

Contents

<u>1. Introduction</u>	<u>Page</u>
1.1 History	3
1.2 Working	4
1.3 Purpose	6
1.4 services compromised	7
<u>2. Malware Analysis</u>	
2.1 Static Analysis	9
2.2 Dynamic Analysis	17
<u>3. Mitigation Strategies</u>	21

1.

Introduction

1.1. A brief History on SnakeKeyLogger

Snake Keylogger is a highly sophisticated malware designed to steal sensitive information by capturing keystrokes, clipboard data, screenshots, and stored credentials. First detected in **2020**, it quickly gained popularity among cybercriminals due to its stealthy nature and ability to evade detection through advanced obfuscation techniques.

Typically distributed via phishing emails containing malicious attachments—such as Microsoft Word or PDF files with embedded macros—Snake Keylogger infiltrates a system once the victim opens the infected document. Once active, it silently records user activity and exfiltrates the stolen data via email, FTP, or messaging platforms like Telegram.

Snake Keylogger gained widespread attention in **2021**, when it became available as **Malware-as-a-Service (MaaS)** on underground hacker forums. This meant that even low-skilled cybercriminals could purchase and deploy the malware easily, leading to an increase in **targeted attacks on businesses, financial institutions, and individual users**. Security researchers have observed that the malware frequently evolves with **obfuscation techniques**, making it harder to detect by traditional antivirus solutions.

Key Features & Capabilities:

- Captures **keystrokes, clipboard data, screenshots, and credentials**.
- Sends stolen data via **SMTP (email), FTP, or Telegram bots**.
- Uses **obfuscation and anti-analysis techniques** to evade detection.

1.2. Working

1. Infection & Initial Access

- The victim receives a **phishing email** with a malicious **Word, Excel, or PDF document** attached.
- The document contains a hidden **VBA macro** that executes **malware upon opening**.
- The user is tricked into clicking "**Enable Macros**" or "**Enable Content**", allowing the malicious script to run.

2. Malware Execution & Installation

- The macro **downloads and executes Snake Keylogger** from a remote command-and-control (C2) server.
- The malware installs itself into **Windows directories** (e.g., AppData or Temp) to persist across reboots.
- It may modify **Windows Registry keys** to launch automatically on startup.

3. Data Collection & Keylogging

- Snake Keylogger begins monitoring **keystrokes**, capturing everything the user types, including:
 - Login credentials (usernames & passwords).
 - Banking and payment details.
 - Messages, emails, and private data.
- It can also steal:
 - **Clipboard data** (copied text, passwords).
 - **Stored browser credentials**.

- Screenshots of active windows.

4. Exfiltration of Stolen Data

- The collected data is sent to the attacker via:
 - **SMTP (email)** – sends logs to an attacker-controlled email.
 - **FTP** – uploads data to a remote server.
 - **Telegram bots** – uses encrypted communication for stealth.

5. Persistence & Evasion

- Snake Keylogger employs **anti-analysis techniques** to evade detection, such as:
 - **Code obfuscation** to bypass antivirus detection.
 - **Disabling security tools** like Windows Defender.
 - **Checking for sandbox environments** to avoid detection by researchers.

How Snake Keylogger Uses Macros²

A **macro** is a script or set of automated instructions embedded within a document, typically used in applications like **Microsoft Word, Excel, and PowerPoint**. Macros are written in **Visual Basic for Applications (VBA)** and are designed to automate repetitive tasks, such as formatting, calculations, and data entry.

Snake Keylogger is often delivered via phishing emails that contain **malicious Microsoft Office documents**. These documents prompt the victim to enable macros, usually by displaying a fake security warning like: *"This document is protected. Enable content to view the full text."*

If the victim enables macros, the embedded VBA script **executes malicious code** that:

- **Downloads and installs Snake Keylogger onto the system.**
- **Runs in the background**, logging keystrokes, stealing credentials, and capturing screenshots.
- **Exfiltrates stolen data** via email, FTP, or Telegram².

The screenshot shows a debugger interface with assembly code. Annotations highlight specific strings:

- initial decryption key string**: A yellow box highlights the string "zbtIwGMyasSqTLx".
- encrypted rsrc data entry**: A yellow box highlights the string "Example".

```

1 //using Max.Properties;
2
3 namespace Novem
4 {
5     // Token: 0x02000007 RID: 7
6     internal class bBAp0
7     {
8         // Token: 0x04000005 RID: 5
9         public static byte[] aKTBl = JVkQP.Abberant(GdDJM.KTGOD, GdDJM.KIGOD, GdDJM.KIGOD, "zbtIwGMyasSqTLx", 0);
10    }
11 }
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55     // Token: 0x17000003 RID: 3
56     // (get) Token: 0x06000005 RID: 5 RVA: 0x000021D8 File Offset: 0x000003D8
57     internal static byte[] bIOxxk
58     {
59         get
60         {
61             object @object = GdDJM.bIOxxk.GetObject("Example");
62             return (@object);
63         }
64     }

```

1.3 Services Compromised

The primary categories of services targeted by Snake Keylogger include the following:

1. Email Services

Snake Keylogger poses a significant threat to widely used email platforms, including **Gmail, Outlook, Yahoo Mail, and ProtonMail**. By intercepting login credentials, the malware facilitates unauthorized access, enabling threat actors to manipulate email communications, disseminate phishing campaigns, and extract sensitive correspondences.

2. Banking and Financial Services

The malware actively targets online banking systems, such as **PayPal, Bank of America, and Wells Fargo**, with the intent of capturing authentication credentials. This form of credential theft exposes victims to financial fraud, unauthorized fund

transfers, and illicit transactions.

3. Social Media Platforms

Popular social networking services, including **Facebook, Instagram, Twitter, and LinkedIn**, are frequently compromised through Snake Keylogger's credential-stealing mechanisms. The exploitation of these accounts can result in identity theft, social engineering attacks, and the propagation of further malware.

4. Cloud Storage and File-Sharing Services

Platforms such as **Google Drive, Dropbox, OneDrive, and Mega** serve as lucrative targets due to their role in storing sensitive documents and proprietary data. Unauthorized access to these repositories heightens the risk of intellectual property theft, data leaks, and corporate espionage.

