Exploiting Windows Systems using Fileless Malware

Athul C K

MCF20004

July 23, 2022

Abstract

Fileless malware emerged in 2017 as a mainstream type of attack where the malware resides prominently in the RAM memory, thus leaving very few artifacts to be found by anti-virus software or by a forensic analyst. Fileless malwares are sneakier and stealthier since traditional detection strategies like file-based whitelisting, signature detection, hardware verification, pattern-analysis, time-stamping, etc., do not work against it. Such malware makes use of the already existing user environment and the tools present in it to launch the attack. One of the reasons attackers favor PowerShell-based malware is because it supports memory-based download and execution, which is perfect for a fileless malware attack. A malware that persists only in memory is often looked down upon since a simple system restart could flush the memory thus eliminating the malware from the system. Attackers have established several other ways to make such malware persist by storing harmless boilerplate code in the harddisks which is later used to only launch the malicious shellcode. The scope of this mini-project includes a detailed study of the lifecycle of a fileless malware, test implementation and analysis of the results thus produced. A live forensic analysis will also be done on the affected system to see whether there exist any forensic artifacts regarding the malware.

Contents

List of Figures

Chapter 1

INTRODUCTION

General Background

Malicious software (or Malware) is designed by malicious attackers whose intention is to bring down the system or to take over the system. Malware is untraceable as attackers use modern obfuscation techniques to build malware to beat the traditional signature-based malware detection systems. The most common way for these malwares to spread is via phishing. When the victim clicks on some genuine-looking link, the malware gets downloaded to the victim's machine. This is the simplest way to get into the victim's machine and the attacker takes control, gains information, and misuses the obtained information. Since the fileless malware runs on the victim's primary memory, live forensic analysis is also performed on the victim system to identify whether we can obtain any artifacts about the malware which can be useful in classifying that process as a malicious one. As technology progresses at an exponential rate, so do the number of cyberattacks. According to a study performed at the University of Maryland in 2007, there is a cyber attack performed every 39 seconds, affecting one in every three Americans every year. These attacks translate to more than 44 records stolen every second of every single day. This number has only increased since Cisco predicts that the number of devices connected to the Internet will exceed the global population by at least three folds by 2023, with an estimate of close to 30 billion connected devices. According to the University of North Georgia, only 38% of organizations worldwide claim they could handle a cyberattack on their organization. The best defense against cyber attacks is human intelligence and security awareness training. Black-hat hackers will always find a new zero-day vulnerability as long as technology keeps expanding and new features are added. It is estimated that around 94% of data breaches start with a phishing email sent to an employee of that company. There is bound to be an exploitable vulnerability lurking with so many different protocols and features implemented on the average network. Unfortunately, even the most advanced security implementations aren't always 100% effective at detecting and blocking threats. A current example of this is the controversial SolarWinds cyberattacks, in which the alleged perpetrator was able to gain access to the company's Orion software, which is what is used to push software updates to their clients. The attackers were able to inject their advanced malware into the software update allowing their malware to be installed completely undetected since it was embedded into a trusted software distributor. This cyber-attack is mentioned because this affected some of the largest organizations in the world, including Microsoft, the Cybersecurity and Infrastructure Security Agency (CISA), FireEye, Homeland Security, the U.S. State Commerce and Treasury, U.S. Energy Department, the National Nuclear Security Administration, and many more. These organizations have some of the absolute best cybersecurity implementations in the entire world, as well as FireEye, being one of the most renowned cybersecurity firms in the globe, and this attack still went undetected for over 8 months. Ever since the origin of malware, one thing that has remained constant is, someone had to create the code and develop the malware by putting significant time and effort into programming. In Early 2002 Microsoft released the framework .NET, which changed the software development industry and unintentionally revolutionized the malware industry to enter into a new phase, the fileless malware phase. The .NET framework made it easy for malware coders to spread malware and achieve the goals of the malware, which replaced the old methods of creating malware from scratch and working to stay ahead of the antivirus companies. While the development of malware on the .NET framework is easier, the popular host platform of attacking is by using PowerShell. PowerShell provides tremendous flexibility and power for all stages of an attack and since it evades most antivirus detection, where the modules used by the malware are whitelisted by the system administrators for legitimate use. PowerShell is tightly integrated into the Microsoft Windows environment and it is hard and impractical for many system administrators to turn it off. PowerShell scripts can be loaded into the memory of the operating system can execute commands without ever writing files to the hard drive. By dynamically loading the PowerShell scripts, the malware is able to attack the system without leaving any file-based evidence. This ability also provides the ability to hide from file-based antivirus protection.

