Expert Systems With Applications 214 (2023) 119133

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges

Ilker Kara

Department of Medical Services and Techniques, Eldivan Medical Services Vocational School Çankırı, Karatekin University, Turkey

ARTICLE INFO

Keywords:
Computer Science
Computer Forensics
Cybersecurity
Malware Detection
Fileless Malware Prevention

ABSTRACT

The rapid advancements in cyber-attack strategies are in parallel with the measures for detection, analysis, and prevention. Attackers have recently developed fileless malware that can simply bypass existing security mechanisms. Researchers publish reports to help discover fileless malware and to better understand the threat's scope to counteract it. However, with the lack of studies on fileless malware regarding the classification and the scale of the threat, they have not been thoroughly analyzed. As a result, in this research, we explored the most recent advancements in fileless malware prevention and detection and highlighted future research challenges. We also propose an analytical approach based on the attack strategies and attributes of the selected sample. Our method simplifies feature extraction and reduces processing load. Furthermore, compared to the static analysis we do not need for decompression and unpacking for the analysis. We applied the proposed method on a real case example. It has been seen that information about fileless malware detection, working mechanism, attack method and attacker named "Kovter" can be accessed. Our approach is advantageous and can be applied as a new technique for fileless malware detection to protect systems from cyber threats. This paper also presents an insight to the fileless malware threat and provides a basic review of the methods and techniques used in the detection and analysis of fileless malware attacks.

1. Introduction

Malware is a term for malicious software that is a tool used to target our devices. It has been so long since the first malware came into being, researchers still struggle with how to protect our systems against malware. Moreover, there have been malware attacks and attackers even if the systems have strong security measures. Every day, new and more sophisticated malware is created, and attacks are planned to make them a game of arm wrestling between attackers and security experts (Beaman, Barkworth, Akande, Hakak, & Khan, 2021).

Cybercriminals now utilize advanced software to launch sophisticated cyber attacks on both essential infrastructures in the public sector and users. Thus, we can categorize malware into two different types such as fileless malware and host-dependent malware. Fileless malware is host-independent malware that does not require a host file to execute. Host-dependent malware is traditional malware that requires a host file to execute. As fileless malware can easily bypass intrusion detection systems, these attacks have recently become the new choice of cybercriminals (Bhardwaj, Al-Turjman, Sapra, Kumar, & Stephan, 2021). Despite the efforts of security companies to create methods and solutions to protect victims from these attacks, they continue to be a source of concern for individual users as well as commercial and government institutions. According to Crowdstrike's research, fileless malware attack techniques were utilized in eight out of ten effective malware attacks in 2016 (CrowdStrike, 2017).

The popularity of fileless malware has been grown as attackers seek to get around traditional antivirus software that has traditionally analyzed data on a hard disk (Kara, 2021). When anti-virus software discovers potentially harmful files, it takes measures to avoid them from file operations such as quarantine and/or deletion, to protect the system's applications, and to ensure the safety of the computer (Falah, Pan, Huda, Pokhrel, & Anwar, 2021). Fileless malware is created to execute in memory-resident apps without requiring any files to be loaded to a disk. The majority of the fileless malware lurks in the Windows registry and then deletes itself after the attack. Without additional protection, the traditional anti-virus tools will not identify these threats (Nuangpookka, Mengistu, & Bafail, 2020).

E-mail address: karaikab@gmail.com.

https://doi.org/10.1016/j.eswa.2022.119133
Received 9 April 2022; Received in revised form 21 October 2022; Accepted 21 October 2022
Available online 26 October 2022

1.1. How fileless malware infect?

When attackers create malicious software, they aim to stay undetected to achieve persistency. An installed malware in Start-Up folder is solid persistency mechanism. Because, every time the machine boots up so the malware. Moreover, the security measure can be easily evaded or halted. Thus, the attackers' latest strategy is to use fileless executable techniques to deploy their malware (Baldin, 2019). For this aim, attackers can employ a variety of approaches for fileless malware attack procedures (Wueest & Anand, 2017). The most common techniques are embedding malicious code into the Windows registry, manipulating Windows Management Instrumentation (WMI), and hollow process injection methods (Kumar, 2020).

The main goal of embedding malicious code in the registry is to achieve persistency. In 2014, the fileless virus "Trojan.Poweliks" became the first example of this approach in the literature (Baldin, 2019; Nuangpookka et al., 2020). Poweliks lives in the registry, hides in legitimate processes and creates a dll on the go. It also creates a registry key that contains a non-ASCII character string. This makes it harder to identify the attack since regular analysis tools are unable to reveal the malicious code.

Attackers use WMI to install posh spy, a persistent backdoor. WMI is used in this approach to connect with the attacker through constructing a backdoor without leaving any traces on the target computer (Afreen, Aslam, & Ahmed, 2020).

Hollow process or process hollowing is a technique that is used for changing executable section of legitimate process. Attackers use this method to get through firewalls and anti-viruses (Aljawarneh, 2011).

1.2. Fileless malware attack lifecycle

Fileless malware is meant to be difficult to detect and analyze. Therefore, standard prevention solutions fail (Afianian, Niksefat, Sadeghiyan, & Baptiste, 2019). Understanding the operational strategy of file-based malware is helpful in a clearer grasp of the attack plan of file-based malware. File-based malware infects a system by storing a malicious file attached to a specifically designed e-mail or a file downloaded from a malicious website to the target machine's hard drive. Fileless malware, on the other hand, infects the system through RAM using tools like PowerShell, which was created by Microsoft to improve control capability on Windows and Windows Server as well as to prepare automated operations. The life cycle activities to classify fileless malware's attack plan processes are shown in Fig. 1.

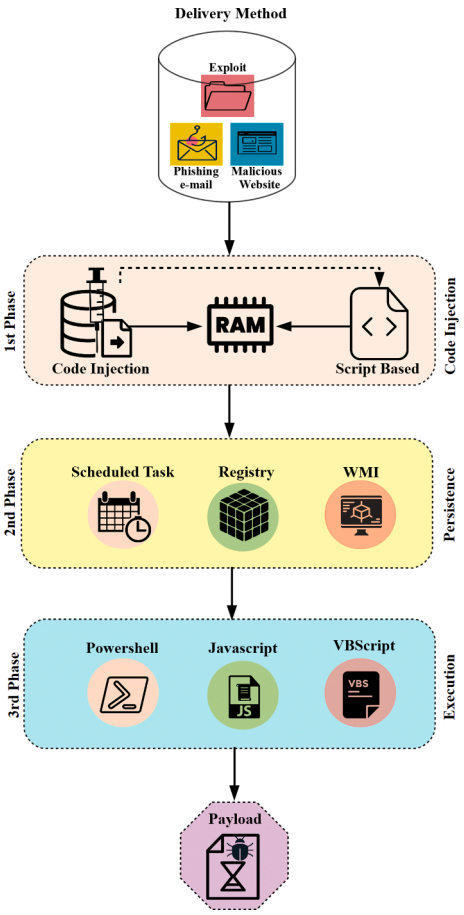

Fig. 1. Lifecycle of fileless malware attack in the literature (Bozkir et al., 2021).