5. E-Commerce and Online Payment Platforms

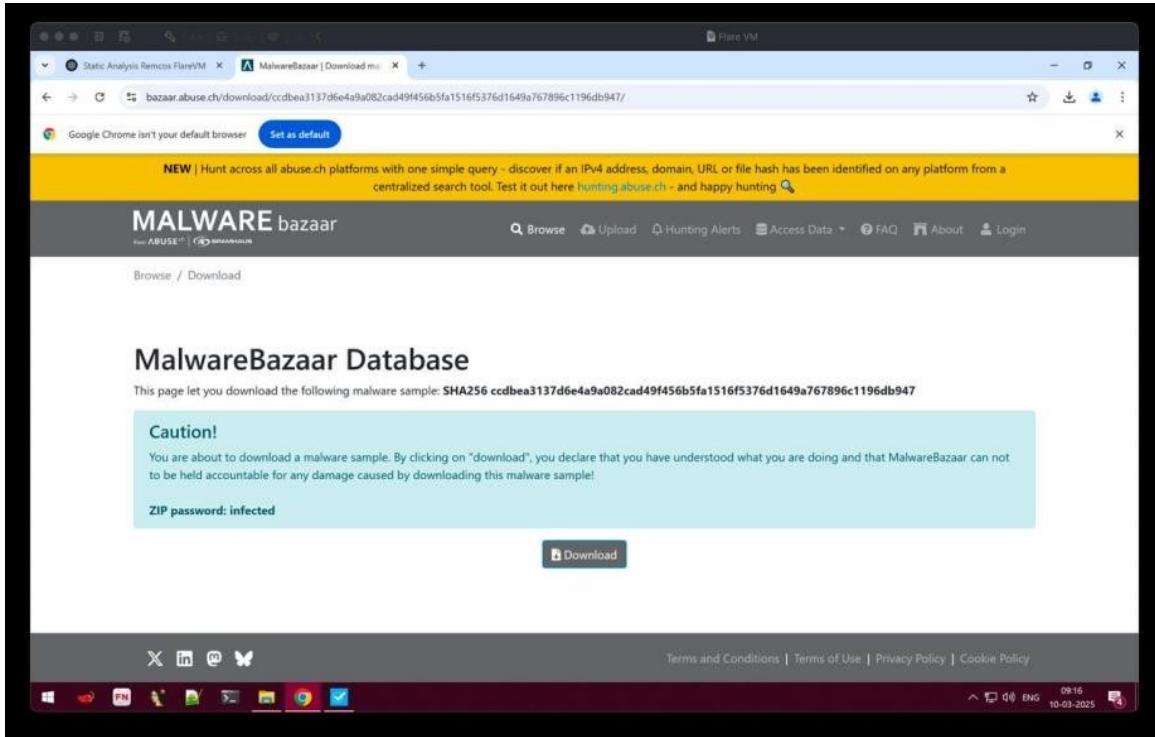
Snake Keylogger also targets e-commerce platforms and online payment services. By intercepting credentials used for platforms such as Amazon, eBay, and payment gateways, attackers can conduct fraudulent transactions, steal user account information, and exploit customer trust. This type of attack can lead to significant financial loss for both consumers and businesses, while also damaging the reputation of the e-commerce platform involved.

² https://www.splunk.com/en_us/blog/security/under-the-hood-of-snakekeylogger-analyzing-its-loader-and-its-tactics-techniques-and-procedures.html

³ : <https://www.fortinet.com/blog/threat-research/deep-analysis-of-snake-keylogger>

2.1 Static Analysis

Static Analysis is the Analysis of malware done without actual execution inside a controlled environment such as a Virtual Machine. Here in order to do a static analysis on the SnakeKeyLogger malware we download a malware sample from www.malwarebazaar.com .



Here we employ a variety of tools using our flareVM to conduct a static analysis on our downloaded malware sample from the Internet.

1. PEView

Indicates that the file is a Windows executable with sections containing encrypted or packed code. Magic number is 5A4D which is .exe.ed

The PE file contains a valid DOS header, starting with the MZ signature (IMAGE_DOS_SIGNATURE).

The screenshot shows the PEview interface with the file '9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688.exe' open. The left pane displays the file structure tree, with the 'IMAGE_DOS_HEADER' node selected and highlighted in blue. The right pane is a table showing the fields of the DOS header:

pFile	Data	Description	Value
00000000	5A4D	Signature	IMAGE_DOS_SIGNATURE
00000002	0090	Bytes on Last Page of File	
00000004	0003	Pages in File	
00000006	0000	Relocations	
00000008	0004	Size of Header in Paragraphs	
0000000A	0000	Minimum Extra Paragraphs	
0000000C	FFFF	Maximum Extra Paragraphs	
0000000E	0000	Initial (relative) SS	
00000010	00B8	Initial SP	
00000012	0000	Checksum	
00000014	0000	Initial IP	
00000016	0000	Initial (relative) CS	
00000018	0040	Offset to Relocation Table	
0000001A	0000	Overlay Number	
0000001C	0000	Reserved	
0000001E	0000	Reserved	
00000020	0000	Reserved	
00000022	0000	Reserved	
00000024	0000	OEM Identifier	
00000026	0000	OEM Information	
00000028	0000	Reserved	
0000002A	0000	Reserved	
0000002C	0000	Reserved	
0000002E	0000	Reserved	
00000030	0000	Reserved	
00000032	0000	Reserved	
00000034	0000	Reserved	
00000036	0000	Reserved	

Viewing IMAGE_DOS_HEADER

2. PEStudio

High VirusTotal Detection Rate (34/73)

Multiple AV engines flagged it as **malicious**, including detections like Win32:MalwareX-gen [Trj], Malicious, and Kryptik.ANKV. The presence of W32.AIDetectMalware.CS suggests AI-based detection of malware behaviors.

