

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 infrastruc- ture defence in relation to emer- ging threats**

Carl Slatter

2020/ 2021

General Computing Hybrid 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  $\langle \text{len} \rangle$   
(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

## Historic Attack & Defence

### 1.1 The Importance Of Computing History

### 1.2 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

### 1.3 Loveletter

### 1.4 SQLSlammer





# Chapter 2

## Modern Security Landscape

### 2.1 The Cat & Mouse Game

Technology is always changing, often to the needs of the growing world. Changes 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 disterous 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 retrated. 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 responce can essas-  
ibated by corporate culture that put strain on the process. Under-  
funded, unmotivated and untrained workers will struggle to work to  
their capability and as a result, security can suffer. Another poten-

tial 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 given to defence policy, strategy and potential response in order to stay ahead of the cat and mouse game that is cyber-security.

## 2.2 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 practice as a response. Scenarios are planned out ahead of time in order of likelihood, 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
- Acquisition of audit trails
- Checking of open connections / malware

### 2.2.1 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.

## **2.3 Emotet**

## **2.4 WannaCry**

## **2.5 NotPetya**

## **2.6 Mimikatz**



# Chapter 3

## Defence Technologies

### 3.1 Variation of Defence Systems

#### 3.1.1 Network Monitoring Capability

Monitoring is rather important. Particually 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 congition 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 implementated 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 benefital is that it has the capacity of content. A flood of HTTPs traffic may look like a DDoS attack, but may actually be a holiday like black friday in which you may expect hightened traffic.

#### 3.1.2 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 independant 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. We use firewall rules to block everything else that is on the same device, such as my dns server. Additonal firewall rules could be used internally to stop priv esc but unlikely in a home lan setup. I only allow local devices to even access the public IP other than for the VPN so it is a whitelist, very strong. 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 anyway.

## 3.2 Threat Definition

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

### 3.2.1 Classification of Threats

Malicious software, often called 'Malware' is an ever growing problem for system administrators.

**Viruses**

**Worms**

**Trojans**

**Rootkits**

**Dossers**

**Adware**

**Ransomware/Cryptors**

**Spyware**

**Fileless/Packetted**

## **3.3 Anomaly Detection**

### **3.3.1 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 dataset 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.

### 3.3.2 Signature & Pattern Matching

An interesting question could be made. What is considered a threat to a computer? The simplest answer is "something that is pre-defined". 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 datasets. 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.

### 3.3.3 Behavioural Dynamic Analysis

Polymorphic encryption and malware versioning has historcly 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. Ultimatly malware across versions aims to do the same task, albiet 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 siganture matching.

### 3.3.4 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 preferntial 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.



# Chapter 4

## Entry Vectors & Profiling

### 4.1 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.

#### 4.1.1 NMAP

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

Host Discovery Scan (Subnet Scoped):

```
nmap -sn 192.168.1.1-254
```

Noisy Host Probing Scan:

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

## 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.

### 4.1.2 Firewalking

### 4.1.3 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 binded to depends on

the architecture and software security level, but is dangerous because it means that the tools on the system are fairly irrelevant.

#### 4.1.4 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 essentially a unique record for a given vulnerability, often used in the cybersecurity industry for disclosures and reference. Each CVE is registered in the database (can be found at [exploit.db](https://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 an 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.

## 4.2 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. (?)

### 4.2.1 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 sanitization, with checks of email header tampering, some form of verification and file/link whitelists.

#### 4.2.2 USB Dropping

USB devices carry data in a convinient 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 usb's are usually either dropped outside randomly and picked up or given in seemingly good faith. Another interesitng 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.

Autorun

## 4.3 Spoofing

### 4.3.1 ARP/MAC Spoofing

### 4.3.2 DNS Spoofing

### 4.3.3 IP Spoofing



# Chapter 5

## Malware Mechanisms

### 5.1 Command & Control

Malware traditionally is static, meaning that it executes a given task and then finishes. Malware that can be maleable 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.

### 5.1.1 Side/Control Channels

#### Bind Reverse Shells

A Bind/Remote shell is you connecting from your machine to the shell. This is more of a backdoor. This is usually blocked by firewalls and can be ruined by change of ports, additionally DHCP and NAT cause IPs to change and as a result you do not know the IP of the listener you have setup. A reverse shell is the shell connecting to a listening service (Netcat) on your machine.

Netcat lets us set up connections between a host and a listener. You can also connect to any listener that is not yours also, which is referred to as banner grabbing, simply using nc on a ip port can do this. A useful.

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

MSFVenom

Reverse shells can be over TLS

### 5.1.2 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 next version can.

The attack I am meaning is the Distributed Denial Of Service or DDOS attack. This 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 than 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. (?)

## 5.2 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 reaquisition 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 useless 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 buisness. The idea being that if an attacker group has 24/7 live chat and customer serivce, 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 ransomw, the key often is not transmitted and it is more liked to corporate rivalry or hacktivism.



# Chapter 6

## Malware Mechanisms - Obfuscation

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.