- Delivery Method: It refers to the methods used to infect the target system of the fileless malware created at this point (Bozkir, Tahillioglu, Aydos, & Kara, 2021).The followings are the most commonly used ways in these attacks: i) The attacker achieves direct access to RAM with remote code execution by exploiting the target system's security flaws. The produced fileless malware takes advantage of the target system's security flaws and loads the harmful code content straight into the RAM. ii) In the phishing e-mail approach, users are directed to a website or URL through a link inside of the phishing email. By clicking the link a malicious file inside the URL is downloaded to the host. The malicious code is then executed in RAM. iii) The malicious website approach loads a file from this website that has been specially designed by the attacker into RAM.
- Code Injection: Infiltrating a target system and silently carrying out activities are known as fileless malware. Fileless malware in RAM masquerades as a harmless file, allowing it to avoid detection by the operating system (Block & Dewald, 2019). Furthermore, it leaves no evidence of its operations on the RAM while it is an effective strategy for avoiding forensic examinations. It is useful for a variety of script-based code injection scenarios.
- Persistence: When fileless malware is not persistent on the target computer, simply restarting the machine will remove the harmful file from RAM (Panchal, 2021). After reaching the target computer's RAM, attackers may wish to remain in the system. Fileless malware's persistence differs depending on the reason for which it was created. At times, attacks aren't even lasting (Wueest & Anand, 2017). In this instance, a method must be developed to ensure that fileless malware gets loaded into RAM each time the machine boots. Using Windows features like the System Registry, Windows Task Scheduler, and the WMI repository are the most common ways to do this (Afreen et al., 2020).
- Execution: Windows PowerShell, Javascript, or VBscript can be connected after ensuring fileless malware persistence. A macro or a script (e.g., Windows PowerShell) will load malicious code into RAM when the downloaded file is opened on the target machine. This code then operates as a downloader, connecting to a malicious file-downloading website or IP address. It relies on a file, this cyber attack method is not fileless; however, it becomes fileless once the payload is placed into RAM. As a result, when malicious code's payload is placed into RAM, it can use the.NET framework, WMI, or Powershell (Afreen et al., 2020). Attackers can use the NET framework to load malicious modules into the target system's RAM. As a result, the attacker can either run the malicious malware or permanently install it on the target machine. Malicious payloads might potentially be loaded directly into RAM using Windows PowerShell.

1.3. Malware analysis

Malware analysis is a common method for figuring out the nature and behavior of malware, including fileless malware (Lee, Shim, Cho, Kim, & Kim, 2021). For malware detection and analysis, many defense methodologies have been presented and may be divided into three

categories: static, dynamic, and memory-based as shown in Fig. 2 (Sihwail, Omar, Zainol Ariffin, & Al Afghani, 2019).

The static analysis involves collecting information about harmful software before its execution. This technique of analysis may be used to determine whether a suspicious file includes malware, and employs a variety of properties derived from the raw bytes of portable executable files (PEs) such as texts, opcodes, API calls, byte arrays, and control flowcharts (Gibert, Mateu, & Planes, 2020). For this reason, it is reasonable to assume that static methods operate with signatures derived from the attributes stated above. The static methods apply far fewer processing resources and provide a fast recognition scheme since the suspicious file does not need to be launched. The methods, on the other hand, frequently have major security flaws since they are subject to methods like code obfuscation and encryption (Or-Meir, Nissim, Elovici, & Rokach, 2019). Furthermore, malware may easily escape security safeguards when techniques like dead code insertion and packaging are used. For this reason, there may be a decrease in malware detection and recognition performance.

Aside from static analysis, dynamic analysis is a method that involves recording behavioral analysis of suspicious files by running them in sandboxes and virtual computers. The domain of the suspicious file, the activities it has performed, the actions it has produced, its movements in file directories, and its IP addresses may all be listed using the dynamic analytical method. One disadvantage of this system is that dynamic malware analysis typically requires significantly more resources (i.e., memory and CPU use) than static malware analysis. In addition, certain executable (i.e., portable executables, PE) files require unpacking and unpacking activities to show their distinct features. However, it is often reported to perform better in terms of accuracy than the static analysis (Santos, Devesa, Brezo, Nieves, & Bringas, 2013). Or-Main et al. also identified some risks of dynamic analysis, including the presence detection of analysis tools, nested virtualization, and logic bombs to disguise malware activities, in their wide survey (Sihwail et al., 2019). Dynamic analysis will need specific tools to keep an eye on questionable file activity if they choose this method. Process Monitor (Monitor, 2019), Process Explorer (Process Explorer, 2019), TDImon (TDImon, 2019), RegMon (RegMon, 2019), and Wireshark (Wireshark, 2019) are some of the analytic tools available.

Apart from the benefits and disadvantages of static and dynamic analysis methods, memory-based analysis (also known as memory forensic) is a useful method for detecting malicious software that has gotten greater attention in recent years. Volatile memory forensics consists of two stages: data collection, analysis, data collection, and converting the data stored in physical or virtual memory into memory dump files via kernel drivers or emulators. The analysis step aims to reveal useful information to detect the presence or behavior of malicious software (Sihwail et al., 2019).

Memory analysis provides some significant benefits in this area, making it a source of motivation for this research. Data in memory, according to Dai et al., is always observed (Dai, Li, Qian, & Lu, 2018). While suspicious files can be hidden via encryption and packing, all processes are visible in memory at run-time and can be inspected because malware must expose the majority of important information (e.g., logs, code, and data segments) in memory to function. In this case, volatile memory analysis allows the malware to be discovered simply by looking at its state in the system RAM.

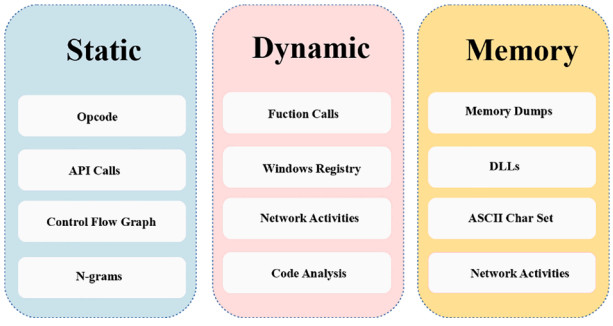

Fig. 2. Commonly employed malware detection methods in the literature.

A good approach to increase accuracy during memory analysis is to compare it with different memory structures or uninfected memories. So we can see differences between an infected and uninfected memory. An approach to memory that discovers the information in memory like function calls, DLLs and registry is crucial for technique mentioned above. So we can compare those features with the legitimate ones and we can see the anomaly more clear. Aghaeikheirabady mentioned a similar technique and, due to his findings, classifying accuracy is above 90 % in almost every feature (function calls, DLLs, API calls) (Aghaeikheirabady, Farshchi, & Shirazi, 2014). Which shows us comparing memory image and features in it with an uninfected memory can give us a better analysis ability finding fileless malwares.

This kind of approach is useful for complex systems such as servers. Due to its duty, servers run distinct and unique programs. If a server is infected with a fileless malware a memory comparison can be crucial since the regular memory operations are already different from well-known OSs. Since fileless malwares can hide in 3rd party tools and applications it would be a better approach to compare memories and see if the regular applications's processes are intact. In this paper we did not use this method. Because, our sample malware is hiding in built in tools and processes. However, a comparative approach can still be mentioned if we think that we already know how a legitimate process should behave.

