

MEMORY FORENSICS USING VOLATILITY FOR STUXNET MALWARE

Kabeer Ahmed, Rehan Mumtaz, Wajahat Ahmed, Moiz Tariq

Department of Software Engineering

NED University of Engineering and Technology

Abstract - This paper explores the potential of memory forensics in analyzing the Stuxnet worm. Using the memory analysis tool Volatility, the processes and network connections of the Stuxnet malware were extracted from a memory dump. The results showed that Volatility is an effective tool for extracting and analyzing information from memory dumps. Furthermore, the paper highlights the importance of memory forensics in the field of digital forensics and explains how it can help in the identification and neutralization of cyber threats. Moreover, this paper provides a brief overview of the Volatility tool and explains how to install and use it. Finally, the paper highlights the work of Michael Hale Ligh in the field of memory forensics and the Volatility Foundation.

Index Terms - Stuxnet, Memory dump, Malfind, DLLs, Registry keys.

INTRODUCTION

Memory forensics is an important tool for digital forensic investigators, as it allows them to analyze a computer's memory in order to uncover evidence of malicious activity. In this blog post, we will explore the use of Volatility on Stuxnet malware and how it can be used in memory forensics investigations.

Stuxnet is a sophisticated piece of malware that was discovered in 2010 and has been linked to attacks against Iranian nuclear facilities. It operates by exploiting vulnerabilities within industrial control systems (ICS) software running on Windows operating systems, allowing attackers to gain access and manipulate the system's operations without detection. As such, Stuxnet poses great risks for organizations who operate ICS-based networks or rely upon ICS components within their infrastructure; making its analysis essential for those tasked with defending these networks from attack or investigating incidents related thereto.

Volatility is an open source framework designed specifically for analyzing volatile memory dumps taken from compromised machines; thereby providing analysts with information about what processes were running at any given time during the course of infection as well as other artifacts left behind by malicious actors which may provide further insight into their activities after gaining access onto a machine via Stuxnet exploitation .[8] The tool works by parsing through raw data stored within RAM using various plugins that are tailored towards specific tasks such as extracting process lists , network connections , DLLs loaded into each process etc., thus allowing investigators greater visibility when attempting to understand what occurred prior/during/after infection so they can take appropriate action accordingly .

By leveraging Volatility when performing analysis on suspected cases involving StuxNet infections , security professionals have better chances at identifying potential indicators associated with this type threat ; thereby enabling them not only detect but also prevent future occurrences thereof should similar patterns arise elsewhere across their environment(s). Furthermore due its wide range capabilities & compatibility across multiple versions Windows Oses – including both 32bit & 64bit architectures – makes volatility even more attractive choice over traditional static methods like disassembly/decompilation etc.. This versatility helps ensure maximum coverage no matter which platform being investigated ensuring all relevant information gathered quickly efficiently possible manner minimizing disruption normal business activities while doing so .

The paper will describe the memory analysis process and explain the importance of the order of volatility. It will discuss the use of memory forensics to extract artifacts from memory dumps

and explain the importance of memory forensics in the field of digital forensics.

It will also discuss the significance of the Stuxnet malware in the context of industrial control systems and the techniques used by Stuxnet to spread across networks. The paper will provide an in-depth analysis of the use of memory forensics to analyze the Stuxnet malware, and provide an overview of the memory forensics field as a whole.

It will discuss the importance of creating a memory forensics strategy, and provide practical tips for memory forensic investigations. Dealing with stuxnet-related attacks will be the focus of this paper.

PROBLEM STATEMENT

The problem could be examined under: “What evidence does memory forensics with Volatility uncover when examining Stuxnet malware?” Through our research into this topic, we hope to gain insight into how effective these techniques are for detecting malicious activity related specifically to Stuxnet infections so that security professionals may better protect themselves against similar threats in future cases. It is to examine how memory forensics can be used to analyze the Stuxnet malware, with a focus on the use of the Volatility suite. The paper will explore the effectiveness of memory forensics in detecting Stuxnet and its components, and will provide an overview of the memory forensics field as a whole. It will discuss the different memory acquisition methods and the best memory forensic tools on the market, and provide examples of how these tools can be used to capture and analyze memory dumps. The paper will highlight the importance of memory forensics in the field of digital forensics and discuss how it can help in the identification and neutralization of cyber threats.

