



Efficient Threat Hunting Methodology for Analyzing Malicious Binaries in Windows Platform

Ahmed M. Elmisery¹, Mirela Sertovic², and Mamoun Qasem¹(✉)

¹ Faculty of Computing, Engineering and Science, University of South Wales, Pontypridd, UK
Mamoun.qasem@southwales.ac.uk

² Threat Defense Unit, Concept Tech Int. Ltd, Belfast, UK

Abstract. The rising cyber threat puts organizations and ordinary users at risk of data breaches. In many cases, Early detection can hinder the occurrence of these incidents or even prevent a full compromise of all internal systems. The existing security controls such as firewalls and intrusion prevention systems are constantly blocking numerous intrusions attempts that happen on a daily basis. However, new situations may arise where these security controls are not sufficient to provide full protection. There is a necessity to establish a threat hunting methodology that can assist investigators and members of the incident response team to analyse malicious binaries quickly and efficiently. The methodology proposed in this research is able to distinguish malicious binaries from benign binaries using a quick and efficient way. The proposed methodology consists of static and dynamic hunting techniques. Using these hunting techniques, the proposed methodology is not only capable of identifying a range of signature-based anomalies but also to pinpoint behavioural anomalies that arise in the operating system when malicious binaries are triggered. Static hunting can describe any extracted artifacts as malicious depending on a set of pre-defined patterns of malicious software. Dynamic hunting can assist investigators in finding behavioural anomalies. This work focuses on applying the proposed threat hunting methodology on samples of malicious binaries, which can be found in common malware repositories and presenting the results.

Keywords: Malicious binaries · Malware · Threat hunting · Digital investigations

1 Introduction

Nowadays, cybersecurity can be considered one of the most crucial topics in the computer science discipline [1]. Enterprises are obligated by law to protect sensitive information. They usually rely on the available security controls such as firewalls, intrusion prevention, and application whitelisting systems. However, some attacks still occur in these organizations even when their systems are secured with these controls. The effect of a single successful attack may be more severe in the organizational and/or personal context. If a malicious software infiltrates the network, it may cause real damage with even

less possibility of attributing the adversary behind that. In enterprise environments, there is a high probability of being targeted with sophisticated attacks. Hence, relying solely on existing security controls is not sufficient. The current system mechanisms rely on the use of predefined characteristics or behavioural analysis of previous/known attacks [2]. Using signature-based detection, only malicious software that is already known and vulnerable can be detected. The behavioural-based detection can help to mitigate so-called zero-day attacks, which are newly created malicious software utilizing zero-day vulnerabilities with no threat information defining it. Only malicious software that is already identified and analysed can be detected using signature-based identification. Behavioural-based identification may help to alleviate the impact of so-called zero-day attacks and any newly evolved malicious software that exploits these vulnerabilities, even without prior threat information describing it. However, given the number of successful attacks and their impact, there is a need for quick and efficient methodologies to analyse malicious files and attachments.

Digital investigations are scientifically proven methodologies that lean towards examining and analyzing digital artifacts. The success of each methodology relies heavily on the competence of the investigators in manually analyzing large amounts of digital data to identify appropriate artifacts. This requires enormous analytical resources given the volume of data collected. The proliferation of cyber-crimes cases, increases the demands and needs for new methodologies and paradigms to cope with such a critical situation. One of the emerging themes of digital investigations is threat hunting. Threat hunting can simply be defined as a preemptive methodology to cyber-defense that encompasses continuous and reiterated searches across internal systems for complex threats and possible vulnerabilities that evade current security controls. Threat hunting differs from the existing digital investigation methodologies, which utilize a passive approach, which detect cyber threats when pre-configured rules deployed within security solutions are violated. Threat hunting can immensely support the creation of a robust digital perimeter, which constantly tracks digital artifacts that could lead to the early detection of unusual threats or destructive activities on the local systems. This, in turn, can have a profound impact on protecting businesses and people.

In the past, threat actors utilized a wide range of malvertising campaigns linked to malevolent exploit kits to target selected victims. This vector resulted in a large number of attacks but less return in success. In recent years, threat actors have begun to utilize targeted attacks for specific high-value industries. These targets are tempting because they store in their servers a gigantic sensitive data and/or their digital operations are critical to the national infrastructure. Example for such common targets are governmental contractors, healthcare providers, and financial institutions. Targeted attacks against other enterprises and individuals are continuing at a less frequent rate [3]. The hunting for malicious activities relies on a combination of different utilities and techniques to gain different views of events occurring in the system. This helps the investigators to analyse and locate evidences regarding the potential maliciousness of a certain process or a file exists in the environment. The realization of the threat hunting relies heavily on the anticipated methodology that the investigators use to conduct a quick and efficient analysis of security threats on large amounts of unlinked data to determine appropriate evidence of misconduct, which in turn will require significant time given the amount of

data involved in the hunt. The main aim of this research is to present an efficient threat hunting methodology that includes a combination of different utilities and techniques to facilitate the immediate detection of malicious binaries against the windows platform. The suggested methodology will have a significant impact on alleviating the need for performing a lengthy binary reverse engineering. Since only high confidence evidence of misconduct related to the malicious binaries is provided to the investigators. The presented methodology will help to resolve the uncertainty of evidence acquisition, as early copies of any malicious binaries or processes are analysed and compared together from multiple internal and external sources. This is considered to be beneficial in detecting malicious software propagation in the internal systems. Finally, the logs obtained from several security controls can collectively support the insights extracted from the proposed methodology, which in turn, makes the extracted evidence much stronger as well as ensures the validity of a threat to the internal systems.

