An Approach to Detect Fileless Malware and Defend its Evasive mechanisms

[#1]Sanjay B N
Department of Computer Science and Engineering,
R.V. College of Engineering,
Bengaluru, India
[#1]sanjay.bommalene@gmail.com

[#2]Rakshith D C
Department of Computer Science and Engineering,
R.V. College of Engineering,
Bengaluru, India
[#2]rakshithdc043@gmail.com

[#3]Akash R B
Department of Computer Science and Engineering,
SJC Institute of Technology,
Bengaluru, India
[#3]akashrb8@gmail.com

[#4]Dr.Vinay V Hegde
Associate Professor,
Department of Computer Science and Engineering,
R.V. College of Engineering,
Bengaluru, India
[#4]vinayvhegde@gmail.com

***Abstract*— Malware, or "malicious program," refers to any malicious program or code that is harmful to systems. Malware Analysis has always been an important topic of security threat research ever since the early days of computers. Over times, many different types of malwares have been evolved increasingly trying to be unnoticed by an antivirus software and attack the systems. Traditionally, malware attacks as we commonly know are files written to disk in one form or another that require execution in order to carry out their malicious activities. Fileless malware, on the other hand, is purposed to be memory resident only rather than writing artifacts to the filesystem, ideally leaving no trace after its execution, leveraging Operating System resident tools namely, Windows Management Instrumentation (WMI) or PowerShell propagate, execute its payload, or otherwise perform the tasks it is designed to perform. The purpose of all this for the attacker is to make post-infection forensics difficult. In addition, this form of attack makes it nearly impossible for antivirus signatures to trigger a detection. In this aspect, this paper will discuss the technical details of Fileless malware and their related attacks in depth. Finally, we will discuss on various Fileless malware detection and mitigation techniques in detail.**

***Keywords—Fileless Malware; Windows Management Instrumentation; PowerShel*l*; Process Injection; Registry entry; Malware Evasio*n

I. INTRODUCTION

Malware is a popular term used to programs that are designed to alter, damage or gain access over the victim's computing systems, those of which are generally known as viruses, trojans and worms. These are software applications that are developed to install themselves in background on a host computer and affect the users in some way. There has been a rapid evolution in malware attack techniques over the past decade, including advanced polymorphic malware and fileless attacks. In specific, security industry has seen a substantial burst in fileless malware. In fact, according to a 2017 Verizon Data Breach Investigations Report [1], 49 percent of attacks involved methods of intrusion, including fileless malware that easily bypasses standard security solutions. Malware are of different forms namely, computer viruses, worms, Trojans, ransomware, scareware, spyware, cryptocurrency miners, adware, and other programs intended to exploit computer resources for unhealthy purposes. The objective of such malwares may be different in terms of its purpose of development, collecting personal data from its residing systems or making your computer involve in a large botnet or even encrypting your data, deleting things off of your computer. Fighting such attacks are antivirus software's.

Antivirus software, sometimes known as anti-malware software, is designed to detect, prevent and take action to block or remove malicious software and its associated files from your computer. Antivirus software will begin by scanning files from specified location and comparing them to specific bits of code that is usually a hash key, against information in its database. If there's found any matching in pattern residing in the database, it is considered a virus, and it will quarantine or delete that particular file. It will also scan your computer for behaviour's that may signal the presence of a new, unknown malware. Keeping in this view, Fileless malwares have been developed by the attackers in a way that it becomes very hard for antivirus software's to detect one. As the name suggests, fileless malware are written directly into memory rather than to a file on a hard drive. After writing malicious content to memory, hackers sometimes try to gain persistence on the system. In addition, hackers often seek out control over legitimate user applications and system administration tools such as PowerShell and Windows Management Instrumentation (WMI) to execute and spread fileless malware. In fact, Fileless malwares leave no traces for antivirus software to detect, thereby making it very hard for antivirus software to detect fileless malware attacks, hence discovering fileless malware attacks is very challenging task for security analysts.

