**Dynamic Analysis of Window’s Based Malware using Reverse Engineering: A Case Study of Exmatter (Cont.)**

**Abstract**

Because malware is always evolving, malware analysis is a necessary yet difficult task. Numerous malware variations damage system resources or take them captive in exchange for ransom demands. Trojans are among the most sought-after by attackers because of their capacity to inflict immediate harm and implant further malware payloads on targets. This research proposes a system for the dynamic analysis of Trojan malware through staged code execution in a regulated virtual environment. Various stages of the framework's code are analyzed using open-source technologies. This study's investigation focuses on finding Indicators of Compromise (IOCs), which are strings that indicate suspicious behavior, and the Tactics, Techniques, and Procedures (TTPs) of the malicious Windows software Exmatter malware.

**Key words**: Dynamic malware analysis, static malware analysis, reverse engineering, .NET SDK libraries.

**1)Introduction**

Malware, sometimes referred to as malicious software, is any application or file created with the intention of causing harm to a server, network,or computer. Trojan horses, worms, spyware, ransomware, and computer viruses are examples of malware. This malicious software alters or hijack necessary computer operations, steals, encrypts, and deletes sensitive data, and keeps track of end users' computer activity .Going over the different types of malwares can help you understand the complexity of the current landscape. The process of malware analysis primarily occurs in an isolated virtual malware environment. One must propagate the process of capturing the malware using different types of techniques available for use in the malware analysis approach. This pertains to both the intent of malware and its application. The categories include viruses, botnets, ransomware, fileless malware, worm trojans, and droppers. Every malware has a unique way of operating. Trojan horses are malicious programs that spread other viruses by disguising themselves as other, trustworthy-looking code. When the Trojan reaches the victim's computer, it unpacks and inserts the code, which takes control and damages the victim's device. One example of malware, or Trojan, is Exmatter. Exmatter malware is a clickjacking type of malware that can be advantageously utilized to inject harmful payloads via files to obtain access to sensitive data belonging to any company and attempt to change it. the victim's machine by shutting it down and destroying it entirely, in addition to extracting as much data is possible from the system. Using a native API like RtlGetVersion, Ex Matter can acquire the OS version number. Additionally, the AlphaFS (Alpha Leonis).NET library's Native Methods class, which has numerous Windows APIs, is utilized by its routine. AlphaFS simulates Win32 file system features. Like System.IO, but supporting advanced NTFS features. Due to its self-destruction processing, the Ex-matter malware makes it more challenging for investigators to track its path.Before the ransomware is really installed on the victim's network, Exmatter was made to steal file types from several pre-selected directories and upload them to a server under the control of the attacker. The tool's developers continued to tweak it to optimize its operation and expedite the exfiltration of a sufficient volume of high-value data in the shortest amount of time, so that even at the time of its discovery, multiple variants were observed. The number of file types that Exmatter tries to exfiltrate has been lowered in this most recent version. Only files with the following extensions will now be exfiltrated by it:

File extensions:.**pdf,.doc,.docx,.xls,.xlsx,.png,.jpg,.jpeg,.txt,.bmp,.rdp,.txt,.sql,.msg,.pst,.zip,.rtf,.ipt,.dwg**

Additional malicious file extensions(FTP protocol):-

Including In former versions, there was a third exfiltration capability (FTP) to SFTP and WebDav. There are two distinct protocols for maintaining and sharing files across a network: SFTP and WebDAV. Here is a quick rundown of each:

**1.SSH File Transfer Protocol, or SFTP:** The Secure Shell (SSH) protocol, which is used for secure remote login and command execution, is extended by SFTP. SFTP offers a dependable and safe way to send files over an encrypted SSH connection between a client and a server. It has functions including file permission management, directory listing, and file transfer resume. In several sectors, such as finance and healthcare, where secure file transfer is crucial, SFTP is extensively utilized.  
2.**Web-based Distributed Authoring and Versioning, or WebDAV:** WebDAV is an addition to the HTTP protocol, which is utilized for data transmission over the Internet. Internet wide. Users can manage and edit files on remote web servers together thanks to WebDAV. Features like property management, versioning, and file locking are included. Wikis, other web-based collaborative platforms, and content management systems are common uses for WebDAV.

To summarize, WebDAV is an HTTP protocol extension that facilitates collaborative file management and editing on distant web servers, whereas SFTP is a secure file transfer protocol that runs over an encrypted SSH connection. Although they are widely utilized in many different industries and applications, the two protocols have different uses. The use case are as follows:-  
**a**)Reports: Possibility of creating a report with a list of all files that were processed.

**b**)Eraser: Processed files may get corrupted (not enabled in the version examined).

**c**)Self-destruct: A configuration option that, if enabled, causes the program to shut down and self-destruct when run outside of a Windows domain or in a non-corporate environment.

**d**)Support for Socks5 was discontinued.

**e**)GPO was used to deploy the tool in at least one attack.  
Furthermore, the malware underwent a significant rewrite, with even its preexisting functions being implemented in a different way. This could have been an attempt to evade discovery. It is unclear if Exmatter was developed by Coreid or a knowledgeable associate of the organization, but when combined with two distinct versions of Coreid's Ransomware is a noteworthy phenomenon. The group's concentration on data theft and extortion is further highlighted by its ongoing evolution, which highlights the significance of this aspect of attacks to ransomware perpetrators today. Agent Tesla is an advanced malware with multiple functions that mostly concentrates on stealing information, such as screenshots, credentials, and keystrokes. It is also capable of acting as a Remote Access Trojan (RAT), which grants remote access to the compromised system. The modular design of Agent Tesla is well renowned for its ability to greatly change and customize it for a wide range of malevolent goals. Usually, exploit kits, phishing emails, or software bundles are used to disseminate it. Since its creation in 2014, cybercriminals have been actively using the malware, which is still a serious danger. Agent Tesla is a common and adaptable malware that can carry out a variety of destructive tasks. It is also known by the names ZLoader and TONYSPY. It is a strong Trojan that steals information because its main purpose is to take sensitive data from compromised systems.   
1. **Information Stealing**: Gathering and removing confidential information from the compromised system is Agent Tesla's main duty. Keystrokes, clipboard information, credentials (including email, FTP, and browser credentials), and screenshots are all included in this. After that, the attacker's command and control (C&C) server receives the stolen data.   
2**. Capabilities of Remote Access Trojans (RATs):** Agent Tesla is capable of acting as a Remote Access Trojan (RAT) in addition to stealing information. This gives the attacker the ability to run commands on the compromised system remotely, install other malware, and more.

3. **Modular architecture:** Agent Tesla is incredibly versatile and flexible thanks to its modular architecture. Because of this, attackers can change and expand its functionality to meet their unique requirements. Furthermore complicating the detection and mitigation of security solutions is the modular design.

4. **Distribution Techniques:** Agent Tesla is usually distributed using a number of techniques, such as exploit kits, phishing emails, and software bundles. Malicious attachments or links that, when clicked or opened, download and run the virus are frequently found in phishing emails. By using software vulnerabilities, exploit kits, on the other hand, infect systems.

5. **Evasion strategies:** To avoid being discovered by security solutions, Agent Tesla uses a variety of evasion strategies. These methods include anti-virtualization, anti-sandboxing, anti-debugging, and encryption of communication channels.