In this research, a threat hunting methodology is suggested to explicitly extract significant information related to any binaries under investigation using multiple utilities that implement various techniques. This methodology has been designed to be carried on windows systems. The presented methodology will help investigators to easily follow a proactive approach for threat hunting tasks. Any suspicious binaries from different windows systems are collected and analysed using our proposed methodology to restrain any probable risks of data breaches or systems outages. During the analysis phases of the threat hunting process, any beneficial patterns can easily be found, which can be preserved to facilitate the classification process related to emerging threats. This paper has been organized as follows, In Sect. 2, relevant works were summarized. In Sect. 3 the proposed threat hunting methodology envisioned in windows systems is outlined along with the study of selected utilities and techniques. Discussions on static and dynamic hunting tools and techniques were presented in Sects. 4 and 5 respectively. Finally, the conclusions and future directions were given in Sect. 6.

2 Related Works

Presently, digital investigations are potent technical approaches used by nearly all large companies. The digital investigations' processes are usually conducted only after a violation has occurred. Recently, the focus of this domain is shifting from reactive approaches towards more proactive ones, where defense strategies would identify risks and flaws in the internal system to deter violations from emerging. An Earlier detection of any malicious actions or possible weaknesses, a greater chance of limiting or preventing any losses that could arise. This strategy can be defined as threat hunting, and has grown rapidly as a hot trend in the context of cyber-forensics. However, there is a noticeable lack of research studies on this modern approach. Each cyber defense provider appears to support its own interpretation of threat hunting in order to distinguish its own offering as a threat hunting system. This in essence, tends to increase the uncertainty regarding its definition. There are several common meanings that describe this concept established on the basis of its interpretation within the cyber-security context. For example, threat hunting can be described as the operation of finding adversaries in local systems before successfully carrying out attacks [4]. Sqrrl describes threat hunting as a systematic and

iterative scanning across networks and databases to identify potential threats that can ultimately bypass current security controls [5]. For the context of this study, threat hunting can be briefly described as preventive practices that finds evidence of misconduct in local systems. The research in [6], a framework was proposed to models multi-stage attacks in a manner that defines the attack methods and the predicted consensuses of each attacks. The basic aim of their study is to simulate misconduct using the cyber kill-chain and intelligence driven defense patterns. Their suggested approach was implemented on Apache Hadoop. In [7], the authors introduced a method that uses text mining techniques to systematically compare the collected evidence of security-related incidents with a database of attack patterns. The goal of their work is to reduce the hunting time and improve the efficiency of attack detection. In [8] an approach was presented to incorporates the identification of systemic abnormalities from communication networks with the psychological profiling of users. Systematic anomaly recognition utilizes graph analysis and learning techniques to recognize systemic abnormalities in diverse knowledge networks, while psychological profiling dynamically blends human psychological characteristics with their behavioural trends. The authors continue to classify threats by connecting and rating the varied findings of systematic anomaly recognition and psychological profiling of the systems' users. The authors in [9], a component of threat identification has been incorporated in their research, which depends on the recognition of irregular variations in the weight of the edges over time. Wavelet decomposition technique was used to separate the transient behaviour from the stationary behaviour in these edges. The research in [10] Introduced the use of data mining techniques with visualization techniques to address various threats. In their study, two new simulation methods have been suggested to visualize threats. The authors in [11], a botnet discovery methodology has been designed, which uses cluster analysis to classify the correlation trends of C&C communication and behaviour flows. The methodology proposed in this work begins with sniffing the network traffic, then conducting two kinds of parallel investigations, one investigation is conducted to identify a group of hosts with identical traffic patterns, while the other examines packet payloads to identify anomalous behaviours. The behaviours are later pooled together to detect a swarm of hosts with similar malignant activity. The cross-correlation method is used to combine the findings of prior analyses into coherent classes of hostile hosts that may constitute a Botnet. In [12] introduced a threat hunting approach to implicitly elicit the relevant threat reputation groups from the multiple feeds of event logs. The proposed approach has been designed to be carried at the smart homeowner side, it also attains security, privacy for event logs which can help smart homeowners to easily adopt a proactive approach for threat hunting in a privacy-preserving manner. Two-stage concealment protocols, which was previously presented in [13–17], used for masking the event logs of smart homeowners when being released for threat hunting. A detailed survey in [18], presented a set of detection techniques for cryptographic ransomware. However, the authors have not centered their research on producing a practical methodology that links these detection algorithms to tools that can assist the investigators in the threat hunting process. Finally, the research work in [19] employed static and dynamic analysis to study WannaCry ransomware. The goal of this research is to uncover crucial information regarding encryption functionality, dynamic link libraries, and Windows API used by WannaCry. This can assist in implementing

effective mitigation mechanisms to detect WannaCry and other strains of ransomware that have a similar profile. Most of the related work in this field has the same direction. The main aim of this work is to propose a general threat hunting methodology that is not tailored to a particular strain. Our proposed methodology provides the building blocks and popular methods that can assist the investigator in identifying evidence of misconduct related to the malicious binaries quickly and efficiently.

3 Proposed Threat Hunting Methodology

Malicious binaries serve the disruptive aims of the threat actors, and these aims range broadly from each payload to the next, from ransomware attacks on small to medium-sized businesses to large state-funded operations. Malicious payloads can also be distributed in a range of manners, such as manipulating browser vulnerabilities, insecure networking services, and social engineering. Cyber defense providers typically supply signatures to their security controls that try to align signatures against data located on the user's device. In order to build these signatures, malignant artifacts must first be detected and then assessed for patterns of misconduct, using two separate examination methods, static analysis, and dynamic analysis [20].