Because fileless malware does not require a file to be downloaded, it is quite difficult to prevent, detect, and remove. Only optimal scenario in case of Fileless malware is that if you reboot your machine, you can halt the breach. Since RAM is a volatile memory, it only keeps its data when your computer is on. Once the power is off, the malicious malware is no longer live. However, hackers can still use that vulnerability to steal data from your computer or even install other forms of malware to give it persistence. For example,

ISBN: 978-1-5386-6078-2 © 2018 IEEE

hackers can set up scripts that run when the system restarts to continue the attack [2].

A. Fileless Malware Attack Life Cycle

Fileless Malware attacks ca be broadly classified into categories of low-observable characteristics (LOC) attacks, whose life cycle involve following stages.

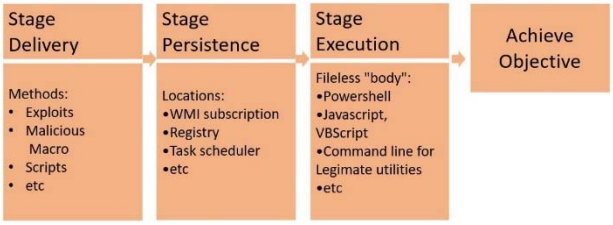

Fig 1.1 Lifecycle of Fileless Malware attack

Stage 1. Initial Delivery: Like most advanced attacks today, fileless attacks often use social engineering to get users to click on a link or an attachment in a phishing email. The malicious script, which may be hiding in Flash on a website or in a document generated by an authorized application, is downloaded in an inconspicuous way and written to memory or some other non-traditional location to evade detection by antivirus signature scanning. At this stage, attackers want to ensure that traditional security monitoring technologies will not inspect any files or activities. There are two main strategies at play here:

- Download to memory and execute:

  This is a big advantage for attackers because they can download the content of files whose signatures would have been detected if the files had downloaded to disk. One of the reasons attackers favor PowerShell-based malware is because it supports memory-based download and execution.

- Use trusted applications:

  In this case, attackers leverage approved, whitelisted applications, which security software won't inspect, since these applications don't usually download malicious content.

Stage 2. Fileless persistence: What happens to such fileless attacks when a targeted system is rebooted? The obvious answer is that malicious code in RAM would be erased. While it's true that most fileless techniques are short-lived, attackers use several evasive techniques to achieve persistence if that's what they are after. They do so by Storing malicious code in unusual locations associated with the operating system or common utilities, such as the Windows registry, WMI Store, SQL tables, or Scheduled Tasks Injecting code into a system process, which helps attackers evade inspection, as the activities will seem to come from legitimate processes. Malicious script directly passed as command line parameter to PowerShell and stored in registry and/or OS scheduler task, and executed by OS scheduler.

Stage 3. Malware Execution: When all the persistent mechanisms takes place, the malware explicitly depend on the windows internals like PowerShell, JavaScript, Macro execution of the office documents and many other legitimate resources of the windows executables. Usually are Dual-use tools like netsh PsExec.exe , Memory only payload e.g. Mirai DDoS and Non-PE file payload e.g. PowerShell scripts.

Fig 1.2 Executing malicious script with the help of mshta application

Fig 1.3 Using rundll32 application to execute malicious javascript script

Fig 1.4 Example of malicious WMI subscription

Stage 4. Achieve objective: When all is said and done, while objectives may vary reconnaissance, credential harvesting, data exfiltration, cyberespionage, or damage—fileless attacks are successful because they are so good at masking their activities and bypassing most security solutions, which use a file-based approach to detection, even if they do monitor behavior.

II. EVOLUTION OF FILELESS MALWARE

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, that 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 development of malware on .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 [3]. 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[4]. This ability also provides the ability to hide from file-based antivirus protection.

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 as depicted in fig 2.1[5]. 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.