Since 2014, Agent Tesla has been in operation as a sophisticated remote access trojan (RAT) and keylogger. It is mostly spread via hacked websites, malware files, and phishing emails. The malware is well-known for its capacity to remotely manipulate compromised devices and steal private data, including browsing data, passwords, and keystrokes.

This is a thorough explanation of Agent Tesla's operation:

**1.Infection:** Spear-phishing emails with malicious attachments, like Word or PDF files, are usually used to distribute Agent Tesla. As soon as the victim opens the attachment, the malware is launched by taking advantage of holes in the victim's system.  
2.**Installation:** Agent Tesla installs itself on the victim's computer following a successful exploit. To guarantee persistence, a scheduled task is created. and proceeds to operate in the background.

3. **Keylogging: Agent** Tesla can record every keystroke made on the compromised system thanks to its sophisticated keylogging features. Passwords, credit card details, and other private information entered by the victim may be captured by it.  
4. **Form grabbing:** Agent Tesla is also capable to extracting sensitive data, such as login credentials, from online forms.  
5. **File stealing:** The malware has the ability to upload files to the attacker's server by stealing them from the victim's computer based on particular file extensions.

6. **Remote access**: The attacker can access the compromised system remotely thanks to Agent Tesla. The victim's computer can be taken over by the attacker, who can also install other software and run commands.

7**. Communication:** In order to obtain commands, Agent Tesla speaks with its command and control (C2) server. directions from

the assailant and to transmit pilfered information.

8**. Evasion**: Agent Tesla employs a number of strategies, including obfuscation, encryption, and behavior modification, to avoid being discovered by security solutions.

To obtain a more comprehensive insight of the behavior and capabilities of the Agent Tesla virus, a variety of analysis types can be employed during the behavioral analysis process. The infection pathways, persistence strategies, communication protocols, and other crucial elements of the malware can be found with the use of these examinations. The following are some important categories of analysis that can be carried out on the Agent Tesla malware:

1. **Static analysis:** In this sort of analysis, the binary code and associated files of the malware are examined without the malware being executed. The structure of the malware, its embedded resources, and any indicators of compromise (IOCs) can all be found with the aid of static analysis.

2. **Dynamic analysis:** In this method, the malware is run in a sandbox or virtual computer to watch how it behaves and interacts with the system, framework. The persistence methods, network communication patterns, and other runtime characteristics of the virus can be found with the aid of this kind of study.

3. **Memory analysis:** This type of study looks at how the malware behaves and interacts with the memory of the system. The memory allocation patterns, data structures, and other memory-related behaviors of the virus can be found with the aid of this kind of study.

4**. Network analysis:** This entails keeping an eye on and examining the network communication activity of the malware. The command and control (C2) communication patterns, C2 server communication protocols, and other network-related behaviors of the virus can be found with the aid of this kind of study.

5**. Code analysis:** If the malware's source code is available, it entails disassembling its binary code and examining it. The methods, features, and other code-related characteristics of the virus can be found with the aid of this kind of study.

6. **Behavioral analysis:** In this method, the behavior of the malware and its interactions with the system are monitored and examined in real time. The evasion strategies, anti-analysis mechanisms, and other behavior-related activities of the virus can be found with the aid of this kind of analysis.

7. **Reverse engineering:** In order to comprehend the malware's inner workings, algorithms, and other crucial elements, reverse engineering entails breaking down and decompiling the binary code. This kind of investigation can assist in determining the malware's susceptibilities, methods of exploitation, and other actions connected to exploitation.  
Through executing these diverse analyses on the Agent Tesla virus, security analysts can acquire a thorough comprehension of the malware's actions, functionalities, and possible security hazards. Using this knowledge, plans for mitigation and effective countermeasures against the malware's dangers can be created.

**Table 1 – Indicators of compromise for exmatter malware**

|  |  |
| --- | --- |
| **SHA256** | **File names** |
| ad5002c8a4621efbd354d58a71427c157e4b2805cb86f434d724fc77068f1c40 | Sync\_nc.exe |
| 8c5b108eab6a397bed4c099f13eed52aeeec37cc214423bde07544b44a62e74a | Without\_cert.exe |
| 18c909a2b8c5e16821d6ef908f56881aa0ecceeaccb5fa1e54995935fcfd12f7 | locker.exe |
| ed6275195cf9fd758fb7f8bce868c14dc9e9d6b7aa6f472f714bce5ed7fabf7f | Vulp.exe |
| 5799d554307906e92749a0c45f21baff28d83b1cedccbf7cb6f2b98ac1b00930 | Ispmegr.exe |

**Table 2 -Indicators of compromise for Agent Tesla malware**

|  |  |
| --- | --- |
| **SHA256** | **File names** |
| cf33b91b38b99505782159948dc62742f5fe9df0ceacdaec3e00c603b080d093 | cf33b91b38b99505782159948dc62742f5fe9df0ceacdaec3e00c603b080d093 |
| 1b0e31ee2445c96bb67fecf748e061b3378cbe913a47240bddacdd8115ff29bf | APT-AE04805Q0024311METP.exe |
| 0c7c1939b48a75524b8ae8f58ca11efb3d2a5a401f100f80fbf956d1fdd6b062 | 74725794.zip |
| a79d7ee94851848b15682a4cbe3ab2721a7bdecaf1ba600eb291abbbd3e4fcf9 | BL.gz |
| 7bc99e45681c54f07827a4beb516cf130666b26ad1244d89dcda0a3e9c983f0f | Nnnnn.exe |

**Table 3 – Strings of exmatter and agent tesla malwares**

**Exmatter**

|  |  |
| --- | --- |
| **String** | **Type of string** |
| sync.exe | UTF-8 |
| Read\_logon | ASCII |
| Advapi32.dll | UTF-8 |

**Agent Tesla**

|  |  |
| --- | --- |
| **String** | **Type of string** |
| Hash key | UTF-8 |
| Registry key | UTF-8 |
| Event window | UTF-8 |

**Table 4 – TTPs for exmatter and agent tesla malware**

**Exmatter**

|  |  |
| --- | --- |
| **TTPs** | **Explanation** |
| Malloc\_dll | It allocates the dll to an extreme pointer of the EDRs to bypass it and get admin access |
| RegKeyCipher\_logon | This is used to login deciphering the hash algorithm and use the key to login as superuser |
| Ntdll\_Bcrypt | It encrypts and decrypts most of the Enums and logon process |

**Agent Tesla**

|  |  |
| --- | --- |
| **TTPs** | **Explanation** |
| GetFileNameEx | It is used to get all the names of different types of files by procrastinating the file types and enhancing its terms |
| GetHashKey | It is simply used to obtain hashes |
| AESDecrypt­\_Algorithm | Thos function can decrypt the AES-256 hashes |