Suspicious Compilation Timestamp (2091)

Malware authors often manipulate timestamps to evade detection by security tools that rely on behavioral analysis.

property	value
file	
file > sha256	9EDB73376E0B1C388EA94BC00B634603F069AA46BE7CA5BAB0E315AFEAB43688
file > first 32 bytes (hex)	4D 5A 90 00 03 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 40 00 00 00 00 00 00 00
file > first 32 bytes (text)	MZ.....@.....
file > info	size: 784384 bytes, entropy: 7.690
file > type	executable, 32-bit, GUI
file > version	1.0.0.0
file > description	Vector International
entry-point > first 32 bytes (hex)	FF 25 00 20 40 00
entry-point > location	0x000C0CC6
file > signature	Microsoft Linker 48.0
stamps	
stamp > compiler	Tue Feb 20 16:36:30 2091 (UTC)
stamp > debug	n/a
stamp > resource	n/a
stamp > import	n/a
stamp > export	n/a
names	
file > name	c:\users\karthik\downloads\9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688
debug > file	n/a
export	n/a
version > original-file-name	yHoL.exe
manifest	MyApplication.app

sha256 > 9EDB73376E0B1C388EA94BC00B634603F069AA46BE7CA5BAB0E315AFEAB43688 | cpu > 32-bit | file > type > executable | subsystem > GUI

High Entropy (7.690)

High entropy suggests encryption or packing, which is a common obfuscation technique used by keyloggers to hide their real behavior.

Presence of .NET Module (yHoL.exe)

Snake Keylogger is known to be written in .NET, and the presence of a .NET module aligns with its structure.

3. VirusTotal

Uploading the malware sample to VirusTotal reveals multiple detections from antivirus engines, confirming its malicious nature. Detection Rate: 31/73 AV engines flagged it as malicious.

The screenshot shows the VirusTotal analysis page for the file 1bb6d872af87193593934e811a5643f153aa52c6d6efbbcb9f87e2cd35cc500. The page displays a community score of 31/71, indicating that 31 security vendors flagged the file as malicious. The threat label is listed as trojan.msl!cdudu. A table below shows detections from various antivirus engines:

Security vendor	Detection	Family labels
Avast	Win32.MalwareX-gen [T]!!	AVG
Avira (no cloud)	TR/AD.SnakeStealer.cdudu	CrowdStrike Falcon
Cylance	Unsafe	DeepInstinct
Elastic	Malicious (High Confidence)	ESET-NOD32
Fortinet	MSIL/SnakeLogger.D5D0ltr.spy	Google
Huorong	HEUUR/VirTecl/MSIL_0fuscator.gen/A	Kaspersky
		Win32/MalwareX-gen [T]!!
		Win/malicious_confidence_100% (W)
		MALICIOUS
		A Variant Of MSIL/Kryptik.ANKV
		Detected
		HEUUR/Trojan.MSIL.Taskun.gen

4. BinText: Extracts suspicious strings such as keylog, sendtoC2, and stealpassword, revealing its malicious intent.

The screenshot shows the BinText 3.0.3 application interface. The 'File to scan' field contains the path C:\Users\admin\Downloads\1bb6d872af87193593934e811a5643f153aa52c6d6efbbcb9f87e2cd35cc500.exe. The 'Advanced view' checkbox is checked. The results table lists memory addresses, memory positions, IDs, and text strings:

File pos	Mem pos	ID	Text
000000000F739	000000411539	0	DataGridViewTriState
000000000F74E	00000041154E	0	Delete
000000000F755	000000411555	0	get_White
000000000F75F	00000041155F	0	Write
000000000F765	000000411565	0	STAThreadAttribute
000000000F778	000000411578	0	CompilerGeneratedAttribute
000000000F793	000000411593	0	GuidAttribute
000000000F7A1	0000004115A1	0	HelpKeywordAttribute
000000000F7B6	0000004115B6	0	GeneratedCodeAttribute
000000000F7CD	0000004115C0	0	DebuggerNonUserCodeAttribute
000000000F7EA	0000004115EA	0	DebuggableAttribute
000000000F7FE	0000004115FE	0	EditorBrowsableAttribute
000000000F817	000000411617	0	ComVisibleAttribute
000000000F828	00000041162B	0	AssemblyTitleAttribute
000000000F842	000000411642	0	AssemblyTrademarkAttribute
000000000F850	00000041165D	0	TargetFrameworkAttribute
000000000F876	000000411676	0	ToolboxItemAttribute
000000000F888	00000041168B	0	AssemblyFileVersionAttribute
000000000F8A8	0000004116A8	0	AssemblyConfigurationAttribute
000000000F8C7	0000004116C7	0	AssemblyDescriptionAttribute
000000000F8E4	0000004116E4	0	XmlSchemaProvideAttribute
000000000F8FF	0000004116FF	0	CompilationRelaxationsAttribute
000000000F91F	00000041171F	0	AssemblyProductAttribute
000000000F938	000000411738	0	AssemblyCopyrightAttribute
000000000F953	000000411753	0	XmlRootAttribute
000000000F964	000000411764	0	AssemblyCompanyAttribute
000000000F97D	00000041177D	0	DesignerCategoryAttribute
000000000F997	000000411797	0	DesignerSerializationVisibilityAttribute
000000000F9C0	0000004117C0	0	RuntimeCompatibilityAttribute
000000000F9DE	0000004117DE	0	ReadByte
000000000F9E7	0000004117E7	0	get_Blue
000000000F9F0	0000004117F0	0	get_AliceBlue

5. GHIDRA

GHIDRA, an open-source reverse engineering tool by the NSA, is widely used for malware analysis. To load a sample, create a new project, import the file (**File > Import File**), and start the analysis. Once complete, open **CodeBrowser** to inspect disassembled code, functions, and imports. Use the decompiler to reconstruct logic, helping to understand the malware's behavior efficiently.