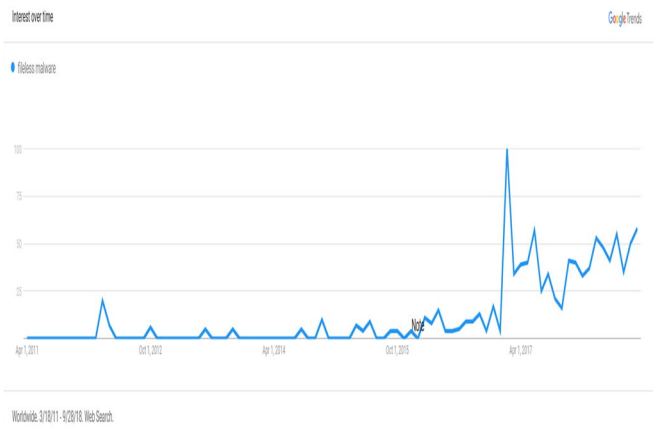

Fig 2.1 A graphical representation of people's interest on Fileless malware, Google Trends, as of April 2011- September,2018

As stated by McAfee, macro malware attacks had increased from 400,000 at the end of 2015 to over 1.1 million during the second quarter of 2017 while PowerShell hosted malware grew by 119% during the third quarter alone [6]. On February 8, 2017 Kaspersky Lab's Global Research & Analysis Team reported variants of this type of malware, and its latest incarnations, effecting 140 enterprise networks across the globe with banks, telecommunication companies and government organizations being the top targets. According to the Ponemon Institute's "State of Endpoint Security Risk Report 2017[8]," 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.

III. Categories of Fileless Malware

There are two main categories of fileless malware depending on the how it is executed:

- RAM-resident Fileless malware
- Script-Based Fileless Malware

A. RAM-resident Fileless Malware

The malware that executes explicitly in RAM is the main advantage for being stealthy. Since most of the antivirus solutions checks are done when a new process starts (verifying digital signatures, digital certificates, searching for malware signatures), processes that are already running on the system are considered unsuspicious for the antivirus solutions. So it does not trigger any event if malware does not create anything on the disk or make changes to file system.

Besides, most antivirus solutions have databases contain signatures for files with malware that basically act as a detection for particular variant of malware. Consequently, antivirus solutions decide how to react for an event based on the files found on the disk or the file system. However, nothing can be done against a fileless malware even if it's detected by runtime protection solution. There're only indirect indicators of fileless malware, for suspecting the instance processes that are currently running. while exiting or killing every suspicious process present in RAM is definitely not an option, since this could cause a damage for the unsaved data of the user.

Recent software applications run long enough for fileless attacks to succeed, and likely background process of the applications are still to be running even after the applications is closed. Thus, fileless malware that reside and run in RAM has more persistence when compared to other malware.

B. Script-Based Fileless Malware

Scripts have been the easiest way to perform any automation for system. It has been a popular attack vector for compromising a system. The main intention behind developing script-based malware is to exploit the vulnerabilities present in the Microsoft Office, Windows applications and Windows Power Shell.

We have seen in the past that VBScript in the Windows Microsoft Office documents was used to download additional second stage malware or the payload. Attackers made use of the VBScript to execute commands with minimal operating system privileges so that it will trigger to download additional malware. However, it was limited for malware because of the difficulty in windows Operating system entities like processors, service, threads, modules, windows registries, and so on. Then in the need of new attack vector they started targeting the Windows Power Shell. Windows Power Shell in a command line utility shell which has been made for the system administrators which include an interactive prompt and environment for scripting which can be made use in combinations of both. Windows Power Shell enables to access to registries, files, kernel configurations and the digital signature certificates as usual file system.

IV. Malware Evasion Techniques

Various evasion techniques malicious writers target:

A. Malicious Documents

Attacks that many professionals classify as fileless often involve document files. In such scenarios, the adversary supplies the malicious document—typically as an email attachment—for one of the following purposes:

- Documents can act as flexible containers for other files. The attacker can embed a JavaScript file in a Microsoft Office document, for instance, and social-engineer the recipient to double-click the embedded file to execute the script. Other document types that can carry files include PDF and RTF. Since this capability is a feature of the respective applications, anti-malware technologies generally don't interfere with its use.

- Documents can execute malicious logic that begins the infection. Modern documents support powerful scripting capabilities, such as Microsoft Office's ability to execute VBA macros. Such features allow the attacker to implement malicious logic without a compiled

executable, taking advantage of the weakness of many anti malware tools at distinguishing between nefarious and benign scripts. Document script abilities include launching programs and downloading malicious code.