**2)Literature Survey**

The author discusses the process of Reverse Engineering: A Technique to Examine the Behavior of Malicious Code Reverse engineering is covered in this paper**[1]** as a crucial technique for examining the behavior of harmful programs. It probably goes deep into dissecting malware to comprehend its features, actions, and possible effects on systems. In this paper the author describes about the PowerShell and.NET Tao of Malware Analysis in which the authors discuss various ways to investigate the analysis of malware written in PowerShell and.NET, two popular languages that hackers are increasingly using. The article**[2]** probably sheds light on the difficulties and approaches associated with examining malware that targets specific settings. The author proposes a machine learning-based method for both static and dynamic malware analysis is put forth by the authors in this paper**[3]**. This probably entails using characteristics taken from malware samples to train models capable of detecting and classifying malicious behavior. In this paper**[4]** a tool named MalAnalyser is used as a a technique for identifying Windows malware based on a series of API calls, is presented by Dabas and Sharma. This method probably entails examining how malware behaves when interacting with the Windows API to spot trends that point to harmful activity. Luo et al. probably talk**[5]** about approaches and difficulties in teaching malware analysis with an emphasis on the newest Windows operating systems. To improve knowledge and abilities in malware analysis, this could involve case studies and hands-on training. The methodical technique to malware analysis known as SAMA is proposed by the author in this paper**[6]**. This strategy probably includes a systematic approach to analyzing and comprehending malware behavior that may include both static and dynamic analysis methods. The author most likely gives a summary**[7]** of the many instruments and methods used in malware analysis and detection. This could use more sophisticated techniques like behavior analysis and sandboxing in addition to more conventional signature-based strategies. The author most likely describes a memory analysis-based malware detection and classification method in this paper**[8]**. This can entail looking through the contents of the memory for signs of malicious activity, including injected code or odd process behavior. A Survey**[9]** of the State-of-the-Art on Adversarial Attacks against Windows PE Malware Detection The most recent developments in hostile attacks against virus detection for Windows PE. This probably entails investigating methods that hackers use to avoid being discovered by antivirus software and other security measures. Most likely, the author of the paper virus detection for Windows PE. This probably entails investigating methods that hackers use to avoid being discovered by antivirus software and other security measures. Most likely, the author of the paper **[10]** suggests a static malware analysis technique that looks at Portable Executable (PE) header files. This method could entail taking elements out of PE files to recognize and categorize harmful software. It is possible that the authors in this paper **[11]** carry out a thorough examination of the anti-analysis strategies used by actual malware. Examining evasion strategies like code obfuscation, anti-debugging, and anti-VM capabilities may be part of this. The author in this paper **[12]**probably shed light on how malware interacts with Windows internals. suggests a static malware analysis technique that looks at Portable Executable (PE) header files. This method could entail taking elements out of PE files to recognize and categorize harmful software. examination and identification. To better detect and reduce malware threats, this may involve learning how the Windows operating system functions from the inside out. It is likely that the authors in this paper **[13]** examine how common Windows malware binaries are in C/C++ GitHub repositories. Examining how malware creators use open-source platforms for development and distribution may fall under this category. The goal of this work**[14]** is to provide the notion of "Malware Sight-Seeing," which uses point-of-interest-beacons (POIBs) to speed up the reverse engineering process. The authors suggest a system for determining and ranking crucial elements in a malware sample using POIBs. This method streamlines and improves the efficiency of the reverse engineering process, enabling security experts to examine and comprehend harmful code more quickly. This study**[15]** investigates the problems and developments in malware analysis and reverse engineering. The authors examine different reverse engineering techniques and resources, with an emphasis on creating the best possible environment for malware investigation. This study contains suggestions for further research and development as well as insightful information about the state of the field today. This study**[16]** offers a novel method for cyber-attack detection of Android malware intelligence about code (CCI). The CCI framework is designed to extract and analyze Android application behavioral patterns to detect potentially harmful activity. The authors show the potential of CCI as a tool for automated malware analysis by demonstrating how well their approach detects different kinds of Android spyware. The present study**[17]** broadens the scope of prior studies on malware sight-seeing by presenting a more comprehensive method known as "Binary Sight-Seeing." The authors provide a framework that leverages POIBs to speed up the process of reverse engineering binary files, both malicious and legitimate. The goal of this effort is to increase reverse engineering jobs' efficacy and efficiency in a variety of fields. This paper**[18]** describes a practical method of utilizing Volatility and Wireshark to analyze malware communications. The writers give examples of how to extract and decode network data from malware samples to reveal important information about the attackers' communication techniques. This study emphasizes how crucial it is to comprehend and examine malware communications to effectively mitigate threats. An extensive examination of protocol reverse-engineering techniques and resources is given in this overview article**[19]**. The authors focus on the creation and use of automated tools for this goal as they examine several methods for recognizing and evaluating the structure and semantics of network protocols. For researchers and practitioners interested in protocol reverse engineering, this book provides an extensive resource. An overview of ransomware, including its types, definition, and protection methods, is given in this article**[20]**. The author talks about how ransomware is becoming a bigger danger and gives tips on how businesses may defend themselves against these kinds of assaults. This piece of work functions as an invaluable tool for people and companies looking to learn more about and reduce the dangers related to ransomware. The malware analysis model presented in this research**[21]** makes use of reverse engineering methods to examine and comprehend the behavior of harmful software. The writers go over the many phases of the malware analysis procedure. In this paper, a new method for deciphering obfuscated Lua bytecode is presented. The authors suggest a methodology for deobfuscating and analyzing Lua bytecode using interpreter semantics testing. This paper **[21]** highlights the potential of interpreter semantics testing as a tool for automated Lua bytecode analysis by demonstrating the efficacy of their approach in recognizing and comprehending the behavior of malicious Lua scripts. This study **[22]** offers a thorough examination of the LockBit 2.0 ransomware. The writers investigate the methods used by this malware type for infection, persistence, and prevention. This research helps security researchers and practitioners better understand and reduce the risks associated with ransomware attacks by providing insightful information on the dynamic threat landscape of this type of malware. To sum up, the research papers mentioned above address a variety of various subjects including malware analysis, cybersecurity, and reverse engineering. These publications offer insightful analysis and significant progress in the field, assisting practitioners and security researchers in comprehending and reducing the ever-changing risks associated. The literature review concludes by highlighting the wide variety of approaches and methodologies used in malware analysis, including machine learning, memory analysis, static and dynamic analysis techniques, and the investigation of anti-analysis evasion tactics. Through comprehension and resolution of the dynamic strategies employed by malware writers, researchers hope to strengthen computer systems' resistance to malevolent attacks.

**3) Malware Analysis**

Malware analysis is an essential step in the process of finding and characterizing the existence and characteristics of malware in cyber security. It is sometimes the responsibility of security specialists to ascertain if a suspicious file is malicious or not. It is crucial because it allows responders to filter out false positives and assess the severity of a malware epidemic. Analyzing malware is useful both in advance of and during an incident. Malware analysis locates and categorizes the malware to offer helpful information during an occurrence. Malware analysis allows you to locate and log the infection, providing you with a wealth of information that can assist thwart future attempts. The malware analysis data is incorporated into the lessons learnt after the occurrence. Newly examined malware teaches analysts about patterns, attack methods, and behavior. We will examine both static and dynamic malware analysis techniques. Static analysis examines the malware code without running it. The executable code is displayed in this type of examination without requiring the application to be run by using pertinent tools. However, dynamic analysis must be carried out if the static analysis is unable to yield enough details about the malware. When performing dynamic malware analysis, the malware is run and its actions inside the computer environment are monitored to see how it behaves. Planning and preparation are essential because the malware's operations can damage the host system. To prevent harm to the host system, dynamic analysis is typically carried out in a virtual environment. The third A hybrid technique is used for malware analysis. Hybrid analysis uses a combination of static and dynamic analysis approaches to evaluate malware. For example, whereas dynamic analysis can be used to watch malware behavior in real time, static analysis can be used to identify possible dangers. Since hybrid analysis provides a thorough understanding of a sample's behavior as well as its code, it is typically considered the most successful form of malware analysis. Malware analysis can be done in two major ways: dynamically and statically. on dynamic analysis, the malware is executed on a virtual machine or other controlled environment, and its activity is monitored while it runs. This method can assist in determining the functions and goals of the malware and offers useful insights into its activity. In contrast, static analysis examines the malware code without running it. With this method, one can gain a better knowledge of the malware's fundamental structure and potentially spot exploitable flaws or vulnerabilities. Some typical methods for analyzing malware are as follows:

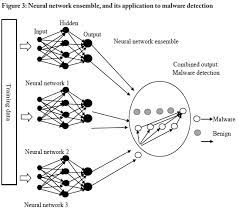
1) **Code analysis** : the process of looking through the malware's code to determine how it works and behaves.