Another major benefit of memory analysis is the ability to identify fileless malware, which avoids detection by leaving no trace or presence on hard drives (Sihwail et al., 2019). Typical signature-based anti-virus solutions have a difficult time detecting fileless malware. Fileless malware, on the other hand, remains in the victim's memory until it is terminated or the victim's machine shuts down, and these actions may be tracked using a memory analytical method.

Considering all these, we use a memory analysis approach in the detection and analysis of new generation fileless malware. To that purpose, the following three contributions are highlighted in this study:

- In this work, we suggest a memory-based(volatile memory forensics) approach for detecting and analyzing fileless malware.
- We also release a new dataset of 1249 pieces, in addition to the fileless malware dubbed Kovter that was used as an example in the study. https://ilkerkara.karatekin.edu.tr/RequestDataset.html
- We thoroughly defined and examined research difficulties.

This article is organized as follows: In Chapter 2, we reviewed several related studies. Chapter 3 details the analysis approach using Windows PowerShell scripts, and presents the application of the memory analysis method of fileless malware on selected examples and the results. Further, Section 4 presents the experimental results and discussion. Finally, Chapter 5 concludes the study and describes possible future directions.

2. Literature review

Fileless malware detection and analysis require a different approach than previous approaches. Many researches have been carried out on the detection and analysis of fileless malware. Although the system does not recognize unusual behaviors such as suspicious login activities, odd working hours, or abnormal network traffic. Anand Khanse suggested that fileless malware can be analyzed as fileless malware (Wueest & Anand, 2017). The high level of false-positive behavior detection is a key downside of this strategy.

Sanjay et al. classified fileless malware attacks into two categories; Memory and script-based attacks (Sanjay, Rakshith, Akash, & Hegde,

2018). Furthermore, they claimed that harmful code elements might be identified using applications like Yara, which can identify malware without downloading it.

Rivera et al. have extensively analyzed fileless malware like Poweliks (Rivera & Inocencio, 2015). They have also analyzed Phasebot, Gootkit, and Emotet malwares, as well as the techniques specific to this malwares.

Kumar et al. analyzed seven different fileless malware that resides in the system memory, persistent in the registry and proposed a framework for countermeasures that may be used when fileless attacks are carried out (Kumar, 2020).

In a similar study, Lee et al. analyzed the fileless malware Poweliks and Kovter (Lee, Kim, & Lee, 2017; Mo, 2020). They revealed how a fileless malware hides in registry and memory. They also provided a way to identify fileless malware using memory analysis.

Graeber used Windows Management Instrumentation (WMI) and Powershell to identify Windows-based fileless malware, which is similar to some of our work (Graeber, 2015). With the Powershell technique, the authors demonstrated an efficient solution to identify fileless malware. Because PowerShell scripts access data from many data stores, such as the file system or registry, they may be presented dynamically in memory (Bulazel & Yener, 2017). This prevents the attacker from remaining on the target system or spying on it (Bulazel & Yener, 2017; Case & Richard, 2017).

It's worth noting that the literature on fileless program identification contains some very unique techniques, such as those described in (Graeber, 2015). To begin (Rivera & Inocencio, 2015) suggests that program tools be used to identify memory and script-based fileless malware attacks. They discovered a new way to detect fileless malware in this way. The authors proved that achieving 100 % detection accuracy while providing a scalable solution against fileless malware is impossible. With the approach described in (Graeber, 2015), they attempted to overcome the problem by developing a model for memory analysis in Windows-based fileless malware attacks. They have shown that this strategy has a high rate of success. In this paper we have shown easy to implement and a well thought out technique that can be applicable on Windows machines.

Most of the analysts analyze fileless malwares considering process anomalies (Khushali, 2020; Varlioglu, Elsayed, ElSayed, & Ozer, 2022). They check for manipulation of legitimate processes (Kim, Alfouzan, & Kim, 2021). With this way they think they will find out which process is harmful and they think they can see the malicious connection or activity or both. For example an excel macro could call msiexec.exe and run the malicious code. In that case an analyst can easily find the source of malicious activity. Moreover, it is considerably easy to find that kind of process since it is not common that an excel process to have a msiexec.exe child process. In addition to that, most of the fileless malwares use similar technique. In contrast to that this kind of approach might make analyst to miss sophisticated techniques.

To detect well thought fileless malwares we need to consider different aspects of the system beside running processes. A good fileless malware analyse method should include:

- Registry control
- Checking for abnormal internet traffic
- Checking command logs
- Abnormal process check

The main goal of an analyst is to make sure if the system is safe. However if an anomaly is discovered in the system, in this case a fileless malware, we need to consider if this malware is persistent. To find out how persistent it is, we need to check Windows Security Logs to see if registries are manipulated. The reason we check registries is to find out how this malware achieve persistency. There can be several ways such as on boot persistency or hiding in another process after the initial run. After the discovery of the technique of the persistency, we need to find out the purpose of fileless malware. Because, a fileless malware can lead to next phase of intrusion or it can harm the system by itself.

In our paper we considered a holistic approach that includes process search, connection check and registry check. Thus, we can clear our concerns that are mentioned above and we can show a more reliable way.

3. Materials and methods

In this section, firstly we will introduce the specifically selected fileless malware samples. Then, the memory analysis stages in the detection and analysis of fileless malware used in the study will be shown in the relevant sub-sections.

3.1. Dataset

Well-reflective and accurate examples are critical for data-driven investigations. We worked with a cyber security firm in Turkey to come up with appropriate examples for this. The organization in issue had a specialized team and system in place to collect a large number of malware samples, and they provided us with actual fileless malware samples. It was very important to us that the chosen samples be among the most commonly observed. In addition, we provide a significant amount of datasets, largely from the Windows operating system, with various examples to provide resources for comparable investigations (https://ilkerkara.karatekin.edu.tr/RequestDataset.html).

3.2. Fileless malware analysis approach

This sub-section presents the proposed method workflow, steps, and results of experiments for fileless malware detection.

3.2.1. Workflow

The proposed method was applied to two different samples. First, the selected samples were run on the work computer. Overall study composes of three phases. The path followed at each stage is described in separate sections (Fig. 3).

3.2.2. Gathering memory data

While looking at current fileless malware detection research, it's clear that the data in volatile memory is often used as memory dumps. It might hold information about the behavior and structure of executable files (PE). Memory dumps, in most situations, provide an efficient answer to malware hiding and packaging techniques. The memory dump method, by definition, is the export of volatile data from the computer's physical and virtual memory. It's mostly used by software engineers and forensic investigators. The Volatility program may be used to analyze the memory dump files on the machine (Bozkir et al., 2021). This Python-based open-source tool can analyze memory dump files from Windows, Linux, OSX, and Android platforms, as well as provide the ability to analyze on Windows, Linux, and OSX systems.

Among the samples in our collection, the fileless virus Kovter was operated in a virtual environment (in an environment that is not linked to the internet). We have also protected our physical system from malicious activities with this way. We used Volatility to get the RAM to dump files. In our tests, we used the Volatility tool first, followed by a 5000-ms intentional delay. The purpose of the planned delay is to guarantee that only Volatility and the malicious program supplied are running. It's worth noting that the dump files' extension changes based on the host's operating system. We used the Windows version of Volatility as an example.