Though documents reside on the endpoint's file system, they offer adversaries an opportunity to avoid placing traditional malicious executables on disk. In many cases, the document leads to the execution of malicious code directly in memory of the endpoint as part of the fileless infection.

B. Malicious Scripts

Consistent with the objective to avoid compiling malicious code into traditional executables, malware authors rely on scripts during attacks that have fileless attributes. Beyond the scripts supported natively by documents, as mentioned above, the scripts that run directly on Microsoft Windows provide adversaries with the following advantages:

- They can interact with the OS without restrictions that some applications, such as web browsers, might impose on the script.
- They are harder for anti-malware vendors to detect and control than compiled malicious executables.
- They offer a flexible opportunity to split malicious logic across several processes to evade Behavorial detection.
- They can be obfuscated to slow down analysts and further evade detection by anti-malware technologies.

Microsoft Windows includes script interpreters for PowerShell, VBScript, batch files and JavaScript. The tools that attackers invoke to run these scripts include powershell.exe, cscript.exe, cmd.exe and mshta.exe. With the addition of the Windows Subsystem for Linux, Microsoft offers even more scripting technologies on the endpoint. For an example of the challenges that enterprises face to restrict the misuse of these tools, see Gal Bitensky's post that discusses the Invoke-NoShell utility.

Attackers can use frameworks that obfuscate scripts without having to implement such evasion tactics themselves. These measures include Daniel Bohannon's Invoke-Obfuscation for PowerShell and Invoke-DOSfuscation frameworks. To see such tactics in actions, review Minerva's analysis of Emotet's script obfuscation.

C. Living off the Land

Discussions of fileless attacks often include the misuse of the numerous utilities built into Microsoft Windows. These tools allow adversaries to trampoline from one stage of the attack to another without relying on compiled malicious executables. This mode of operation is sometimes called "living off the land."

Once the adversary's malicious code can interact with local programs, possibly by starting the infection with a document, the attacker can misuse the utilities built into the OS to download additional malicious artifacts, launch programs and scripts, steal data, move laterally, maintain persistence, and more. The many tools that attackers invoke for these purposes include regsvr32.exe, rundll32.exe, certutil.exe and schtasks.exe. For a comprehensive listing and description of such built-in binaries, libraries and scripts that attackers misuse in this manner see Oddvar Moe's LOLBAS project.

Windows Management Instrumentation (WMI), built into the OS, offers attackers additional opportunities to live off the land. WMI allows adversaries to interact with most aspects of the endpoint with the help of the wmic.exe executable (and some others) as well as by using scripts (e.g., PowerShell). Since these actions involve only trusted, built-in Windows capabilities, they are difficult for anti-malware technologies to detect and restrict. For a comprehensive explanation of how WMI can assist with fileless attacks, see Matt Graeber's paper Abusing WMI to Build a Persistent, Asynchronous, and Fileless Backdoor.

D. Malicious Code in Memory

While examining files on disk is the strength of many anti-malware products, they often struggle with malicious code that resides solely in memory. Memory is volatile and dynamic, giving malware the opportunity to change its shape or otherwise operate in the blind spot of antivirus and similar technologies. Once the attacker starts executing malicious code on the endpoint, possibly using the methods outlined above, the adversary can unpack malware into memory without saving artifacts to the file system. This can involve extracting the code into the process' own memory space. In other cases, malware injects the code into trusted and otherwise benign processes.

Examples of in-memory attack techniques include the following:

- Memory injection utilizes features of Microsoft Windows to interact with the OS without exploiting vulnerabilities. For instance, API calls often abused by malware for injection include VirtualAllocEx and WriteProcessMemory, which allow one process to write code into another process. To see these techniques in action, read Gal Bitensky's overview of the AZORult attack.
- Attackers can wrap compiled executables into scripts that extract malicious payload into memory during runtime. One example of such toolkits is PowerSploit, which you can see in action by reading the GhostMiner analysis by Asaf Aprozper and Gal Bitensky. Chris Truncer's Veil Framework is another example.
- Process Doppelganging is an example of an approach to avoiding the file system that doesn't involve memory injection in a classic sense. Instead, the attacker misuses NTFS transaction capabilities built into Microsoft Windows to temporarily modify a trusted file in memory without committing changes to disk. SynAck malware used this evasion technique, as described by Anton Ivanov, Fedor Sinitsyn and Orkhan Mamedov.