BACKGROUND STUDY

Memory forensics is a technique used to examine the contents of a computer's memory in order to extract information about the system's state at the time of acquisition. This information can be used

to identify and track malware, as well as understand its behavior and persistence on a system. Memory forensics is becoming increasingly important as advanced malware is developed to evade traditional detection methods.

One of the most well-known and sophisticated malware to have been discovered is the Stuxnet worm. The Stuxnet malware was first discovered in 2010 and was found to have been used to target specific industrial control systems. The malware was designed to exploit a vulnerability in the Windows operating system, and was able to propagate itself to other systems on a network.

The Stuxnet malware was notable for its advanced capabilities, including the use of a rootkit to hide its presence on a system and the ability to propagate itself to other systems on a network. Additionally, the malware was found to be targeting specific industrial control systems, which suggests that it was designed for a specific purpose. The malware's intended targets were specifically the centrifuges used in the Iranian nuclear program, and it was able to cause damage to them by manipulating the speed of the centrifuges.

The analysis of the Stuxnet malware is a complex task, and several tools have been developed to aid in the process. One of the most popular open-source memory forensics tools is Volatility, which can be used to analyze memory dumps from Windows and Linux systems. Volatility is a command-line tool that can be used to extract a wide range of information from memory dumps, including process listings, network connections, and registry keys. It is also able to extract data from the memory of a live system, which makes it a powerful tool for incident response.

The Volatility tool can be used to identify the presence of malware in memory, as well as extract information about the malware's behavior and persistence on a system. This can include information about the malware's network-related capabilities and targeting of specific industrial control systems.

The use of memory forensics and the Volatility tool allows for the identification of malware that may not be detected by traditional detection methods. This is important because advanced malware such as Stuxnet is often able to evade traditional antivirus software.

The use of memory forensics also allows for the extraction of information about the malware's behavior and persistence on a system, which can be used to improve the detection and response to similar threats in the future. Additionally, the ability to extract information about the malware's network-related capabilities and targeting of specific industrial control systems can be used to understand the malware's intended purpose and potential targets.

In this research paper, we will investigate the use of memory forensics and the Volatility tool in analyzing the Stuxnet malware. The goal of this research is to understand the malware's behavior and persistence on a system, as well as its network-related capabilities and targeting of specific industrial control systems. Additionally, we will investigate the Volatility tool's ability to extract information from the memory of a live system, which makes it a powerful tool for incident response.

As the use of advanced malware such as Stuxnet becomes more prevalent, the importance of memory forensics in incident response and threat hunting will continue to grow. Understanding the behavior and persistence of such malware is crucial for developing effective detection and response methods. This research aims to contribute to the field by investigating the use of the Volatility tool in analyzing the Stuxnet malware

RELATED WORK

Hacking is a serious issue in today's environment. Hackers can do it for pleasure or for nefarious purposes. In 2008, the FBI estimated that Internet fraud had cost 264.6 million dollars. Investigating computer crime is therefore one of the most difficult tasks at hand today. The efficiency of the memory acquisition

instrument is a key factor in memory acquisition success. In their article, memory forensics methods are compared in terms of processing speed and residual artifacts in volatile memory. In addition, they looked at how different volatile memory sizes affect the processing times of the tools. They employ the following tools to carry out their work: FTK Imager, Pro Discover, Nigella32, Helix3(dd), OSForensics, and Belkasoft RAM Capturer were among the forensic tools whose performance was assessed. According to the findings, Belkasoft RAM Capturer processed data the quickest and produced the fewest artifacts overall. With a 95% confidence level, the study also discovered that there were substantial differences between the analyzed tools' residual artifacts in volatile memory.[1]

