones we hadn't seen yet such as the spatial approach looking at the entropy of time zones related to A or NS records. The second interesting set of features we attempted to use in our thesis were the timing-based featured analyzing delays related to different actions (network, processing and document fetching) but we realized that we didn't have the capacity of building the infrastructure necessary to analyze them.

Perdisci et al. [82] is a project tested using various geographical inputs that allows the authors to detect fast-fluxing actors long time, multiple weeks, before they appear in usual blacklists. The study clusters the groups of IPs resolving to the same domains in a specific range of time then tries to analyze if those clusters are a flux network. What is really interesting about the features of this paper is that they aren't all novel but their application to these cluster provides a lot of insights.

DNS tunneling

Another evasion technique where botnets exploit the DNS protocol is the concealing of data exchanges through DNS tunneling. We explore below some of the studies that have proposed features and methods to detect such behavior.

In [83], Dietrichyz et al. analyze the use of TXT RR with segmented and encrypted data. Their study resulted in a set of features mostly analyzing the strings characteristics. Rdata features: we look for the Shannon entropy of the strings. Measures the randomness of the string. Since encrypted data as a high level of entropy this is one of the things we'll be looking for. We are looking for "high byte entropy". Because of inherent reasons this entropy for a small string can't reach the max, we are looking at the "statistical byte entropy" instead. They expect these behavioral communication features to be effective enough in order to extend a classifier based on the rdata features.

In this article written by G. Farnham [84], he proposed a visual approach to detecting DNS tunneling, by plotting the set of features proposed, you can detect by "visual anomaly detection" the presence of DNS tunneling in your network. He uses a simple chart where he output the following:

- On the x-axis: destination IP
- On the y-axis: character count

He then uses the radius of points to showcase the hostname length and their color to identify the request type.

Born et al. [85] [86] present 2 papers on DNS tunneling detection focused on the n-gram frequency analysis. They also detect tunneling by comparing the distribution of domains to the zipf's law patterns. This paper shows how similar some analysis of dns tunneling are very similar to DGA analysis. This is because some of the encoded data used in the transfers has the same properties. In the second paper they combine a visual and quantitative approach using the same type of features used in the first one.

Domain shadowing

Research keeps improving their detection capabilities and botmasters are struggling. Domain shadowing is the latest trend used by malicious actors which provides really high levels of stealthiness. For this reason we think that adding the best features able to intercept this technique could be very beneficial.

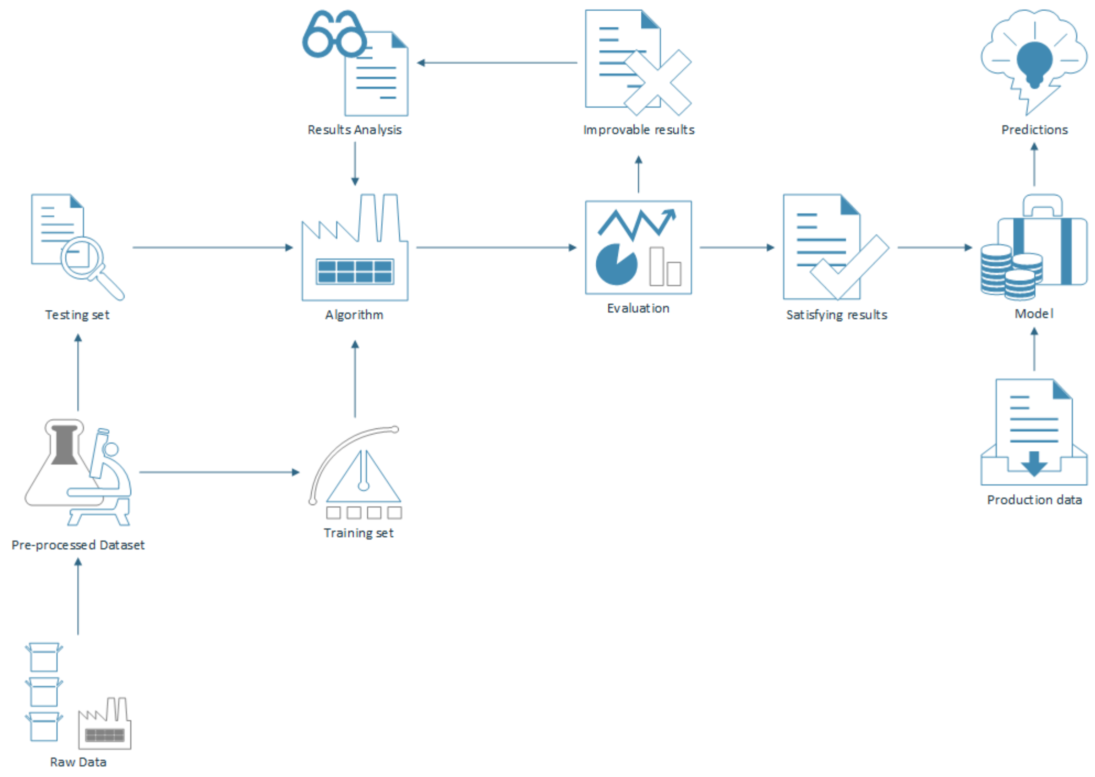
Liu et al. [87] are the first to propose an approach against this new trend. They focused on the 3 RR associated with subdomain creation: A, AAAA(IPv6) and CNAME (alias). Their goal is to detect the bulk creation of shadowed domains. They exploit 2 indicators differentiating these subdomains from normal ones: Deviation from legitimate domains under the same apex and correlation amongst shadowed domains under different apex. By apex, we are talking about the compromised domain used for the creation of subdomains.

Chapter 3

The experiment

During this thesis, the ideas for the experiment changed a lot, we started by mimicking the architecture and features from Exposure to see if we could reproduce their experiment. We rapidly realized that their features were very complicated to obtain for almost all the datasets we could find. We also realized that we didn't have a continuous improvement of our classifier because we didn't have access to a live environment passive DNS capture as Bilge et al. did. That is when we realized a gap that we could try to fill. We decided to use the most pertinent aspects of Exposure's architecture and features and combine them with the research we did on all the current techniques used to abuse the DNS protocol.

