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 research 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: 01/01/2019

"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

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

keywords:botnet detection; botnet detection model; machine learning-based botnet detection;**[ref1]** domain generation algorithm botnet detection; fast flux botnet detection

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

List of Abbreviations

CnC	**Command and Control**
DoS	**Denial of Service**
DDoS	**Distributed Denial of Service**
IRC	**Internet Relay Chat**
HTTP	**Hyper Text Transfer Protocol**
IDS	**Intrusion Detection System**
RR	**Resource Record(DNS)**
DNS	**Domain Name System**
TLD	**Top Level Domain**
TTL	**Time-to-Live**
CDN	**Content Delivery Network**
FFSN	**Fast Flux**
DGA	**Domain Generation Algorithm**
FQDN	**Fully Qualified Domain Name**
ML	**Machine Learning**
WSF	**W**hat (it) **S**tands **F**or

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Research context

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

Article: [2] [**CRC_Botnets**] [3] [4] [5] [6] [7] This is a compilation of Botnet news by Trend Micro: [8] Compilation of Botnets variants: [9] [10] [**mylobot**]

1.2 Research question

My research question is : "**How can we detect Botnets through passive DNS traffic analysis?**" TODO: define proper sub-questions that build the paper. 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 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.

[11] 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, Honeypots, Law and future threats. The authors technical experience will also be used to answer the research questions. It is hoped that others can benefit from the knowledge that this thesis provides to implement better security measures within their own organisations. The information provided in this thesis will provide the reader with in depth knowledge on the subject of botnets, and how they the threat of them within an organisation can be migrated.

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.

TODO

Chapter 2

State of the art

[12]

2.1 Botnets

[13] As stated in the introduction botnets are an important problem for anyone involved somehow with the internet. They can result in great economic damage. [**report1**] Especially with their continuous improvement to become more resilient and powerful which makes them an even more important threat.

[14] Botnets can become very lucrative and can infect very large amount of devices resulting in scary tool. 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.

[13] 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

[**memoire1**] [15]

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. 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, exfiltrate data and more actions explained down below. [16]

[17]

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. [**tracking**] [18] 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. 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.

[19]

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.

What is the purpose of botnets? All these resources make botnets very powerful to execute network attacks. [20] 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. [21]

[22]

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. This BaaS is possible with decentralized architectures that can subdivided into smaller botnets to sold and then reintegrated to the parent botnet after use. [**tracking**]

[**memoire1**]

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) [21]

- **Spam email campaigns** Bots are set as proxy nodes and then used to send large amounts of spam and phishing emails.[21]
- **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, ...) [21]
- **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.[21]
- **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$. [23] [**tracking**]

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

- **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 presented in these studies: [13] [25] [26] [27]

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. [**bot-appr**] 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. [28] [29] [30] [31]

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. [32] [33]

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. [34]

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. [35]

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. [20] 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? [18] 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.
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. [36]

How do botnets achieve such channels? [32] [37] To stay invisible and persistent channels have gone through different methods, here are the main ones:

- **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. Their structure will result as mentioned above in trade-offs for its different specifications. [24] [20] As explained in this paper, there are two typical botnet topologies 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.[18]
- **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.[38]

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

[39] [40] [41] [42] [43]

Star [18] 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. 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. [20]

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.

TODO: topologies

Hybrid

P2P

What design to pick ? Here is a summary 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.

IMAGE topologies (cf folder)

[phoenix]

2.2 Uses and abuses of DNS protocol

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

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.

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. [44]

Protocol example to present usual behavior and better understand later, where the malicious actors abuse the protocol.
[45] 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.
TODO: DIAGRAM [46]

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:

```
                               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 (A,AAAA,CNAME,MX,NS,PTR,...)
q_class	Class of the request (often IN for internet)

[47] The server will respond by filling the **Answer section**:
[48]

```
                               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.
[49] Here is a list of the main RR types 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

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.

IMAGE: ADD picture explaining dga

[18]

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).[50]
- **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[51], 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.[**robin_dns**] Round-robin 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.[52]

IMAGE: IP flux

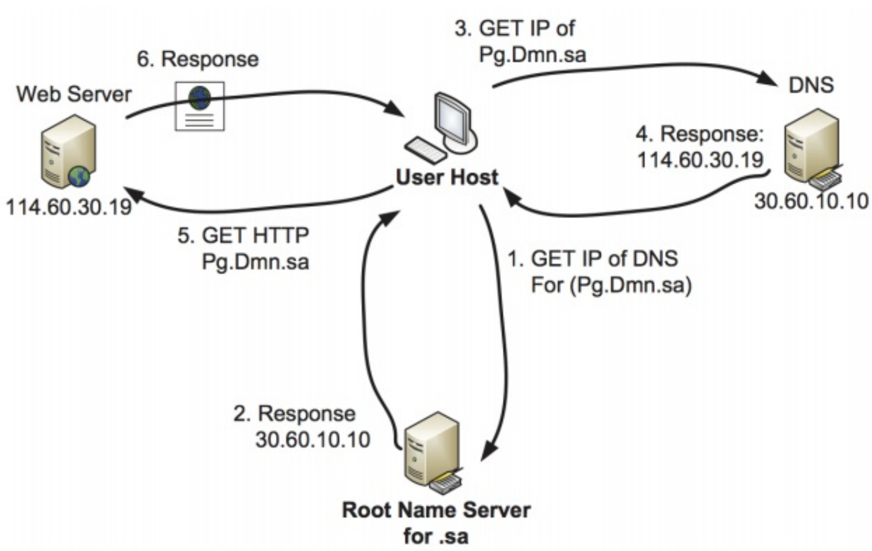
How do botnets exploit this feature ? Fast-Flux is mostly used by botmasters to hide a malicious network behind a large amount of dynamic proxies(flux-agents).[53] When a bot tries to connect with the CnC his request to the FQDN goes through a DNS server that returns one of these proxies which is picked from an immense list of rotating addresses. After that, the flux-agent relays the client's request to the mothership.[**wiki_ff**] 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][54]. 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.[18]