Import Results Summary	
Project File Name:	9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688.exe
Last Modified:	Tue Apr 01 20:53:41 IST 2025
Readonly:	false
Program Name:	9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688.exe
Language ID:	x86:LE:32:default (3.0)
Compiler ID:	windows
Processor:	x86
Endian:	Little
Address Size:	32
Minimum Address:	00400000
Maximum Address:	004c41ff
# of Bytes:	784384
# of Memory Blocks:	4
# of Instructions:	0
# of Defined Data:	375
# of Functions:	147
# of Symbols:	345
# of Data Types:	1626
# of Data Type Categories:	14
Compiler:	cli
Created With Ghidra Version:	11.0.1
Date Created:	Tue Apr 01 20:53:25 IST 2025
Executable Format:	Portable Executable (PE)
Executable Location:	/C:/Users/Karthik/Downloads/9edb73376e0b1c388ea94bc00b634603f069aa46be7099cbd2b5a48d7a9a59319077f2719e9
Executable MD5:	9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688
Executable SHA256:	file:///C:/Users/Karthik/Downloads/9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688
FSRL:	file:///C:/Users/Karthik/Downloads/9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688
PE Property[Assembly Version]:	1.0.0.0
PE Property[Comments]:	
PE Property[CompanyName]:	Microsoft
PE Property[FileDescription]:	Vector International
PE Property[FileVersion]:	1.0.0.0
PE Property[InternalName]:	yHoL.exe
PE Property[LegalCopyright]:	Copyright © Microsoft 2024
PE Property[LegalTrademarks]:	
Additional Information	
Loading file:///C:/Users/Karthik/Downloads/9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315afeab43688	
Delay imports detected...	

Entry Point & Main Function :

_CorExeMain() confirms it's a .NET-based executable.

The main logic is embedded in **CIL (Common Intermediate Language) bytecode**.

Possible use of **obfuscation or packing** to evade analysis.

Import Descriptor Table :

The **IMAGE_IMPORT_DESCRIPTOR** section contains references to **suspicious DLLs**.

Some fields hold **unusual values (0x2000)**, indicating **import table manipulation**.

Malware may be using **runtime API resolution** to conceal critical functions.

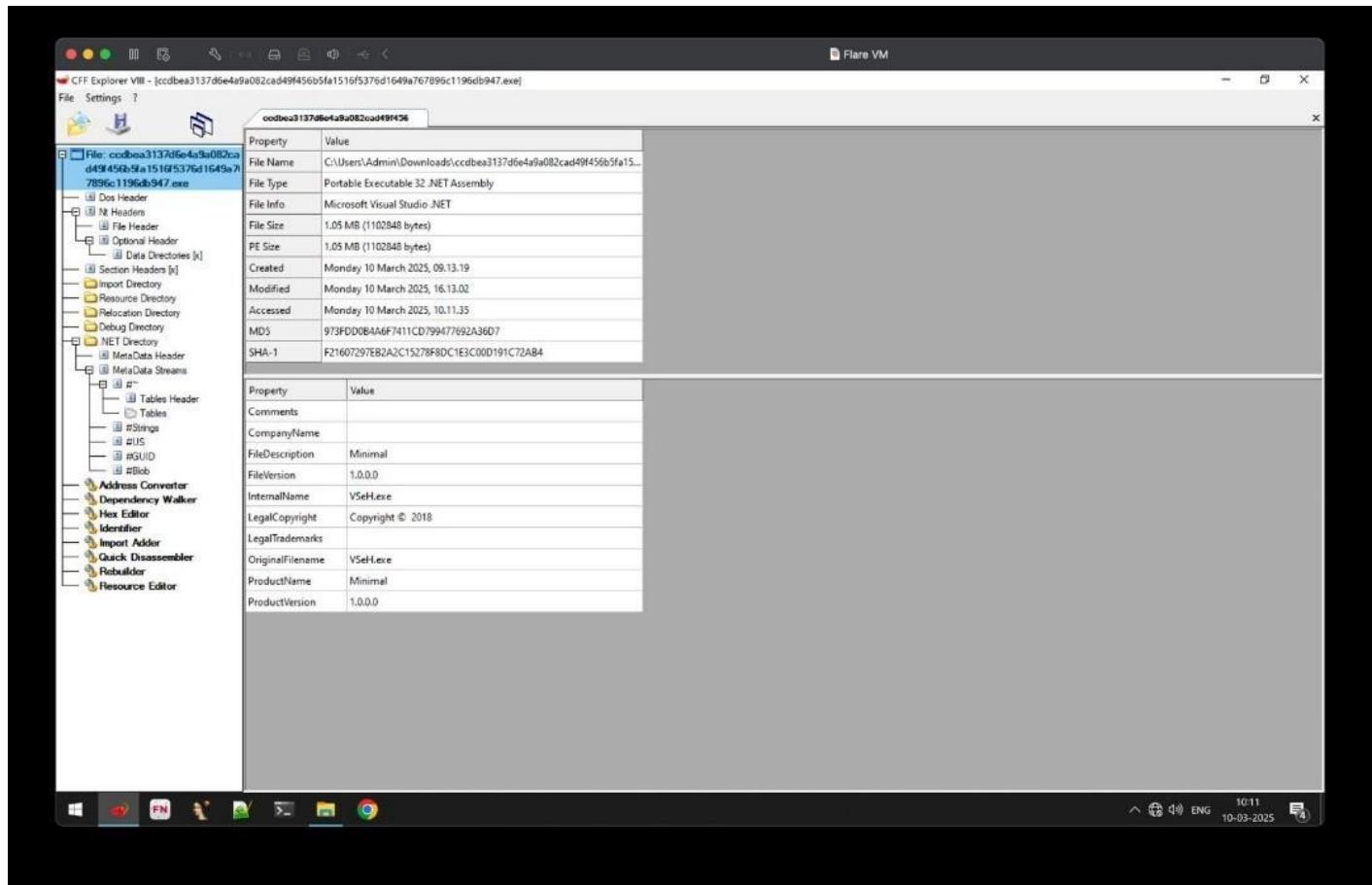
The screenshot shows the CodeBrowser interface with several panes:

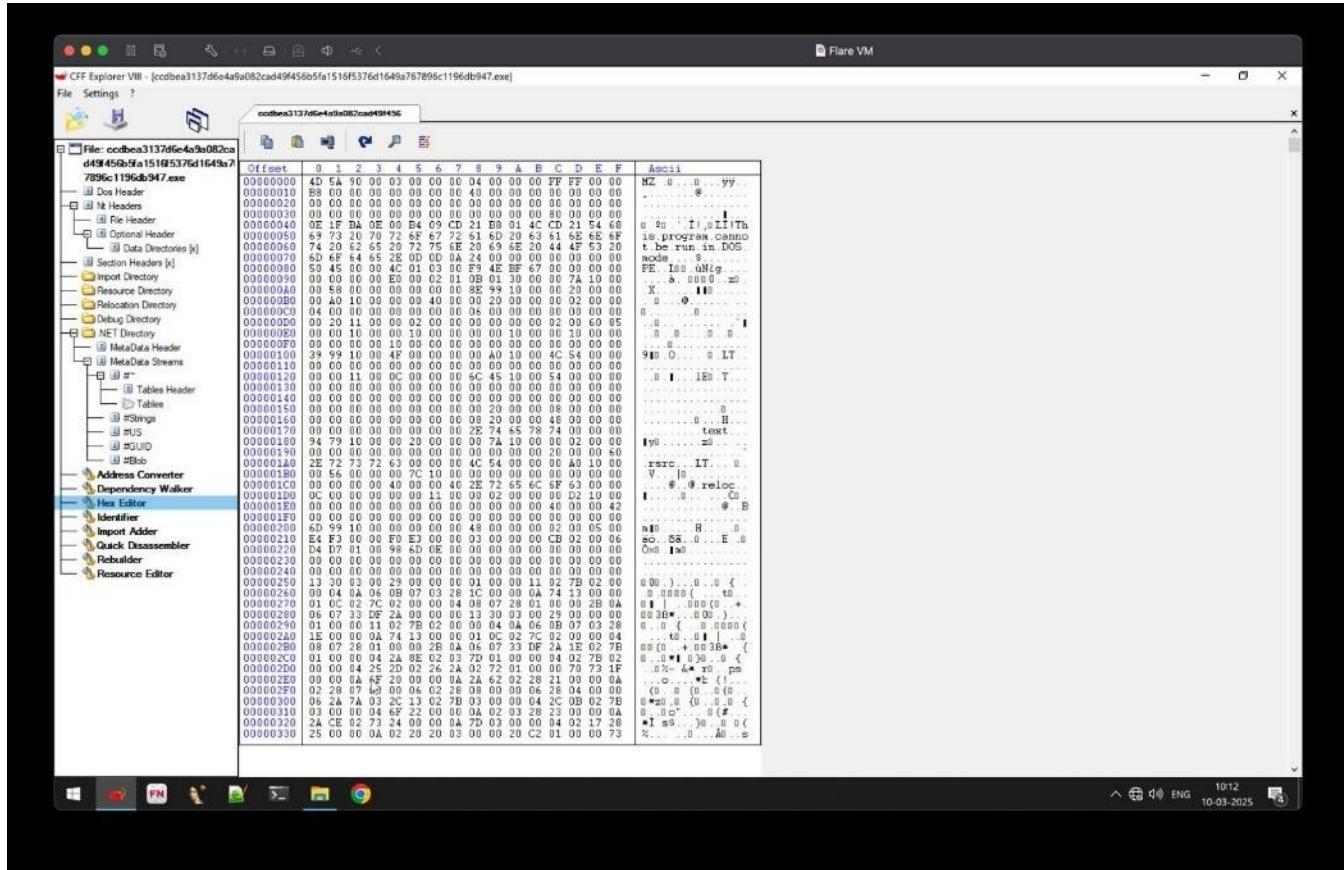
- Program Trees**: Shows the file structure of the executable.
- Symbol Tree**: Displays symbols such as `CLT_METADATA_HEADER`, `CLT_Stream_#Bob`, `CLT_Stream_#GUID`, etc.
- Listing**: Shows the assembly listing for the file `9edb73376e0b1c388ea94bc00b634603f069aa46be7ca5bab0e315feeb43688.exe`. It includes fields like `IMAGE_DOS_HEADER` and `IMAGE_NT_HEADERS`.
- Decompile**: Shows the decompiled CIL code for the `_CorExeMain()` function.
- Data Type Manager**: Shows the data types defined in the executable, including `CLT_METADATA_HEADER` and `CLT_Stream`.
- Console - Scripting**: An empty console window.
- Status Bar**: Shows the current temperature (27°C), search bar, system icons, and date/time (01-04-2025).

6. CFF Explorer

CFF Explorer is a powerful tool for analyzing Portable Executable (PE) files. To examine a malware sample, open CFF Explorer and load the executable. Navigate through headers, imports, and sections to inspect its structure.

Check the **Optional Header** for subsystem info and security flags, and review **Import Directory** to identify external dependencies. This helps in detecting suspicious API calls and potential evasion techniques.





This image displays an analysis of a **.NET Portable Executable** using CFF Explorer. It reveals key attributes like **file version**, **internal name**, and **hash values (MD5 & SHA-1)**. The presence of a **MetaData Header** suggests it's a .NET-based file

Shows imports like `GetAsynckeyState`, `WriteFile`, and `IsDebuggerPresent`, which confirm its keylogging and anti-analysis functions.

The `.text` section (likely containing executable code) and `.rdata` section (likely containing strings and imports) are visible in the dump..reloc suggests the presence of relocation data, indicating it might be dynamically loaded into memory.

Dynamic Analysis

Dynamic analysis is about running a program in a controlled setup to study its real behavior. This helps in detecting malware actions like process injection, registry changes, and network activity.

Using Hybrid Online Malware Sandboxing

VMRay

VMRay was used to analyze the behavior of a suspicious executable in a controlled sandbox environment. It helped trace the process tree, detect code injection activities, and identify keylogging capabilities by extracting critical artifacts like encryption keys, SMTP credentials, and C2 server URLs.