3.1 Pipeline of the research



The pipeline we created follows the machine learning workflow of a supervised classification problem. We started by curating the dataset, transforming the captures into bro logs. From the bro logs we extracted the list of features and computed all the features. The next processing is the labeling, we used the information related to the malicious hosts and used it to label each one of the features sets. At this point, we have data that can be ingested by supervised algorithms. Before starting to train

our models, we do a first analysis of the features and go through the feature selection process. We do a visual analysis of the distributions of the features and a correlation analysis.
During this feature selection we create a couple of different sets of features that will will use to train and compare our selections. We follow by balancing our dataset and we split it into 2 sets, the training set and the testing set. We train the classifiers with the different algorithms using the training set. During this training process, we use a feedback loop to change the features used and test out different hyper-parameter optimization techniques. We then test the predictions of our classifiers against the testing set. And finally, we analyze our results and draw some conclusions on our experiment.

3.2 Environment

To build our experiment we used, Python 2 as our programming language. This choice was done because the machine learning libraries available for this programming language are very extensive and they have been adopted largely by the data science community. The libraries related to data science that were used in the project are the following: pandas, numpy, scikit-learn, bro, pcap.
All the machine learning algorithms used were part of the scikit-learn library except for XGBoost which was installed with separately.
Some of the features needed some very specific data. To compute the n-grams [88] features we used a repository called the Natural Language Corpus Data that provided all the necessary stats around the most common used n-grams in the internet for legitimate sites.
All the datasets were extracted with a 16 Gb or RAM laptop with an i5 8th Gen processor. All features were computed with the same laptop.

3.3 Datasets

This is a crucial part of the research, based of the approach datasets will be different and the dataset used will influence all the results obtained and will be a big part of the discussions.
For supervised approaches, there is a need of labeled data. We have searched for DNS traffic coming from Botnets combined with Normal traffic. This is a very difficult task because there aren't many datasets that focus on DNS data generated by botnets. It is also difficult because those datasets are complicated to produce and they would take a lot of time if they had to be labeled manually.
The datasets used in this thesis come from different sources that try to provide labeled data or precise data from botnets. These different sources are the CIS (Canadian institute of Security) that provide datasets created for the research related to security on traffic, specifically for machine learning techniques. They have labeled data from a lot of different sources and protocols. The one they provided me focused on different types of botnets.
The ISOT Botnet dataset [89] which consists in a large dataset regrouping 9 malicious botnets traffic in a controlled environment.
And finally, the Czech Technical University (CTU) scenarios. They created a project called the Malware Capture Facility (MCFP) to provide researchers with botnet captures. Their repository holds currently more then 300 captures of botnets.

Now that we had the labeled malicious traffic we needed normal traffic to use for our classifiers. We decided to use the traffic provided by CTU.

3.4 Purpose of datasets

The idea of having different datasets allows to test out our classifiers against completely new data. If some techniques work against a large variety of evasion techniques or on specific ones. It is a way to propose a more robust set of features for complete detections.
Finding proper datasets with botnet traffic with malicious DNS traces is not an easy thing to achieve. Luckily, the CIC team were kind enough to allow me to use their dataset designed for botnet traffic analysis. We also found a dataset proposed by the CTU which is a set of 13 scenarios that represent different types of malicious traffic generated by botnets. The CTU dataset was not as rich in DNS traffic as the other ones, but provided with fast flux traffic to test some of the features proposed to detect FF.

3.4.1 CTU

This is a set of datasets provided by the malware capture facility project[90]. It captures malicious traffic from different malwares then provide them to the public for research. They also provide their own analysis on most of the datasets which provides insight on how the malware act and how it can be detected.
Their main project is called MCFP-13 which is a dataset of 13 scenarios using botnet traffic analyzed and labeled. Their captures provide a lot of different formats which is very helpful when trying to use different datasets and group them together. They also provide hundreds of additional datasets but that are not as well documents and presented which makes their research more complicated.

3.4.2 CIC

This is a dataset maintained by the Canadian Institute for Cybersecurity[91] (CIC). The have compiled malicious traffic from different sources and for most of them the traces are labeled which is very convenient for machine learning supervised approaches.
The dataset they provide for Botnet traffic analysis is composed of 9 different Botnets and of 2 datasets, a training and a testing dataset. Unfortunately, even being the largest labeled dataset, the DNS traffic was a very low proportion of the malicious traffic.

3.4.3 ISOT

This is a dataset specifically build around DNS traffic by the The ISOT Lab from the University of Victoria. It regroups 9 exploit kits ran in virtual environments in a closed network with a custom DNS server to sniff all the traffic. This dataset provides us with 3 sets of traces, malicious, benign and a mixed[89].

3.5 Dataset Processing

Because of the amount of different features tested in the thesis. I decided to use 2 pcap extractors. The first was provided by our teacher Prof. J. Colin which is a

very effective extractor and summarizes very well the relevant data for most of the features but for labeling and some of the features, more information was required. What we did to obtain a second extractor to complement the information was to run the traffic datasets through Bro, the network IDS. This provided us with DNS.logs, that we then converted using the python "bat" library which converts directly bro logs to dataframes.
Most of the labeled datasets actually only provided the IP addresses of the malicious traffic, this is why working with the bro logs was essential to label the datasets.

3.5.1 cleaning

For the cleaning of the dataset we checked for any data that was corrupted or missing, luckily the datasets used didn't have any of these problems.

3.5.2 Scaling

To solve the problems of scaling introduced when analyzing multiple features we decided to use a panel of techniques and assess which one of them would perform better.
We know that normalization and standardization are the main techniques used to solve this issue but this article [57] proposed to actually test other techniques because depending on the data they might provide better results.
The techniques we tested are the following: StandardScaler (standardization), MinMaxScaler(normalization), RobustScaler, QuantileTransformer (uniform and normal distributions), PowerTransformer and Normalizer.

3.5.3 balancing the datasets

It is very important in supervised learning to balance the dataset or the majority class would mistrain the classifier and output biased results. You could obtain really good accuracy scores but then fail totally at predicting any new data, because you would actually simply predict the majority class. To balance the dataset we decided to down-sample the data from the majority class in a 50:50 ratio.

3.5.4 Assessment model (for features and models)

To improve our results we did the following, we used a first loop scaling the data in one of the 7 techniques. We initiated the different algorithms we wanted to test in the second loop, each one of them assigned a parameter grid for the hyper-parameter optimization. Finally, we created different subsets of data with different features to test out the best combinations and our feature selection choices.
We then used the metrics explained in chapter 5 (results) to assess the different parameters, options and sets used.