Memory forensics is a technique that includes looking through a computer's memory dump to find out details about the user's activities. The memory of the computer, sometimes referred to as volatile memory or random access memory, holds a multitude of system data, including information on network connections, process activity, and more. This data may be utilized to understand the user's behavior and perhaps even turn up proof of any nefarious activities. Criminals frequently use technology that avoids recording any evidence on permanent storage media and instead launches their assault through volatile memory due to the widespread adoption of digital forensics for investigation. Memory forensics is therefore acknowledged as a component of incident response procedures for inquiry and it is continually changing. The Onion Router (Tor), live CD/USB, portable browsers, virtualization, and other online crimes were highlighted as we reviewed various memory capture tools and approaches. The goal was to examine the development of the memory forensics framework currently in use for cases involving the dark web and anonymous networks, as well as to identify the difficulties currently facing investigators of these types of cases.[2]

Authors performed the comparisons of tools in memory forensics because effectiveness of the memory acquisition tool is a major factor in memory acquisition success. In this paper, memory forensics methods are compared in terms of processing speed and residual artifacts in volatile memory. In addition, research was done to see how different volatile memory sizes affected how well forensic tools worked. FTK Imager, Pro Discover, Nigella32, Helix3(dd), OSForensics, and Belkasoft RAM Capturer were the tools employed in the investigation. According to the results, Belkasoft RAM Capturer processed data the quickest and left the

fewest artifacts behind. The study also came to the 95% confidence level conclusion that there are substantial variations between the tested tools in terms of residual artifacts in volatile memory. The study also discovered that the processing speed of the tools is unaffected by an increase in memory size.[3]

Volatility is a framework that is used worldwide to analyze the RAM of computers by many investigators. Currently, The Volatility Framework, a programme with a command line interface alone, is given a graphical user interface (GUI) and enhancements in this study. The program will be more approachable and user-friendly for investigators thanks to its GUI and extensions, which also bring more capability. The additional capabilities allow for the database-based storage of findings, the creation of shortcuts for difficult Volatility Framework actions, and the development of new commands based on database data correlation.[4]

For simple malware, conventional techniques for studying memory forensics may work, but not for sophisticated malware. Because it is more productive, this article recommends a virtual machine introspection approach as a potential fix. Microsoft Office files, portable document format files, and executable files have been reported to have a detection rate of up to 90% when using memory forensics. However, because network and process activity disappears fast in volatile memory, the detection rate of script files is just 75%. By utilizing memory forensics timing, frequency modification, and other heuristic techniques, this study offers a solution to the issue at hand. This approach, which is an agentless solution that supports several VM hypervisor types, solves the issue of excessive dependence on VM hypervisor types.[5]

Traditional procedures are efficient in finding and analyzing dead forensics and common computer forensic techniques, but they are unable to find live forensics, which can, in comparison, yield a lot more information. Malware that operates fully in RAM or memory has the ability to steal sensitive data like passwords, encryption keys, and network activities and is challenging to detect or nearly impossible to stop. Using keywords and default hex values, a signature-based artifact identification approach is put forward in this work. Investigators can effectively locate many possible artifacts with less traces on the physical hard drive by employing memory forensics.[6]

In paper [7] author focused on analyzing the Stuxnet worm in the Metasploit framework would

likely cover the technical details of how the malware works, its command and control infrastructure, and its propagation methods. Additionally, it would also include an analysis of the potential motivations behind the creation and deployment of the malware and the impact of it, as well as a discussion of the broader implications of the incident for cyber security. Metasploit is an open-source framework that is widely used for penetration testing and vulnerability assessment. It includes a collection of tools and modules that can be used to exploit known vulnerabilities, as well as a powerful scripting language for creating custom payloads and modules.

Stuxnet is a highly advanced and sophisticated computer worm that was discovered in 2010 and is believed to have been used to attack industrial control systems (ICS) in Iran's nuclear program. The malware was specifically designed to target Siemens Supervisory Control and Data Acquisition (SCADA) systems, which are widely used in critical infrastructure such as power plants and factories.

Research papers on Stuxnet worm analysis in Metasploit covers the technical details of how the malware works, its command and control infrastructure, and its propagation methods. They also include a discussion of the broader implications of the incident for cyber security, as well as an analysis of the potential motivations behind the creation and deployment of the malware.