2)**Behavior analysis**: the process of tracking the actions of the malware when it runs in a safe environment.

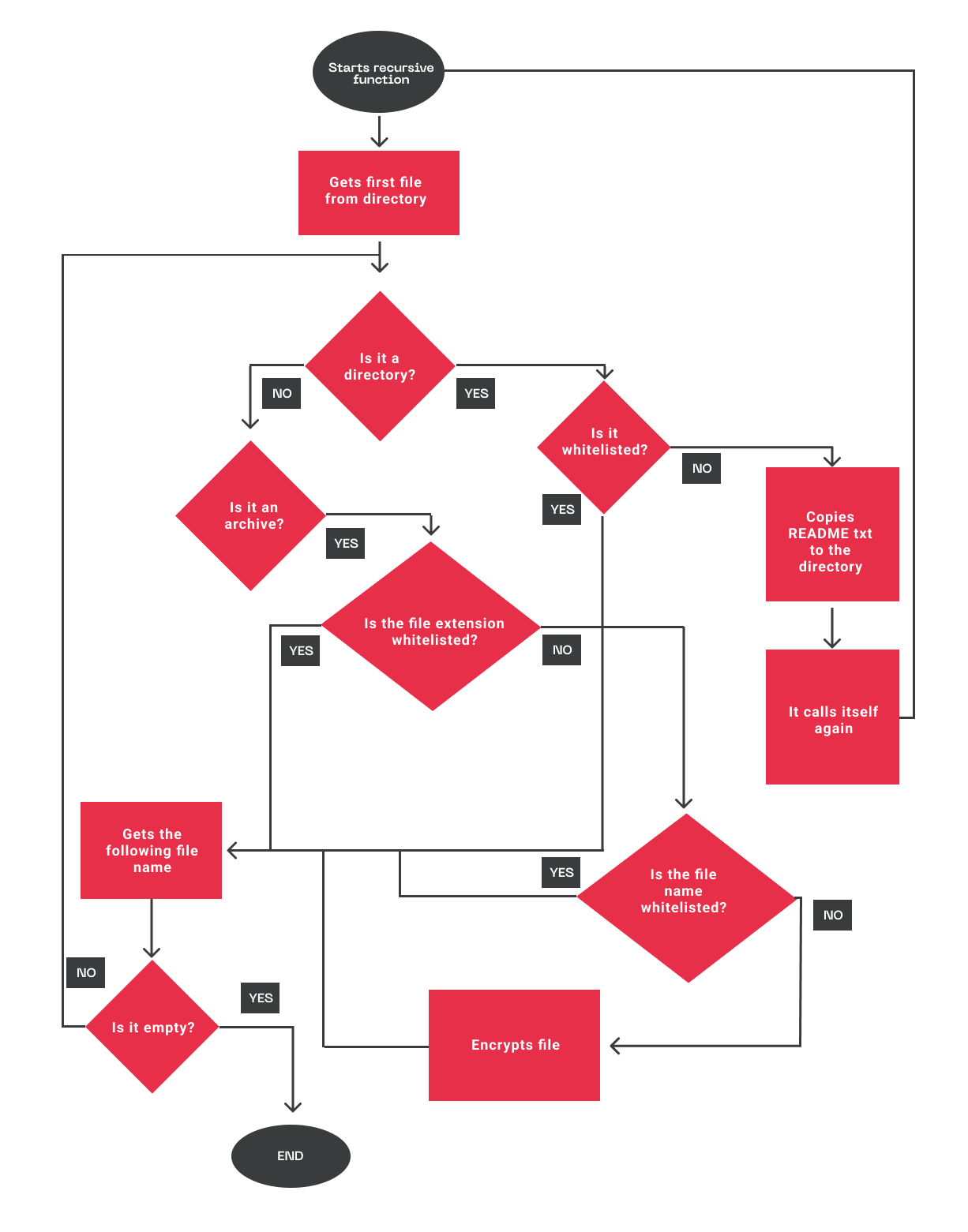
3)**Memory analysis**: looking at the malware's memory footprint to determine its behavior and operations;

4)**network analysis:** tracking the malware's interactions with other systems and devices to learn how it disseminates and gathers data.

Because malware analysis offers vital information that may be utilized to thwart future assaults, it is a crucial component of cybersecurity. Security researchers and organizations can create new defense techniques and tools to fend against malware by knowing how it operates. Furthermore, malware analysis can assist in determining the origin of an attack, which can be useful data for businesses and law enforcement organizations trying to stop such assaults in the future.



**Fig 1** – **malware analysis on a nvidia driver using neural networks(from symanec’s blog post)**



**Fig 2** – **Attack vector of the colonial pipeline attack(from Metabase Q)**

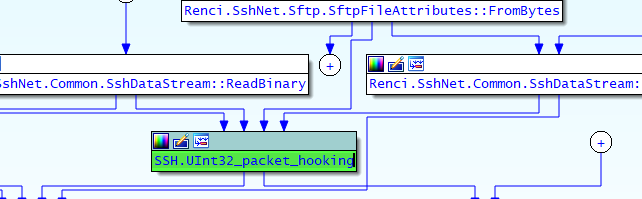
**4)Proposed methodology**

**Figure 3 illustrates the process flow of malware analysis and shows the standard strategy that is suggested**

Malware analysis for.NET malware entails a methodical approach to analyzing its behavior and code to comprehend its possible effects and functionality. First, disassemblers or decompiles are used to analyze the malware's binary to extract the source code or representation in an intermediate language. This makes it possible for analysts to understand the algorithms and logic that the malware uses. Next, the malware's runtime behavior—including its interactions with the system and network—is observed using dynamic analysis techniques like sandboxing and debugging. This aids in spotting malicious activity related to network interactions, file manipulation, and registry changes. Furthermore, the malware's structure and metadata are examined using static analysis techniques, which also help to spot potential exploits and indicators of compromise (IOCs). As the analysis progresses, specialists to extract useful insights and ultimately support the creation of efficient detection signatures and mitigation measures to counter the danger posed by the.NET malware, tools and skills are employed. In this process flow diagram for performing malware analysis for a .NET malware we can use tools like ghidra, ida pro, cutter, ILspy, dnSpy. Etc. and many more tools can be used. The decompilation can be done on Microsoft visual studio community edition.