3.6 Machine algorithms

3.6.1 Algorithm selection

Based on popular algorithms used to solve this type of problems and popular algorithms used for classification we decided to test the following algorithms to create the best classifier.

Gaussian Naive Bayes

This comes from the know probability and statistics Bayes' theorem. "It describes the probability of an event , based on prior knowledge of conditions that might be related to the event"[92]. In ML learning the model saves the probabilities of each feature to belong to which category and when it is asked to predict new data, it does so by computing the event's probability to belong to each one of the categories. It bases its predictions on previous experience.

K-Nearest Neighbors

k-Nearest Neighbors (kNN) is a very simple supervised ML algorithm. kNN classifies new objects based on their nearest neighbors. The parameter k represents the amount of nearest neighbors it looks for before it assigns the majority. The model created by kNN is actually the entire training dataset since it will use all the neighbors to determine the classification of a new item. kNN algorithm works well with a low amount of features, with datasets with large dimensions computing the distances for the k-nearest neighbors becomes computationally expensive.

Decision Trees

Decision Trees(DTs) are supervised algorithms that based on observations of labeled data creates a model of decision that is represented by a tree where all the nodes of the tree are a condition for the features and where all the leaves are the label they are classified as. The classification process starts at the root and ends in one of the leaves.
DTs are very simple to visualize and perform really well even in higher dimensions. The most common problem of DTs is the overfitting of the data. Because outliers and unbalanced datasets will create branches that don't generalize the data correctly.

Logistic Regression

Logistic regression (LR) is a simple algorithm that is very effective in binary classification. Its core is the logistic function which transforms ranges of real-values into [0-1] ranges. What the logistic regression algorithm will do is through the use of weights of features, create the logistic function (model) for 1 of the classes and then for new values, it will output a probability which can be seen as a prediction of the input belonging to the class or not [93].

Support Vector Machine

Support Vector Machine (SVM) is a supervised algorithm that excels in classification problems. The algorithm looks locally between the classes what hyper-plane separates them. Its hyper parameters are very interesting too, they allows to adapt to the dimension of the features to allow for transformations of the hyper-planes relative to the dimension. They also allow to tune how much we accept outliers. The tuning has to be done with care because we could end in an overfitting situation that doesn't generalizes enough and therefore could perform poorly with new data.

XGBoost and Adaboost

Both algorithms, eXtreme Gradient Boosting (XGBoost) and Adaptive Boosting (AdaBoost), are based on a similar concept which is boosting. Boosting is a technique that modifies weak learners into strong learners. It is done by training multiple of these weak learners sequentially and each improving based on the previous one.[94] Both use this technique with different approaches to it.

AdaBoost The boosting used with Adaptive Boosting(AdaBoost) is simpler than XGBoost, it uses decision trees with a single decision node, they are called decision stumps. Each mistake made by stumps during classification are forwarded to the next stump by carrying more weight. This will allow the errors to be corrected of the numbers of stumps and obtain a cleaned up result in the end. This concept is mostly applied with DTs but it could be applied to any supervised algorithms. As are trees, AdaBoost are victims of outliers but they do a good job not to overfit the models as its peer.

XGBoost The type of boosting used with XGBoost [95] is Gradient Boosting on steroids. The algorithm improved is the RFs which is already a really strong supervised classifier. Like the AdaBoost, it sequentially improves the current tree using its predecessor but instead of adding more weight to the misclassified observations, it tries to train the next predictor to those observations. But this isn't all, XGBoost was created for fast and high performance to solve the slow characteristic of gradient boosting. That is where the eXtreme part comes in and where they introduced solutions to its shortcomings: parallelization of the tree construction, distributed computing, out-of-core computing and cache optimization. Due to this, it is one of the most popular algorithms because it provides the best performance on a range of difficult machine learning tasks. Its only drawback is its overfitting but that can be solved fine-tuning its hyper-parameters and its results' interpretation which can be complicated.

Random forest

The Random Forest (RF) algorithm is an improvement of the DTs. The concept is to randomly create low correlated models (forest) and give them the same input. It then picks the majority result as the prediction for the input. This technique provides a good way to protect from the overfitting that DTs are known for and to protect against individual errors of the trees. RF takes advantage of Bagging (Bootstrap aggregation) and Feature randomness to achieve this uncorrelated forest. Bagging consists in giving all the trees the same amount of data but with replaced data (duplicates). Secondly feature randomness as its name implies, at each node the trees only take decision based on a subset of features[96].

Artificial Neural Networks

Artificial Neural Networks (ANN) is an algorithm created similar to how the brain works and its learning capabilities by modeling neurons and their synapses. It works as a black box system with inputs on one side and and outputs on the other which are dependent on the inputs. The black box is a network of neurons grouped in layers, neurons of each layer connect with the ones of the next layer with weighted connections that will determine when the neurons fire. The modeling is around finding the values of the neurons and the connections inside the black box (hidden layers). The way ANNs hidden layers are computed are through gradient descent, a way of automatically updating the weights step by step into a direction that will make them less wrong, based on the output desired[97].

3.6.2 Feature selection

The goal of this step is to avoid using multiple features that provide the same information. This adds more time to the extraction and to the training of the classifiers. As explained in the Machine learning section of the state of the art, there are some common techniques one can use to achieve this: PCA, LDA are really good but introduce difficulties of interpretation of when downsizing features. Univariate feature selection modules from scikit-learn selects the features based on scoring. Manual combination of features and visual analysis can help too understand the features and their relevance.
The techniques used will be detailed in the next chapter.

3.6.3 Training and testing model

For the training and testing part of the experiment, we followed the norm and divided the dataset into a training and testing sets with a 80:20 ratio to provide as much data possible to train the classifier but enough data as well for the testing set to provide substantial statistical value when using it for prediction scores.
Furthermore, we used the 10-k fold cross validation technique for the training set to use the best generalization of the data possible [98].
We decided to leave KNN and SVM out of the algorithms sets because we realized during the training period that the dimension of features was to high for those algorithms to build models in a reasonable time.

Chapter 4

Combined solution process

4.1 Features extraction

After curating the dataset, we can now start to extract the value that will allow us to train our classifiers. The extraction of the data and the computation of the features of the dataset is a very long process. There are a lot of possible features possible for each problem and this is why the state of the art is so important, it will provide you with the current information that can be extracted and the value that it provides. In this section we'll present all the features we extracted from our research and all the features that we selected for the final set.
The more features you have the more precise your classifier becomes but it induces a lot of other problems explained in the machine learning section. Mainly, overfitting and complexity issues.