The static analysis mainly relies on decompilers where a reverse engineering process is conducted to convert the malicious binary into assembly or C code. Subsequently, the generated code can be rigorously examined to gain deeper insights into these patterns and artifacts that can be considered malicious. The malicious binary will never be executed during the various static analysis tools. Although static analysis alone generates useful insights regarding various functions of the binary under investigation, it can be used as an entry point for dynamic analysis, for example, it can be used to identify main functions that need further investigation using dynamic analysis tools [21]. Analyzing malicious binaries using static analysis is a complicated process that requires certain skill sets. Malicious software programmers use concealing strategies that can make it incredibly difficult for static analysis tools to read generated code and assess the control flow of its functionality, as well as resolve API calls at the run-time to make it harder to identify the main ones that can attribute any binary as malicious.

Dynamic analysis is conducted by executing the malicious binary in a restrained environment. It offers useful information about the binary under investigation, as the conducted examination can intercept API calls, and evaluate the memory contents assigned to the binary. Other valuable insights can also be gained when examining the requested URLs, files generated/updated during/after the execution, and registry keys inserted/edited during/after the execution. Currently, there are numerous research efforts underway to integrate artificial Intelligence with conventional dynamic analysis tools. This is achieved by using sandbox environments to construct detailed activity reports that could be used to detect harmful payloads that are not commonly identified with signature-based monitoring [22]. Performing dynamic analysis has multiple benefits, it especially aids in detecting any concealment process supported in the malicious binary, as it is not susceptible to evasion techniques. However, it does have a significant downside, as it requires the malignant binary to run on the system, and this is an essential step to ensuring that the malicious payload is fully implemented and the occurrence of all

that all modifications into the system or changes to the file system are present. Due to that, it is highly recommended to carry out the dynamic analysis in a protected and isolated environment, such as containers or virtual systems, which can always be reverted to its original state. The workflow of the proposed methodology is presented in Fig. 1 depicted below. Although there are numerous tools that can be allocated to each stage of the proposed methodology, we have decided to select a set of tools, that have been found efficient and easy to handle in most of the scenarios encountered. The presented methodology combines a set of static and dynamic hunting tools to help investigators to easily follow a proactive approach for analyzing any suspicious binaries from different Windows systems. Beneficial patterns can easily be preserved, which can be utilized to facilitate the future classification of emerging threats.

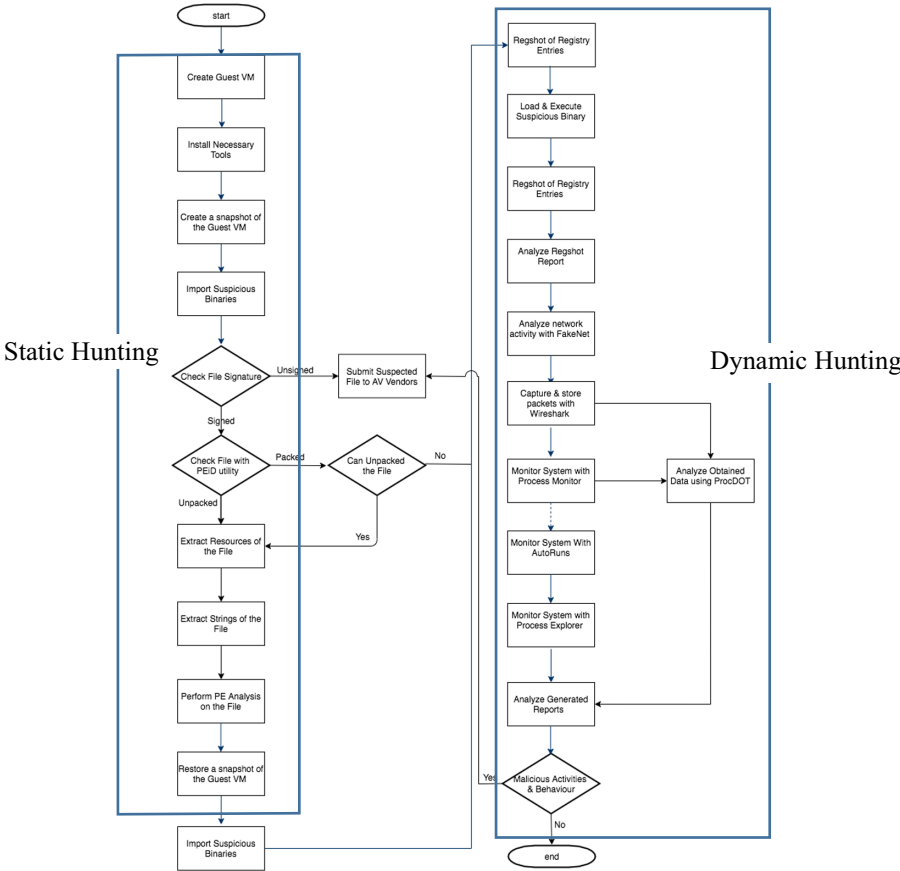


Fig. 1. Workflow of the proposed methodology

3.1 Selected Static Hunting Tools

A selected set of static hunting tools was employed to retrieve unique information from the binaries under investigation. Although there are multiple tools that can be utilized in the different stages of static hunting, only efficient tools and utilities that operate in a similar manner have been selected, and which can also extract useful data to complement other data generated by other selected tools as well. PE header extractors were used to collect header information of the portable executable samples, these tools are also capable of recognizing signatures of packers used to obscure the binary sample. The PEiD utility was used to identify the used packers, and compilers. The latest version of PEiD is equipped with detection algorithms that are capable of detecting more than 600 unique signatures of popular packers and compliers in binaries [23]. PE Tools is a lightweight executable control utility. It appeared for the first time in 2002 and contains a small set of tools such as process viewer and manager, HEX editor, EXE rebuilder and so on [24]. The PEview utility was utilized to retrieve a list of imported Windows APIs required to implement the binary. This will provide insights regarding the various functions inside the binary. for example, it can reveal if some functions within the binary were employed to manipulate the system's registry and file system. PEview can be used to inspect Windows 32-bit and 64-bit binaries to retrieve a list of imports required by its functions. It also presents the essential dependencies required to implement the binary as well as its debugging information and paths [25].