3.3. Fileless malware analysis

Botacin et al investigated RAM dump data and used it to detect and analyze fileless malware (Botacin, Grégio, & Alves, 2020). Similarly, we

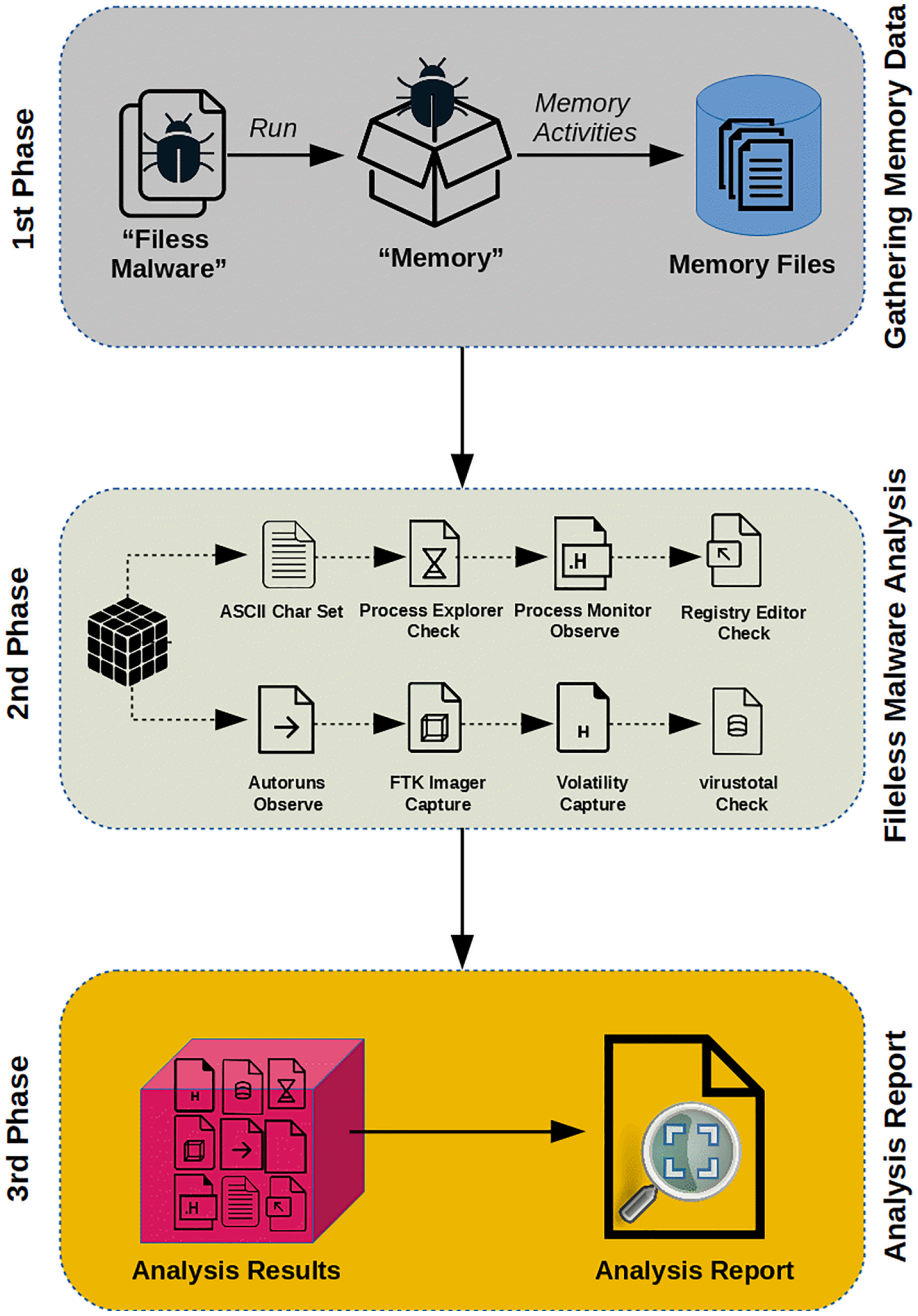

Fig. 3. The overall workflow of the proposed approach.

employed the RAM dump we received using the approach suggested in the paper in the investigation of fileless malware. In the analysis phase, we used the "strings" tool to see whether anything interesting happened, mostly using the ASCII character set. The processes operating on the system were checked using the "process explorer" tool without launching any fileless dangerous malware on the work machine. It was tracked whether the fileless virus created a new process or attached itself to an existing one as soon as it was launched. Suspicious processes were found and watched by watching their actions after running the fileless malware using the "process monitor" tool. We used the "netstat" tool to identify any unknown port or connections that the fileless malware could try to communicate with.

According to research, attackers tried to connect with fileless malware that has entered the target machine (Kara, 2021). To track down the attacker, it is critical to capture and analyze suspicious network activity. To identify unfamiliar or suspicious communications, we employed the "Wireshark" application. Virustotal was used to conduct a signature-based search of the data collected. So we double-checked the URLs, IP addresses, and files to be sure they were harmful.

The registry contains a huge number of configuration settings for all programs and applications on the computer, including the operating system. This data contains sensitive information about the computer user's identity as well as the programs that are now operating on the machine. That's why we employed the "autoruns" function to look through registry activities and look for unusual behavior.

Behavior analysis may access information such as the domain, the operations it does, the operations it produces, and IP addresses such as the command and control server when fileless malware is executing in memory. To get this key info, we used the "ollydbg" program.

Forensic specialists will need to look at the memory dump of the suspicious machine. The only correct way to get a snapshot (copy) of the contents of the system's physical memory is to do a system memory

dump. That's why, when the fileless virus was launched, we employed the "FTK Imager" program to capture the real-time memory image. The memory dump file was examined using the "volatility" program.

3.4. Analysis report

The definitions of the concepts should be made clear in the analytical reports that summarize the outcomes of the forensic investigations. The investigative methods and technical analysis of the incident should be understood in forensic analysis reports in a way that does not cause conceptual confusion. Another crucial element of the analysis reports is to clearly explain the tools and techniques used in the technical tests; this is necessary to make sure that the same findings are obtained when the same tools are used by different experts and the same processes are followed on the sample.

3.5. Case study

In this section, we analyzed two fileless malware to describe the techniques used in our entire approach, and in the sections below we provide an in-depth analysis of each type of attack.

3.6. Kovter

It's classified as fileless malware with a variety of tasks, similar to Kovter Poweliks (Lee et al., 2021), which was identified in 2016. According to 2016 research done by security firm Check Point (Kim et al., 2021), Kovter is executing a malware file on the victim's computer that includes an attached file. This malware file uses JavaScript to run through Mshta, which then executes the file and records harmful codes in the registry.

In an actual scenario, with the fileless malware sample Kovter, we used the analysis method suggested in our paper. Three winters were studied, and the methods outlined in Fig. 3 were followed.

3.7. Analysis of case study

Kovter is a well developed fileless malware and it is highly disguised since it heavily depends on registry. After initial execution Kovter injects its script to registry. Then the binary in the registry is injected in to a process (regsvr32.exe in this case). After that, malicious URL connections will start. In the following parts we analyzed every phase of Kovter.

In the VMWare environment, we first executed the Kovter fileless malware Workstation. We used the "process explorer" program (Fig. 4) to see what sort of process the suspicious file was executing at the time of execution.

