REPUBLIC OF ALBANIA
NATIONAL AUTHORITY FOR CYBER SECURITY
DIRECTORATE OF CYBER SECURITY ANALYSIS

Technical Analysis of the Malicious File

Lumma Stealer

Version: 1.0
Data: 02/07/2025

CONTENTS

LIST OF FIGURES

This report contains limitations and should be interpreted with caution.

Some of these limitations include:

Phase One – Information Sources:
This report is based on the information available at the time of its preparation. As such, certain aspects may differ from current or future developments.

Phase Two – Depth of Analysis:
Due to resource constraints, some components of the malicious file may not have been analyzed in depth. Any unknown or additional information may lead to updates or changes in the report's conclusions.

Phase Three – Information Security:
To protect sources and confidential information, certain details may be redacted or omitted from the report. This decision is made to preserve the integrity and security of the data used.

The National Authority for Cyber Security (AKSK) reserves the right to modify, update, or revise any part of this report without prior notice.

This document does not constitute a final or definitive report.
The findings are based on the information available at the time of investigation and analysis. No guarantee can be made regarding potential changes or updates to the reported information in the future. The authors of the report assume no responsibility for any misuse or decisions based on the content of this document.

Technical Information

Lumma Stealer (also known as **LummaC2**) is a malicious file belonging to the **Infostealer** category, designed to steal sensitive data from compromised devices. This malware has been developed and distributed across various forums (including the dark web), often provided as **Malware-as-a-Service (MaaS)**. It is equipped with advanced capabilities to collect information such as: browser credentials, cookies, autofill data, password manager records, cryptocurrency wallets, and profiles from well-known applications like **Telegram** and **Discord**.

Analiza e skedarit Setup.exe

setup.exe is an executable file with the hash value: **974a6af4f91d5d99d7501059907d64aa3882981dab350ad3f654ece13ed18f1f.** If this file is renamed and its extension is changed from .exe to .7z, and then extracted, it reveals a file with the .nsi extension and a directory named $TEMP.

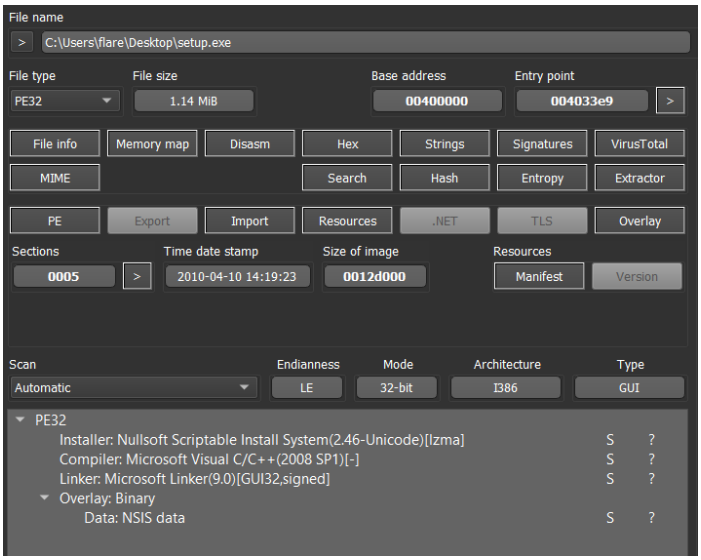


Figure 1: Nullsoft Scriptable

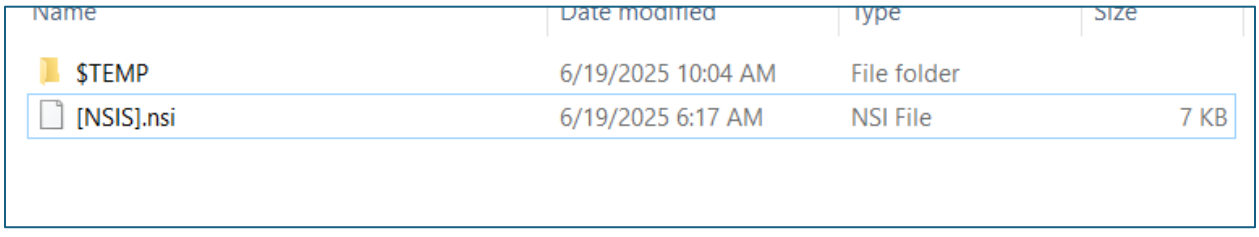


Figure 2: Extraction of setup.exe

If we analyze the file [NSIS].nsi, we will observe that it is an **NSIS (Nullsoft Scriptable Install System)** script — a system commonly used for creating installers for Windows. This file is heavily obfuscated and contains variable names with no meaningful context. However, it clearly uses commands such as ExecShell ... SW_HIDE, which are designed to execute files without

displaying any output to the user — a widely used technique in the distribution of malicious files.

```
label_137:
  IntOp $1 $1 + 1
  IfRebootFlag label_140 label_140
  ExecShell open ChaseDefeat LatelyNo SW_HIDE    ; "open ChaseDefeat"
label_140:
  Nop
  IntCmp $1 12754 label_144 label_114 label_144
  GetTempFileName $9 MindsWooden
  GetTempFileName $9 ShoppingcomFleshStatusKimPhotographDestruction
label_144:
  StrCpy $R8 "Invited Surprise Relates Appropriations Strengths Todd Deals "
  IfErrors label_148 label_148
  RegDLL QuotesPrecipitation
  SetErrors
label_148:
  GetInstDirError $R7
SectionEnd
```