Additionally, the papers provide a detailed analysis of how the Metasploit framework can be used to analyze the Stuxnet worm and its behavior, by using the different modules and tools provided by Metasploit. This includes the use of the Metasploit's payloads and payload modules, as well as its exploitation and post-exploitation modules, to study the worm's behavior. Also, they would provide an overview of the advantages of using Metasploit as a tool for analyzing the worm and its impact on the smart grid systems security.

In general, stuxnet worm analysis in Metasploit covers the various techniques, tools and methodologies used to analyze the worm and its behavior, the advantages of using Metasploit as a tool for analyzing the worm, and the future research directions and recommendations in this area[7]

One of the key findings of the paper is that the Stuxnet malware was specifically designed to target industrial control systems (ICS) and supervisory control and data acquisition (SCADA) systems. Langner argues that the use of such a sophisticated and targeted malware in an industrial setting is a significant departure from traditional cyber attacks, which have typically targeted general-purpose computer systems.[9] The paper provides evidence that the malware was specifically designed to target the specific type of programmable logic controllers (PLCs) used in the Iranian nuclear program, and was able to manipulate the PLCs in such a way as to cause physical damage to the centrifuges used in the nuclear enrichment process.

















The paper also notes that the Stuxnet malware was able to evade detection for a significant period of time, and argues that this is a testament to the malware's advanced capabilities. Langner states that the malware was able to evade detection by using a number of different techniques, such as using a complex and highly-encrypted command and control infrastructure, and using a number of different zero-day vulnerabilities in order to propagate itself.

The paper presents a well-researched and well-argued case for the use of shadow memory as a technique for memory analysis in commodity operating systems. The authors also demonstrate the effectiveness of MACE in detecting memory-related security vulnerabilities, and provide a clear roadmap for future research in this area. One limitation of the paper is that it focuses solely on the Windows operating system, and it would be beneficial to see similar studies done on other operating systems such as Linux and macOS. Additionally, the authors do not provide a detailed evaluation of the performance overhead introduced by MACE, which would be important for practical implementation. Overall, the paper provides a valuable contribution to the field of memory analysis and highlights the importance of developing robust and high-coverage memory analysis tools for commodity operating systems.[10]

COMPARATIVE ANALYSIS

S No	Paper Title	Technique Proposed	Issue Highlighted	Proposed Architecture	Security Schemes Applied
1	Memory forensics tools: Comparing processing time and left artifacts on volatile memory	Tools examined based on processing time and memory size	With the increasing demand in forensics to do the comparison of tools which is best for memory forensics	The positive points of this is we found the details of different tools that how they behave with different artifacts of memory. Negative Point is they perform in windows 7 using 1GB memory	This study does not propose or use any network security scheme.
2	Simplifying RAM Forensics: A GUI and Extensions for the Volatility Framework	Create GUI for volatility CLI with extensions to make its features more enhanced	CLI is difficult for investigators, GUI will provide easiness to the user with more functionality like <u>store data in database</u> , <u>create shortcuts for volatility framework</u> .	The positive point is volatility will be easy to use for everyone, Negative point is the core process and architecture of volatility will be not known to <u>user</u> which can irritate <u>user</u> when error comes.	This study does not propose or use any network security scheme.
3	Memory Forensics Using Virtual Machine Introspection for Malware Analysis	A Virtual Machine introspection technique is suggested	Traditional techniques are effective with files such as Microsoft Word etc but not with script files. This technique addresses this problem.	The positive point is that analysis is more thorough, detailed and accurate than with traditional methods. The negative point is <u>that expensive setup</u> is required to execute it.	This study does not propose or use any network security scheme.
4	Signature based volatile memory forensics	A Signature based artifact identification method is used to detect memory or RAM data	Some malwares store and read data from RAM and are harder to detect for forensic analysis	The positive point is <u>a memory</u> forensic technique discussed in this paper is better than conventional forensic tools. Negative point is more research is required.	This study does not propose or use any network security scheme.
5	Memory Forensics Analysis for Investigation of Online Crime - A Review	Importance of physical memory forensics is obvious but at the same time, it is also clear that	Online crimes where live CD/USB, portable browsers, virtualization, The Onion Router (Tor) is involved.	The objective was to study the existing growth of the memory forensics framework for investigation of cases involving dark web and anonymous <u>network</u> and find out the existing challenges in investigation of such cases.	This study does not propose or use any network security scheme.