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.[29]
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:

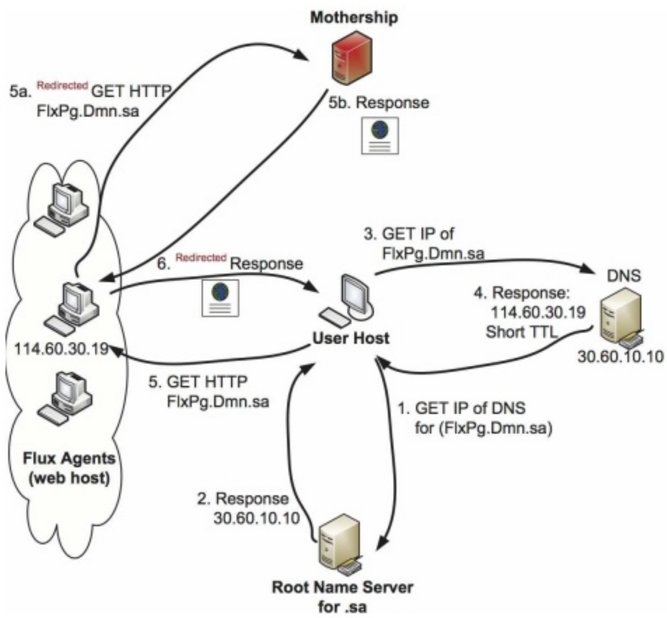
- **Single-flux**: Multiple IP addresses are assigned to the same domain (either CNAME or A records). The IP addresses of the bots are constantly registered and unregistered to the domain record. They have low TTL and most are proxies for master servers.[**wiki_ff**].
- **NS flux**: Multiple NS records 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.

IMAGE replace these below by own pictures

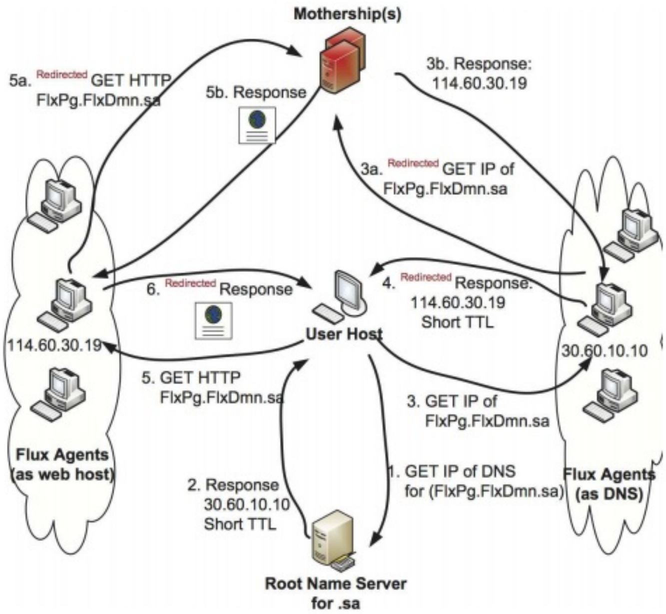
Normal FF[24]



single flux



double flux



ADD IMAGE HERE: [55] [56]

DNS tunneling

DNS tunneling implies the act of embedding other protocols or data through 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 while exfiltrating data. [57] What is positive about this abuse is that there are almost no legitimate reasons for this application, making obvious malicious behavior stands behind if this type of traffic is discovered.[58]

Paloalto's research unit provided a nice medium level overview of DNS tunneling[59]. They explain how the protocol can be used either to exfiltrate or infiltrate data with infected machines. They show how different parts of the protocol can be exploited. Using the request queries names to embed information directly in the domain name queried with encoded data, using the type of RR to indicate the CnC the status or type of response expected, and finally, using the type of the response codes (NXDOMAINS or NOERROR) to define the malicious response.

IMAGE tunneling (cf /img)

Domain shadowing

[60][61] TODO: explain domain shadowing

2.3 Machine learning approach

We use machine learning in our thesis to create models around certain behaviors botnets show to helps us detect 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. [62]

2.3.2 Machine learning for botnet detection

The goal of botnet detection is to distinguish which traffic is normal and which is malicious. We are interested in one of the tasks that machine learning algorithms excel at which is classification. After training the models, classification algorithms are able to predict the belonging of new data.

IMAGE: DIAGRAM OF THE WORKFLOW (DONE)

2.3.3 Machine Learning workflow

Machine learning has a specific workflow that needs to be followed to obtain the best results: [63]: Gathering the data, feature extraction, choice of model, training and testing of the model, and assessment of the prediction capabilities of the model.

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 2 processes: 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.

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 chose to select features based on patterns or features based on data type. Pattern based means through statistical analysis of the data and using unsupervised learning algorithms to extract new tendencies from their results. The goal is to find patterns for each label, 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 been analysed by other researchers and will be based on the nature of the traffic coming from malicious and normal labels.

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 (300 seconds for the TTL of a record), 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' 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]). 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. Finally, we normalize the features[64] by dividing them by the maximum for that feature. The goal of the normalization is to avoid features unbalanced analysis by the algorithms(i.e. TTL is around