The "regsvr32.exe" process created a file that appeared like a normal Windows process when the Kovter program was executed. Malware kills itself and makes regsvr32.exe its parent process as soon as the process is created. To further understand the generated process, right-click on "regsvr32.exe" (in this example, the process id is 2080) and use the "Environment" function of Process Explorer to discover the path of the process (Fig. 5).

Once Kovter's.exe has been identified, the registry values must be confirmed before the file can be further investigated. "HKLM/HKCU-SoftwareMicrosoftWindowscurrentversionRun/ RUNONCE" is the first place to look. When the device is restarted or powered on, the given registry path defines the files that run. However, since the virus modified the registry values when the stated registry location was attempted to be read, the most up-to-date data could not be detected. To identify registry

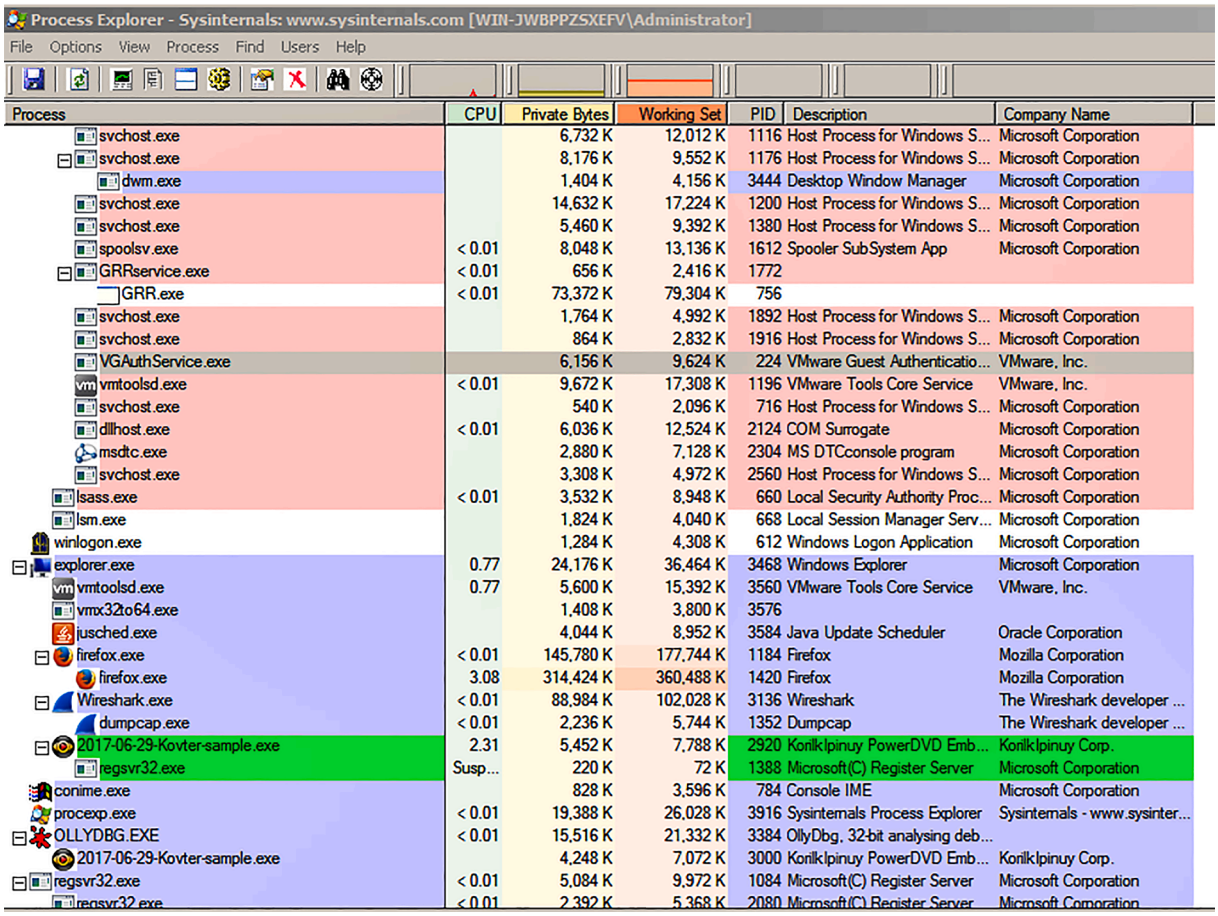

Fig. 4. Analysis results with process explorer program.

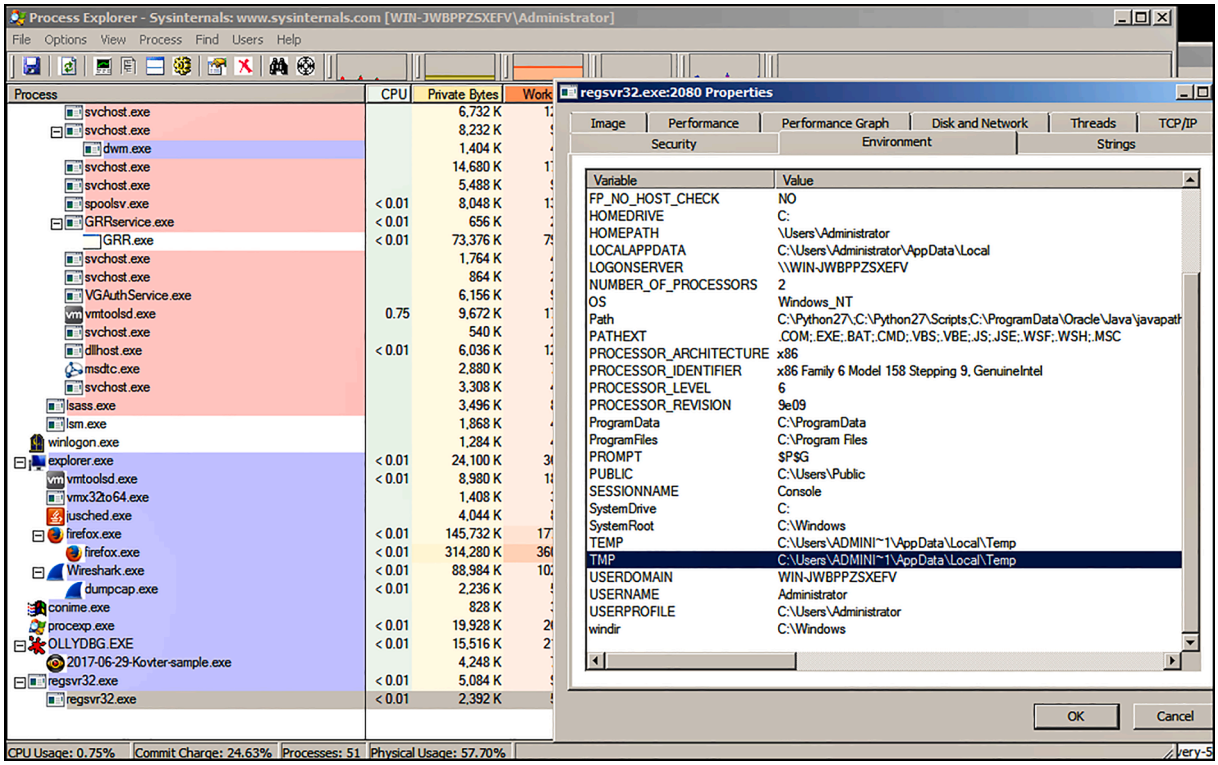

Fig. 5. "regsvr32.exe" analysis results with the process monitor program.

changes, we used an old but still effective Windows system tool which is "Autoruns". According to Autoruns' outputs, Kovter is the file in this location: "C:\Users\Administrators\appdata\local\ybybat\ybybat.exe". It was seen that it created another file under it.