We took two dump and start our analysis :
 One with normal dump of RAM, one with injected malware dump
 We organized the result in form of chart as shown

Memory Attributes	Normal Memory Dump	Malware Affected Memory Dump
Suspicious Isass.exe Process		
Calling logon.exe with LS process as Parent ID.		
Unlink DLL		
Malicious Remote Connection		
Injected Code		
Detecting API Calls		
Malicious Drivers		
Scanned Hash on Virus Total		

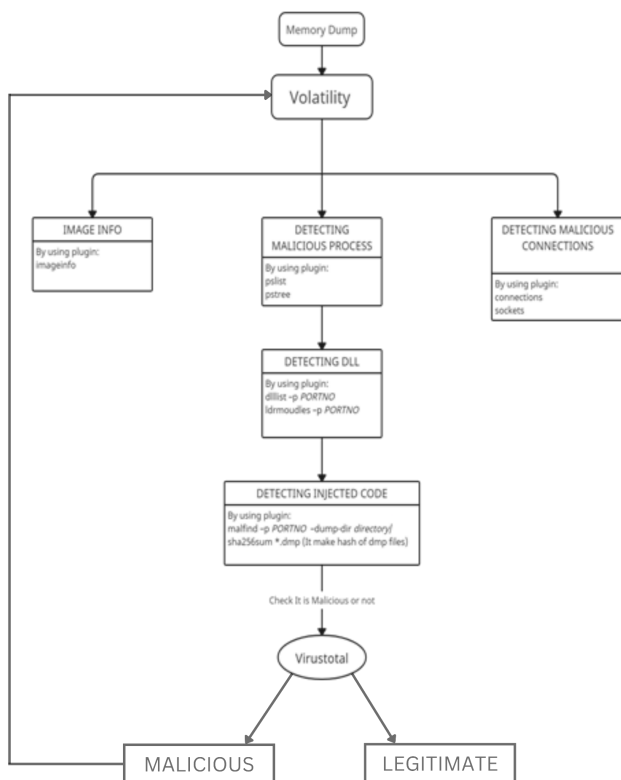
METHODOLOGY

Data collection: We obtained a sample of the Stuxnet malware and acquired an infected virtual machine image for analysis. The sample and virtual machine image were acquired from known sources such as malware repositories and security researchers.

Memory acquisition: We used the Volatility framework to acquire the memory dump of the infected virtual machine. The Volatility framework was run on a separate host machine and the memory dump was acquired via the physical memory acquisition method.

Memory analysis: Using Volatility plugins, we analyzed the memory dump to extract information

about the malware and its actions on the system. This included identifying and analyzing malicious processes, network connections, and file system activity. Specifically, we used the following plugins: pslist, pstree, psscan, netscan, connscan, and dlllist.



It gives a clear visual depiction of how it is affecting the entire architecture. In this way we can see that first the memory dump file is entered into the volatility tool that provides different types of plugins which is useful for the whole investigation of the memory dump file. Here we used imageinfo plugin to show the information of the file and, connections and sockets to detect malicious connections of the file. The plugins pslist and pstree are used to detect the malicious process in the file. After that we used dllist to find dll files for a particular process and ldrmodules to show the details of that dll. After that using malfind the infected code is saved in the directory after that we make hashes of all the files in the directory then by using VirusTotal website we check the file using their hashes that the file is malicious or not. By using so we identified malicious API calls, further detected abused registry for malicious code behavior in our memory dump

Results and discussion: We present our findings on the Stuxnet malware's actions and its impact on the system, as well as discuss the implications of our analysis for memory forensics and incident response. Our findings were validated by comparing

them with previous research on the same topic. The results were presented in tables, figures and graphs for better understanding and were discussed in the light of existing literature on the topic.

RESULT AND ANALYSIS

Memory forensics is a technique used to examine the contents of a computer's memory in order to extract information about the system's state at the time of acquisition. This information can be used to identify and track malware, as well as understand its behavior and persistence on a system. Memory forensics is becoming increasingly important as advanced malware is developed to evade traditional detection methods.