### 6.1 Signature & Behavior Evasion

Antivirus systems rely on signatures. What is a way to fingerprint a piece of data? Hashes; they are unique to the dataset, assuming we are not using a broken hash algorithm. MD5 is a broken hash algorithm because it can be manipulated in a way that more than one data set gives the same hash. A MD5 collision In the same way, malware can alter it's own composition in a way that still works, but gives a different signature. If the AV does not have this new signature, it cannot be stopped assuming a blacklist approach. There are malware out there will add randomness each time, purely for this purpose. AV will only ever catch broad attacks, or default payloads due to this fact. AV will also sometimes scan a file, beyond a simple hash check for strings only normally used in malware, dynamic analysis. This process can be made harder be by splitting up regonizable strings. The new result becomes a new detection vector, hense the cat and mouse game.

## 6.2 Encoding & Compression

Base64 is a popular option. Used for compression, even further than hex. Can be decoded, is not secret but can help. Youtube URLs provide plenty of unique and random spaces that are hard to brute force unlisted videos.

- Can be used to hide malware, to provide it in a compressed way -  
Uses less bandwidth - Other compression includes gzip and deflate.  
alongside ROT13

```
cat /etc/passwd > /tmp/secret.txt ----->  
↪ Y2F0IC9ldGMvcGFzc3dkID4gL3RtcC9zZWNyZXQudHh0
```

### 6.2.1 Shellcode

Simply programmatic code that is compressed down to it's essentials, and then encoded to be unreadable by anything until it's decoded. Can be used to bypass IDS/IPS and antivirus. This is an encoded string, it represents a given value without a transformative key, that is simply hidden to our understanding. Each part here represents a character, H is 72 for example (in both ASCII and unicode). This can also be represented in a hex format, . a and b being hex values that resolve to the number in decimal relating to the character.

```
\x48 for H. "Hello World!" ---->  
↪ "\x48\x65\x6c\x6c\x6f\x20\x57\x6f\x72\x6c\x64\x21".
```

Useful CyberChef Decoders:

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



- Magic

### 6.2.2 Steganography

### 6.2.3 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



# Chapter 7

## Malware Mechanisms - Exfiltration

### 7.1 Proxy Chains & VPNs

#### 7.1.1 ToR Routing

### 7.2 Protocol Payload Abuse

#### 7.2.1 ICMP

#### 7.2.2 DNS

### 7.3 TLS

### 7.4 Data Breaching



# Chapter 8

## Malware Mechanisms - Persistence

Important for malware to have continued access, even after a reboot. I will try a few of these with a reverse shell. We need triggers, these triggers should be fairly legit, with malformed payloads. These triggers should be automatic, or at least on something that would be done normally.

### 8.1 Windows Systems

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. We can prove that the criminal acted at a given time, on a given file etc... These are some useful areas. This is windows 10, but older ones are still out there, 7 upwards are very similar. We need to be able to recreate.

#### 8.1.1 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. //expand upon with real scripting.

**Startup Keys** Keys that point to folders, that can launch shortcuts and executables as the given user, often during login or reboot

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User
↳ Shell Folders //will default to carl in my case
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell
↳ Folders
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell
↳ Folders //for public
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User
↳ Shell Folders //for users

Windows Startup folder - Anything here auto execs on startup, even
↳ shortcuts. run ----> shell:startup

C:\Users\USERNAME\AppData\Roaming\Microsoft\Windows\Start
↳ Menu\Programs\Startup //seems to run as logged in user - waits
↳ 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.

```
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
```

```
//temp or user folders would be very sus! It's about looking for
```

```
→ anomalous locations.
```

```
\subparagraph{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.
```

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

```
→ Helper Objects
```

```
HKLM\SYSTEM\CurrentControlSet\Control\hivelist
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet002\Control\Session Manag
```

### 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 this, loads this on boot. This is the view of an
```

```
→ online sandbox analyzer
```

```
autocheck autochk * aHdqEPamx
```

### 8.1.2 Symbolic Locations

**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<sub>DLLs</sub>** Everytime 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 loadup from multiple initilized processes.

```
User (For then priv esc):
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\Policy\
//depending on architecture
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce

reg add
↪ "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
↪ /v Pentestlab /t REG_SZ /d "C:\Users\pentestlab\pentestlab.exe
reg add
↪ "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
↪ /v Pentestlab /t REG_SZ /d "C:\Users\pentestlab\pentestlab.exe
reg add
↪ "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
↪ /v Pentestlab /t REG_SZ /d "C:\Users\pentestlab\pentestlab.exe
reg add
↪ "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
↪ /v Pentestlab /t REG_SZ /d "C:\Users\pentestlab\pentestlab.exe
```

Run RunOnce Keys

## 8.2 Linux Systems

### 8.2.1 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.

### **8.2.2 Cron Manipulation**

### **8.2.3 SUID Manipulation**

### **8.2.4 User Account Creation**

## **8.3 Process Hijacking**



# Part III

## Synthesis



# Part IV

## Evaluation



# Part V

## Appendices





# Appendix A

## Terms of Reference

### A.1 Ethics Form

If you scan the Ethics form on one of the multifunction printers, you can get a pdf copy. This can then be included with the  $\text{\LaTeX}$  command

```
%\includegraphics{ethics.pdf}
```

### A.2 Risk Assessment Form

Likewise you can scan and include the Risk Assessment Form

```
%\includegraphics{risk-assesment.pdf}
```