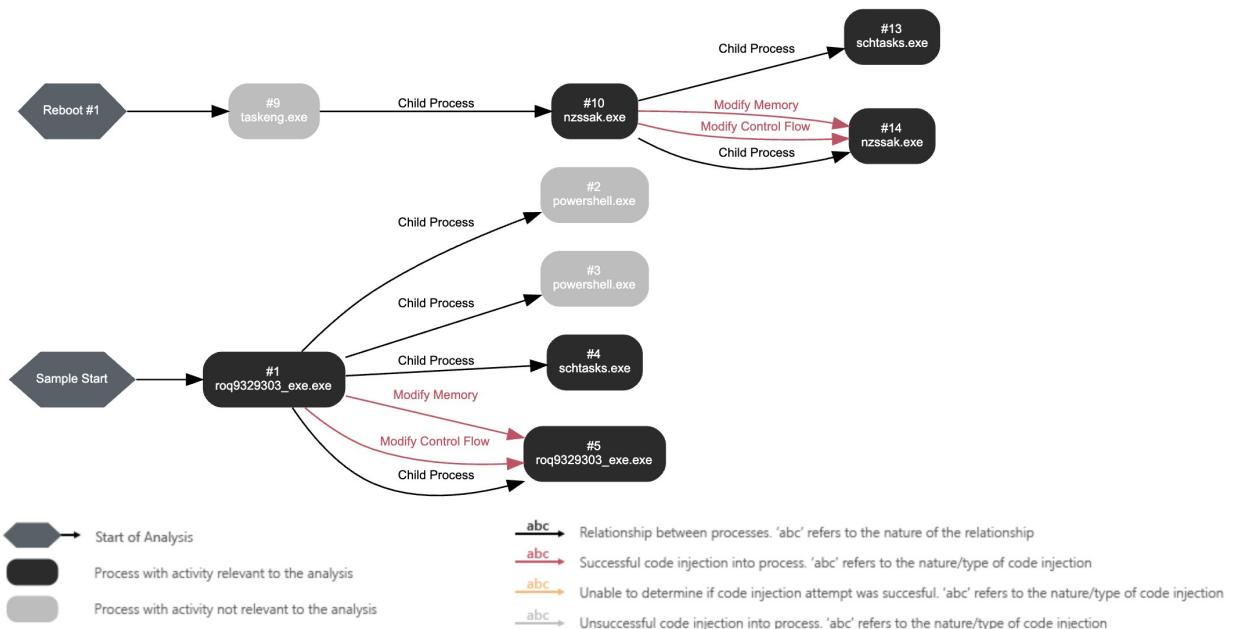
VMRay Threat Identifiers (22 rules, 72 matches)

Score	Category	Operation	Count	Classification
5/5	YARA	Malicious content matched by YARA rules	2	Spyware
5/5	Extracted Configuration	VIPKeylogger configuration was extracted	1	Spyware
4/5	YARA	Malicious content matched by YARA rules	2	Spyware
4/5	Reputation	Malicious file detected via reputation	1	-
4/5	Reputation	Malicious host or URL detected via reputation	3	-
4/5	Defense Evasion	Modifies Windows Defender configuration	2	-
3/5	Network Connection	Performs DNS request for known DDNS domain	1	-
2/5	Discovery	Searches for sensitive mail data	3	-
2/5	Data Collection	Reads sensitive mail data	2	-
2/5	Discovery	Searches for sensitive browser data	22	-

YARA Matches (6)

Ruleset Name	Rule Name	Rule Description	File Type	File Name	Classification	Score	Actions
Malware	VIPKeylogger	VIPKeylogger strings	Memory Dump	-	Spyware	5/5	...
Malware	VIPKeylogger	VIPKeylogger strings	Memory Dump	-	Spyware	5/5	...
Malware	VIPKeylogger	VIPKeylogger strings	Memory Dump	-	Spyware	5/5	...
Generic	InfoStealer_Generic	Generic InfoStealer detection	Memory Dump	-	Spyware	4/5	...
Generic	InfoStealer_Generic	Generic InfoStealer detection	Memory Dump	-	Spyware	4/5	...
Generic	InfoStealer_Generic	Generic InfoStealer detection	Memory Dump	-	Spyware	4/5	...

Monitored Processes



OLLYDBG

OllyDbg was utilized to inspect the dynamic behavior of the suspected malware. It revealed the loading of several standard DLLs, including `mscoree.dll` and `mscoreei.dll`, suggesting the application is built using .NET.

The debugger logs also highlighted thread creation and module loading activities within system directories like `C:\Windows\System32`, which are indicative of potential malware behavior—especially linked to threats like Snake Keylogger.

The OllyDbg CPU window displays the assembly code and registers for the main thread of the `KERNELBA` module. The assembly code shows various instructions, including `ADD BYTE PTR DS:[EBX+ESI]*2+EB1,CL` and `JNC EBK`. The Registers (FPU) pane shows the current state of CPU registers. The Registers pane also includes a stack dump section showing memory locations from `0049C000` to `0049C080`. The Registers pane also includes a stack dump section showing memory locations from `0049C000` to `0049C080`.

Exception 04242420 - use Shift+F7/F8/F9 to pass exception to program

L Log data

Address	Message
	Command Line plugin v1.10 Written by Oleh Yuschuk
File 'C:\Users\admin\Downloads\1bb6d872af87193593934e811a5643f1533aa52c6d6efbbcb9f87e20	
0049B90E	New process with ID 000010E0 created
00400000	Main thread with ID 00001FC4 created
74100000	Module C:\Users\admin\Downloads\1bb6d872af87193593934e811a5643f1533aa52c6d6efbbcb9f87e2
74240000	Module C:\Windows\SYSTEM32\apphelp.dll
76460000	Module C:\Windows\System32\MSCOREE.DLL
76460000	Module C:\Windows\System32\KERNEL32.dll
76630000	Module C:\Windows\System32\KERNELBASE.dll
77460000	Module C:\Windows\SYSTEM32\ntdll.dll
76AD0000	Module C:\Windows\System32\ADVAPI32.dll
75ED0000	Module C:\Windows\System32\msvcrt.dll
77495940	New thread with ID 00001060 created
76010000	Module C:\Windows\System32\sechost.dll
77495940	New thread with ID 000006A0 created
76550000	Module C:\Windows\System32\RPCRT4.dll
72DE0000	Module C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscoreei.dll
75E80000	Module C:\Windows\System32\SHLWAPI.dll
742A0000	Module C:\Windows\SYSTEM32\kernel.appcore.dll
75150000	Module C:\Windows\SYSTEM32\VERSION.dll
706B0000	Module C:\Windows\Microsoft.NET\Framework\v4.0.30319\clr.dll
76D10000	Module C:\Windows\System32\USER32.dll
76320000	Module C:\Windows\System32\win32u.dll
77495940	New thread with ID 0000134C created
76EB0000	Module C:\Windows\System32\GDI32.dll
72D20000	Module C:\Windows\SYSTEM32\wcvtbase_clr0400.dll
742B0000	Module C:\Windows\SYSTEM32\VCRUNTIME140_CLR0400.dll
769E0000	Module C:\Windows\System32\gdi32full.dll
75D00000	Module C:\Windows\System32\msvcp_win.dll
76340000	Module C:\Windows\System32\wcruntime.dll
75CD0000	Module C:\Windows\System32\IMM32.DLL
7276E920	Function 04242420