3.2 Selected Dynamic Hunting Tools

A selected set of dynamic hunting tools was employed to audit the behaviours of various processes and modules operating in the system environment. The process monitoring utility was utilized to check the behaviour of the current operating system. It can also track the behaviours of every main and child processes running in the system. It is also attainable for the process monitoring utility to document in real-time, file system manipulations, as well as changes to the registry. Process monitoring is a very useful utility in dynamic hunting. it would be beneficial to implement it in an isolated environment prior to executing the malicious binary, as it can be used to record system changes made by the malicious binary such as the filesystem or registry manipulation [26]. Process Explorer is another utility that has been utilized in dynamic hunting. It displays the currently active processes along with their profiles, called API and loaded DLLs. It is a valuable tool for detecting the type of APIs and DLLs are assigned to the binary under investigation to implement particular tasks [27]. ProcDOT utility [28] is used to take the output of Process Monitor and Wireshark, then visualize how a specific process is executed in the form of a flowchart. This present and interpret the collected pieces of data in order. This aids the investigator to know what has happened and in which order. Registry monitoring is an open-source utility that can be utilized to record a snapshot of the registry entries to be evaluated at a future stage with a second snapshot to highlight the modifications made to the registry entries due to certain events. It can be used to detect which updates in the registry entries have materialized when the malicious binary implemented [29]. FakeNet is a network emulation utility that has been used to mimic the communication infrastructure for tracking DNS requests and outbound IP connections. It is a valuable

tool in assessing the network activity of binaries under investigation. It was employed in dynamic hunting to monitor and track DNS requests, detect any dropper URIs, and classify malignant external IP connections to C&C services [30]. Wireshark is a network packet sniffing that has been employed to record the IP traffic of various modules. It has a broad variety of functions, such as live logging and off-line examination. It is a valuable tool for recording traffic information from binaries under investigation for future evaluation or for use with other tools as well [31].

3.3 Selected Dataset for Threat Hunting

It is important to assess the effectiveness of the proposed methodology in detecting malignant artifacts of misconduct in any binary file under investigation, these artifacts can be preserved later to facilitate the classification process related to emerging threats. It is imperative to make use of a dataset made up of different malware families that are known to have many variants. Since there are no common data sets for malware families, VirusTotal [32] was employed to gather our own sample set. Few malwares and their reports can be obtained using the VirusTotal API. Hence, a selected sample set was gathered and analysed for the suitability of the inclusion on a weekly basis for the period of 3 months. We believe that this is a fair amount of time to gather a sample of malicious software that reflects the different Windows malware families that are known to have many variants. The following malware families were selected for our sample set, which includes OnlineGames, Bifrose, Delf, Lmir, Zbot, Zlob, Banload, Vundo, Yimfoca, Farfli, Banker, Vapsup, Swizzor, PcClient, and NSAnti. Some of these malware families have mechanisms to evade specific analysis techniques, which have been determined to be an advantage for testing the accuracy of the proposed methodology in real-world situations.

4 Discussion on Static Hunting Tools

The analysis platform comprised an open-source hosted hypervisor for x86 virtualization entitled Oracle VM VirtualBox with a virtual network in host-only network setting. Two virtual machines were created, one is a Windows 8.1 machine, and the other is Windows 7 machine. Windows 8.1 was selected to permit the proper execution of various analysis tools required for the proposed research. Moreover, minimal configurations in the selected tools will successfully modify them to report the results properly. The following programs will be employed for the path of the required analyses: sigcheck, 7zip, strings, PEview, ultimate packer for EXcutables, fakenet, regshot, autoruns, process monitor, and process explorer. The previously mentioned programs have been copied on specified virtual machine then installed. The sample files were installed in the pair of Windows machines. In addition, these Windows machines were fully backed up and isolated from the internet to assure their digital hygiene. Finally, the network was configured to connect to the internet through an intermediary VM running Parrot OS.

On one sample file, the signature can be extracted with a utility named 'sigcheck'. Extracting the file signature from an executable file is a quick and easy way of determining whether the file is malicious. If the file is not digitally signed, it is a clear indication

that the executable has been modified. Running sigcheck against any benign file, will show legitimate information related to the digital signature of the file. However, running sigcheck against the sample file appearing to be a legitimate file indicates that the executable has not been digitally signed and is a strong indicator that the file may be malicious.

Malicious executables are usually packed by obfuscating their malicious payload, then combines the packed executable with deobfuscation code to create a self-extracting archive. When the obfuscated executable is executed, the deobfuscation code recreates the original code before executing it. Most obfuscation packers deobfuscate the original code into memory. As the obfuscated payload is not readable, packed executables are commonly used by malware to evade detection by security controls and to make it harder to be analysed. PEiD utility can be used to determine if obfuscation packer was used to obfuscate the executable, applying the PEiD utility on one of the sample files, obfuscation using *'ultimate packer for executables'* utility was detected. PEiD also shows other information such as the entry point and file offset of the original executable. The sample file can be deobfuscated using *'ultimate packer for executables'* utility.

To extract the sample file, an archiving program such as 7zip was used to display the file structure of the executable. Many archiving programs have the ability to extract the resources of an executable file. Using this technique on legitimate executables displays its structure. However, applying the same technique against one of the sample files, displays its contents. Continuing to extract these executables to a separate folder, facilitates displaying their resources. The extracted resources can further be analysed.