$$[300 - 86400]$$

and number of resolved IPs

$$[0 - 15]$$

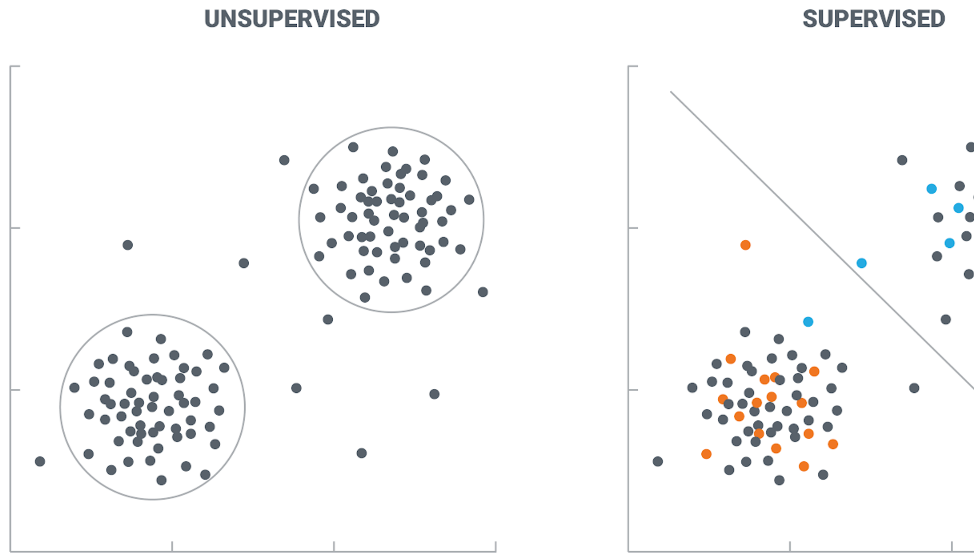
, TTL could have more weight in the algorithm due to the scale difference of the features).

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.
IMAGE: ADD HERE CAPTION OF DIFFERENT CATEGORIES OF ALGORITHMS

(supervised -> labeled(colors) then line between, unsupervised -> clusters, semi-



super -> dunno yet) [65]
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 the typical classification problem as mentioned above where supervised algorithms prevail. To find the best algorithm to model the problem we will test out different algorithms and compare them under the same conditions.

Supervised algorithms have been chosen for the modeling part of the problem but unsupervised of algorithms will be used as well for the research of new features in the **features selection** step.

Training and testing

We now have a balanced and clean dataset with the chosen features. The way models are tested is by dividing a dataset into 2 sets, one for training and one for testing. In addition to this approach, it is combined with a second technique which is the 10-fold cross validation[66]. The training set will be used initially by the algorithm to learn and the testing set will be used to contrast the accuracy of the learning and create a feedback loop. The way datasets are divided into the 2 sets are up to us, the standard is 90-10 to train on a larger portion of the dataset but have a testing set big enough to include most of the variations. The cross validation technique will repeat the experiment changing the the training and testing sets for 10 loops equally and then average out the results, this allows to obtain more realistic results and unbiased by the randomness of sets chosen.

TODO: A CAPTION EXPLAINING 10-fold

Analysis of the model's prediction abilities

Finally, we need metrics that will provide us information of the experiment. A very common way to do so is using the confusion matrix. This provides 4 main metrics that can then be exploited for more detailed information. There are other metrics [67] that can be introduced as well such as:

1. Classification Accuracy.
2. Logarithmic Loss.
3. Area Under ROC Curve.
4. Classification Report.

These metrics will allow us to understand what can be improved and the reason of the score obtained. From there we will adapt the features or the experiment to improve the results obtained. The goal is to reach the best results possible.

2.3.4 How will machine learning be implemented for this Thesis?

A big part of the work will be the feature selection. Going through the papers that have already done research on the subject, doing statistical analysis and using unsupervised learning to possibly extract more features and pick the ones we consider most effective. The algorithms used for detection as well as their parameters are an important part. Furthermore, the challenge in machine learning is to find the features with enough strength to create the model and the right settings for the classifiers so the fit the data correctly, it is very easy to fall in the over-fitting or underfitting problem.

2.3.5 Algorithm selection?

Here we present the algorithms we will use.
TODO: CHECK HOW HE PRESENTS THE ALGORITHMS

Naive Bayes

The Gaussian Naive Bayes name comes from the Bayes' theorem. This theorem "describes the probability of an event, based on prior knowledge of conditions be related of conditions to the event" [51]. So basically this machine learning algorithm bases its prediction on previous experience. It is a probability algorithm, which means that it tells us not only the target but also the degree of certainty, which is one of its advantages.

K-Nearest Neighbor

kNN (k-Nearest Neighbor) is one of the simplest supervised machine learning algorithms. The idea of kNN is that it will classify the new object based on its k nearest neighbors, where k is a predefined positive integer [16]. kNN algorithm is comprehensively described in the following steps: - Step 1: Determine the parameter value k. - Step 2: Calculate the distance between the new object that needs to be classified with all objects in the training dataset. - Step 3: Arrange computed distances in the ascending order and identify k nearest neighbors with the new object. - Step 4: Take all the labels of the k neighbors selected above. - Step 5: Based on the k labels that have been taken, the label that holds the majority will be assigned to the new object. The model created by the k-Nearest Neighbors (k-NN) algorithm is, in fact, the entire training dataset. To make the predictions, the algorithm looks for the k nearest neighbors that are similar to the new instance we want to predict and summarizes instances found in the model [13]. The k-NN algorithm works well for small datasets but tends to struggle with high dimensional data and reach the “Curse of Dimensionality“ that we talked about in chapter 5.

Decision Trees

3.1.2.2. Decision tree Decision tree is a prediction model that is a mapping from observations of a thing, or a phenomenon to the conclusions about the objective value of things, or phenomena. Decision tree creates models that allow the classification of an object by creating a set of decision rules. These rules are extracted based on the set of characteristics of the training data. In a decision tree, leaves represent classes and each child node in the tree and its branches represent a combination of features that lead to classification. Thus, classifying an object will begin with checking the value of the root node, and then continuing downward under the tree branches corresponding to those values. This process is performed repeatedly for each node until it cannot go any further and touch the leaf node. For the best model, the decision to select the root node and sub-node while building the decision tree is based on the Information Gain (IG) and Gini Impurity measurements [16]. The decision tree algorithm used for experiments in this paper is C4.5.

Decision tree

The decision tree algorithm takes its name from the way it creates a tree-like model of decisions. Each internal node corresponds to a condition while each leaf is the decision (e.g. "Botnet" or "Normal" traffic). The main advantages of decision trees are that they are easy to understand as we can visualize trees. The usage of the model for predictions is logarithmic, thus cheap in term of performance. One of the drawbacks is that the algorithm could create models that do not generalize the dataset well, resulting in overfitting. It is recommended to balance the dataset before the training as not doing so could result in trees being biased in the case of a class dominating another [18].