Chapter 2

PROBLEM STATEMENT

To exploit windows systems and gain control over them using fileless malware. There exists ordinary malwares that can be easily detected by existing antivirus systems. Fileless malware is a new addition to the malware arena where it is very difficult to be detected and quarantined by most of the existing antivirus systems. Fileless malware has been gaining a lot of attention at industry events, private meetings, and online discussions. Indeed, according to Google Trends, people's interest in this term blipped in 2012 and 2014, began building up toward 2015 and spiked in 2017. PowerShell-hosted attacks surged by 432% in 2017 compared with the previous year and by 267% in the last three months of the year alone. According to the Ponemon Institute's "State of Endpoint Security Risk Report 2017," 29% of the attacks in 2017 were fileless. Ponemon predicts that this rate will increase to 35% in 2018. In fact, the Ponemon Institute claims that they are 10 times more likely to succeed than file-based attacks.

Chapter 3

LITERATURE SURVEY

The literature survey consists of 5 papers which focus on various aspects of the Fileless malware. The work done by M. Kaushik, M. Malik and B. Narwal [1], explains how malware can be disguised by binding it with an image using steganography. The study showed that OSINT virus detection platforms like VirusTotal showed a significant drop in its confidence level while scanning the steganographed malware, when compared to the actual malware. The limitations of the study is that they have only used free OSINT engines to detect the malware, and the literature does not really specify how exactly the steganography process is done. The takeaway from this paper is that the traditional steganography techniques still works in making the malware undetectable by the AntiVirus softwares.

The work done by B. N. Sanjay, D. C. Rakshith, R. B. Akash and D. V. V. Hegde [2], studies the fileless malwares quite extensively. It discusses the fileless malware attack cycle which is similar to the attack cycle for most malwares, but also states various methods and attack vectors for each stage in the life cycle. It discusses two main strategies which can be used to ensure that traditional security monitoring technologies will not inspect the files and activities done by the malware, and they are: Download to memory and execute: Downloading directly to memory and executing will not leave any files behind for antivirus software to scan and detect. Use trusted applications: Using applications that are already trusted by the victim system will allow the malware more freedom and power while evading suspicion of both user and anti-malware programs.

The study also classifies fileless malwares into two:

- RAM-resident Fileless Malware: which solely resides in RAM and runs as a background process.
- Script-Based Fileless Malware: which uses VB Script in Microsoft Office (macros),

PowerShell scripts to run the malware.

It also states many evasion techniques which are commonly used by malware writers like:

- Malicious Documents: Use of Office document files to hide malicious JavaScript or VB Scripts.
- Malicious Scripts: There are scripts that can directly run on Windows in the memory. The tools that attackers invoke to run these scripts include powershell.exe, cscript.exe, cmd.exe, mshta.exe, Invoke-NoShell, Invoke-Obfuscation and Invoke-DOSfuscation.
- Living Off-the-Land: regsvr32.exe, rundll32.exe, certutil.exe, schtasks.exe and wmic.exe will all help the malware to rely solely on these already present tools and not to package in it's own tools and utility programs for the execution of malicious purposes.
- Malicious Code In Memory: API calls often abused by malware for code injection include VirtualAllocEx and WriteProcessMemory, which allow one process to write code into another process.

Their work discusses some Fileless malware detection techniques, which include:

- Sandboxing: Windows Powershell programs should be run in a specialized sandbox and the APIs used should be carefully monitored.
- Execution emulation: Running the Powershell scripts in emulation is better to identify any malicious indications.
- Heuristics: Heuristics like a Powershell script running from a browser or a word document can be considered suspicious. Heuristics cannot make clear conclusions whether the suspected script is malicious or not.
- Yara: An open-source tool that can be used to design rules which will help in identifying malicious instructions and strings specific to a memory-run malware.