'Strings' is command-line utility used to extract the ASCII representation of the hex values within the sample file. Only ASCII strings that are found to be 3 consecutive ASCII characters or longer are displayed. While this technique does extract a lot of false-positive information, it is a simple way of quickly extracting information from an executable. Running the strings command on the sample file extracted previously provides 3360 matches. Quickly looking at the output shows that sample file is likely to be some sort of keylogger as shown in Fig. 2. Examining the output in detailed, a set of IRC commands can be found within the extracted strings. It can be suggested that the keylogger will periodically log into an IRC server and upload some log files. Other

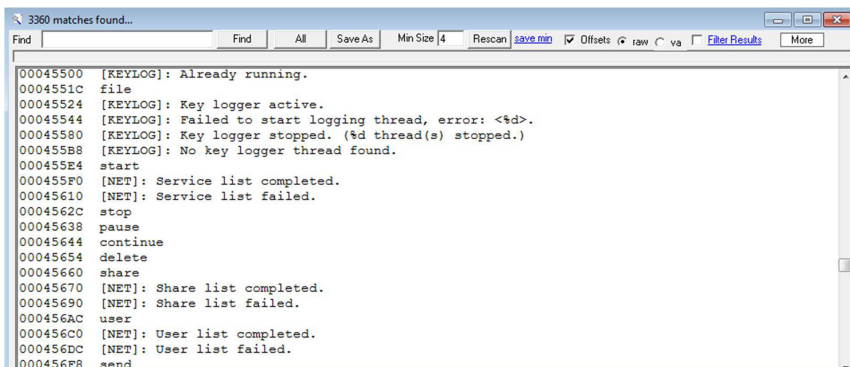
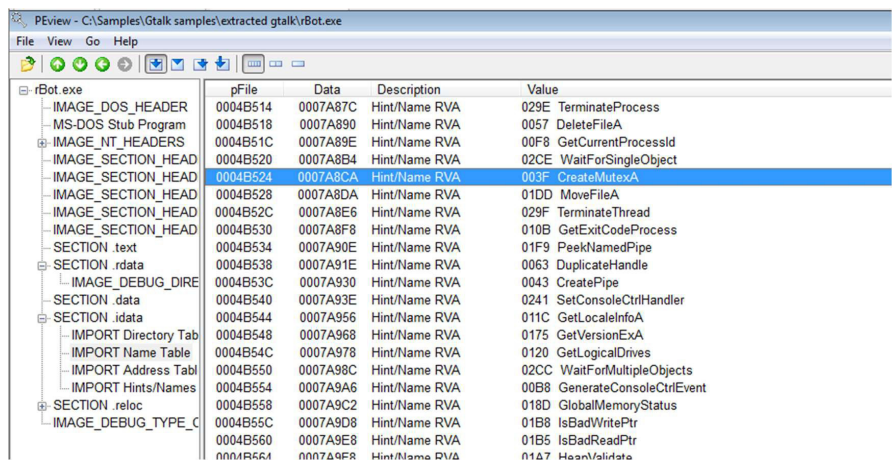


Fig. 2. Extracting the strings of the sample file

commands such as ‘*RegCreateKeyExA*’ were spotted as well, which suggests that a set of registry keys are installed to provide persistence on the system PE analysis can be performed using PView, that was used to extract the PE header information. Expanding the ‘.idata’ section and selecting the import name table, a list of imports can be displayed as seen in Fig. 3.



pFile	Data	Description	Value
0004B514	0007A87C	Hint/Name RVA	029E TerminateProcess
0004B518	0007A890	Hint/Name RVA	0057 DeleteFileA
0004B51C	0007A89E	Hint/Name RVA	00F8 GetCurrentProcessId
0004B520	0007A8B4	Hint/Name RVA	02CE WaitForSingleObject
0004B524	0007A8CA	Hint/Name RVA	003F CreateMutexA
0004B528	0007A8DA	Hint/Name RVA	01DD MoveFileA
0004B52C	0007A8E6	Hint/Name RVA	029F TerminateThread
0004B530	0007A8F8	Hint/Name RVA	010B GetExitCodeProcess
0004B534	0007A90E	Hint/Name RVA	01F9 PeekNamedPipe
0004B538	0007A91E	Hint/Name RVA	0063 DuplicateHandle
0004B53C	0007A930	Hint/Name RVA	0043 CreatePipe
0004B540	0007A93E	Hint/Name RVA	0241 SetConsoleCtrlHandler
0004B544	0007A956	Hint/Name RVA	011C GetLocaleInfoA
0004B548	0007A968	Hint/Name RVA	0175 GetVersionExA
0004B54C	0007A978	Hint/Name RVA	0120 GetLogicalDrives
0004B550	0007A98C	Hint/Name RVA	02CC WaitForMultipleObjects
0004B554	0007A9A6	Hint/Name RVA	00B8 GenerateConsoleCtrlEvent
0004B558	0007A9C2	Hint/Name RVA	018D GlobalMemoryStatus
0004B55C	0007A9D8	Hint/Name RVA	01B8 IsBadWritePtr
0004B560	0007A9E8	Hint/Name RVA	01B5 IsBadReadPtr
0004B564	0007A9F8	Hint/Name RVA	01A7 IsBadVolatile

Fig. 3. List of imports of the sample file

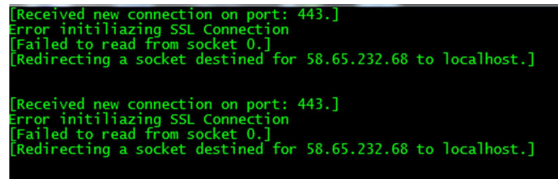
There are several imports that can be found in the sample files that could be interesting for further analysis as. For example, the *CreateMutexA* import is commonly used by malware to set a value in a predetermined location on disk, in order to not infect the same target multiple times. It is important to note that legitimate programs may use *CreateMutexA* to ensure that only a single instance of the program is running on the system to prevent conflicts. Other imports such as *SetHandleCount* and *SetStdHandle* are used to create a handle to a process and redirect input, which can be used to redirect keystrokes from the user to a file. Additionally, the *ReadProcessMemory* import is used to interact with victim processes and extract information.