Support Vector Machine

Logistic Regression

Logistic regression is a very basic algorithm that can be used when the target is categorical (e.g. "botnet" or "normal" traffic). The model created bases itself on a threshold to predict unknown data. The main advantage of Logistic Regression is the informative output it gives us as it explicitly expresses the relationship between the feature and the target or, in other words, which direction it takes (e.g. "botnet" or "normal" traffic). The drawback resides in the quantity of data needed to obtain stable and meaningful results [22].

XGBoost

XGBoost stands for eXtreme Gradient Boosting. It is an open-source implementation of gradient boosted decision trees which have been created for performance reasons. The boosting technique is based on ensemble where we use new models to correct previously trained ones. The models are added until no further improvements are possible [11]. Gradient boosting comes from the gradient descent algorithm it uses to obtain minimal losses when adding new models. XGBoost is considered to be the fastest implementation of gradient boosting [11] and is very popular on the Kaggle competition platform. Despite it being an algorithm that gives good results, we have to take care of overfitting when using it, which is its main drawback. However, the algorithm provides multiple parameters that allow to tune it correctly.

AdaBoost

AdaBoost, which is a short name for "Adaptive Boosting", is another ensemble learning algorithm that combines multiple weak classifiers in order to conceive a stronger one. It was the first algorithm created for binary classification. We could use any machine learning algorithm to boost it with AdaBoost but most frequently it is a Decision tree classifier with one level that is used [12]. The main drawback of AdaBoost is that it is sensitive to noise and outliers but it can be less susceptible to overfitting than other algorithms. Finally, it is still an algorithm that performs well on diverse machine learning tasks

Random forest

3.1.2.3. Random forest Random forest (RF) is a member of the chain of decision tree algorithms. The idea of this algorithm is to create some decision trees. These decision trees will run and give independent results. The answer predicted by the largest number of decisive trees will be chosen by the random forest [16]. To ensure that the decision trees are not the same, random forest randomly selects a subset of the characteristics of each node. The remaining parameters are used in the random forest as those in the decision trees. Future Internet 2018, 10, 43 5 of 11

Gaussian Naive Bayes

3.1.2.4. Naïve Bayes Naive Bayes is a conditional probability model based on the Bayes theorem [16]. In the classification problem, the data includes: D is the set of training data that has been vectorized as x = (x1, x2, ..., xn), Ci is subclass i, with i = 1, 2, ..., m. Assume that all properties are conditionally independent of each pair together. According to Bayes theorem, we have: P(Ci | X) = P(X | Ci)P(Ci)P(X)(1) Based on the conditionally independent characteristic, we have: P(X | Ci) = n k=1 P(xk | Ci) (2) where, P(Ci | X) is the probability of class i given sample X, P(Ci) is the probability of class i and P(xk | Ci) is the probability of the kth property that has the value of xk given X in class i.

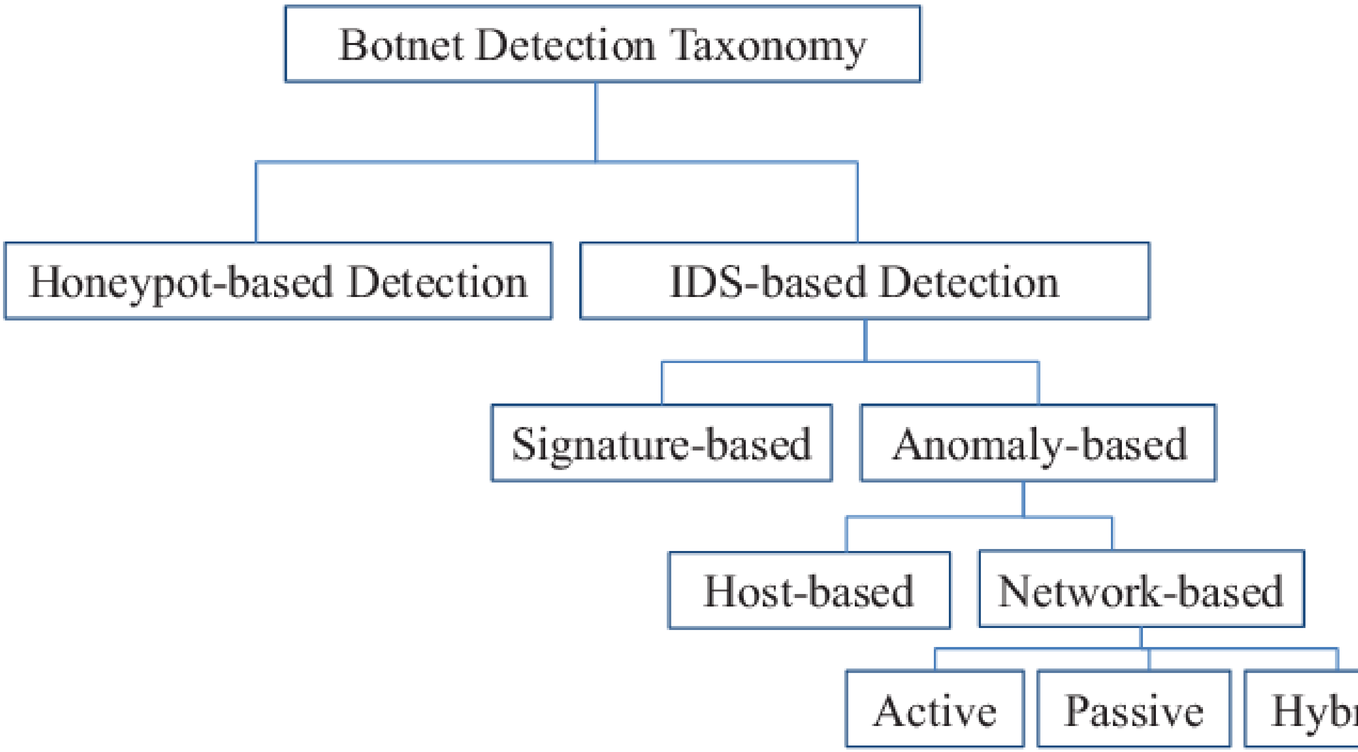
Neural Networks

In theory, neural networks are inspired by the brain and try to reproduce the way humans learn. In practice, they work very differently: there is an input and an output layer with some hidden layers in between. The hidden layers transform the input and gives it to the output layer. To train the model, neural networks use a technique called "back-propagation" to adjust the hidden layers when the output is not the one desired. The main drawback of neural networks is the black box model generated by this algorithm. In fact, it is very difficult to interpret the result and to identify the relationship between an input and its target.