In-memory techniques allow attackers to bypass many anti-malware controls, including application whitelisting. Though antivirus tools try to catch memory injection, adversaries' consistent ability to infect endpoints highlights their limitations. Asaf Aprozper's CoffeeShot tool demonstrates the brittle nature of such detection attempts by implementing an injection method in Java.

V. FILELESS MALWARE DETECTION AND MITIGATION

A. Fileless Malware Detection

The challenge raised in the detection of fileless malware is finding a new approach to the executable in the computer. Traditional approaches include sandboxing, execution emulation using tools like ProcMon, and heuristics based.

- Sandboxing: When the Windows Power Shell runs in the system, it should be run in the sandbox so that the executions done by it are clearly monitored for the API calls which are wrapped by the layer of sandbox and all the susceptible, dangerous calls are detected, blocked and informed in case of potential threat.

- Execution Emulation: As Windows Power Shell became to be an open source, it is possible now to design an emulation for the execution and interpretation of the Power Shell scripts. The emulation engine can be used for verification of the scripts before it starts in actual hosts PowerShell. It can also be used to deobfuscate the scripts and search for any string constants that are used to flag any script as a malicious indication.

- Heuristics: The main heuristics detection mechanism can be applied to the Power Shell scripts are monitoring the scripts for suspicious calls and the sources from where they are executed. It is necessary to look into the processes that started the Windows Power Shell process as it is always suspicious for a word document or browsers to run the Power Shell all of a sudden. Restrictions can be done on limiting the processes which are allowed to trigger a Power Shell. Heuristics cannot make clear conclusions whether the suspected script is malicious or not.

- Yara: With the extensive application usage, YARA is tool which is open-source designed to help the researchers and malware analysts to identify and classify malwares. Using which it is possible to create descriptions or yara rules for different malware families based on the binary or string patterns. YARA is also multi-platform which can run on windows, Mac OS, Linux. It is used via the command line or from the Python scripts with the built in Python-YARA extension.

Mentioned is an example of a detection using YARA rule. The example rule is completely random and was not made to detect any malware sample.

```
rule ExampleDetection
{
    strings:
        $hex_string = {AA (BB | CC) [3] FF [2-4] 00}
        $string1 = "malString" wide ascii fullword
        $hex2 = {CC DD 33 DD}
    condition:
        $hex_string and #string1 > 3 and $hex2 at entrypoint
and filesize > 200KB
}
```

Yara Rules for ShellTea + Poslurp Malware

```
rule inRegShellTea : ShellTea {
meta:
description = "ShellTea + PoSlurp PoS Malware in Registry
ShellTea"
version = "1.0"
last_modified = "2017-06-27"
strings:
$hex1 = {48 83 EC 28 E8 F7 03 00 00 [1015] 48 89 5C 24
18 48 89 4C 24 08 55 56 57 41 54 41} // Binary registry
value with variable content for ShellTea config
condition: $hex1
}

rule PoSlurpFile : PoSlurp
{
meta:
description = "ShellTea + PoSlurp PoS Malware on Disk
PoSlurp executable"
version = "1.0"
last modified = "2017-06-27"
strings:
$hex1 = {81 C2 FF 5C F3 22 52 56 E8} // outer layer
custom function resolver
condition:
uint16(12) == 0x5A4D and uint32(uint32(0x3C)) ==
0x00004650 and $hex3
}
```

Running all the above yara rules along with the susceptible executable will detect the PoSlurp and ShellTea Variants of the fileless malware, yara does this by comparing the binary executable, for attributes like creation date, author name, strings present in the executable. Multiple yara rules can be put in place to detect a single variant of malware.

A good static signature still allows you to be dynamic and detect malware even when a writer modifies his code. The following are examples of how legitimate applications are used to execute malicious scripts which are not stored on a disk. These techniques used for their persistence approach, become a true challenge for security solutions.