Memory forensics is used to do further study on registry values. First, we used FTK Imager, a free and capable application, to collect the memory dump. The memory dump will be analyzed using the volatility tool. Before using the volatility tool, the operating system should be determined after the memory dump procedure with the FTK Imager tool. It may change based on the operating system, as we mentioned in the previous section. The analytic environment employed in the study is "Windows server 2008 service pack 1 with x86 architecture (Win2008SP1x86)", according to our findings. The next step is to finish the ongoing processes (Fig. 6).

According to the volatility output, processes with the pid of 1084–2080, and 3000–2600 and has run "regsvr32.exe" which were suspicious processes at the beginning of the investigation (Fig. 6). To analyze details, the procdump plugin of the volatility will be used.

First, we looked into the 1084–2080 procedure. We used the strings tool for suspicious transactions for this. Malware has affected multiple registry values and paths, according to the Strings tool's findings. However, due to the malware's nature, this path may be changed. The paths that have been defined are listed below;

HKLM\Software\RANDOM VALUE
HKCU\Software\RANDOM VALUE
HKLM\Software\Classes\RANDOM VALUE.
HKCU\Software\Classes\RANDOM VALUE.

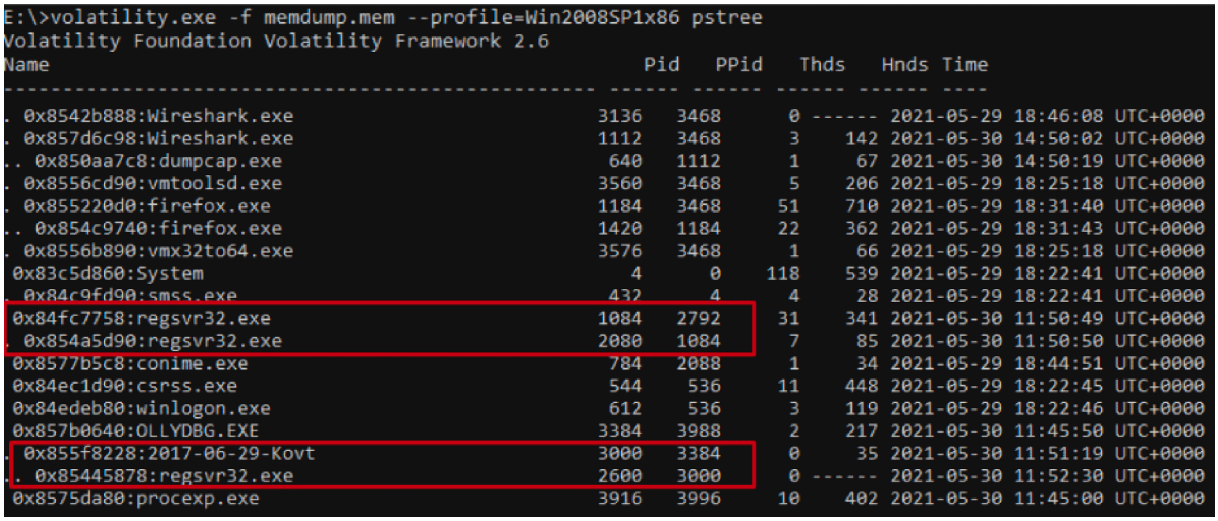

Fig. 6. The "regsvr32.exe" file is a screenshot of the list of dlls running on pid numbers 1084–2080, and 3000–2600.

We used the Registry Editor analysis tool to examine suspicious files in these defined paths. As a result of the analysis,"HKCU\Software\LocalSettings\Software\Microsoft\Windows\Shell\MuiCache", "HKLM\Software\mtez" and "HKCU\Software\mtez". records seen. As a result of the analysis made on these records, no anomaly was observed. However, upon examination with the Process Explorer tool, it was seen that the "regsvr32.exe" process was trying to communicate with some unknown web services (Fig. 7).

To identify which URLs or IP addresses "regsvr32.exe" malware tries to communicate with, process with pid 3000 also dumped from memory image. As a result of the analysis, the websites belonging to "HTTP, HTTPS or httx" with which the malware communicated were reached (Fig. 8).

4. Discussion

From the results of our literature review and experiments, we can make several observations about the detection, analysis, and limits of fileless malware. The dynamic analysis approach or the static analysis approach, or a mix of both, is used in fileless malware analysis. This is understandable given that static analysis can typically defeat code obfuscation and polymorphic/metamorphic attacks (Shaukat & Ribeiro, 2018).

However, several studies have shown that attackers may discover and avoid dynamic analytic tools (Kharaz, Arshad, Mulliner, Robertson, & Kirda, 2016). Both forms of analysis have limits, thus they can't be used to identify and analyze fileless malware variants in the same way. As a result, memory forensics offers significant benefits in terms of detecting and analyzing fileless malware (Bozkir et al., 2021). Our suggested approach includes memory dump file analysis, similar to Dai. et al's work memory dump file analysis has several advantages, including the ability to identify or even analyze an encrypted file or program known as fileless malware, which is a hazardous and complex sort of malware that injects itself without leaving any traces on the disk (Dai et al., 2018).

However, some research has proven that dynamic analytic methods may be discovered and tracked. Obtaining memory dumps, on the other hand, is a challenge. First and most importantly, this procedure requires that it be started in the operating system, which puts the system in danger. Its length is dependent on several environmental circumstances. It also requires that it be started in the operating system, which puts the system at risk. As a result, many experts prefer to deal with this danger in a virtual computer or sandbox. This slows down the process of extracting a memory dump and takes the system out of real-time mode (Korkin & Nesterow, 2016).

On the other hand, when clever malware detects that it is running in a sandbox, it has been known to shut down instantly. Memory dump-based malware techniques have additional problems that need time and effort to overcome.

Using the memory forensic technique, we discovered valuable information about how the malware hides its actions in the victim system, and the websites it connects with based on the findings of the study on the genuine fileless malware case sample named Kovter.

Another interesting problem is the potential for machine learning algorithms to help us. Through suspicious behavior or particular underlying processor design patterns, these models may be trained to detect general behavior patterns of fileless malware. Because ransomware is always developing and changing, machine learning's capacity to recognize the general behavior of fileless malware is critical. Moreover, a machine learning algorithm can also be applied for a comparison technique that is mentioned before in this paper. Comparison of a clean memory and an infected memory can be automated with a machine learning algorithm. Another advantage of machine learning algorithm when we are investigating a memory image is a machine learning algorithm can catch evolving technology.