2.4 Classification of botnet research and detection

In this section, we are going to present the current state of the botnet detection taxonomy, show the weaknesses and strengths of each method and end with the current state of the art of the research on passive traffic analysis for botnet detection which is the method this thesis is dedicated to.

2.4.1 Taxonomy of Botnets



Detection systems In a recent survey[12] of the state of the art regarding Botnet detection based on DNS traffic analysis[**survey**], they present a taxonomy of botnet detection techniques. As we can see on the figure (botnet_detection_taxonomy)there are 2 type of systems used for detection: Honeypots and IDS. Honeypots aim at creating an environment specially forged to attract malicious traffic and extract information out of the behavior of the malicious actor on the host. IDS aim at analyzing the network and alert the security analysts of any malicious/suspicious traffic. IDS are 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 was a signature-based detection in their Intrusion Detection Systems (IDS)[61]. However, these signatures have shown to be ineffective against bots that are constantly getting updated with new code and evasion techniques. Furthermore, they are ineffective against any new type of emerging botnet. Signature-based detection is great for known botnets but remains weak to any new ones[68]. The recent introduction of other methods have been mostly motivated by the goal to solve this issue.

Where are botnets detected? In the anomaly detection technique, we have different locations that researchers have found relevant data. Directly on the host affected and throughout the network the host belongs to. 3 types of behaviors observed from the bots[69]:

- **network based behavior**: observable network traffic between botmaster and bots, it is used to detect individual bots and their CnC server.
- **host based behavior**: observable activity on the hosts infected by the botnets.
- **global correlated behavior**: global behavior characteristics, structure will be similar to current structures; same for all mechanisms

What are the types of approaches for detection? They are classified into 2 categories, passive and active detection[70]. **Passive detection** consists 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 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.

This taxonomy applies to all botnets detection using different protocols. Let us now analyze the particularities the passive traffic analysis of the DNS protocol approach.

The interest to study this part of the taxonomy comes from the progress it has shown and the large amount of possibilities around those techniques.

2.4.2 Passive DNS detection Techniques

What are the different passive detection methods? With the logs or packets captured, researchers have proposed different approaches to utilize them, most of the techniques involve a statistical analysis or the use of machine learning algorithms others have opted for detection through visual representations.

The reason we have decided to focus on the machine learning approach as mentioned in the introduction is to develop our machine learning experience and methodology but also because regarding botnet detection it has shown to be the most effective method, achieving better scores then the other techniques in the taxonomy.
In the machine learning realm, these are some of the techniques used: clustering, tree classification, neural networks, entropy, statistical,..

As explained in the abuses of the DNS protocol, a new type of detection techniques have appeared that aim at finding bots evading the more usual detection techniques.

2.5 Related work

Now that we know the detection techniques used against botnets for different topologies and evading techniques.

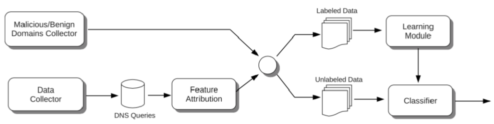
Since the end goal is to find the best features for an combined solution, we have structured the current state of research as follows: First, we will present the combined experiments. These experiments don't focus on a single type of detection mechanism but aggregate the different detection techniques used for 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.

2.5.1 Presentation of the Exposure and the Local botnet experiments

The first step in the research was to find the models that would be the current baseline for all-in solutions: because it is considered as the best all-in solution paper we could find, we are going to compare and extend the work of the EXPOSURE team.

Our first objective during our research was to

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



Time-based When we analyse many requests to a particular domain over time, patterns indicative of malicious behaviour may emerge.
These were supposed to be the features with the most weight, unfortunately due to lack of the same caliber of capture available to the authors of Exposure, we could not test out the 4 features related to time. Either because the datasets are compositions of smaller datasets, or because the timestamps are too short.

DNS answer based Here are some domain-flux features: A domain name can map to multiple IP addresses. In such cases,the DNS server cycles through the different IP addresses in a round robin fashion and returns a different IP mapping each time. 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.

- the number of different IP addresses that are resolved for a given domain during the experiment window
- the number of different countries that these IP addresses are located in
- the reverse DNS query results of the returned IP addresses
- the number of distinct domains that share the IP addresses that resolve to the given domain (false positive can be reduced with google reverse DNS which will have hosting providers in top answers)

TTL value based Low TTL and Round-Robin DNS:

- high availability (Content Delivery Networks (CDNs))
- botnets using this, makes them resistant to DNS Blacklists(DNSBL) and take downs. Often using Fast-Flux Service Networks (FFSN).

Because FFSN are usually detectable because of low TTL and growing list of distinct IP addresses for a domain, it explains the purpose of the TTL features.

Domain name based Finally 2 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).
This can be noticed with 2 simple features:

- ratio numerical chars to length of domain name
- length of the longest meaningful substring to length of domain name

Heuer et al.[71] did a case study on local infections related to botnets. They propose a very large set of 24 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.
TODO get access to the paper!!! (visuals aren't enough)

features:

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

TODO: add missing ones

2.5.2 Weaknesses and possible improvements

TODO: rework this section. These features are very specific to each mechanism and only work on the botnets that use them. That is why in the second part, we presented solutions with combined approaches. We want to experiment if a combined solution could provide better results and have other advantages such being more effective against new families of botnets. This type of project already exist such as the project Exposure detailed above. 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. Furthermore, in their study they showed that these features were the strongest features of the set. This means that without the access to these features their experiment doesn't show the same level of success. What we aim to do in our experiment is to provide smaller environments with similar detection capabilities as Exposure but with the features available in smaller environments.