Joe Sandbox

Joe Sandbox was employed to perform a static and dynamic analysis of the suspicious executable rROQ9329303.exe. It provided crucial metadata such as the file hashes (MD5, SHA1, SHA256) and tagged the sample as associated with **Snake Keylogger**, indicating its malicious nature. The use of tags like `exe`, `SnakeKeylogger`, and `user-Porcupine` further supports its classification as a known threat, enabling threat intelligence correlation across systems.

Mitre Att&ck Matrix

Reconnais...	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Gather Victim Identity Information	Acquire Infrastructure	Valid Accounts	1 Scheduled Task/Job	1 DLL Side-Loading	1 DLL Side-Loading	1 Disable or Modify Tools	1 OS Credential Dumping	1 File and Directory Discovery	Remote Services	1 Archive Collected Data	1 Web Service	Exfiltration Over Other Network Medium	Abuse Accessibility Features
Credentials	Domains	Default Accounts	Scheduled Task/Job	1 Scheduled Task/Job	1 Process Injection	1 Deobfuscate/decode Files or Information	LSASS Memory	1 System Information Discovery	Remote Desktop Protocol	1 Data from Local System	3 Ingress Tool Transfer	Exfiltration Over Bluetooth	Network Denial of Service
Email Addresses	DNS Server	Domain Accounts	At	Logon Script (Windows)	1 Scheduled Task/Job	3 Obfuscated Files or Information	Security Account Manager	1 Security Software Discovery	SMB/Windows Admin Shares	1 Email Collection	1 Encrypted Channel	Automated Exfiltration	Data Encrypted for Impact
Employee Names	Virtual Private Server	Local Accounts	Cron	Login Hook	Login Hook	1 Software Paking	NTDS	1 Process Discovery	Distributed Component Object Model	Input Capture	1 Non-Standard Port	Traffic Duplication	Data Destruction
Gather Victim Network Information	Server	Cloud Accounts	Launchd	Network Logon Script	Network Logon Script	1 Timestamp	LSA Secrets	3 Virtualization/Sandbox Evasion	SSH	Keylogging	3 Non-Application Layer Protocol	Scheduled Transfer	Data Encrypted for Impact
Domain Properties	Botnet	Replication Through Removable Media	Scheduled Task	RC Scripts	RC Scripts	1 DLL Side-Loading	Cached Domain Credentials	1 Application Window Discovery	VNC	GUI Input Capture	2 Application Layer Protocol	Data Transfer Size Limits	Service Stop
DNS	Web Services	External Remote Services	Systemd Timers	Startup Items	Startup Items	1 Masquerading	DCSync	1 System Network Configuration Discovery	Windows Remote Management	Web Portal Capture	Commonly Used Port	Exfiltration Over C2 Channel	Inhibit System Recovery

```

    {
      "Exfil Mode": "SMTP",
      "Email ID": "587",
      "Password": "matias@repsider.com.ar",
      "Host": "Lanus090909$",
      "Port": "587"
    }
  
```

The configuration data of **VIP Keylogger** reveals that it uses **SMTP** for exfiltrating stolen information. Credentials such as matias@repsider.com.ar with the password Lanus090909\$ are hardcoded, along with the SMTP port 587. This shows the malware is configured to automatically send logged keystrokes to an attacker-controlled email server, making it a clear example of credential theft and data exfiltration via email.

Process Monitor

Process Monitor was utilized to observe real-time system-level behavior of the suspicious executable roq9329303_exe.exe. It revealed the creation of multiple child processes such as sctasks.exe and powershell.exe, often associated with persistence and script execution. The tool also captured frequent file and registry operations in critical directories like System32 and AppData, along with network activity over port 587, reinforcing its role in detecting keylogger behavior and data exfiltration attempts via SMTP.