5 Discussion on Dynamic Hunting Tools

In the dynamic hunting step, the selected sample files were executed 10 times, clean snapshots of the VMs were acquired before each execution plan. After completion, all VMs are restored to the saved snapshots to obtain immaculate states in every attempt. The analysis platform was prepared as stated in the subsection, with two virtual machines, one is running Windows 7 and the other runs Windows 8.1. Both are running on a host only network, through an intermediary VM running Parrot OS. The Windows VMs were prepared with these IP addresses 172.16.22.10/70, while the Parrot VM was prepared with the IP address 172.16. 22.40.

To simulate a network, the FakeNet utility was used to allow the analyst to observe the network activity of the sample file within a safe environment. When FakeNet runs, it

listens to multiple ports of multiple protocols. After running one of the sample files, an executable suspected to be a malware sample shows that a connection was attempted to IP address 58.65.232.68 on port 443 as shown in Fig. 4. Conducting a who-is lookup on the extracted IP address shows that it is an IP address originating from foreign network. An unsolicited outbound request to an external network suggests malicious activity and is a good indication that the executable is indeed a malicious binary.



```
[Received new connection on port: 443.]
Error initializing SSL Connection
[Failed to read from socket 0.]
[Redirecting a socket destined for 58.65.232.68 to localhost.]

[Received new connection on port: 443.]
Error initializing SSL Connection
[Failed to read from socket 0.]
[Redirecting a socket destined for 58.65.232.68 to localhost.]
```

Fig. 4. FakeNet shows an unsolicited outbound request

To extract and monitor network traffic on a specific interface, the Wireshark utility is used within a safe environment. The utility has the ability to capture webpages and traffic and then store this data in a file that dumps all network elements that were transferred while this utility was running.

Regshot utility was employed to determine which registry values were added, altered or modified by the malicious binaries. All unnecessary services and applications running on the VM have been terminated before running of RegShot. A registry snapshot was taken from the registry before running the malicious binaries. After, the complete run of the malicious binary, a second registry snapshot was taken. The output of the second registry snapshot is compared with the output of the previously taken snapshot to determine which registry entries were modified, as well as folders and files within the operating system that have been added, altered or modified.

An example of a report of changes generated by Regshot is provided in Fig. 5, which shows that two registry keys were deleted, 1346 keys were created, 13 files have been modified and 246 folders have been added. Examining this report in detail, it is possible to detect where malicious files have been installed and which registry entries have been maliciously modified.

AutoRuns utility was used as start-up observer to monitor and list all processes that are set to run on the start-up programs. Snapshots of the current system configuration can be made and saved to a file then later compared to previous snapshots to determine changes. After the complete run of the sample file, the saved snapshot can be compared to the current start-up programs.

Process monitor utility is an advanced monitoring tool for windows, it is a vital utility to understand the changing behaviours of any running binaries. It excessively offers different views of the running processes by swapping between different filters that can depict multiple insights about the binary file in runtime. The process monitor can analyse and watch in real-time each running process, created files, accessed registries entries, etc. The utility is equipped with various filters to search for patterns of interest, it also has the ability to filter out all other processes that can interfere with the required results. Process monitor can be used to record all operations completed by any running

```
-----
Keys deleted: 2
-----
HKLM\SYSTEM\ControlSet001\services\PROCMON23\Enum
HKLM\SYSTEM\CurrentControlSet\services\PROCMON23\Enum

-----
Keys added: 1346
-----
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\AutoplayHandlers\Handlers\VLCPplayCDaudioonArrival
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\AutoplayHandlers\Handlers\VLCPplayDVDaudioonArrival
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\AutoplayHandlers\Handlers\VLCPplayVCDmovieonArrival
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\AutoplayHandlers\Handlers\VLCPplayMusicFilesonArrival
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\AutoplayHandlers\Handlers\VLCPplayVCDmovieonArrival
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\AutoplayHandlers\Handlers\VLCPplayVCDmovieonArrival
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\AutoplayHandlers\Handlers\VLCPplayVideoFilesonArrival

-----
Files [attributes?] modified: 13
-----
C:\Users\sc1lentwolf\AppData\Local\Microsoft\Windows\History\History.IE5\index.dat
C:\Users\sc1lentwolf\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\index.dat
C:\Users\sc1lentwolf\AppData\Local\Microsoft\Windows\UsrClass.dat
C:\Users\sc1lentwolf\AppData\Local\Microsoft\Windows\UsrClass.dat.LOG1
C:\Users\sc1lentwolf\AppData\Roaming\Microsoft\Windows\Cookies\index.dat
C:\Users\sc1lentwolf\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\1b4dd6f729cb1962.automaticDestinations-ms
C:\Users\sc1lentwolf\NTUSER.DAT
C:\Users\sc1lentwolf\ntuser.dat.LOG1
C:\Windows\AppCompat\Programs\RecentFileCache.bcf
C:\Windows\ServiceProfiles\LocalService\AppData\Local\Microsoft\Windows\History\History.IE5\index.dat
C:\Windows\ServiceProfiles\LocalService\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\index.dat
C:\Windows\ServiceProfiles\LocalService\AppData\Roaming\Microsoft\Windows\Cookies\index.dat
C:\Windows\System32\LogFiles\Scm\eca24ff-236c-401d-a1e7-b3d5267b8a50

-----
Folders added: 246
-----
C:\Program Files (x86)\VideoLAN
C:\Program Files (x86)\VideoLAN\vlc
C:\Program Files (x86)\VideoLAN\vlc\locale
-----
```

Fig. 5. Report of changes generated by Regshot

processes. By setting process monitor to capture data and then running the malicious binary sample, all the performed operations can be captured. If the malicious payload is hidden within a legitimate installer, filters could be applied to filter out any unneeded process operations. The ‘process tree’ view can be selected where the legitimate process is displayed along with the malicious payload.