We initiated identifying malicious processes running before taking a memory dump. Since there is a lot to look for i focused on looking specifically at *lsass.exe*, able to obtain malicious workout present other than the normal process of lsass

1872	856	wmiprvse.exe	0x81fa5390
868	668	lsass.exe	0x81c498c8
1928	668	lsass.exe	0x81c47c00
968	1664	cmd.exe	0x81c0cda0 0
304	968	lsass.exe	0x81f14038

Obtaining the process id make our suspicion true as we dig more into it. The "lsass.exe" with Pid 868 and 1928 was started by the "services.exe" process. It isn't normal behavior. They could be malicious processes. We have just discovered two suspicious processes.

```

> sudo vol -f stuxnet.vmem windows.pstree | egrep '(services.exe|lsass.exe|winlogon.exe)'
** 624ss376100.0winlogon.exe 0x81da5650 19 570 0 False 2010-10-2
*** 680 624 lsass.exe 0x81e70020 19 342 0 False 2010-10-2
*** 668 624 services.exe 0x82073020 21 431 0 False 2010-10-2
**** 868 668 lsass.exe 0x81c498c8 2 23 0 False 2
**** 1928 668 lsass.exe 0x81c47c00 4 65 0 False 2
  
```

Best approach is to narrow down your scope as it helps to maintain your research and look better and dig in a prosperous way, so we picked down the process id **1928**. We started looking for dlls attached to that process. It has contacted these following unlinked dlls which is not normal.

```

> vol -f stuxnet.vmem ldrmodules --pid 1928
Volatility 3 Framework 2.4.1
Progress: 100.00 PDB scanning finished
Pid Process Base InLoad InInit InMem MappedPath
1928 lsass.exe 0x00000000 False False False N/A
1928 lsass.exe 0xc0000000 True True True \WINDOWS\system32\ntdll.dll
1928 lsass.exe 0x10000000 True False True N/A
1928 lsass.exe 0x87000000 True True True N/A
1928 lsass.exe 0x7c000000 True True True \WINDOWS\system32\kernel32.dll
1928 lsass.exe 0x77d00000 True True True \WINDOWS\system32\advapi32.dll
  
```

Furthermore, we look at the injected code and stunned by uprising to see api-call-hooks which were doing the following

- Decrypt the configuration data used by the threat
- Drop two .sys files and install them as a kernel level rootkit
- Access files created by the Siemens Step 7 software package
- Update itself
- Drop more .dll and .dat files
- Infect removable drives with custom .lnk files
- Inject into the lsass.exe process and execute custom code
- Inject into the iexplore.exe process
- Check if certain antivirus applications are running
- Scan the network for servers
- Remove itself
- Communicate with the C&C server

```
Process: svchost.exe Pid: 940 Address: 0xbf0000
Vad Tag: Vad Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6
0x00bf0000 90 06 bf 00 c6 07 bf 00 24 00 bf 00 a5 04 00 00 .....$.
0x00bf0010 f2 04 bf 00 48 06 00 00 c9 04 bf 00 29 00 00 00 ....H.....)
0x00bf0020 00 00 b7 00 e8 13 00 00 00 5a 77 4d 61 70 56 69 .....ZwMapVi
0x00bf0030 65 77 4f 66 53 65 63 74 69 6f 6e 00 5a 51 81 c1 ewOfSection.ZQ..
```

These calls are directly linked to the Stuxnet worm. Moreover, we move towards fiddling with malicious drivers embedded in the operating system to abuse it

```
> ./Vol -f stuxnet.vmem modscan | grep "mrx"
Volatility Foundation Volatility Framework 2.6
0x000000001c2a530 mrxnet.sys 0xb21d8000 0x3000 \\?\C:\WINDOWS\system32\Drivers\
0x00000000218cb60 mrxcls.sys 0xf895a000 0x5000 \\?\C:\WINDOWS\system32\Drivers\
```

In the last stage of the analysis, we marched to registry keys, examination of registry keys is an important step in identifying the presence of the Stuxnet malware and understanding its behavior and persistence on a system. One of the registry keys that is commonly associated with the Stuxnet malware is the "**HKLM\Software\Microsoft\Windows\CurrentVersion\Run**" key. This key is used by the Windows operating system to automatically run programs during startup. The Stuxnet malware is known to add a value to this key in order to run the malware's malicious code automatically when the system is started.