Figure 3: Use of ExecShell and SW_HIDE

In the **$TEMP** directory, several files with the **.midi** extension can be found, whose names are also referenced in the previously mentioned **.nsi** script.

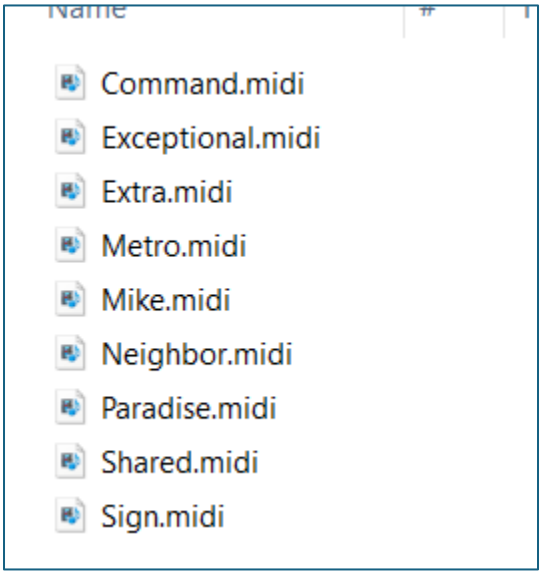


Figure 4: .midi files

The file that contains hidden text — and is in fact a **batch (.bat) file**, not a true .midi file — is named **Mike.midi**. Within this file, there are several key commands such as set, start, and findstr, which clearly indicate that we are dealing with a **batch script**.
This assumption is confirmed in the original .nsi file, where a line of code appears as follows:
ExecShell open $SYSDIR\c$_2_d $_3_M$_4_e.midi.bat SW_HIDE.

```
hDLQLearning
set %Fm%p ="MZ" > %Stands%\%JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd% <n%Lou%l
sTrjValid Marcus Cage Fwd Taken Gg Toolkit Salem Hwy
milAirfare Sms Require Nigeria Ntsc Better
dNeoWriters Promotes Mercedes Sku Curious Reductions Unlikely Approval
findstr %Fm%V "FCC" Sailin%Enrolled% >> %Stands%\%JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd%
mPfCooperative Anticipated Size Charge Cult Photoshop Surely Urban Bridal
EgbiOrganizational
tlrxCrown Objective
NkBin Orientation
XfLTutorials Ladies Nominations Slide Leeds Variable Arch Myth
kXgOSri
%Canon%%Battle%py %Fm%b %Stands%\%JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd% + %Aaa%axi + V%Er
```

Figure 5: Mike.midi.bat

set %Fm%p ="MZ": Creates a variable with the value "MZ" (the beginning of every executable in Windows).

> %Stands%\%JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd%: Writes this value into a file at a path constructed using variables. This file may be a **dropper**, meaning a file that will later expand into shellcode or a full payload.

MZ is the **magic number** for .exe files in Windows, so this is a first step in building an executable on disk from a script.

If we attempt to execute the file Mike.midi.bat, several other files will appear in the directory, such as:

Bahrain, Couple, Disney, Frame, Grew, Hostles, Maintain, Taxi, Turtle, Vg — which for the moment have no specific meaning.

To see exactly what happens with the .bat file, we need to modify its code by adding parts such as **echo** and **pause**.

```
FLARE-VM Thu 06/19/2025  9:48:22.18
C:\Users\flare\Desktop\974a6af4f91d5d99d7501059907d64aa3882981dab350ad3f654ece13ed18f1f.exe\$TEMP>copy /b .\Metroo.midi + .\Commando.midi + .\Shared.mid
di + .\Exceptionalonal.midi + .\Neighborghbror.midi AeKGyRcAMwUmEbxrKMkrfnYhKy
.\Metroo.midi
.\Commando.midi
.\Shared.midi
.\Paradise.midi
.\Signgn.midi
.\Exceptionalonal.midi
.\Neighborghbror.midi
       1 file(s) copied.
```

Figure 6: Creation of New File (1)