In this section we will go through some pre-processing we did for the features computation, a presentation of the features extracted in our state of the art research and finally, an analysis of the features for their selection and the final set used for the classifier.

4.1.1 Features pre-processing

To reduce time during the computation of the features we pre-processed some values that are used later by our features. There are 3 main values that we provided for the features:

- N-grams: a lot of features analyzing the domain name use n-grams to detect if the domain name is part of the usual distributions of n-grams for the most reputable sites. To help with their computation, we used tables that provided the frequency of each bi-gram and tri-gram in the top 1 million sites of alexa.com.
- Autonomous Systems (AS): Fast flux have a weakness in their wall of proxies which is that by nature they are located in different ASs based on the randomness of the infections. Therefore, for each IP returned by A queries, we computed the country and the ASN related to their AS.
- Black/white lists: Since Exposure used an architecture that would start by labeling traffic based on white lists and black lists we decided to adopt the idea and introduce them as features instead.

4.1.2 Black lists and white lists

We have done a large research on the blacklists available. The first idea was to use basics lists with an additional check with the virusTotal API to provide up to date information on the domain but with the limited API, it is a very long process it becomes not a viable solution. Instead, we researched the blacklists used by virusTotal and other DNSBL providers and have a very long static list instead. At first we only focused on botnets domains and IPs, but then realized that bots can query their CnC but also try to access any malicious domain or IP, either to upload or download relevant data for the bot. Therefore, the final blacklist used is a very large combination of blacklists that go from domains linked to malware or CnC to domains linked to suspicious phishing/adware campaigns. One of the websites that really helped us in this process is `https://firebog.net/`. Finally, all the lists have been combined into a big unique blacklist used to compute certain features.
For the white lists, the most common resource used is the alexa.com top 1 million sites. We tried to find other resources available but they were all already comprised in the alexa.com list.

4.1.3 The DNS features

A lot of features for botnets, need an aggregation of packets to be computed, that is why we decided to group together all queries under the same query. This simplified a lot of the features specially, for the features related to DNS answers and TTLs. But this could in rare occasions result in aggregating normal and botnet queries under the same scope if they are accessed by both parties, this is to keep in mind if odd results appear because of it.
In the sections below we list the different features presented in the state of the art that we thought could improve Exposure results without the time-based features. In the list below is a large gathering of different ideas where a lot of features have already been rejected to lack of data or resources to obtain them.

Domain-flux Features

length of the domain name excluding TLD (top level domain)
Number of vowels in the Second Level Domain (SLD)
Number of consonants in the SLD
Number of digits in the SLD
SLD tri-gram entropy
SLD tri-gram conditional probability

Fast-flux features

Numerous unique A records for a domain
Numerous unique NS records for a domain
Different Autonomous Systems (AS) for the IPs of linked to the same domain
Different countries for the IPs of linked to the same domain
Short Time-To-Live (TTL)
Ratio of different ranges of IPs
Fluxiness, is the total of unique A records for a domain divided by the number of A records returned for each lookup. This measures consistency in the unique A records returned
Flux-score, is an hyper feature that combines Unique A records, AS numbers and NS records into a single vector.
Number of resolved IPs
Number of domains resolving to the same IPs
Avg. TTL
Network prefix diversity, is a ratio of the /16 network prefixes
IP Growth Ratio, represents the average number of new IP addresses discovered per each DNS response related to any domain
Autonomous System (AS) diversity
Organization diversity
Country Code diversity
DNS packet size
Edit Distance
KL (Kullback-Leibler) Divergence (unigrams and bigrams)
Jaccard Index (unigrams and bigrams)
Time Zone Entropy of A records
Time Zone Entropy of NS records
Number of distinct autonomous systems
Number of distinct networks

DNS tunneling features

As mentioned in the research, a lot of the features proposed for tunneling are the same as DGA because they both involve either payloads or requests that have non-human readable characteristics. To avoid redundance, we have omitted to repeat most of the features similar to DGA.

NXDomain count
TXT RR present
TC Flag is set (truncation of the packet)
Request type (categorical for the different RRs used in tunneling)
Character count
Statistical byte entropy
DGA features

Shadowing

This sections is divided into 2 main sets of features. When mentioning Apex in this section we are referring to the domain name (2LD + TLD).

- Deviation from legitimate domains under the same apex

- Correlation among shadowed domains under different apex

This spawns the following set of features, all related to the subdomains characteristics:

Usage (creation data, ratio of popular domain, web connectivity)
Hosting (hosting deviation, correlation ratio of subdomain)
Activity (first seen date distribution, resolution count and active days)
Name(diversity of name levels and subdomain name length)

4.1.4 Combined solution features

For Exposure features, we have removed the time-based features since we couldn't use them efficiently with the datasets used due to time capture times being too short for those features to provide insightful data.

Number of distinct IP addresses for a given domain during the whole capture.
Number of distinct countries of the IP addresses for given a domain
Number of domains sharing the IP
Reverse DNS query results. The outputted list of IPs from the reverse DNS query is used to extract 5 atomic features
Average TTL
Standard Deviation of TTL
Number of distinct TTL values is the simple TTL count for the entry.
Number of TTL change is the different TTLs values for the same IP/domain.
Percentage usage of specific TTL ranges
Ratio of numerical characters compared to length of domain name.
Ratio of the length of the LMS, where LMS stands for longest meaningful string.

Same as with Exposure, we removed the time-based features.

LMS
Nb of numerical / character
Ngram
DBForwardBackwardSimilarity (FBS)
DBDialup
Nb distinct IP addresses
Nb distinct Countries
Nb distinct Domain Share IP
NXDomain
MXRecordPresent
SOA (min, exp, refresh, retry)
TXTRecordLength
Nb distinct AS
Reputation AS
Size of AS
Average TTL
Nb distinct TTL values
Nb TTL changes
% usage of TTL ranges
Std deviation TTL

4.1.5 Features analysis

As explained before, it would be great if we could use all these features together to create the classifier but that would introduce a lot of different questions. If all features aren't independent, this would introduce a bias towards that group of features giving them more weight. It also would make testing and computing of features extremely costly and long. For this reason, it is important to select a subset of the features proposed and introduce strong features. This can be done through statistical analysis of the features behavior with the different classes, it can also be done visually, this can provide a good idea of features that do not bring relevant information to the problem.