The paper by A. Johnson and R. J. Haddad [3] tries to create malware using the Metasploit framework. This paper exposes new customized shellcode exploitation of the Windows

64-bit operating system that can successfully evade almost all Antivirus software using signature-based detection algorithms. This is achieved by source code obfuscation of a typical Metasploit reverse shellcode exploitation, which renders its signature undetected by Antivirus software. A standard shell is a computer process that presents a command prompt interface that allows the user to control the machine (computer) using keyboard commands instead of using a GUI (Graphical User Interface). A reverse shell is a type of shell in which the target/victim machine communicates back to the attacker's machine and spawns a shell that has control over the target machine. This usually works by having what is called a "listener" server on the attacker machine that waits for incoming connections, such as the reverse shell calling back to initiate a connection. The main tool that is used in this project is the Metasploit-Framework. Msfvenom is a tool incorporated within the Metasploit framework that allows individuals to generate different types of shellcode exploits for various operating systems. Shellcode is usually a small piece of code that is written in hexadecimal or commonly referred to as machine code,that is used as a payload in the exploitation of a software vulnerability. The term shellcode is derived from its original purpose; it was the specific portion of an exploit used to spawn a root shell. The shellcode used in this projects pawns a staged meterpreter reverse shell that exploits theWindows 64-bit operating system. A meterpreter shell is what would be considered a more "powerful" shell in the respect that it has built-in functions that automate common post-exploitation jobs, such as escalating privileges and transporting data between the victim and attacker machines. Kali Linux was spun up on a virtual machine, and it is important that the virtual machine network connection is bridged to the Local Area Network (LAN) so that the attacking machine and the target machine could communicate with each other without having to tunnel the connection or set up port-forwarding on the gateway. This proposed approach was able to reduce its ability to be detected by the Antivirus software by 97% compared to a typical reverse shell program.

The work done by A. Afreen, M. Aslam and S. Ahmed [4], defines fileless malware attacks as something where attackers use techniques where malicious files are not written to the physical drive, they will use built-in legitimate tools such as PS or WMI to gain persistence and do network reconnaissance to know where they have landed in the network. The paper does point out some common and powerful tool used in fileless malware attacks as:

- .NET Framework: It comes preinstalled on all Windows OSs, has built-in functionality to dynamically load memory-only modules and thus proving to be a very handy tool in crafting malwares.
- Windows Management Instrumentation: WMI is useful in code execution, lateral movement and persistence of the malware. It can be used to execute malicious JavaScript/VB script payloads directly into the memory to evade traditional antivirus solutions which are based on signature detection. WMI allows threat actors to interact with the target system using wmic.exe since it is a pre-built in windows capability, therefore it is difficult for antimalware/antivirus solutions to detect and restrict access from adversaries.
- Powershell: It is probably the most powerful scripting language supported by Windows that integrates to almost every nook and corner of the OS. Payloads can be loaded into the memory of the OS using Powershell and can execute instructions without writing anything to the disk, without leaving any evidence on disk.

Fileless malware attack techniques include:

- Persistence Techniques: by storing malicious code in unusual locations associated with the operating system or common utilities, such as the WMI Store, SQL tables,Windows registry or tasks scheduling to inject malicious code into a system process, which helps threat actors to evade detection, as the activities would seem to be coming from legitimate processes.
- Memory Resident Malware: Memory resident malware loads the payload directly into the memory-making detection and sign of infections difficult to identify. Manual analysis of such type of attack could be done by using Volatility and for memory capture FTK Imager is a tool which can be used for this purpose. Once a memory dump is captured it can be analyzed in a controlled environment for analysis , Volatility Framework can be used for such kind of analysis.
- Windows Registry Malware: It includes JavaScript code added into the registry and is executed by a legitimate Windows application, a PS script which decodes the encoded shellcode and injects it to the legitimate windows process to execute, using a technique called Process Hollowing.