Another key that is commonly associated with the Stuxnet malware is the

"**HKLM\System\CurrentControlSet\Services**" key. This key contains information about the services that are installed on the system. The Stuxnet malware is known to create a new service in this key in order to maintain persistence on the infected system.

```
-f stuxnet.vmem printkey -K 'ControlSet001\Services\MrxNet'
Volatility Foundation Volatility Framework 2.6
(S) = Stable (V) = Volatile
Path: \Device\HarddiskVolume1\WINDOWS\system32\config\system
Name: MRxNet (S)
Created: 2011-06-03 04:26:47 UTC+0000
Type:
Description : (S) MRXNET
DisplayName : (S) MRXNET
ErrorControl : (S) 0
Group : (S) Network
ImagePath : (S) \\?\C:\WINDOWS\system32\Drivers\mrxnet.sys
Start : (S) 1
Type : (S) 1
```

```
-f stuxnet.vmem printkey -K 'ControlSet001\Services\MrxCls'
Volatility Foundation Volatility Framework 2.6
(S) = Stable (V) = Volatile
Path: \Device\HarddiskVolume1\WINDOWS\system32\config\system
Name: MRxCls (S)
Created: 2011-06-03 04:26:47 UTC+0000
Type:
Description : (S) MRXCLS
DisplayName : (S) MRXCLS
ErrorControl : (S) 0
Group : (S) Network
ImagePath : (S) \\?\C:\WINDOWS\system32\Drivers\mr
Start : (S) 1
Type : (S) 1
Data : (S)
000 8f 1f f7 6d 7d b1 c9 09 9d cc 24 7a c6 9f fb 23 ...m}....
010 90 b8 9d bf f1 d4 51 92 2a b4 1f 6a 2e a6 4f b3 .....Q..
020 cb 69 7c 0b 92 3b 1b c0 d7 75 17 a9 e3 33 48 dc .1|...;...
030 ad f6 da ea 2f 87 10 c4 21 81 a5 75 68 00 2e b1 ...../...
040 c2 7b eb dd bb 72 47 dc 87 91 14 a5 f3 c4 32 b0 {...rG...
050 cc 93 38 36 6b 49 0a f2 6f 1f 1d a1 4a 15 05 80 ...86kI..
060 4b 13 a8 aa 82 41 4b 89 dc 89 24 a2 ed 16 37 f3 K...AK...
070 42 a9 a0 6a 7f 82 cd 90 e5 3c 49 cc b2 97 ca cb B...j....
080 7b 64 c1 48 b2 4c f5 ae 54 42 74 0f 00 31 fd 80 {d.H.L..
```

Lastly we take a hash of a dump of injected code and scan with **VirusTotal-API** to analyze the behavior of what the global sources have to say.

Ad-Aware	① Win32.Worm.Stuxnet.D
Alibaba	① Worm:Win32/Stuxnet.92c45335
Antiy-AVL	① Worm/Win32.Stuxnet
Avast	① FileRepMalware [Trj]
Avira (no cloud)	① WORM/Stuxnet.ywlzs

Same method we go for suspicious drivers dump hash and results are as frightening as declared

Ikarus	❗ Trojan.WinNT.Stuxnet
MAX	❗ Malware (ai Score=100)
McAfee-GW-Edition	❗ GenericRXFO-ZPI74387C6AB0F2
Panda	❗ Trj/CI.A
Sangfor Engine Zero	❗ Trojan.Win32.Stuxnet.B
Symantec	❗ Trojan.Gen.2
VBA32	❗ Trojan.Stuxnet

Our analysis came to a conclusion that the dump showed us how it affects the memory and it is worth it finding traces of *stuxnet*.

The results of the analysis revealed that the Stuxnet malware had several malicious components, including a rootkit and a worm. The rootkit was used to hide the malware's presence on the system, and the worm was used to propagate the malware to other systems on the network. The malware also had several persistence mechanisms, including the creation of a service and the modification of the system's registry.