4.1.6 Features discussion

We realized during our work that a lot of these feature we complicated to obtain or the datasets we were working with didn't show any evidence of that feature being relevant due to the technique not being present in the dataset. This selection resulted of an manual analysis of the features against the dataset. This could be improved using automated techniques and testing out more feature selection tools.

4.2 Features selection

In the current status of feature analysis our selection of features is the following:

In Alexa top 1 million
In Blacklist
Nb unique IP addresses
Nb domains sharing IPs
Nb unique Countries
Nb unique ASNs
Nb different IP ranges
Nb distinct TTLs
Ratios of TTL ranges
Standard deviation TTLs
Average TTLs
Length of FQDN but TLD
Nb vowels 2LD
Nb consonants 2LD
Nb digits 2LD
Tri-gram entropy 2LD
Tri-gram conditional probability 2LD
LMS Ratio

Chapter 5

Sets run and result discussion

5.1 Results metrics analysis

The metrics used to assess the quality of the features are the following:

- Confusion Matrix
  - True positives(TP): we predicted "class" and it is "class"
  - True negatives(TN): we predicted "not class" and it is "not class"
  - False positives(FP): we predicted "class" and it is "not class"
  - False negatives(FN): we predicted "not class" and it is "class"
- Accuracy: the proportion of predicted true results (both true positives and true negatives) in the population, that is

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

- Precision: the proportion of predicted positive cases that are indeed real positive, that is

$$\frac{TP}{TP + FP} \tag{5.2}$$

- F-Measure: the harmonic mean of precision and recall. It measures the quality of these 2 metrics but does not take into account the True negatives that is why we introduce Matthews correlation coefficient

$$\frac{2 * precision * recall}{precision + recall} \tag{5.3}$$

- Matthews correlation coefficient(MCC): is a coefficient that is used to assess the quality of binary classifiers. Because it takes into account all the parts of the confusion matrix it is considered to be a balanced metric:

$$\frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.4}$$

- Area Under the Curve(AUC) - ROC: provides a measure of how well a parameter can distinguish between the 2 classes, similar to the accuracy metric. The higher the AUC the better the classifier performs.

We used the Accuracy score of the confusion matrix to decide which algorithm would be the best version of itself.

5.2 Interpretation of the results

5.2.1 Results

Because of the nested loops testing all the scalers, with all the algorithms optimizing the hyper-parameters, here is the summary for each algorithm with their best performance. The different scalers were: StandardScaler (STD), MinMaxScaler (MMS), RobustScaler (RS), QuantileTransformer (QTU QTN), PowerTransformer (Power) and Normalizer (NORM).

Classifier	Best Score	TPR	FPR	Acc	Prec	Err	F-M	MCC	AUC
Xgboost(RS)	0.898	0.925	0.128	0.898	0.101	0.880	0.902	0.798	0.898
DT(NORM)	0.902	0.455	0.050	0.698	0.3013	0.902	0.605	0.463	0.702
RF(STD)	0.884	0.877	0.118	0.879	0.120	0.884	0.880	0.759	0.879
NB(PWR)	0.705	0.851	0.445	0.705	0.294	0.663	0.746	0.4265	0.703
Aboost(PWR)	0.849	0.892	0.195	0.849	0.151	0.8246	0.857	0.7004	0.848
LR(PWR)	0.805	0.832	0.206	0.813	0.186	0.806	0.819	0.626	0.813
NN(PWR)	0.847	0.923	0.171	0.876	0.123	0.847	0.883	0.756	0.876

5.2.2 Features discussion

To understand what features have the most importance and towards what class they pushed the classification towards we used a couple of different tools: Random Forest, LIME, Decision Trees are awesome for so many things, they can help you in the feature selection, you can use them to classify data and by nature their structure allows to understand how they classifies the data into the different classes.
Local Interpretable Model-agnostic Explanations (LIME) and Shapley values and additive explanations (SHAP) are options that provide a lot of information as well in regards to interpretation.

5.2.3 Results discussion

We obtain scores that do not surpass the 90% accuracy score, which for this type of problem is a low score but could be explained by a lot of different factors.

It is interesting to see that our approach to test out different scalers showed its value, since we can see that the best scores use multiple different scalers for each classifier. We can see that the PowerScalers seems to provide the best format of data for most classifiers and the interesting bit is that the most common scaler (MMS) doesn't provide the best scores for any algorithm.

The focus of our work is to detect the moment where the bots will use the DNS protocol as the channel with their CnC, because we know that whatever protocol they are using as a communications mean they have to go through the DNS protocol. They need to resolve domains, if they do not use DNS and directly use the IPs, they will be spotted rapidly by most IDS or security tools.
When looking at the predictions made by our classifier, we realized that a lot of the DNS traffic of the bots labeled as normal was related to normal traffic. What was interesting is to note that our approach isn't able to detect traffic related to the attacks performed by the bots. One of the examples was a bruteforce to Wordpress sites, all the DNS traffic generated from that attack looked completely normal. The reason it is important is because most traffic is labeled by IP of the infected host, therefore all

DNS traffic from that host is counted in the predictions as well. We could argue that the attack traffic was used in the training part as well and could be recognized but since it looks exactly like legit traffic because there was no interactions with the rest of the botnet the weight of the normal traffic would take over.

The high predictions from the Exposure and Heuer et al. came from 2 different things from our research. Exposure used a very large sample making it more reasonable in proportion to obtain higher scores, specially as they were manually verifying the results each day to improve the classifier.

In our initial implementation our first goal was to introduce 2 additional evasion techniques to Exposure, DNS tunneling and DNS shadowing but we didn't realize that finding datasets for those 2 categories would be so hard, especially mixed with the other evasion techniques which are more slightly more noisy. In this regards, an anomaly detection method would probably have suited these 2 techniques better. That is also sustained by the visual effectiveness of DNS tunneling detection which can be easily spotted.

Chapter 6

Conclusion

In this project, we realized an experiment to provide smaller environments with the capabilities of solutions such as Exposure. We did so by removing the complicated features from Exposure and tried to find other sets of feature to improve the solution. In addition to providing new features, we also tested other machine learning algorithms such as XGBoost and did a wide testing of scalers and hyper-parameters.