5. Conclusion

The severe issue of the fileless malware cyberthreat is one that could block or restrict people from accessing their system or data. Some shortcomings that lower performance have been found out when the detection strategies of fileless malware in the literature and the analysis

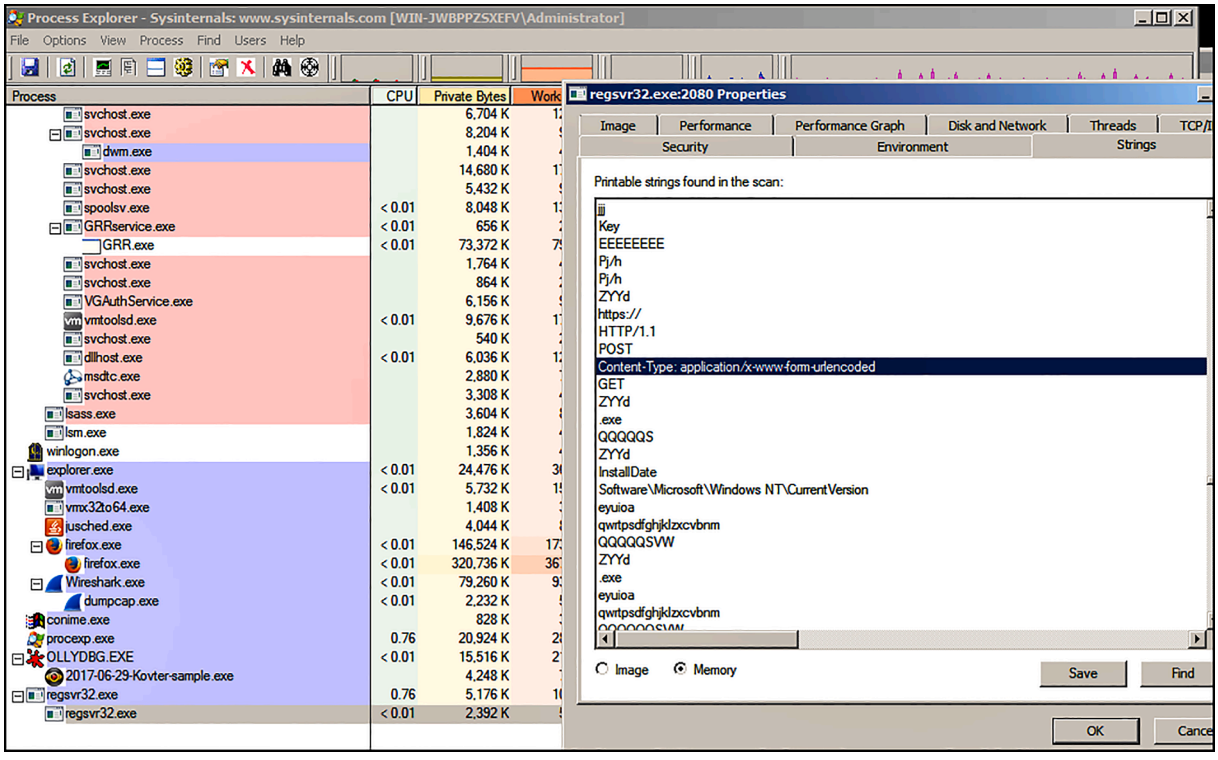

Fig. 7. "regsvr32.exe" analysis results with the process monitor program.

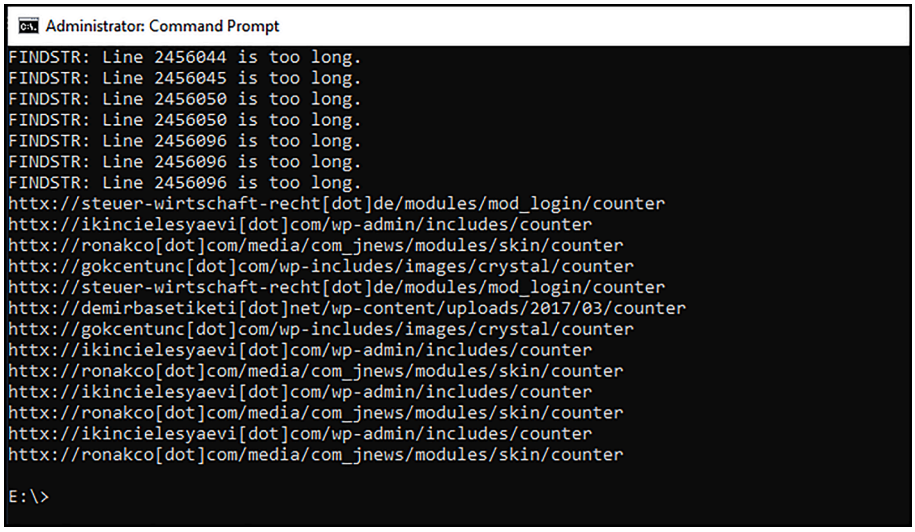

Fig. 8. The query of suspect IP address.

methods created utilizing these strategies are reviewed. These shortcomings can be listed as follows:

i. Automated analysis methods are unable to find fileless malware signatures.
ii. The behavior of fileless malware is unclear
iii. Fileless malware has numerous qualities that are unrelated to one another.
iv. Inadequate fileless malware next-generation detection
v. Fileless malware can easily go around the analysis done in the virtual environment and it is not impervious to attacks and cloaking methods

The memory analysis approach suggested in this study took these shortcomings into consideration, and any necessary adjustments were done at each stage. Following is a list of helpful tips for the identification and analysis of fileless malware in this situation:

- Opposing fileless malware cloaking strategies calls on the use of robust analysis methodologies. Because cloaking techniques mask fileless malware's true behavior, complicating investigation and raising false alarms.
- It is necessary to execute the examined software in several contexts in order to obtain the proper behavior of fileless malware.
- More efficient and comprehensive analysis approaches should be provided by combining the beneficial elements of the fileless malware analysis approaches (manual-automatic analysis).

Fileless malwares are hard to detect but we can still find them using 3rd party tools such as process explorer since they are living in the memory. Moreover, LOLBAS (Living Off the Land Binaries and Scripts) can be used to find how legitimate processes are used to achieve harm.

In conclusion, fileless malwares are widely used by attackers to evade security, achieve persistency or starting of next pahse of an attack. However, with yara rules, Event Data Recorders (EDRs), Living Off the Land Binaries (LOLBins), and LOL Scripts it is getting easier to detect and understand how those fileless malwares work. As a result, the suggested method holds promise for use in the analysis and identification of fileless malware utilized in similar intrusions. We believe that forensic investigation and analysis mentioned in this paper will become more significant in the near future as a result of the rise in fileless malware threats.

CRediT authorship contribution statement

Ilker Kara: Conceptualization, Methodology, Data curation, Software, Formal analysis, Writing – original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eswa.2022.119133.

References

Afianian, A., Niksefat, S., Sadeghiyan, B., & Baptiste, D. (2019). Malware dynamic analysis evasion techniques: A survey. *ACM Computing Surveys (CSUR), 52*(6), 1–28. https://doi.org/10.1145/3365001

Afreen, A., Aslam, M., & Ahmed, S. (2020). Analysis of Fileless Malware and its Evasive Behavior. In *2020 International Conference on Cyber Warfare and Security (ICCWS)* (pp. 1–8). IEEE. https://doi.org/10.1109/ICCWS48432.2020.9292376.

Aghaeikheirabady, M., Farshchi, S., & Shirazi, H. (2014). A new approach to malware detection by comparative analysis of data structures in a memory image. In *International Congress on Technology, Communication and Knowledge* (pp. 1–4).