Process Explorer is a powerful utility for managing windows processes. It can be used to present insights about all the processes running in the system. Every running process is displayed in a tree-like structure that shows the relations between parent and child processes.

Process Explorer is considered an advanced task manager with enhanced features. Some of these features are the hierarchical coloured view of processes which simplify the analysis, the ability to identify input/output operation of a specific process such as opening or locking specific file or folder and loading a DLL, the ability to terminate or suspend any specific process tree with all of its spawned processes, and smooth integration with VirusTotal database that permits submitting and comparing cryptographic hashes of all running executables against those stored on VirusTotal then displays the number of detections without needing to isolate each executable file and then upload it separately for review. The granularity of information offered by process explorer can help the analyst to trace DLL versioning and memory leaks problems for any executable file. An example of an output generated by process explorer is provided in Fig. 6, a process name ‘newbos2.exe’ (that is a new child process of ‘explorer.exe’ process) has 47 out of 55 detections. This sharp detection rate confidently indicates that this executable is malicious. The Properties window of the malicious binary can provide additional useful information to the investigator such as, the character strings in memory, the user under whom the process is being executed, the active network connections, active threads, and the location of the executable pertaining to the process on the desk. In order to reduce the interpretation time for the investigator, the aggregated traces of process monitor and the collected network traffic dump can be visualized by ProcDOT as seen in Fig. 7. This application can generate a call graph that represents behavior of the sample under investigation by combining the previously collected data.

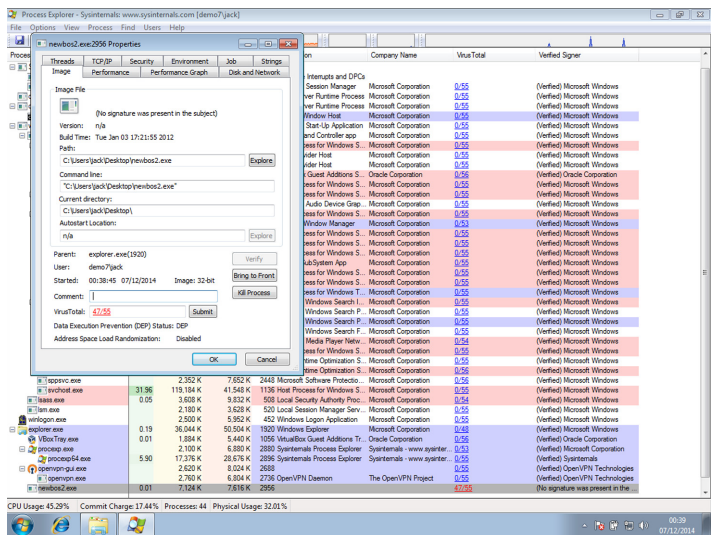


Fig. 6. An output generated by ‘Process Explorer’

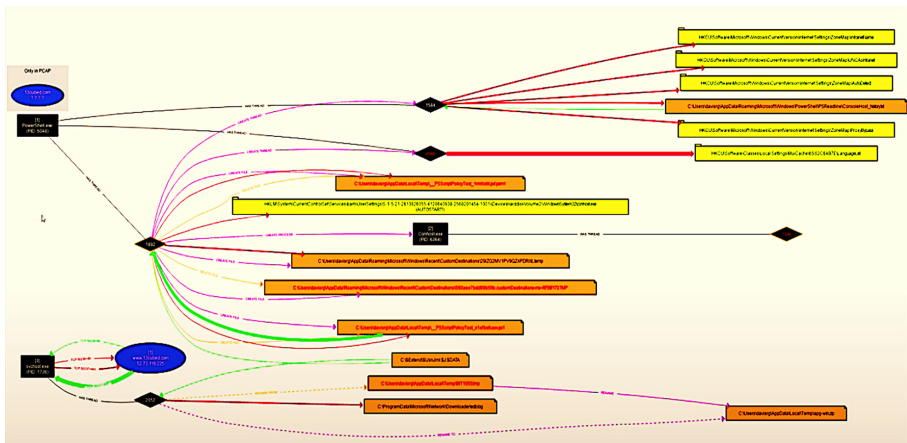


Fig. 7. Call Graph Generated by ‘ProcDOT’

6 Conclusions and Future Work

This research presents our efforts to propose a methodology to distinguish malicious binaries from benign binaries in a quick and efficient manner. The proposed methodology consists of static and dynamic hunting techniques. Using these two techniques, the proposed methodology is not only capable of identifying a range of signature-based anomalies but also to pinpoint behavioural anomalies that arise in the operating system when malicious binaries are triggered. The proposed threat hunting methodology was

applied to samples of malicious binaries, which can be found in common malware repositories. The findings presented in this paper can be used to construct other methodologies and incident response plans for other emerging threats. Full reverse engineering of the malicious binary can be extremely beneficial. However, it is not feasible to do this for all binaries on the internal systems. Additionally, there are specific skill sets required to properly perform reverse engineering for a binary. More research is required on automatic detection of advanced malware using artificial intelligence techniques. Future research will include automated approaches to uncover various concealment algorithms and evasion methods used by malware developers. The integration of Cuckoo Sandbox with the proposed methodology should also be an area of investigation.