Unfortunately, the results obtained didn't reach the expected results from Exposure solution, and even if we obtain results in the lower end of the 90% range, those results are considered very low when it comes to supervised malicious traffic classification because the FP rate will generate a huge amount of alerts and result in too much noise for analysts.

6.1 Improvements propositions

We realized that the detection of botnet traffic is very complicated and to hope catch all types of evasion techniques, a combined approach might not be the best solution. But that anomaly detection techniques that are given the right features and that create a really good baseline for normal traffic could be a better approach.

Another approach that we considered but didn't test is a special architecture for botnet detection: Training 4 classifiers based each on a single evasion technique. This could provide a better detection of the individuals techniques and because they are piped after each other, malicious traffic would have a harder time not being detected.

The feature selection process and the interpretation of the models in the project haven't been exploited enough, to obtain better results a reduction of features using PCA or selecting the best features would have provided with some improvement and an understanding to our results.

Using larger datasets such as the ones provided by ISC SIE (caida.org) with additional manual labeling could be an option as well

Another improvement could be the pipeline with 5 classifiers following each other with each one of it trained specifically on 1 DNS evasion technique. The experiment would be to compare the efficiency and comprehension of the results provided by both architectures.

Use better hardware such as hydra to test out more options and algorithms. (SVM, xgboost, knn) The reason those algorithms took long is because we tried them under a lot of different scaling techniques and optimizing their hyper-parameters.

Improve the labeling of datasets used going through a manual review, this would ensure that the final results are focused on the detection of the channel communication part of the infection and not on the rest. Our goal was able to improve the detection of botnets.
Trying an anomaly detection approach instead, this would reduce lot of problems related to normal traffic seen from the botnets and focus on the parts that are unusual and which make botnets discoverable through DNS analysis.

Bibliography

[1] M. Mendoza. *Vulnerabilities reached a historic peak in 2017*. Feb. 2018. URL: https://www.welivesecurity.com/2018/02/05/vulnerabilities-reached-historic-peak-2017/.

[2] *New Mirai Botnet Variant Targets IoT TV, Presentation Systems*. Mar. 2019. URL: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/new-mirai-botnet-variant-targets-iot-tv-presentation-systems.

[3] *Botnets*. 2018. URL: https://www.cyber.nj.gov/threat-profiles/botnet-variants/.

[4] A. Manasrah M. M. Khadhum K. Alieyan A. ALmomani. "A Survey of Botnet Detection Based on DNS". In: *Neural Comput. Appl.* 28.7 (July 2017), pp. 1541–1558.

[5] D. O'Brien M. Wielogorska. "DNS Traffic analysis for botnet detection". In: *Irish Conference on Artificial Intelligence and Cognitive Science* 25th.issue number (Dec. 2017).

[6] N. Goodman. "A Survey of Advances in Botnet Technologies". In: *CoRR* abs/1702.01132 (2017). arXiv: 1702.01132. URL: http://arxiv.org/abs/1702.01132.

[7] R. Danford W. Salusky. "The Honeynet Project: Know your enemy: Fast-flux service networks". In: (2007).

[8] G. Ollmann. "Botnet Communication Topologies". In: *Damballa* GIAC (GSEC) Gold Certification (June 2009).

[9] R. C. Pinto S. S. Silva R. M. Silva and R. M. Salles. "Botnets: A survey". In: *Eslervier* 57.2 (2013), pp. 378–403.

[10] J. Bardin. *Introduction to Botnets*. June 2016. URL: https://ccdcoe.org/cycon/2012/workshops/Intro_to_Botnets.pdf.

[11] *Uses of botnets*. June 2018. URL: http://www.honeynet.org/node/52.

[12] Y. Emre. "A literature survey about recent botnet trends". In: (June 2011). URL: http://geant3.archive.geant.net/Media_Centre/Media_Library/Media%20Library/botnet_trends_M2.pdf.

[13] KAFEINE. *Smominru Monero mining botnet making millions for operators*. Jan. 2018. URL: https://www.proofpoint.com/us/threat-insight/post/smominru-monero-mining-botnet-making-millions-operators.

[14] M. Nir M. Mahmoud and A. Matrawy. "A Survey on Botnet Architectures, Detection and Defences". In: *International Journal of Network Security* 17 (Jan. 2014).

[15] B. Sayed W. Lu S. Saad A. Ghorbani D. Garant D. Zhao I. Traore. "Botnet detection based on traffic behavior analysis and flow intervals". In: *Elsevier Computers and Security* 39.issue number (Nov. 2013), pp. 2–16.

[16] S. Ramadass M. Feily A. Shahrestani. “A survey of botnet and botnet detection”. In: *IEEE 3rd international conference on emerging security information* systemsand technologies (2009), pp. 268–273.

[17] Y. Chen Z. J. Fu P. Roberts K. Han Z. Zhu G. Lu. “Botnet research survey”. In: *32nd Annual IEEE International Conference on Computer Software and Applications* (Aug. 2008), 967–972.

[18] Z. A. M. Noh M. Z. Mas’ud S. R. Selamat R. Yusof R. S. Abdullah M. F. Abdollah. “Revealing the criterion on botnet detection technique”. In: *IJCSI International J Computer Science* 2.10 (2013), 208–215.

[19] O. A. Abouabdalla S. Ramadass A. M. Manasrah A. Hasan. “Detecting botnet activities based on abnormal DNS traffic”. In: *CoRR* abs/0911.0487 (2009).

[20] V. Yegneswaran P. Barford. “An inside look at botnets”. In: *Advances in Information Security* 27.Springer (Mar. 2007), pp. 171–191.

[21] V. Yegneswaran M. Fong W. Lee G. Gu P. Porras. “Bothunter: detecting malware infection through ids-driven dialog correlation”. In: *USENIX Security Symposium on USENIX Security Symposium* 16 (2007), pp. 1–16.

[22] H. Lee H. Kim H. Choi H. Lee. “Botnet detection by monitoring group activities in DNS traffic”. In: *7th IEEE international conference on computer and information technology* CIT 2007 (2007), 715–720.

[23] G. Yan Z. Zhang L. Liu S. Chen. “Bottracer: executionbased bot-like malware detection”. In: *Springer, Information security* CIT 2007 (2008), 97–113.

[24] P. Garcia-Teodoro R. A. Rodriguez-Gomez G. Macia-Fernandez. “Survey and taxonomy of botnet research through lifecycle”. In: *ACM Comput Survey* 45.4 (2013).