**5)Results and Discussions**

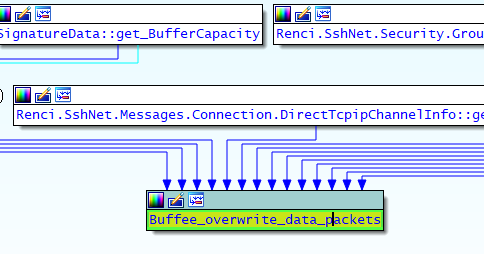
**a)Exmatter Malware Results**



**Fig 3 -**  **SSH packet hooking**

The act of intercepting and modifying Secure Shell (SSH) packets while they are being transmitted over a network is known as SSH packet hooking. Attackers attempting to gain unauthorized access to private data or systems frequently use this strategy. Attackers may try to get passwords, carry out commands, or introduce malicious payloads into the communication channel via intercepting SSH packets. Network devices such as switches, routers, and even hacked hosts are susceptible to packet hooking. Attackers may employ specialized instruments or scripts, like packet sniffers

or man-in-the-middle (MITM) assaults, to intercept SSH communication.



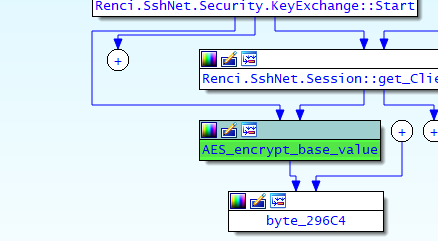
**Fig 4 – Packet overwrites (buffer underflow)**

A network security vulnerability known as packet overwrite occurs when an attacker modifies packet data to replace information that already exists in a packet. This can result in a few harmful effects, like changing control flags in the packet header, manipulating payload information, or changing the destination or source IP address. Attackers use vulnerabilities in packet overwrite to perform a variety of attacks, such as denial-of-service (DoS) assaults, packet injection, and session hijacking. Attackers can interfere with network traffic, get around security measures, or run arbitrary commands on target systems by altering packet contents. Network administrators should use intrusion detection systems (IDS) to identify and stop unusual packet activity, traffic filtering, and robust network segmentation to reduce the danger of packet overwrite attacks.



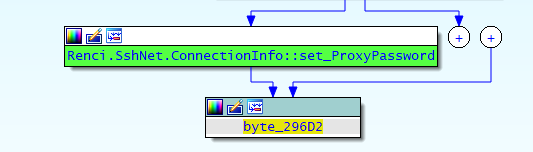
**Fig 5 – String payload (command injection using regex values)**

Creating input strings that match patterns defined by regular expressions (regex) is the process of creating string payloads using regex values. Web application security testing frequently uses these payloads, particularly for input validation and injection threats. An attacker could, for instance, create a string payload for a SQL injection attack that matches a regex pattern used for input validation. This would allow the attacker to get over client-side validation and take advantage of flaws in server-side code. In a similar vein, attackers may employ regex payloads in cross-site scripting (XSS) attacks to circumvent detection systems or input filters and insert malicious scripts into websites. Developers should put strong input validation and sanitization procedures in place on both the client and server sides to ward against such attacks.



**Fig 6 – AES Encryption (hash impersonation)**

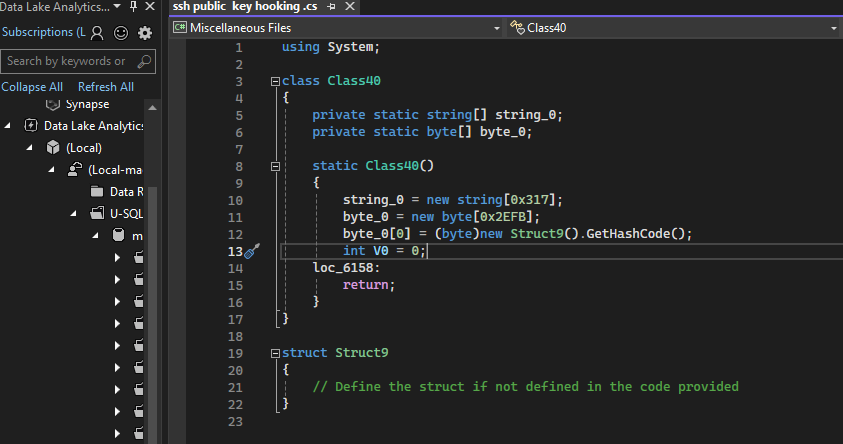
The symmetric encryption algorithm known as AES (Advanced Encryption Standard) is frequently employed to protect data. When used appropriately, AES is immune to assaults; but, when combined with subpar key management or cryptographic implementations, it can become vulnerable to attacks. Hashing impersonation is one possible attack vector in which a hacker tries to take advantage of flaws in the hashing technique used in AES implementations. With a hash function impersonation, an attacker can modify input to cause a collision—a situation in which two distinct inputs result in the same hash value. This collision could result in several security flaws, such as the capacity to pass for legitimate data or evade authentication procedures. Use of strong hash functions and adherence to appropriate key management procedures are essential for mitigating such attacks.



**Fig 7 – bypassing proxy password**

Usually, exploiting holes or weaknesses in the proxy server's setup is how a proxy password is bypassed. Several techniques, including brute-force assaults, misconfiguration exploits, and the use of known flaws in the proxy software, can be used to accomplish this. Credential brute-forcing is a popular technique in which an attacker repeatedly tries various username-password combinations until they discover the right one. This technique is dependent on users' or administrators' usage of default or weak passwords. Exploiting incorrect proxy server configurations is another technique. Inadequate logging and monitoring, improperly configured authentication methods, or lax access controls could all make it possible for an attacker to get around the proxy password without authenticating. To mitigate follow the MSDN docs entra protection to safeguard your data.

<https://learn.microsoft.com/en-us/entra/identity/authentication/howto-password-ban-bad-on-premises-deploy#install-and-configure-the-proxy-service>



**Fig 8 – Exploit for SSH key hooking (GetHashCode)**

This C# code defines a struct called **‘Struct9’** and a class called ‘**Class40’.** Now let us dissect the code:

1.**{Class40}:**  
- It has two private static arrays: one for bytes, called `byte\_0}, and the other for strings, called `string\_0}.  
- `static Class40()` is a static constructor that initializes `byte\_0} with an array of 12,091 (0x2EFB in hexadecimal) bytes and `string\_0} with an array of 791 (0x317 in hexadecimal) empty strings.

- The hash code of an instance of `Struct9} is set as the first element of `byte\_0} within the static constructor.

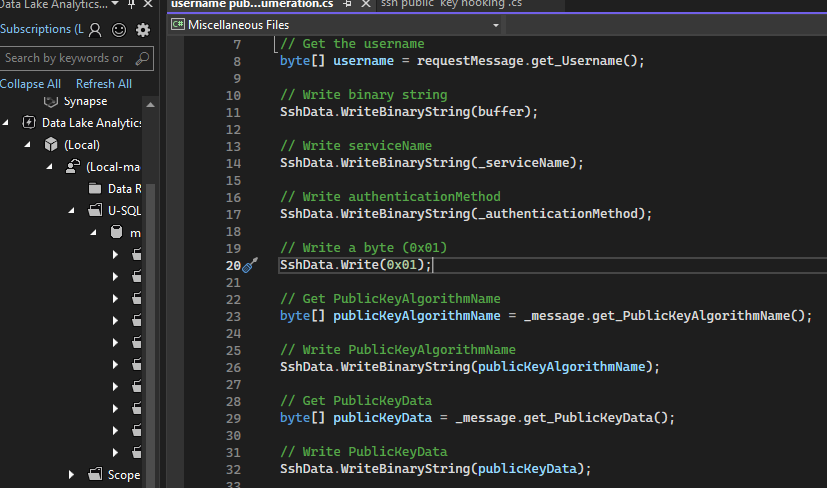
2. **{Struct9}:** - The code defines a struct type, but it does not supply its members.