- Rootkits: Although they are not completely fileless, they often reside at a level below what antimalware solutions can detect and can successfully evade detection. Rootkits can access a computer remotely while staying undetected by security programs or administrator privileges.

- Process Hollowing/Injection Attacks: Process hollowing is a technique of hiding a process behind a legitimate process to hide in plain sight. An application creates an empty/suspended process swapping out the original code from the process, then the legitimate process is injected with an adversary malicious payload. Once the payload is loaded in memory, the process is then resumed with the entry point of the new payload which was injected.

- Reflective DLL Injection: Attackers insert malicious code into a legitimate process through DLL injection. With reflective DLL injection, attackers are able to copy an entire DLL into process memory, which avoids having that DLL reside on disk (making it hard to detect) and being registered with the process it's being injected into. A malicious payload can reflectively load portable executable without getting registered as module in the process and hence can perform actions without leaving forensics footprints.

- Dynamic Data Exchange (DDE) Attacks: DDE is a client-server protocol for interprocess communication between Microsoft office applications that allows different office applications e.g. Word, Excel to exchange data between themselves by using shared memory. DDE attacks are the next generation of macro-based attack as they continue to use Office documents as an entry point into the target victim. Traditional antimalware solutions will not detect these kinds of attacks and requests for command execution.

- Dual-use Tool Attacks: Due to major change in threat landscape, adversaries are exploiting legitimate or dual-use tools like PS, netsh, PsExec.exe, etc. to deliver memory only payload. These are much less suspicious since these tools are commonly used by users as well as malicious programmers.

Fileless malware detection techniques like behavioral analysis, logging, least privilege rule, content filtering, applying proper patches and updates and the common user awareness programs are also discussed in their work.

The paper by F. A. Garba, K. I. Kunya, S. A. Ibrahim, A. B. Isa, K. M. Muhammad and N. N. Wali [6], proposes an experiment which tests several antivirus evasion tools against popular antivirus softwares currently available in the market. The disadvantage is that the paper only proposes the experiment but has not conducted it. So, it lists out various antivirus evasion tools which can be useful, but the results have to be worked out by implementation.

Chapter 4

PROPOSED SYSTEM

The proposed system consists of a malware boilerplate module which can be easily used to spread the malware by using it as a payload in the phishing attacks. The good thing about this module is that it contains no exploits, shellcodes or anything harmful which an antivirus software can detect. Once it is run by the victim, it contacts the C&C server, which is run by the attacker, and downloads the actual shellcode which is malicious. This way, we can ensure that the shellcode never gets stored and it just runs directly in the memory. Execution of the shellcode will actually spawn a reverse meterpreter session on the attacker's machine, using which they can have complete access to the target system. The shellcode instructions never touch the disk and are instead downloaded and executed straight into the memory, thus reducing detection chances. Additionally, in order for the program to download the shellcode instructions from a remote server, the number of changes and features that will be implemented in the boilerplate module will drastically change the signature of the modified source code, which is the main goal of this proposed exploit.

Here, the boilerplate code is written in C++ using the .NET framework and exported as an executable, but this can also be done using a Powershell script or any other scripting language. Powershell scripts, Visual Basic Macros in Microsoft Office softwares are famous for how they can be used to launch malicious programs. So the attacker can also do what the boilerplate does using those attack vectors and still achieve the same result.

There is a way to make the boilerplate malware code look like a PNG or a JPG file. Note that this is not a proper steganographic obfuscation, but it will just make the executable looks like an image file. For this to work we need the victim machine to have the WinRAR application already installed, and it works better if the user has no antivirus software, or has the antivirus software disabled.