[25] M. Shiraz S. S. A. Shah I. Awan N. B. Anuar A. Karim R. B. Salleh. “Botnet detection techniques: review, future trends, and issues”. In: *J Zhejiang Univ Sci C* 15.11 (2014), 943–983.

[26] T. Holz J. Goebel. “Rishi: identify bot contaminated hosts by IRC nickname evaluation”. In: *Proceedings of the first Conference on First Workshop on Hot Topics in Understanding Botnets* (2007), p. 8.

[27] *Taxonomy of botnet threats*. Nov. 2006. URL: `https://sites.cs.ucsb.edu/~kemm/courses/cs595G/TM06.pdf`.

[28] H. B. Jethva J. Vania A. Meniya. “A Review on Botnet and Detection Technique”. In: *IJCTT* 4.1 (2013), pp. 23–29.

[29] L. Cavallaro S. Zanero S. Schiavoni F. Maggi. “Phoenix: DGA-based Botnet Tracking and Intelligence”. In: *Springer* Detection of Intrusions and Malware, and Vulnerability Assessment (July 2014), pp. 192–211.

[30] T. Wijsman. *What is DNS and which Server do I choose?* Feb. 2012. URL: `https://blog.superuser.com/2012/02/16/what-is-dns-and-which-server-do-i-choose/`.

[31] *What is the DNS Protocol*. URL: `https://ns1.com/resources/dns-protocol`.

[32] *How DNS Protocol Works*. May 2018. URL: `https://www.hack2secure.com/blogs/how-dns-protocol-works/`.

[33] P. Mockapetris. *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION*. Nov. 1987. URL: `http://www.zytrax.com/books/dns/apd/rfc1035.txt`.

[34] *DNS Messages*. July 2017. URL: http://www.zytrax.com/books/dns/ch15/#answer.

[35] *Domain Name System (DNS) Parameters*. June 2019. URL: http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml.

[36] J. Wang X. Li and X. Zhang. "Botnet Detection Technology Based on DNS". In: *MDPI, future internet* 9.4 (Sept. 2017), p. 55.

[37] *About fully qualified domain names*. 2019. URL: https://kb.iu.edu/d/aiuv.

[38] A. Abakumov. *dga_algorithms*. Feb. 2016. URL: https://github.com/andrewaeva/DGA/tree/master/dga_algorithms.

[39] *Round-Robin DNS*. Feb. 2019. URL: https://en.wikipedia.org/wiki/Round-robin_DNS.

[40] *Round-Robin Scheduling*. Apr. 2019. URL: https://en.wikipedia.org/wiki/Round-robin_scheduling.

[41] T. Holz J. Nazario. "As the net churns: Fast-flux botnet observations". In: *The 3rd International Conference on Malicious and Unwanted Software* (Oct. 2008), 24–31.

[42] *Fast flux*. Apr. 2019. URL: https://en.wikipedia.org/wiki/Fast_flux.

[43] S. Sparks P.Wang and C. C. Zou. "An advanced hybrid peer-to-peer botnet". In: *Proceedings of the first Conference on First Workshop on Hot Topics in Understanding Botnets* (2007), p. 2.

[44] M. Kotter G. Wicherski P. Bacher T. Holz. "Know your Enemy: Tracking Botnets". In: *The Honeynet Project* (Oct. 2008).

[45] D. O Brien M. Wielogorska. "DNS Traffic analysis for botnet detection". In: *25th Irish Conference on Artificial Intelligence and Cognitive Science* (Dec. 2017), pp. 7–8.

[46] *What is DNS tunneling?* Mar. 2006. URL: https://www.infoblox.com/glossary/dns-tunneling/.

[47] A. Hinchliffe. *DNS Tunneling: how DNS can be (ab)used by malicious actors*. Mar. 2019. URL: https://unit42.paloaltonetworks.com/dns-tunneling-how-dns-can-be-abused-by-malicious-actors/.

[48] J. Segura. *Domain Shadowing With a Twist*. Mar. 2016. URL: https://blog.malwarebytes.com/threat-analysis/2015/04/domain-shadowing-with-a-twist/.

[49] *domain-shadowing*. Feb. 2018. URL: https://www.normshield.com/domain-shadowing/.

[50] F. Z. Mohamad I. Zakira A. Shahid Z. M. Jasni. "A Review Paper on Botnet and Botnet Detection Techniques in Cloud Computing". In: *ISCI* (Sept. 2014).

[51] Q. C. Nguyen X. D. Hoang. "Botnet Detection Based On Machine Learning Techniques Using DNS Query Data". In: *Future Internet* (May 2018).

[52] N. Biasini. *Threat Spotlight: Angler Lurking in the Domain Shadows*. Mar. 2015. URL: https://blogs.cisco.com/security/talos/angler-domain-shadowing.

[53] Team RiskIQ. *Domain Shadowing: When Good Domains Go Bad*. June 2016. URL: https://www.riskiq.com/blog/external-threat-management/domain-shadowing-good-domains-go-bad/.

[54] A Melegari M. Varone D. Mayer. *What is Machine Learning? A definition*. URL: https://www.expertsystem.com/machine-learning-definition/.

[55] A. Pant. *Workflow of a Machine Learning project*. Jan. 2010. URL: https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94.

[56] U. Jaitley. *Why Data Normalization is necessary for Machine Learning models?* Oct. 2018. URL: https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94.

[57] S. Geller. *Normalization vs Standardization — Quantitative analysis*. Apr. 2019. URL: https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf.

[58] F. Malik. *What Is Dimension Reduction In Data Science?* Dec. 2018. URL: https://medium.com/fintechexplained/what-is-dimension-reduction-in-data-science-2aa5547f4d29.

[59] *Supervised vs. Unsupervised Learning. Which is better?* Apr. 2019. URL: https://lawtomated.com/supervised-vs-unsupervised-learning-which-is-better/.

[60] D. Soni. *Supervised vs. Unsupervised Learning*. Mar. 2018. URL: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[61] D. Senapati. *Grid Search vs Random Search*. Aug. 2018. URL: https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318.

[62] *10-fold Crossvalidation*. URL: https://www.openml.org/a/estimation-procedures/1.

[63] Jason Brownlee. *Metrics To Evaluate Machine Learning Algorithms in Python*. May 2016. URL: https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/.

[64] L. Hulstaert. *Interpreting machine learning models*. Feb. 2018. URL: https://towardsdatascience.com/interpretability-in-machine-learning-70c30694a05f.