Let us now discuss the possible weakness in this code:

**Vulnerability to Security:** It is the line "byte\_0[0] = (byte)new Struct9()" that is vulnerable .**GetHashCode()**;. It is attempting to initialize an instance of **‘Struct9’**s hash code as the first element of `byte\_0.Nevertheless, there is no guarantee that `**GetHashCode()`** will behave in a secure or surprising manner, particularly if `Struct9} does not override the `**GetHashCode()`** method. An attacker could be able to take advantage of the hash code's predictability.Potential security flaws like hash collision attacks or the nefarious use of predictable hash codes may result from this.

**In order to lessen this risk:**

- Should hash codes be required, make sure they are created correctly in a random and safe way, perhaps by overriding the **`GetHashCode()`** function in `**Struct9’.**



**Fig 9 – Exploit for Username enurmeration using public key authentication (PublicKeyAlgorithm)**

The following are the weaknesses in the code:

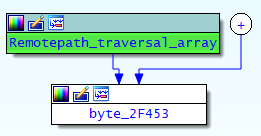
1.**Injection Vulnerabilities:** To obtain the username, the code calls the `**requestMessage.get\_Username**()` method. This approach may be susceptible to injection attacks, such as SQL injection or command injection, if the input is not being properly validated.

2. **Buffer Overflow:** The code stores and processes data, including the **{username}** variable, using byte arrays. These byte arrays may be subject to buffer overflow attacks if they are not appropriately sized and inspected.  
3. **Inadequate Input Validation**: The code uses functions like **`SshData.WriteBinaryString()’** and `SshData.Write()` to write different kinds of data to a buffer. These techniques may be subject to XSS and CSRF attacks if insufficient input validation is done.

The following are some countermeasures for the possible vulnerabilities found in the code:

1. **Injection Vulnerabilities:** You should check all input data, including the username, in order to reduce injection vulnerabilities. Libraries or frameworks for input validation can be used to make sure the data is in the correct format and is free of dangerous code.

2**. Buffer Overflow:** You should appropriately size and validate all byte arrays before utilizing them to guard against buffer overflow attacks. Secure coding techniques, including input validation and bounds checking, can be used to make sure that the byte arrays do not grow larger than they are supposed to.  
3. **Inadequate Input Validation: All** data written to the buffer should have enough input validation performed on it to thwart attacks like XSS and CSRF. Libraries or frameworks for input validation can be used to make sure the data is in the correct format and is free of dangerous code.

  
 **Fig 10 – Remote path injection**

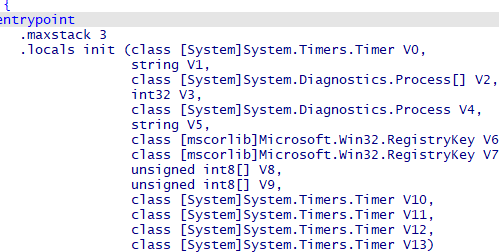
Vulnerability’s such as Remote Path Injection (RPI) allow attackers to run arbitrary code or obtain unauthorized access to a system. RPI happens when a program or service takes user-supplied URLs or file paths without properly validating or sanitizing them. Attackers can get around security measures or introduce malicious code to take advantage of this vulnerability.

**Use these recommended procedures to reduce RPI vulnerabilities:**

1. Carry out appropriate input validation: User-supplied input data, including file paths and URLs, should always be validated. Libraries or frameworks for input validation can be used to make sure the data is in the correct format and is free of dangerous code.

2. Put file path sanitization into practice: Before using user-supplied file paths, always sanitize them. You may utilize the file Use frameworks or path sanitization libraries to make sure that the file paths are free of dangerous characters or code to ensure its safety.

**b)Agent Tesla Malware Results**



**Fig 11 – Main function**

Malicious software, or malware, primarily aims to gain unauthorized access or cause harm to a target system or device by infiltrating it. Malware can have many distinct purposes and behaviors, but some typical ones are as follows:  
1. **Unauthorized access:** To enter a target system, malware frequently takes advantage of flaws in operating systems, applications, or hardware. This can be accomplished in several ways, including by using weak passwords, phishing attempts, and software flaws.  
2**. Running malicious code:** After the virus has successfully penetrated the target system, it will run a specific malicious code that can do several different things, like encrypting files for ransomware, stealing confidential information, or initiating a denial-of-service attack.

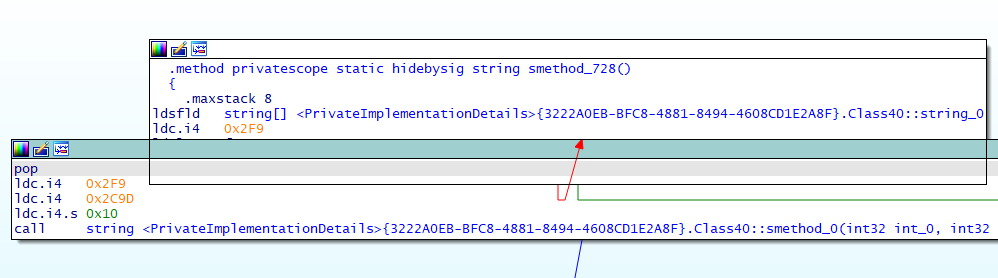


**Fig 12 – UnhookWindowsHookEx vulnerability**

A security flaw known as the UnhookWindowsHookEx vulnerability occurs when a program uses the UnhookWindowsHookEx API call to unhook a global hook (such WH\_KEYBOARD or WH\_MOUSE). Malicious actors may be able to take advantage of the application's vulnerability if it is unable to correctly unhook the global hook.

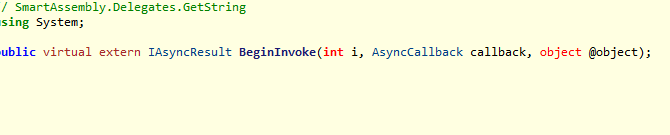
The following information relates to this vulnerability:

a)**Vulnerability:** An application may keep the hook method in place if it is unable to correctly unhook a global hook using the UnhookWindowsHookEx API function. Malicious actors may take advantage of this vulnerability and use the hook technique to do a variety of nefarious tasks, such stealing confidential information or inserting malicious code.  
b) **Mitigation**: mitigation strategy is for applications to make sure that they When global hooks are not needed, unhook them correctly.



**Fig 13 – Private implementation (0x2F9 and 0x2C90 vulnerable endpoints)**

The SAP NetWeaver vulnerability **CVE-2022-22535** is caused by inadequate user input encoding, which can result in unauthenticated code injection. The private implementation of the endpoints **0x2F9** and **0x2C90**, which are normally exposed via the network, is impacted by this issue. A successful exploit of this vulnerability may give an attacker access to private information like user IDs and passwords, which could have a partial impact on the application's confidentiality. This vulnerability's primary cause is inadequate user input validation and sanitization, which leaves open the possibility of an attacker infecting the system with malicious code. An attacker would have to make a specially crafted request to one of the affected endpoints to take advantage of this issue. There would be harmful code in the request that is not adequately sanitized or verified by the program, giving the attacker the ability to run any code on the target system.