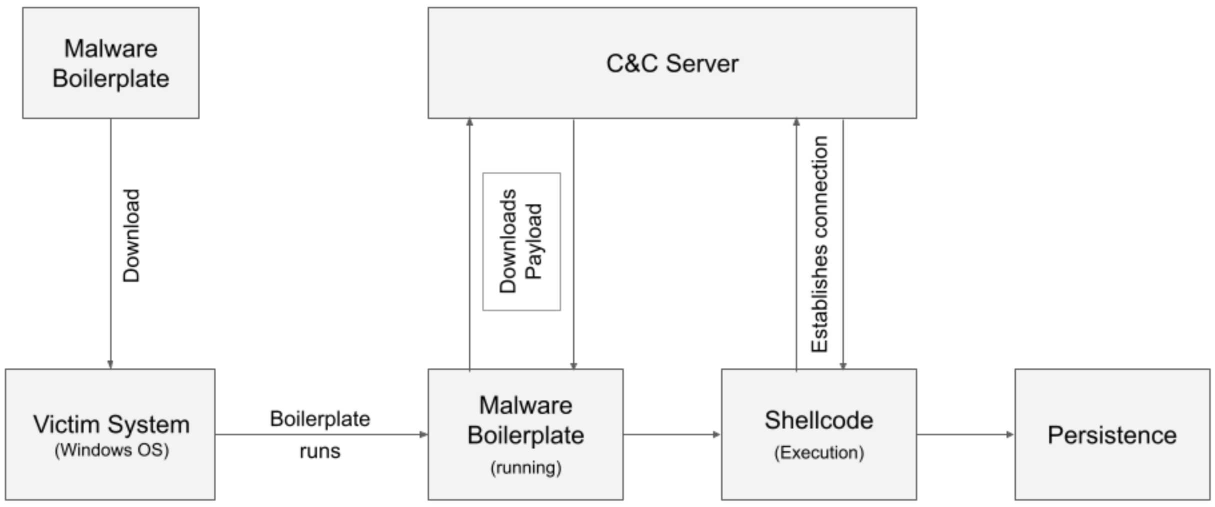

Figure 4.1: Block diagram of the proposed system

The following steps explains how to craft the malicious executable:

Step 1: Select both the files, and right click on it and select “Add to archive”.

Step 2: In the General tab under the Archiving Options, check “Create SFX Archive”.

Step 3: Go to the Advanced tab and click on SFX options.

Step 4: Under the Setup tab, add the names of the image file and the exe file.

Step 5: Go to the Modes tab, under Silent Mode, select “Hide all”.

Step 6: Go to the Text and icon tab, and load the image icon.

Step 7: Go to the Update tab, and under Overwrite mode, select Overwrite all files.

Step 8: Click OK and OK.

Now, the victim gets a boilerplate download that looks and functions just like an image file, but also executes the shellcode in the back.

Chapter 5

HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements

- **CPU**: Intel i7 or above / AMD Ryzen 7 or above
- **Memory**: 8 - 16GB
- **Hard Disk**: 500GB - 1TB

Software Requirements

- Windows 10 Host Machine
- Oracle VM Virtual Box v6.1
- Metasploit Framework
- .NET Framework