Aljawarneh, S. (2011). A web engineering security methodology for e-learning systems. *Network Security, 2011*(3), 12–15. https://doi.org/10.1016/S1353-4858(11)70026-5

Baldin, A. (2019). Best practices for fighting the fileless threat. *Network Security, 2019*(9), 13–15. https://doi.org/10.1016/S1353-4858(19)30108-4

Beaman, C., Barkworth, A., Akande, T. D., Hakak, S., & Khan, M. K. (2021). Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & Security, 111*, Article 102490. https://doi.org/10.1016/j.cose.2021.102490

Bhardwaj, A., Al-Turjman, F., Sapra, V., Kumar, M., & Stephan, T. (2021). Privacy-aware detection framework to mitigate new-age phishing attacks. *Computers & Electrical Engineering, 96*, Article 107546. https://doi.org/10.1016/j.compeleceng.2021.107546

Block, F., & Dewald, A. (2019). Windows memory forensics: Detecting (un) intentionally hidden injected code by examining page table entries. *Digital Investigation, 29*, 3–12. https://doi.org/10.1016/j.diin.2019.04.008

Botacin, M., Grégio, A., & Alves, M. A. Z. (2020). September). Near-Memory & In-Memory Detection of Fileless Malware. In *The International Symposium on Memory Systems* (pp. 23–38). https://doi.org/10.1145/3422575.3422775

Bozkir, A. S., Tahillioglu, E., Aydos, M., & Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer

vision. *Computers & Security, 103*, Article 102166. https://doi.org/10.1016/j.cose.2020.102166

Bulazel, A., & Yener, B. (2017). November). A survey on automated dynamic malware analysis evasion and counter-evasion: Pc, mobile, and web. In *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium* (pp. 1–21). https://doi.org/10.1145/3150376.3150378

Case, A., & Richard, G. G., III (2017). Memory forensics: The path forward. *Digital Investigation, 20*, 23–33. https://doi.org/10.1016/j.diin.2016.12.004

CrowdStrike. (2017). Who needs malware? How adversaries use fileless attacks to evade your security. Retrieved from https://go.crowdstrike.com/rs/281-OBQ 266/images/WhitepaperFilelessAttacks.pdf?aliId=820125.

Dai, Y., Li, H., Qian, Y., & Lu, X. (2018). A malware classification method based on memory dump grayscale image. *Digital Investigation, 27*, 30–37. https://doi.org/10.1016/j.diin.2018.09.006

Falah, A., Pan, L., Huda, S., Pokhrel, S. R., & Anwar, A. (2021). Improving malicious PDF classifier with feature engineering: A data-driven approach. *Future Generation Computer Systems, 115*, 314–326. https://doi.org/10.1016/j.future.2020.09.015

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications, 153*, Article 102526. https://doi.org/10.1016/j.jnca.2019.102526

Graeber, M. (2015). *Abusing Windows Management Instrumentation (WMI) to Build a Persistent, Asyncronous, and Fileless Backdoor* (pp. 31–48). Las Vegas, NV, USA: Black Hat.

Kara, I. (2021). Cyber-Espionage Malware Attacks Detection and Analysis: A Case Study. *Journal of Computer Information Systems, 6*, 1–18. https://doi.org/10.1080/08874417.2021.2004566

Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., & Kirda, E. (2016). {UNVEIL}: A {Large-Scale}, Automated Approach to Detecting Ransomware. In *25th USENIX security symposium (USENIX security 16)* (pp. 757–772).

Khushali, V. (2020). A Review on Fileless Malware Analysis Techniques. *International Journal of Engineering Research & Technology (IJERT), 9*, 46–49.

Kim, K., Alfouzan, F. A., & Kim, H. (2021). Cyber-Attack Scoring Model Based on the Offensive Cybersecurity Framework. *Applied Sciences, 11*(16), 7738.

Korkin, I., & Nesterow, I. (2016). Acceleration of statistical detection of zero-day malware in the memory dump using CUDA-enabled GPU hardware. *arXiv preprint arXiv:1606.04662*, (pp. 1-27). doi:10.48550/arXiv.1606.04662.

Kumar, S. (2020). An emerging threat Fileless malware: A survey and research challenges. *Cybersecurity, 3*(1), 1–12. https://doi.org/10.1186/s42400-019-0043-x

Lee, G., Kim, K., & Lee, S. (2017). Analysis and detection methods for the fileless in-memory malwares. In *2017 Conference on Information Security and Cryptography-Summer* (pp. 5–12).

Lee, G., Shim, S., Cho, B., Kim, T., & Kim, K. (2021). Fileless cyberattacks: Analysis and classification. *ETRI Journal, 43*(2), 332–343. https://doi.org/10.4218/etrij.2020-0086

Mo, B. (2020). The classification model of fileless cyber attacks. *Journal KIISE, 47*, 454–465.

Nuangpookka, P., Mengistu, Z., & Bafail, G. (2020). Fileless malware and programmatic method of detection. *Journal of Computing Sciences in Colleges, 36*(3), 161–162.

Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era-A state of the art survey. *ACM Computing Surveys (CSUR), 52*(5), 1–48. https://doi.org/10.1145/3329786

Panchal, R. (2021). A Review On Protection Against Fileless Malware Attacks Using Gateway. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12*(10), 7302–7307. https://doi.org/10.17762/turcomat.v12i10.5620

Process Explorer., (2019). https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer.

Process Monitor., (2019). https://docs.microsoft.com/en-us/sysinternals/downloads/procmon.

(RegMon., 2019). https://docs.microsoft.com/en-us/sysinternals/downloads/regmon.

Rivera, B. S., & Inocencio, R. U. (2015). Doing more with less: a study of fileless infection attacks. *Virus Bulletin*, 2–56.

Sanjay, B. N., Rakshith, D. C., Akash, R. B., & Hegde, V. V. (2018). An approach to detect fileless malware and defend its evasive mechanisms. In *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)* (pp. 234–239). IEEE. https://doi.org/10.1109/CSITSS.2018.8768769.

Santos, I., Devesa, J., Brezo, F., Nieves, J., & Bringas, P. G. (2013). Opem: A static-dynamic approach for machine-learning-based malware detection. In *International joint conference CISIS'12-ICEUTÉ 12-SOCÓ 12 special sessions* (pp. 271–280). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-33018-6_28.

Shaukat, S. K., & Ribeiro, V. J. (2018). RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)* (pp. 356–363). IEEE. https://doi.org/10.1109/COMSNETS.2018.8328219.

Sihwail, R., Omar, K., Zainol Ariffin, K. A., & Al Afghani, S. (2019). Malware detection approach based on artifacts in memory image and dynamic analysis. *Applied Sciences, 9*(18), 3680. https://doi.org/10.3390/app9183680

(TDIMon., 2019). https://sysinternals.d4rk4.ru/Utilities/TdiMon.html.

Varlioglu, S., Elsayed, N., ElSayed, Z., & Ozer, M. (2022). The Dangerous Combo: Fileless Malware and Cryptojacking. *SoutheastCon, 2022*(5), 125–132. https://doi.org/10.1109/SoutheastCon48659.2022.9764043

(Wireshark., 2019). https://www.wireshark.org/download.html.

Wueest, C., & Anand, H. (2017). Internet security threat report-living off the land and fileless attack techniques. *An Istr Special Report*. (pp. 4-9).