**Fig 14 – IAsyncResult vulnerable driver**

The Microsoft Windows TCP/IP Driver has a buffer overflow problem that is connected to vulnerability **CVE-2021-21559**. The **IAsyncResult object**, which is used by the driver to control asynchronous activities, is vulnerable. When the driver neglects to appropriately verify the size of a buffer used in an **IAsyncResult** action, a buffer overflow happens. An attacker can take advantage of this weakness by targeting the compromised system with a specially constructed network packet. The buffer overflow may occur if the packet is handled by the weak driver, which would give the attacker access to run arbitrary code on the system. This vulnerability is important because it can be used by an unauthorized attacker over the network. If this vulnerability is successfully exploited, remote code execution may result, which can lead to the impacted system being completely compromised.



**Fig 15 – winapi intptr vulnerable endpoint**

The Microsoft Windows operating system has the Windows API method {SessionCreate`, which facilitates the creation of a new session object for user session management. If this function is not implemented or secured appropriately, it may be open to exploitation.This is a synopsis of the function Several parameters are entered into the function:  
1. {**pszSessionName**}: A Unicode string containing the session name.  
2. {**pszProfilePath{:** The path to the user's profile is specified by this Unicode string.  
3. The path of the provider's DLL is indicated by the Unicode string "pszProviderPath{."  
4. The name of the provider's DLL is specified by the Unicode string {pszProviderDll{.  
5. {**dwFlags}:** An 32-bit variable that indicates which flags should be utilized when creating a new session.  
6. **{dwReserved**{: A 32-bit value set aside for potential usage in the future.  
An integer pointer (intptr\\_t) to the recently formed session object is the function's return value. When input parameters are not checked correctly, there is a chance that the `SessionCreate` method will include vulnerabilities that could allow an attacker to craft a malicious request to the vulnerable endpoint. Numerous issues could arise from this, including remote code execution, privilege escalation, and unauthorized access. It is crucial to adhere to secure coding principles, such as input validation, output encoding, and appropriate access control, in order to reduce these vulnerabilities. Regular penetration tests and security assessments can also aid in locating and addressing any possible weaknesses in the system. To sum up, the function `SessionCreate` If not adequately guarded, vulnerabilities in the Windows API may be exploited. To avert possible attacks, developers should take the required safety measures to verify input parameters and put in place appropriate access controls.



**Fig 16 – Allocation function vulnerable drivers**

Memory on the stack is allocated with the help of the `stackalloc` keyword. Using `stackalloc`, a 29-byte buffer is allocated on the stack in an given formt. In this case, the vulnerability associated with `stackalloc` is not a direct flaw in the code itself, but rather a possible problem with how `stackalloc` is used generally. The primary issue with `stackalloc` is that if the allocated buffer size is more than the stack space that is available, it may result in stack overflow. This may occur when the code is run in a context with insufficient stack space, like a recursive function or a thread with a tiny stack size, or when the buffer size is incorrectly determined. An Crashing, corrupted data, and even uncontrolled code execution can result from stack overflows when attacker-controlled data overwrites the stack. It is advised to take the following actions to lessen the possible vulnerability associated with stackalloc:-

1.Verify that the allocated buffer size is accurately computed and does not surpass the available stack space.  
2. To avoid buffer overflow vulnerabilities, use appropriate memory management and bounds checking techniques.  
3. Steer clear of utilizing {stackalloc` in situations when there isn't much stack space available, like recursive functions or threads with tiny stack sizes.  
4. Use appropriate error checking and exception handling to deal with situations where the stack space is full.  
5. If the allocated memory is not sufficient, think about using different memory allocation techniques like the `Marshal.AllocHGlobal} function if there is a limited amount of stack space available, a big buffer size is anticipated.  
**Note:** The above-mentioned `stackalloc} vulnerability is a possible problem that could occur from using `stackalloc` generally. In order to precisely detect and evaluate vulnerabilities associated with {stackalloc{, the code should undergo a comprehensive security analysis and penetration testing.

**Table 5 – Vulnerabilities found and its types**

|  |  |  |
| --- | --- | --- |
| **Vulnerability** | **Type of vulnerability** | **Exploit State** |
| SSH packet hooking | Packet Obfuscation | It can be exploited using valid SSH key credentials to bypass the root user of the system |
| Packet overwrites | Buffer underflow | In this the packet has been assigned wrong ASCII keys and weak cryptography mechanism to weaken the encoded credentials |
| AES Encryption | Hash malfunction | The hashes are being not encrypted properly |
| bypassing proxy password | Arbitary write | In this the function of bypasses the proxy and gets the real address of the user even though he is using the faulted credentials |
| Remote path injection | Path traversal | In this simply we use or annotate the path by misplacing the file state and adding a malicious DLL in place if it |
| Main function(system32.dll) | Stack overflow | In this we misplace the stacks and create another function so that we can enter the victim’s system |
| UnhookWindowsHookEx | Sisals | This function simply is used to bypass the systems EDR/AV endpoints |
| Private implementation (0x2F9 and 0x2C90 vulnerable endpoints) | Authentication error | These two endpoints are the vulnerable points from which we will get the authentication error while entering our login credentials because it has already bypassed all the authentication mechanism |
| IAsyncResult | Driver Loader(buffer overflow) | In this function the driver will be affected and packets are being misplaced and tweaked to create functional errors |
| winapi intptr vulnerable endpoint | Dereference pointers | It simply initializes the dereference values and gain the same point of functions. |
| Allocation function vulnerable drivers | Stack loader | It allocates manipulated data function to get the value of the credentials by acquiring tokens |

**6)Conclusion**

To sum up, the examination of the malware samples ExMatter and Agent Tesla indicates their possible influence on the computers they aim to infect. These malwares exhibit advanced methods for avoiding detection, pilfering confidential data, and remaining persistent. ExMatter presents a serious problem for security experts because of its sophisticated encryption and obfuscation techniques. Conversely, Agent Tesla concentrates on credential theft, screenshot capture, and keylogging. Organizations should adopt a multi-layered security strategy that includes advanced threat detection and response systems, frequent software updates, and security awareness training for staff members to reduce the risks posed by these malware families. Monitoring and examining system logs and network traffic on a regular basis can aid in the early identification of malicious activity related to ExMatter and Agent Tesla. In the fields of penetration testing, bug bounty, and Security researchers can be vital in locating and disclosing vulnerabilities that these malware families might exploit through responsible disclosure. Organizations are therefore able to improve their entire security posture and proactively address these vulnerabilities. In conclusion, the examination of the ExMatter and Agent Tesla malware emphasizes the necessity of ongoing watchfulness, preventative security steps, and cooperation between enterprises and security researchers to successfully counter these dynamic threats. To sum up, the Agent Tesla malware is a dangerous and ever-evolving menace that endangers both individuals and companies. It is a dangerous foe because of its capacity to avoid detection, obtain private data, and persist on compromised computers. Using a multi-layered security approach is essential for individuals and organizations to reduce the threats related to Agent Tesla. This entails frequently backing up crucial data, utilizing reliable antivirus and anti-malware programs, and maintaining all operating systems and applications up to date. Users should also use caution when opening attachments from emails or clicking on links that come from unidentified or unreliable sources. Using two-factor authentication and changing passwords on a regular basis can also help lower the chance of compromise.For businesses, recurring security evaluations and penetration tests Testing can assist in locating and fixing security holes that Agent Tesla or other malware threats might use against you. To make sure that users are aware of the risks and know how to defend themselves and their organizations from these kinds of threats, employee security awareness training is also crucial. In conclusion, even though the Agent Tesla malware is a serious threat, individuals and businesses can lower their chance of infection and better secure their sensitive data by putting in place a thorough security policy and remaining vigilant.