TODO: weaknesses and improvements found in the other combined approaches.

2.5.3 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. [72] 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. [73] 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. [**phoenix**] 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. [74] 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. [75], in this paper they propose a different system called 'Notos' based on dynamic reputation 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. [76] 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. [77], 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.

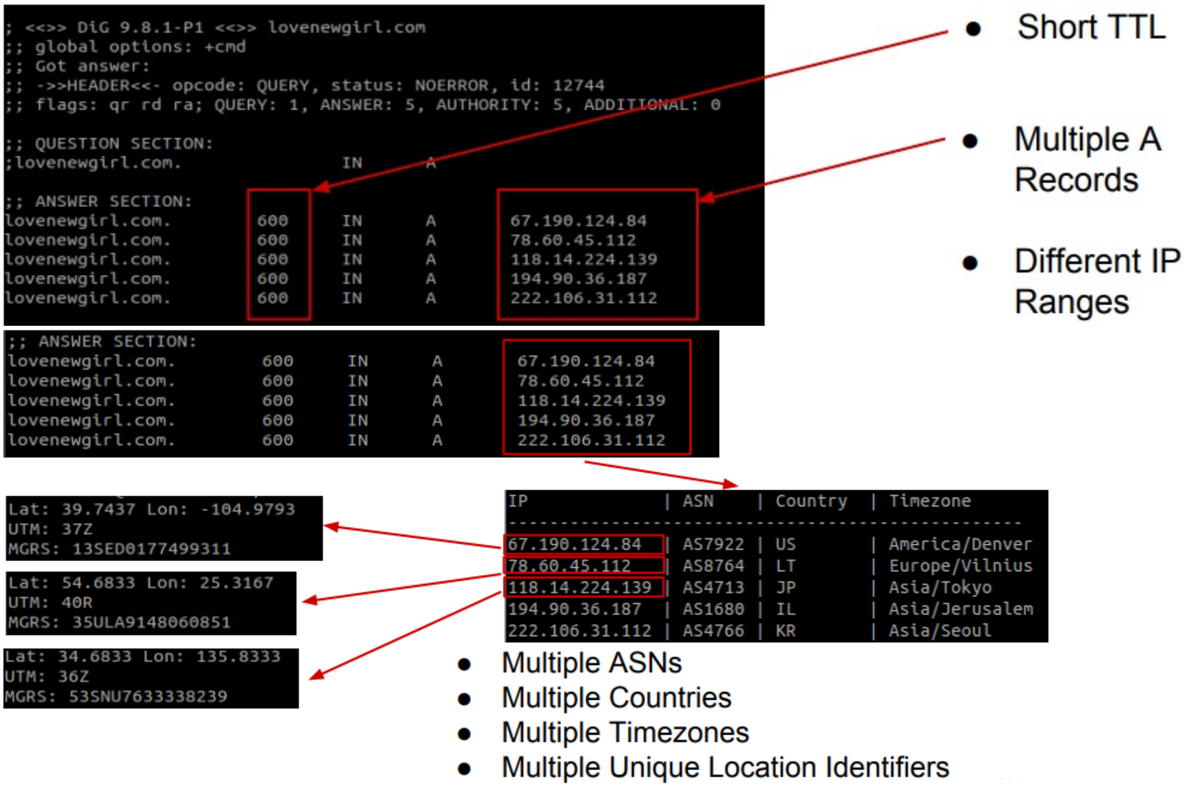
A similar idea was proposed by Antonakakis et al. [78] creating Pleiades, their second big botnet project after the Notos experiment. First, following the same machine learning idea of Notos but applied to the DGA algorithms, meaning they applied the X-means clustering algorithm to DGA algorithms. Secondly, 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.[17], Nazario et al.[53], Perdisci et al.[79], Holz et al.[80] and Stalmans et al.[81] 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. [53] showed visually how the FF traffic features present themselves.



Perdisci et al. [79] aimed at improving the Honeynet's features [17]. 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 [82] 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. [80] 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. [83] 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.

TODO: Perdisci et al. [84]

DNS tunneling

Another evasion technique of botnets that exploits 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 [85], 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 [86], 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..

TODO: This last paper [**dns-tun**]

Domain shadowing

TODO: Liu et al. [87] are the first to propose an approach against this new trend.

2.5.4 Value summary

2.6 Our botnet detection contribution

In the first part of the related works, we presented Bilge et al. [88] where they propose a large-scale system that uses passive DNS monitoring of 15 features to detect malicious activity. The interesting parts of this paper were some of the unusual features proposed among the 4 categories of features presented and their pipeline for the project which we used for inspiration. What we look add to this paper is the low-scale part that is missing, some of the features are only available to ISPs and the idea is to scale it to any company with DNS monitoring capabilities.

The second part presented very interesting studies and projects that allowed the detection of the different evasion mechanisms using the DNS protocol. They allowed us to extract valuable features and techniques for each mechanism.

TODO: Explain our possible contribution: "lower FPR, stronger features, better applicability of features, different modules"

Chapter 3

Model creation

Experiment Dataset Pre-Processing = feature extraction (I LIKE HOW HE EXPLAIN BI-TRI GRAMS)

3.1 Pipeline of the research

3.1.1 Environment

3.1.2 Datasets

TODO: reformat the dataset section.

Since some of the implementations that we are going to compare have a blacklist step, we have done a large research on the blacklists available. At first we only focused on botnets domains and IPs, but then realized that bots can query their C2 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 C2 to domains linked to suspicious phishing/adware campaigns. One of the websites that really helped in this process is https://firebog.net/. All these lists have been combined into a big unique blacklist used in some of the approaches presented here.

The first idea was to use basics lists with an additional check with the virusTotal API but it is a very long process et the amount of lookups is limited therefore not a viable solution. Instead we researched the blacklists used by virusTotal and other DNSBL providers and have a very long static list instead.

Let us talk about the datasets used to train the different classifiers and algorithms used in the thesis.