[65] *SNORT*. Mar. 2006. URL: https://www.snort.org/.

[66] M. F. Zolkipli Z. Inayat S. Anwar J. M. Zain. "Taxonomy of botnet threats". In: *Trend Micro Incorporated* (Nov. 2006).

[67] E. Gerhards-Padilla D. Plohmann and F. Leder. "Botnets: measurement, detection, disinfection and defence". In: *ENISA Workshop* (Mar. 2011).

[68] D. Balzarotti E. Kirda L. Bilge S. Sen. "EXPOSURE: a Passive DNS Analysis Service to Detect and Report Malicious Domains". In: *In Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)* (Feb. 2011).

[69] I. Schiering T. Heuer et al. "Recognizing Time-Efficiently Local Botnet Infections - A Case Study". In: *International Conference on Availability, Reliability and Security (ARES)* (Aug. 2016), pp. 304–311.

[70] G. Cheng D.-T Truong. "Detecting domain-flux botnet based on DNS traffic features in managed network". In: *Security and Communication networks* 9.14 (May 2016), pp. 2338–2347.

[71] A. Reddy S. Ranjan S. Yadav A. K. K. Reddy. "Detecting algorithmically generated malicious domain names". In: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement* (Nov. 2010).

[72] K. Ganame N. Agarwal A. Ahluwalia I. Traore. “Detecting broad length algorithmically generated domains”. In: *In Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments: First International Conference, ISDDC 2017* (Oct. 2017), p. 19.

[73] D. Dagon W. Lee N. Feamster M. Antonakakis R. Perdisci. “Building a Dynamic Reputation System for DNS”. In: *Proceedings of the 19th USENIX Conference on Security* USENIX Security.10 (Aug. 2010), p. 18.

[74] A. Mohaisen M. Thomas. “Kindred domains: detecting and clustering botnet domains using DNS traffic”. In: *Proceedings of the companion publication of the 23rd international conference on World wide web companion* (Apr. 2014).

[75] A. L. N. Reddy E. Yadav. “Winning with DNS Failures: Strategies for Faster Botnet Detection”. In: *SecureComm 2011: Security and Privacy in Communication Networks* (2011), pp. 446–459.

[76] Y. Nadji N. Vasiloglou S. A.-N. W. Lee D. Dagon M. Antonakakis R. Perdisci. “From Throw-Away Traffic to Bots: Detecting the Rise of DGA-based Malware”. In: *In Proceedings of the 21st USENIX Conference on Security Symposium* (2012).

[77] D. Dagon R. Perdisci I. Corona and W. Lee. “Detecting malicious flux service networks through passive analysis of recursive DNS traces”. In: *In Computer Security Applications Conference. ACSA 09, Annual. IEEE.* (2009).

[78] K. Rieck T. Holz C. Gorecki and F. C. Freiling. “Measuring and detecting fast-flux service networks”. In: *NDSS* (2008).

[79] B. Irwin E. Stalmans. “Spatial Statistics as a Metric for Detecting Botnet C2 Servers”. In: *Botconf* (2013).

[80] L. Martignoni D. Bruschi E. Passerini R. Paleari. “FluXOR: Detecting and monitoring fast flux service networks”. In: *In Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (July 2008), pp. 186–206.

[81] Oktug Z. B. Celik. “Detection of Fast-flux Networks using Various DNS Feature Sets”. In: *In IEEE Computers and Communications Symposium (ISCC)* (2013).

[82] G. Giacinto R. Perdisci I. Corona. “Early detection of malicious flux networks via large-scale passive DNS analysis”. In: *IEEE Transactions on Dependable and Secure Computing* (2012), 714–726.

[83] F. C. Freiling H. Bos-M. van Steen N. Pohlmannz C. J. Dietrichyz C. Rossowz. “On Botnets that use DNS for Command and Control”. In: ().

[84] G. Farnham. *Detecting DNS tunnelling*. Feb. 2013. URL: `http://armatum.com/blog/2009/DNS-part-ii/`.

[85] Dr. D. Gustafson K. Born. “Detecting DNS Tunnels Using Character Frequency Analysis”. In: *In Proceedings of the 9th Annual Security Conference* (Apr. 2010).

[86] Dr. D. Gustafson K. Born. “NgViz: Detecting DNS Tunnels through N-Gram Visualization and Quantitative Analysis”. In: *In Proceedings of the the 6th Annual Cyber Security and Information Intelligence Research Workshop* (Apr. 2010).

[87] K. Du H. Wang B. Liu-H. Duan D. Liu Z. Li. “Don’t Let One Rotten Apple Spoil the Whole Barrel: Towards Automated Detection of Shadowed Domains”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 537–552.

[88] Peter Norvig. *Natural Language Corpus Data: Beautiful Data*. July 2008. URL: https://norvig.com/ngrams/.

[89] Ganame K. Woungang I Alenazi A. Traore I. "Holistic Model for HTTP Botnet Detection Based on DNS Traffic Analysis". In: *ISDDC* (2017).

[90] Sebastian Garcia. *Malware Capture Facility Project*. URL: https://stratosphereips.org.

[91] Elaheh Bigla B. "Towards effective feature selection in machine learning-based botnet detection approaches." In: *Communications and Network Security* (2014).

[92] *Bayes' theorem*. July 2019. URL: https://en.wikipedia.org/wiki/Bayes%27_theorem.

[93] J. Brownlee. *Logistic Regression for Machine Learning*. Apr. 2016. URL: https://machinelearningmastery.com/logistic-regression-for-machine-learning/.

[94] J. D'Souza. *A Quick Guide to Boosting in ML*. Mar. 2018. URL: https://medium.com/greyatom/a-quick-guide-to-boosting-in-ml-acf7c1585cb5.

[95] J. Brownlee. *A Gentle Introduction to XGBoost for Applied Machine Learning*. Aug. 2016. URL: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/?source=post_page.

[96] T. Yiu. *Understanding Random Forest*. June 2019. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[97] S. Arzt. *Explained In A Minute: Neural Networks*. Mar. 2018. URL: https://arztsamuel.github.io/en/blogs/2018/EiaM-NeuralNetworks.html.

[98] S. Joglekar. *Overfitting and Human Behavior*. Jan. 2018. URL: https://medium.com/@srjoglekar246/overfitting-and-human-behavior-5186df1e7d19.