**7)Future Scope**

Future work on ExMatter and Agent Tesla malware research will focus on a few important areas for development. Among them are:

1. **Advanced Artificial Intelligence (AI) and Machine Learning (ML) Techniques**: By incorporating ML and AI algorithms into malware analysis tools, such as ExMatter and Agent Tesla, malware detection and classification can be greatly enhanced. By identifying novel and developing malware variants, these cutting-edge approaches can facilitate quicker response and mitigation.

2. **Improved Dynamic Analysis:** Creating more complex dynamic analysis environments can shed light on the capabilities and actions of the viruses ExMatter and Agent Tesla. To further understand the malware's infection and dissemination methods, this can involve adopting sophisticated sandboxing techniques and mimicking real-world network and system configurations.

3**. Better Static Analysis**: By improving static analysis methods like code emulation, decompilation, and disassembly, it is possible to find hidden features and ways that the ExMatter and Agent Tesla virus evades detection. This may result in the identification of vulnerabilities and exploitation strategies that were not previously known. The future range of Agent Tesla malware is probably going to change and adjust to the constantly shifting cybersecurity environment. It is anticipated that Agent Tesla and other malware threats will keep coming up with new ways to get around security measures as new technology and protections are introduced. The cybersecurity sector must keep coming up with new methods and technologies to identify, evaluate, and eliminate malware threats in order to keep up with these ever changing dangers. This could involve developments in machine learning and artificial intelligence to more effectively recognize and react to new malware variations. Agent Tesla and other malware threats may also focus more on targeting particular sectors or companies that might have laxer security measures in place in the future. This might encourage the creation of more focused and specialized malware strains.  
Furthermore, Agent Tesla and other malware threats might start targeting these connected devices as the Internet of Things (IoT) grows. Because IoT devices sometimes have low processing and memory capacities, putting in place strong security measures may be more challenging, which could provide new issues for cybersecurity specialists. In conclusion, it is likely that Agent Tesla virus and other malware threats will continue to evolve and adapt in the future, requiring constant innovation and improvement in the cybersecurity sector in order to effectively counter these threats.

**8)References**

1. *Reverse Engineering-A Method for Analyzing Malicious Code Behavior*. (n.d.). Reverse Engineering-A Method for Analyzing Malicious Code Behavior | IEEE Conference Publication | IEEE Xplore. 2022
2. Pontiroli, Santiago M., and F. Roberto Martinez. "The tao of. net and powershell malware analysis." *Virus bulletin conference*. 2021.
3. Ijaz, Muhammad, Muhammad Hanif Durad, and Maliha Ismail. "Static and dynamic malware analysis using machine learning." *2020 16th International bhurban conference on applied sciences and technology (IBCAST)*. IEEE, 2020.
4. Dabas, Namita, and Prabha Sharma. "MalAnalyser: An effective and efficient Windows malware detection method based on API call sequences." *Expert Systems with Applications* 230 (2023): 120756.
5. Luo, Lan, et al. "On teaching malware analysis on latest windows." *Journal of The Colloquium for Information Systems Security Education*. Vol. 9. No. 1. 2022.
6. Bermejo Higuera, Javier, et al. "Systematic approach to malware analysis (SAMA)." *Applied Sciences* 10.4 (2020): 1360.
7. Talukder, Sajedul. "Tools and techniques for malware detection and analysis." *arXiv preprint arXiv:2002.06819* (2020).
8. Sihwail, Rami, Khairuddin Omar, and Khairul Akram Zainol Arifin. "An Effective Memory Analysis for Malware Detection and Classification." *Computers, Materials & Continua* 67.2 (2021).
9. Ling, Xiang, et al. "Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art." *Computers & Security* (2023): 103134.
10. Aggarwal, Naman, Pradyuman Aggarwal, and Rahul Gupta. "Static malware analysis using pe header files api." *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2022.
11. Kim, Minho, Haehyun Cho, and Jeong Hyun Yi. "Large-Scale Analysis on Anti-Analysis Techniques in Real-World Malware." *IEEE Access* 10 (2022): 75802-75815.
12. Mohanta, Abhijit, et al. "Windows internals." *Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware* (2020): 123-162.
13. La Cholter, William, Matthew Elder, and Antonius Stalick. "Windows Malware Binaries in C/C++ GitHub Repositories: Prevalence and Lessons Learned." *ICISSP*. 2021.
14. See, August, et al. "Malware Sight-Seeing: Accelerating Reverse-Engineering via Point-of-Interest-Beacons." *arXiv preprint arXiv:2109.04065* (2021).
15. Shin, Kangsik, et al. "A Study Protocol Reverse Engineering Research Trend and Construction of Optimal Environment for Malware Analysis." *Journal of the Korea Institute of Information Security & Cryptology* 31.2 (2021): 175-186.
16. Qiu, Junyang, et al. "Cyber code intelligence for android malware detection." *IEEE Transactions on Cybernetics* 53.1 (2022): 617-627.
17. See, Richard August, et al. "Binary Sight-Seeing: Accelerating Reverse Engineering via Point-of-Interest-Beacons." *Proceedings of the 39th Annual Computer Security Applications Conference*. 2023.
18. Acosta, Jaime C., Daniel E. Krych, and US Army Combat Capabilities Development Command Army Research Laboratory Adelphi. "Hands-on cybersecurity studies: uncovering and decoding malware communications-initial analysis with Wireshark and volatility." *US Army Combat Capabilities Development Command Army Research Laboratory Adelphi* (2021).
19. Huang, Yuyao, et al. "Protocol reverse-engineering methods and tools: A survey." *Computer Communications* 182 (2022): 238-254.
20. Chambers, Brandon. "Ransomware: What Is Ransomware, and How to Prevent It." (2023).
21. Amiruddin, Amiruddin, et al. "Utilizing Reverse Engineering Technique for A Malware Analysis Model." *Scientific Journal of Informatics* 8.2 (2021): 222-229.
22. Luo, Chenke, et al. "Reverse Engineering of Obfuscated Lua Bytecode via Interpreter Semantics Testing." *IEEE Transactions on Information Forensics and Security* (2023).
23. Eliando, Eliando, and Yunianto Purnomo. "LockBit 2.0 Ransomware: Analysis of infection, persistence, prevention mechanism." *CogITo Smart Journal* 8.1 (2022): 232-243.
24. <https://www.bankinfosecurity.com/minimize-effects-malware-a-733>
25. <https://socradar.io/exmatter-tool-provides-a-new-strategy-for-extortion>
26. <https://purplesec.us/resources/cyber-security-statistics/>
27. <https://learn.microsoft.com/en-us/entra/identity/authentication/howto-password-ban-bad-on-premises-deploy#install-and-configure-the-proxy-service>
28. <https://www.mcafee.com/blogs/other-blogs/mcafee-labs/agent-teslas-unique-approach-vbs-and-steganography-for-delivery-and-intrusion/>
29. <https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/noberus-blackcat-ransomware-ttps>
30. <https://intezer.com/blog/malware-analysis/the-role-of-malware-analysis-in-cybersecurity/>