This is going to be very dependent on the approach, for unsupervised algorithms, we are going to train them with clean traffic and see its reaction to malicious traffic. For classification algorithms we are going to go for model training for manual tagging of traffic that is either malicious or clean. The idea will be to test the algorithms on parts of the sets but also try them on completely different traffic and compare the results.

Depending on the algorithms the datasets aren't the same, for supervised approaches, there is a need of labelled data. For that we have searched for unique DNS traffic coming from Botnets combined with clean generic traffic. We create different combinations of traffic to improve our understanding of what works in what scenarios.

The datasets used in this thesis come from different sources that try to provide labelled 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 labelled data from a lot of different sources and protocols. The one they provided me is focused on different types of botnets. The ISOT Http Botnet dataset consists of 2 different datasets: malicious DNS traffic generated by multiple botnets and benign traffic generated by generic software.

The ISOT Botnet dataset which consists in a large dataset regrouping 9 malicious botnets traffic in a controlled environment. And finally the CTU scenarios provided by the malware capture facility project.

3.2 Purpose of datasets

The idea of having different datasets allows to understand if some techniques work against a large variety of evasion techniques or on specific ones. This also can be a way to propose complete feature set 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.

3.3 Presentation of the datasets

3.3.1 CIC

This is a dataset maintained by the Canadian Institute for Cybersecurity. The have compiled malicious traffic from different sources and for most of them the traces are labelled 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 labelled dataset, the DNS traffic was low.

3.3.2 CTU

This is a set of datasets provided by the malware capture facility project[**CTU**]. 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.
The scenarios that interest us are: the 5th and the 13th that include Fast-Flux labelled traffic. It also provides hundreds of additional datasets but that are not as well documents and presented which makes the research for a particular dataset more complicated.

3.3.3 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 snif all the traffic. This dataset provides us with 3 sets of traces, malicious, bening and a mix[**ISOT**].

3.3.4 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 labelling 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 labelled 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.

Information regarding the datasets

Sources
Content
Labelling

balancing the datasets

3.4 Assessment model (for features and models)

3.4.1 Machine algorithms

Supervised

List of algorithms (to be decided)

Unsupervised for feature extraction

List of algorithms (to be decided)

3.4.2 Training and testing model

[72] Experiment 1 – 10-fold CV: 10-fold CV has become the standard method in practical terms 27. The dataset is divided into 10 subsets; for each of these 10 subsets, a model is built using the remaining nine sets, and this model is evaluated on the 10th subset. Store the performance of the model, and repeat this process for all remaining subsets. In the end, we have 10 performance measures, all obtained by testing a model on data not used for its construction. The 10-fold CV estimate is the average of these 10 measures. Experiment 2 – percentage split: To ensure unbiased results in training phase, the dataset is divided into two portions. The first is 75

3.4.3 Results metrics

List of metrics used to analyse the models

Chapter 4

All-in solution process

4.1 Features extraction and features analysis

4.1.1 Domain-flux Features

[74] They propose these features to detect DGAs:

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 trigram entropy
SLD trigram conditional probability

4.1.2 Fast-flux features

<table>
<tr><td rowspan="5">[17] [79] [80] [81]</td><td>Numerous unique A records for a domain</td></tr>
<tr><td>Numerous unique NS records for a domain</td></tr>
<tr><td>Different Autonomous Systems (ASN) for the IPs of linked to the same domain</td></tr>
<tr><td>Different countries[74] They propose these features to detect DGAs:
<table>
<tr><td>length of the domain name excluding TLD (top level domain)</td><td rowspan="6">for the IPs of linked to</td></tr>
<tr><td>Number of vowels in the Second Level Domain (SLD)</td></tr>
<tr><td>Number of consonants in the SLD</td></tr>
<tr><td>Number of digits in the SLD</td></tr>
<tr><td>SLD trigram entropy</td></tr>
<tr><td>SLD trigram conditional probability</td></tr>
</table></td></tr>
<tr><td>Short Time-To-Live (TTL)</td></tr>
</table>

[53] Some features for FF detection: nb of A records returned: 1-3 normal, 5 or more ff nb of NS: normal -> small, ff -> several NS several A records for the NS AS: small nb of A from 1 AS -> normal, located in different AS -> ff Hardware and IP: range of IP is diverged -> ff No physical agent -> ff no guarantee uptime -> ff

[80] **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 an hyperplane that separates benign from malicious fast flux where $x = (n_A, n_{ASN}, n_{NS})$ (unique A records, ASN and SN records) and the plane is defined as follows

From $F(x)$ they induce a metric $f(x) = w^T x$ with w the weight of the vector and b a bias. $f(x) > b$ would mean x is a FFSN. By empirically testing this on a labelled dataset they determined the value of w and b. $w = (1.32, 18.54, 0)$ and $b = 142.38$. We can notice that n_{NS} does not have any impact.

[82]	Number of resolved IPs
	Number of domains (in a cluster = domains with similar IPs)
	Avg. TTL per domain in a cluster
	Network prefix diversity = ratio between the number of distinct /16 network prefixes and the total
	number of IPs (measures the scattering)
	Number of distinct domain names that resolved to at least one of the IP addresses in the considered cluster
	IP Growth Ratio. This represents the average number of new IP addresses discovered per each DNS response related to any domain

Then the active ones:

Autonomous System (AS) diversity (ratio between the number of distinct ASs where the IPs of a cluster reside and the total number of resolved IPs. Same for the following diversities)
BGP prefix diversity
Organization diversity
Country Code diversity
Dynamic IP ratio (ratio of dynamic vs total IPs using keywords in reverse DNS lookups)
Average Uptime Index (average uptime for the IPs in a cluster, Uptime tested through probing)

	Type	Features
[ff5]	DNS Answer-based	Number of unique A records Number of NS records DNS packet size TC (Tnmcated) Flag is set
	Domain name-based	Edit Distance KL (Kullback-Leibler) Divergence (unigrams and bigrams) Jaccard Index (unigrams and bigrams)
	Spatial-based	Time Zone Entropy of A records Time Zone Entropy of NS records Minimal service distances (mean and standard deviation)
	Network-based	Number of distinct autonomous systems Number of distinct networks
	Timing-based	Round Trip Time of DNS request Network delay (mean and standard deviation) Processing delay (mean and standard deviation) Document fetch delay (mean and standard deviation)

4.1.3 DNS tunnelling features

4.1.4 All-in

[88]

Time-based When we analyse many requests to a particular domain over time, patterns indicative of malicious behaviour may emerge.