CONCLUSION

Overall, this research has highlighted the importance of memory forensics in incident response and threat hunting, especially when dealing with advanced malware such as Stuxnet. The use of memory forensics tools like Volatility can provide valuable insights into the behavior and persistence of malware, which can aid in the development of effective detection and response methods.

REFERENCE

- [1] A. Chetry and U. Sharma, "Memory Forensics Analysis for Investigation of Online Crime - A Review," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019, pp. 40-45.
- [2] K. M. A. Kamal, M. Alfadel and M. S. Munia, "Memory forensics tools: Comparing processing time and left artifacts on volatile memory," 2016 International Workshop on Computational Intelligence (IWCI), Dhaka, Bangladesh, 2016, pp. 84-90, doi: 10.1109/IWCI.2016.7860344.
- [3] K. M. A. Kamal, M. Alfadel and M. S. Munia, "Memory forensics tools: Comparing processing time and left artifacts on volatile memory," 2016 International Workshop on Computational Intelligence (IWCI), Dhaka, Bangladesh, 2016, pp. 84-90, doi: 10.1109/IWCI.2016.7860344.
- [4] S. Logen, H. Höfken and M. Schuba, "Simplifying RAM Forensics: A GUI and Extensions for the Volatility Framework," 2012 Seventh International Conference on Availability, Reliability and Security, Prague, Czech Republic, 2012, pp. 620-624, doi: 10.1109/ARES.2012.12.
- [5] C. -W. Tien, J. -W. Liao, S. -C. Chang and S. -Y. Kuo, "Memory forensics using virtual machine introspection for Malware analysis," 2017 IEEE Conference on Dependable and Secure Computing, Taipei, Taiwan, 2017, pp. 518-519, doi: 10.1109/DESEC.2017.8073871.
- [6] Mistry, N.R., Dahiya, M.S. Signature based volatile memory forensics: a detection based approach for analyzing sophisticated cyber attacks. Int. j. inf. tecnol. 11, 583–589 (2019).
- [7] R. Masood, Um-e-Ghazia and Z. Anwar, "SWAM: Stuxnet Worm Analysis in Metasploit," 2011 Frontiers of Information Technology, Islamabad, Pakistan, 2011, pp. 142-147, doi: 10.1109/FIT.2011.34.
- [8] McRee, Russ. "Memory analysis with DumpIt and volatility." *ISSA Journal*, September (2011).
- [9] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon," IEEE Security & Privacy, vol. 9, no. 3, pp. 49–51, 2011.
- [10] Feng, Qian, Aravind Prakash, Heng Yin, and Zhiqiang Lin. "Mace: High-coverage and robust memory analysis for commodity operating systems." In Proceedings of the 30th annual computer security applications conference, pp. 196-205. 2014.

BIOGRAPHY

Rehan Mumtaz, a software undergraduate at NED University holds a strong interest in the chain of development and breaking of the whole loop set of technologies. I am a great fan of cyber-security and loves to research about this domain and takes this as my passion as my expertise solely lies in Penetration testing(especially web and operating system exploitation), Cryptography & Exploit development. I am looking forward to move ahead with my passion and progress with it in the future InshaAllah

Wajahat Ahmed is an undergraduate student of Software Engineering Program at the Ned University of Engineering and Technology. His main areas of interest are data science ,cyber-security and software development .He has experienced of developing full stack applications and data analytics using cloud like Azure. He has also hand on experience in cyber security tools like wireshark,nmap,cryptool etc.His future aim is to become a data scientist and cloud engineer.

Moiz Tariq is an undergraduate student studying Software Engineering at NED University of Engineering and Technology in his senior year. His interests lie in the field of Cloud Computing, Computer Networks, DevOps, Cryptology and Virtualization. He is also an avid supporter of the FOSS (Free & Open-Source Software) movement. He enjoys reading and doing puzzles such as Sudoku, Chess and Jigsaw Puzzles.

Kabeer Ahmed is an undergraduate student at NED University of Engineering and Technology in Software Engineering. His main areas of interest are Offensive Security like OS Exploitation, Web Exploitation and Defensive Security like Digital Forensics, Cryptography. He has hands-on experience in Data Extraction and web penetration testing. He is keenly interested in making this career in this Cyber Security field.