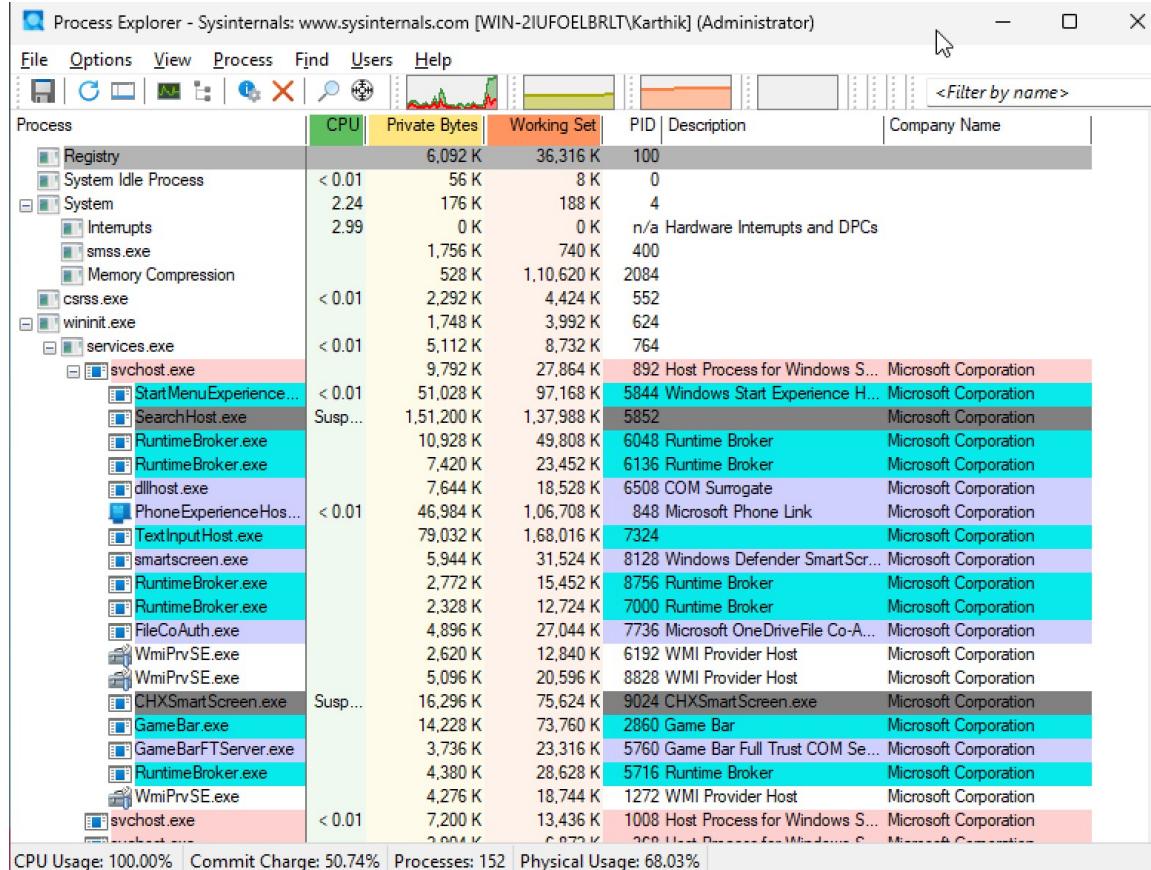
Time ...	Process Name	Sess...	PID	Arch...	Operation	Path	Result	Detail	Date & Time	Image Path
12:42...	svchost.exe	0	3132	64-bit	RegCloseKey	HKLM\SYSTEM\Setup	SUCCESS		5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Desired Access: M...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Query: HandleTag...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM\SYSTEM\Setup	SUCCESS	Desired Access: R...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS		5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM\SYSTEM\Setup	SUCCESS	Type: REG_DWO...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM\SYSTEM\Setup	SUCCESS		5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Desired Access: M...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Query: HandleTag...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM\SYSTEM\Setup	SUCCESS	Desired Access: R...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS		5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegQueryValue	HKLM\SYSTEM\Setup\SystemSetupIn...	SUCCESS	Type: REG_DWO...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegCloseKey	HKLM\SYSTEM\Setup	SUCCESS		5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,766,144...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,864,448...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 11,190,272...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,856,256...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,749,760...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,897,216...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,782,528...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,823,488...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,807,104...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,733,376...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 23,044,096...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,880,832...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,692,416...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,651,456...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,889,024...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 22,036,480...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 23,543,808...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,790,720...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,774,336...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,954,560...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,643,264...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 20,332,544...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,757,952...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,921,792...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,831,680...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Repository	SUCCESS	Offset: 21,848,064...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Desired Access: M...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...	5/25/2021 12:42...	C:\Windows\system...
12:42...	svchost.exe	0	3132	64-bit	RegQueryKey	HKLM\System\Setup	SUCCESS	Desired Access: R...	5/25/2021 12:42...	C:\Windows\system...

Showing 125,034 of 366,792 events (34%)

Backed by virtual memory

Process Explorer

Process Explorer displayed the creation of multiple child processes under the main executable but only for a second before the process was hid, and this makes the mitigation tough.



Mitigation and Removal Steps:

- Open Process Explorer or Task Manager.
- Identify and terminate all instances of the keylogger executable.
- Use Regshot to confirm that no permanent registry changes were made.
- Manually delete the executable file from the desktop or its original location.
- If pop-ups or symptoms persist, revert the VM to a pre-execution snapshot.

Additional Solutions and Preventive Measures:

1. **Reboot in Safe Mode:** If pop-ups prevent user interaction, reboot the system in Safe Mode to disable non-essential services and processes. From there, navigate to the file's location and delete it manually.
2. **Use an Antivirus Tool:** Many antivirus programs detect this file as prankware. Running a scan with a reputable antivirus tool can assist in automatic detection and removal.
3. **Create a System Restore Point:** Before executing suspicious files, always create a restore point. In cases where the malware alters settings or causes UI denial, system restore can revert the machine to a prior, clean state.
4. **Use AppLocker or Software Restriction Policies:** Prevent execution of unauthorized applications in enterprise or lab environments by enforcing these group policies.
5. **Educate on Safe Handling:** Ensure malware samples are only run in isolated virtual environments with snapshots and restricted access. Disable file sharing between host and guest VMs to avoid accidental host infection.
6. **Monitor Network Traffic:** Although this sample does not exhibit network behavior, using tools like Wireshark can ensure there is no covert data exfiltration in other malware cases.

Conclusion:

The keylogger in question is a malicious software designed to record keystrokes and exfiltrate sensitive data. While its primary function is covert data theft, it also demonstrates key malware behaviors such as code injection, process spawning, and file manipulation. The keylogger might not cause immediate visible damage, but its potential for data exfiltration and compromising user privacy is significant. This analysis highlights the importance of using static tools like **PEiD** and **OllyDbg** to analyze executable files and dynamic tools like **ProcMon** and **Process Explorer** to observe real-time behaviors. The sample also showcases how seemingly simple malware can exhibit complex tactics, techniques, and procedures (TTPs) often associated with advanced threats.

This project serves as a practical case study in identifying, analyzing, and documenting

malware in a controlled lab environment, further emphasizing the importance of proactive cybersecurity measures.