These were supposed to be the features with the most weight, unfortunately due to lack of the same caliber of capture available to the authors of Exposure, we could not test out the 4 features related to time. Either because the datasets are compositions of smaller datasets, or because the timestamps are too short.

DNS answer based Here are some domain-flux features: A domain name can map to multiple IP addresses. In such cases,the DNS server cycles through the different IP addresses in a round robin fashion and returns a different IP mapping each time. 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.

- the number of different IP addresses that are resolved for a given domain during the experiment window
- the number of different countries that these IP addresses are located in
- the reverse DNS query results of the returned IP addresses
- the number of distinct domains that share the IP addresses that resolve to the given domain (false positive can be reduced with google reverse DNS which will have hosting providers in top answers)

TTL value based Low TTL and Round-Robin DNS:

- high availability (Content Delivery Networks (CDNs))
- botnets using this, makes them resistant to DNS Blacklists(DNSBL) and take downs. Often using Fast-Flux Service Networks (FFSN).

Because FFSN are usually detectable because of low TTL and growing list of distinct IP addresses for a domain, it explains the purpose of the TTL features.

Domain name based Finally 2 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).
This can be noticed with 2 simple features:

- ratio numerical chars to length of domain name
- length of the longest meaningful substring to length of domain name

4.1.5 Features analysis

Distribution of the features for the different datasets,Correlation between features

4.1.6 Features discussion

4.2 Features selection for All-in solution

4.2.1 Feature optimization cycle (adding/removing features, changing or modifying thresholds)

4.3 Comparison of the results

Chapter 5

Sets run and result discussion

5.0.1 Results analysis

5.1 Current All-in solution assessment

5.1.1 Exposure

5.1.2 Recognizing Time-efficiently Local Botnet Infections (Heuer et Al.)

5.1.3 More to come

Chapter 6

Conclusion

6.1 This will depend on results

6.2 Advantages of solution

6.3 Shortcomings

6.4 Improvements propositions

6.5 Welcome and Thank You

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or

`\hypersetup{citecolor=green}`, or

`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:

`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.

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] T. Seals. *Newsmaker Interview: Troy Mursch on Top Botnet Trends*. Dec. 2018. URL: https://threatpost.com/troy-mursch-botnet-trends/140010/.

[3] M. Korolov. *What is a botnet? And why they aren't going away anytime soon*. Feb. 2019. URL: https://www.csoonline.com/article/3240364/what-is-a-botnet-and-why-they-arent-going-away-anytime-soon.html.

[4] R. Nigam. *Unit 42 Finds New Mirai and Gafgyt IoT/Linux Botnet Campaigns*. July 2018. URL: https://unit42.paloaltonetworks.com/unit42-finds-new-mirai-gafgyt-iotlinux-botnet-campaigns/.

[5] *Botnets remain a persistent cyberthreat*. 2018. URL: http://news.centurylink.com/2018-04-17-Botnets-remain-a-persistent-cyberthreat.

[6] M. Grooten. *VB2018 preview: IoT botnets*. Sept. 2018. URL: https://www.virusbulletin.com/blog/2018/09/vb2018-preview-iot-botnets/.

[7] E. Targett. *Top 10 Malware Families in 2018: Botnet Analysis*. Sept. 2018. URL: https://www.cbronline.com/news/kaspersky-botnet-activity.

[8] *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.

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

[10] J. Leyden. *OMG, that's downright Wicked: Botnet authors twist corpse of Mirai into new threats*. June 2018. URL: https://www.theregister.co.uk/2018/06/01/mirai_respun_in_new_botnets/.

[11] B. Weymes. "Recognising Botnets in Organisations". In: *Department of Computer Science, Radboud University* volume number.issue number (Aug. 2012).

[12] A. Manasrah M. M. Khadhum K. Alieyan A. ALmomani. "A survey of botnet detection based on DNS". In: *Neural Computing and Applications* (Dec. 2015).

[13] 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.

[14] 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.

[15] 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).

[16] 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.

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

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

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

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

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.

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

[34] 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).

[35] 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.

[36] 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.

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

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

[39] W. Lee D. Dagon C. Zou. "Modeling botnet propagation using time zones". In: *Proceedings of Network and Distributed System Security Symposium* (Feb. 2007), 111–123.

[40] M. J. Jacobson R. Vogt J. Aycock. "Army of botnets". In: *Journal Name* volume number.issue number (year).

[41] W. H. Sanders E. Van Ruitenbeek. "Modeling peer-to-peer botnets". In: *Fifth International Conference on Quantitative Evaluation of Systems* (Sept. 2008), 307–316.

[42] D. Daly D. Deavours S. Derisavi J. M. Doyle W. H. Sanders P. Webster G. Clark T. Courtney. "The Möbius Modeling Tool". In: *Proceedings of the 9th International Workshop on Petri Nets and Performance Models* (Apr. 2011).

[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] 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/.

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

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

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

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

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

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

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

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

[53] 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.

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

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

[56] M. Thompson. *Moving Target Defense and DNS Fast Flux*. Feb. 2018. URL: https://blog.domaintools.com/2018/02/moving-target-defense-and-dns-fast-flux/.

[57] 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.

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

[59] 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/.

[60] 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).

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

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

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

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

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

[66] *10-fold Crossvalidation*. URL: https://www.openml.org/a/estimation-procedures/1.

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

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

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

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

[71] 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.

[72] 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.

[73] 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).

[74] 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.

[75] 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.

[76] 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).

[77] 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.

[78] 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).

[79] 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).

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

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

[82] 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.

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

[84] 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.

[85] 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: ().

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

[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] 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).