Chapter 6

PROJECT TIMELINE

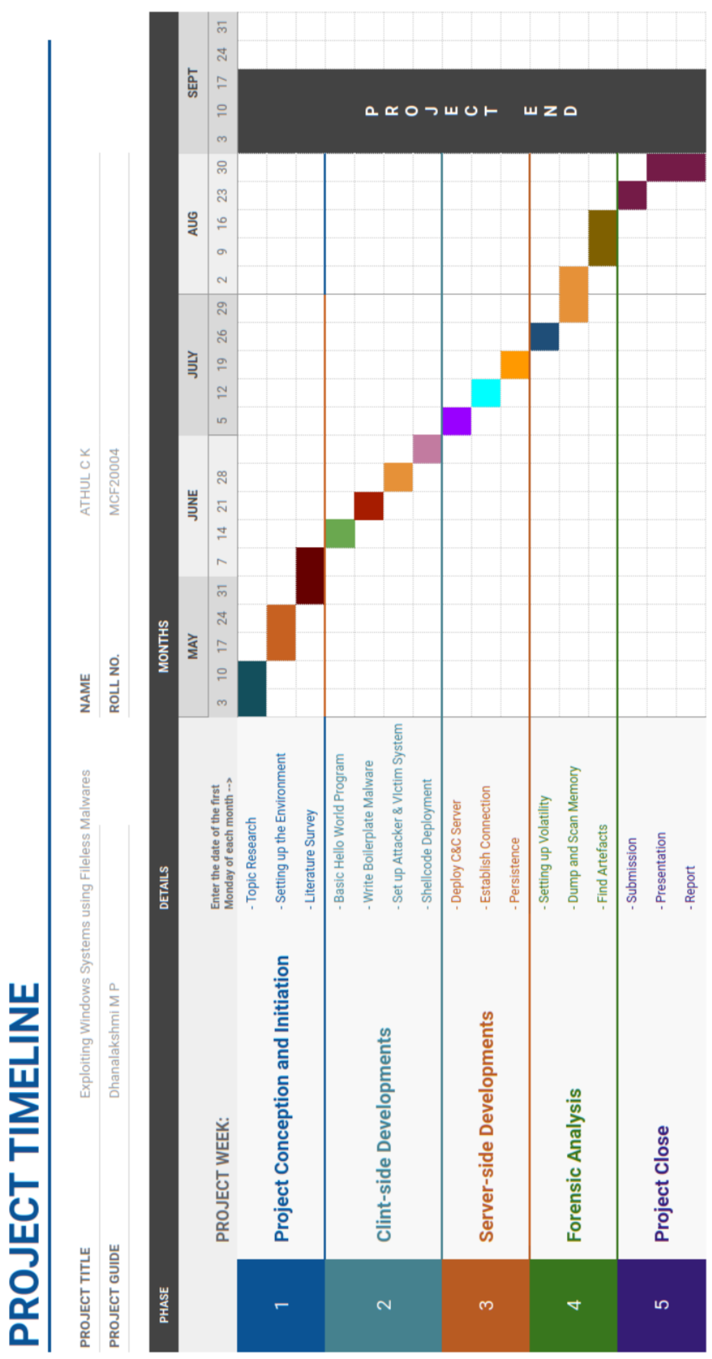

Figure 6.1: Project timeline represented by a Gantt Chart

Chapter 7

CONCLUSION

In this study, plenty of research papers and online resources were explored to study about the fileless malwares and its various attack approaches. After going through various approaches, the approach depicted in [3] seems exciting to work with. Although it does not heavily rely on the traditional fileless malware features and does not use much of the already available Windows tools like PS and WMI, this approach will surely serve as a starting point in further research. The use of two-staged fileless malware will be improved upon by integrating more Windows tools and also a more robust way of persistence will be looked into. A forensic analysis will also be conducted on the primary memory of the affected system for traces of malicious programs, which helps in identifying malicious artefacts related to the malware.

Bibliography

[1] M. Kaushik, M. Malik and B. Narwal, "Developing Malware and Analyzing it Afore & After Steganography with OSINTs," 2020 IEEE International Conference for Innovation in Technology (INOCON), 2020, pp. 1-4, doi: 10.1109/INOCON50539.2020.9298288.

[2] B. N. Sanjay, D. C. Rakshith, R. B. Akash and D. V. V. Hegde, "An Approach to Detect Fileless Malware and Defend its Evasive mechanisms," 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), 2018, pp. 234-239, doi: 10.1109/CSITSS.2018.8768769.

[3] A. Johnson and R. J. Haddad, "Evading Signature-Based Antivirus Software Using Custom Reverse Shell Exploit," SoutheastCon 2021, 2021, pp. 1-6, doi: 10.1109/SoutheastCon45413.2021.9401881.

[4] A. Afreen, M. Aslam and S. Ahmed, "Analysis of Fileless Malware and its Evasive Behavior," 2020 International Conference on Cyber Warfare and Security (ICCWS), 2020, pp. 1-8, doi: 10.1109/ICCWS48432.2020.9292376.

[5] F. A. Garba, K. I. Kunya, S. A. Ibrahim, A. B. Isa, K. M. Muhammad and N. N. Wali, "Evaluating the State of the Art Antivirus Evasion Tools on Windows and Android Platform," 2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf), 2019, pp. 1-4, doi: 10.1109/NigeriaComputConf45974.2019.8949637.

[6] Matt Graeber, "Abusing Windows Management Instrumentation (WMI) to Build a Persistent, Asynchronous, and Fileless Backdoor"(Online), Black Hat 2015.