B. *Fileless Malware Mitigation*

- Ability to detect Windows Power Shell, white-listed applications, scripts and parameters against the file less attacks.
- Periodically check the advisories for recent patches for the vulnerabilities in the application and conduct Security checks frequently.
- Perform OS patches and updates periodically.
- Making sure to use other tools such as the MEME(Microsoft Enhanced Mitigation Experience Toolkit) to detect any activity upon a baseline.
- Restrict the Power Shell usage policy with restricted access to run the scripts through Windows Group Policy .
- Application Restriction for the control of the browser and other applications like Microsoft Office , Microsoft applications from spawning interpreters like Power Shell, JAVA and WMIC.
- Extensive utilization of machine learning and Artificial intelligence for the prevention of the exploit also the virtualization as sandbox mechanism to limit the scope of the scripts to create or download additional malware within the environment.
- Ability to monitor the activities in the Windows Power Shell or other engines which are accessing the thread data gaining information user activities.
- Ability to control the targeted system or the compromised system halting all the process, remediation processes and isolating the compromised devices.
- Gaining visibility into malicious behavior Reacting at scale in a fast and flexible way Forward-thinking third-party security vendors are in a better position than most OS providers to offer robust checks and balances that can improve your security posture.
- Endpoint hardening: Since file less attacks originate at the endpoint, essential defenses like vulnerability assessment, exploit/memory protection, desktop
  firewall, and URL filtering help minimize the attack surface and prevent the encounter from happening in the first place.
- Reputation enforcement :enable reputation to override and prevent processes in targeted applications from executing—even if the applications are authorized and whitelisted.
- Machine learning, pre- and post-execution analysis of custom-built payloads driven by machine learning can prevent secondary zero-day payloads from executing.
- Behavioral analysis allows efficient detection of fileless threats on execution stage. Behavior-based heuristics are analyzing execution patterns of any process in the system (including legitimate utilities) to detect attempts to perform malicious actions.

VI. CONCLUSION

Successfully interrupting fileless attacks requires a multilayered, integrated approach that spans the entire threat lifecycle. While it may seem daunting to tackle fileless attacks, it's important to remember that they are one category of tactics that are part of a larger, more complex campaign. Fileless malware will continue to evolve over years and get more disruptive with the availability of open source tools. Over period, security solutions should go beyond file detection and come up with more robust layered protective solution that can defend and mitigate all types of attacks.

REFERENCES

[1] Verizon Labs,"Data Breach Investigations Report-2017,Verizon." [Online]. Available: https://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-Breach-Investigations-Report.pdf

[2] Ellen Zhang, "What is Fileless Malware (or a Non-Malware Attack)? Definition and Best Practices for Fileless Malware Protection" [Online]. Available: https://digitalguardian.com/blog/what-fileless-malware-or-non-malware-attack-definition-and-best-practices-fileless-malwareI. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Chickowski E. (2016, December 27). "Fileless Malware Takes 2016 By Storm". Available: http://www.darkreading.com/vulnerabilities-threats/fileless-malware-takes-2016-by-storm/d/d-id/1327796

[4] Pontiroli S & Martinez F. R. (2015). "The Tao of .NET and PowerShell Malware Analysis". Virus Bulletin Conference.Retrieved from Secure list.

[5] Googletrends.Available:https://trends.google.com/trends/embed/explore/TIMESERIES?req={%22comparisonItem%22%3A[{%22keyword%22%3A%22fileless%20malware%22%2C%22geo%22%3A%22%22%2C%22time%22%3A%222011-03-18%202018-09-28%22}]%2C%22category%22%3A0%2C%22property%22%3A%22%22}&tz=240&eq=date%3D2011-03-18%25202017-04-18%26q%3Dfileless%2520malware

[6] McAfee Labs, McAfee Labs Threats Report June 2018, USA, 2018 Available:https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2018.pdf

[7] Kaspersky Lab, "Fileless attacks against enterprise networks," 2017. [Online]. Available: https://securelist.com/fileless-attacks-against-enterprise-networks/77403/.

[8] Charlie Osborne,Fileless attacks surge in 2017, security solutions are not stopping them" Available:https://www.zdnet.com/article/fileless-attacks-surge-in-2017-and-security-solutions-are-not-stopping-them/