References

1. Dowdy, J.: The cyber-security threat to us growth and prosperity. In: *Cyberspace: A New Domain for National Security* (2012)
2. Friedberg, I., Skopik, F., Settanni, G., Fiedler, R.: Combating advanced persistent threats: from network event correlation to incident detection. *Comput. Secur.* **48**, 35–57 (2015)
3. Connolly, L.Y., Wall, D.S.: The rise of crypto-ransomware in a changing cybercrime landscape: taxonomising countermeasures. *Comput. Secur.* **87**, (2019)
4. Lord, N.: What is threat hunting? The emerging focus in threat detection. In: *Digital Guardian* (2018)
5. Sqrrl. Cyber Threat Hunting. www.sqrrl.com
6. Bhatt, P., Yano, E.T., Gustavsson, P.: Towards a framework to detect multi-stage advanced persistent threats attacks. In: *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pp. 390–395 (2014)
7. Scarabeo, N., Fung, B.C., Khokhar, R.H.: Mining known attack patterns from security-related events. *Peer J. Comput. Sci.* **1**, (2015)
8. Mahyari, A.G., Aviyente, S.: A multi-scale energy detector for anomaly detection in dynamic networks. In: *2013 Asilomar Conference on Signals, Systems and Computers*, pp. 962–965. IEEE (2013)
9. Miller, B.A., Beard, M.S., Bliss, N.T.: Eigenspace analysis for threat detection in social networks. In: *14th International Conference on Information Fusion*, pp. 1–7. IEEE (2011)
10. Bhardwaj, A.K., Singh, M.: Data mining-based integrated network traffic visualization framework for threat detection. *Neural Comput. Appl.* **26**(1), 117–130 (2015)
11. Gu, G., Perdisci, R., Zhang, J., Lee, W.: Botminer: clustering analysis of network traffic for protocol-and structure-independent botnet detection (2008)
12. Elmisery, A.M., Sertovic, M.: Privacy preserving threat hunting in smart home environments. In: Anbar, M., Abdullah, N., Manickam, S. (eds.) *Advances in Cyber Security (ACeS 2019) Communications in Computer and Information Science*, vol. 1132, pp. 104–120. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-2693-0_8
13. Elmisery, A.M., Botvich, D.: Privacy aware recommender service using multi-agent middleware-an IPTV network scenario. *Informatica* **36**(1) (2012)
14. Elmisery, A.M., Rho, S., Botvich, D.: A fog based middleware for automated compliance with OECD privacy principles in internet of healthcare things. *IEEE Access* **4**, 8418–8441 (2016)
15. Elmisery, A.M., Rho, S., Botvich, D.: Collaborative privacy framework for minimizing privacy risks in an IPTV social recommender service. *Multimedia Tools Appl.* **75**(22), 14927–14957 (2016)

16. Elmisery, A.M., Botvich, D.: Enhanced middleware for collaborative privacy in IPTV recommender services. *J. Converg.* **2**(2), 10 (2011)
17. Elmisery, A.M., Doolin, K., Roussaki, I., Botvich, D.: Enhanced middleware for collaborative privacy in community based recommendations services. In: Yeo, S.S., Pan, Y., Lee, Y., Chang, H. (eds.) *Computer Science and its Applications. Lecture Notes in Electrical Engineering*, vol. 203, pp. 313–328. Springer, Dordrecht (2012)
18. Berrueta Irigoyen, E., Morató Osés, D., Lizarrondo, M., Izal Azcárate, M.: A survey on detection techniques for cryptographic ransomware. *IEEE Access* **7**, 144925–144944 (2019)
19. Akbanov, V.G., Vassilakis, I.D. Moscholios, Logothetis, M.D.: Static and dynamic analysis of WannaCry ransomware
20. Aman, W.: A framework for analysis and comparison of dynamic malware analysis tools. *Int. J. Netw. Secur. Its Appl.* **6**(5), 63–74 (2014). arXiv preprint [arXiv:1410.2131](https://arxiv.org/abs/1410.2131)
21. Wichmann, B.A., Canning, A., Clutterbuck, D., Winsborrow, L., Ward, N., Marsh, D.: Industrial perspective on static analysis. *Softw. Eng. J.* **10**(2), 69–75 (1995)
22. Firdausi, I., Erwin, A., Nugroho, A.S.: Analysis of machine learning techniques used in behavior-based malware detection. In: 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies, pp. 201–203. IEEE (2010)
23. Snaker (ed.): Softpedia (2008). <https://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/PEiD-updated.shtml>
24. Petoolse (ed.): Github (2018). <https://github.com/petoolse/petools>
25. Miller, S. (ed.): Dependency walker (2015). <http://www.dependencywalker.com>
26. Microsoft (ed.): Process explorer (2019). <https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer>
27. Microsoft (ed.): Process monitor. <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon>
28. Wojner, C. (ed.): ProcDOT, a new way of visual malware analysis. Austrian National CERT (2015). <https://www.procdot.com/>
29. Maddes, X. (ed.): Regshot download (2018). <https://sourceforge.net/projects/regshot/>
30. Hungenberg, T., Eckert, M. (ed.): INetSim: internet services simulation suite (2013)
31. Wireshark, F.: Wireshark-Go Deep, vol. 15. Retrieved Oct 2011
32. Sistemas, H. (ed.): VirusTotal (2004). <https://www.virustotal.com/gui/>