Thus, a file is dynamically created, resulting from the merging of several files into a single one. Additionally, in the next phase, the merging of other previously shown files will also be observed.

```
FLARE-VM Thu 06/19/2025  9:47:57.35
C:\Users\flare\Desktop\974a6af4f91d5d99d7501059907d64aa3882981dab350ad3f654ece13ed18f1f.exe\$TEMP>copy /b 122648\JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd + Taxi + Vg + Frame + Maintain +
ey + Hostels + Turtle + Couple + Bahrain + Grew 122648\JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd
122648\JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd
Taxi
Vg
Frame
Maintain
Disney
Hostels
Turtle
Couple
Bahrain
Grew
       1 file(s) copied.
```

Figure 7: Creation of New File (2)

Thus, during execution, two files are dynamically created — namely **AeKGyRcAMwUmEbxrKMkrfnYhKy** and **JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd** — along with a directory named **122648**.
If we follow the execution, we will also observe a "start" command, where it is evident that the file **JNEoIBWjfqiMdvrUUInXwxnOkIbkjyMUWd** takes the file **AeKGyRcAMwUmEbxrKMkrfnYhKy** as a parameter.

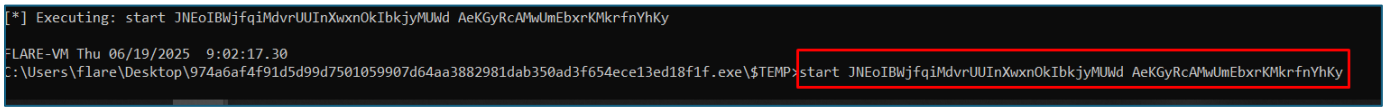


Figure 8: Start of File Execution

To understand what is happening, we rename the first file and change its extension to **.exe**, and we observe that it automatically adopts the **AutoIT** application icon.
AutoIT is a simple scripting language that allows:

- Automation of tasks in Windows
- Simulation of keyboard, mouse, and window interactions
- Creation of GUIs (graphical windows) and installers

However, it is often abused for the creation of malicious files.

*** How does it work?**

- Scripts are written in **.au3** files
- They can be compiled into **.exe** using **Aut2Exe** (to convert them into a Windows application)
- They can be integrated with the **SciTE editor** for easier programming

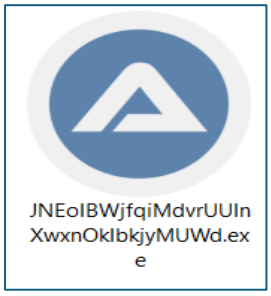
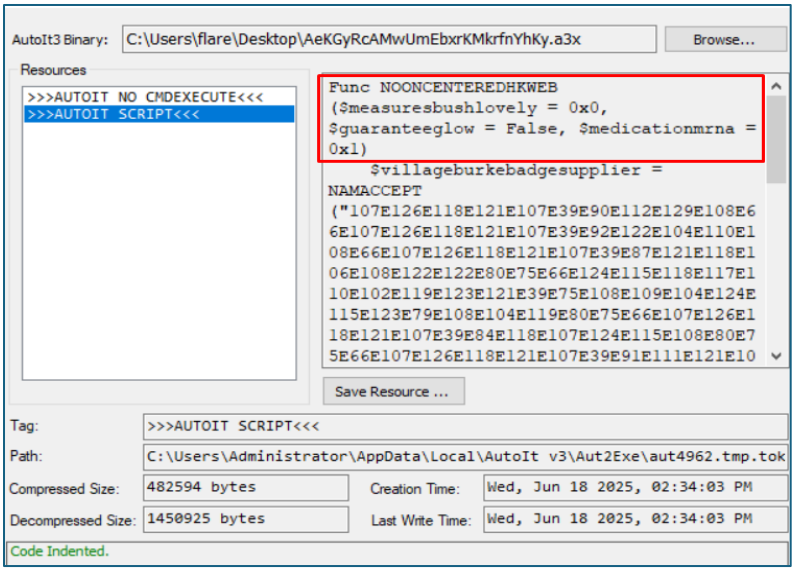


Figure 9: AutoIT

This gives us a clearer idea of the role of the second file, **AeKGyRcAMwUmEbxrKMkrfnYhKy**.
While analyzing the file's strings, we observe a specific string segment: 00075FD2 AU3!EA06, indicating that this is a file compiled with **AutoIT**.
Therefore, we need to **decompile** it in order to analyze the code being executed.



Figur 10: Detection of the AutoIT File Type

The decompiled code is heavily obfuscated, and many functions serve the purpose of hiding logic and making the analysis more difficult.
The most frequently used function is **NAMACCEPT**, which takes two parameters.
To observe the behavior of this function, we write our own identical code in **Python** to display the outputs.
Additionally, the file contains calls to DLLs (Dynamic Link Libraries) using **DllCall**.

```
Func NAMACCEPT($interference, $rebates)
    $richmondsport = ""
    While 0x34b
        $franchisebeerdidheat = 0x7586
        Switch $franchisebeerdidheat
            Case 0x7584
                Exp(0x1dd5)
                Chr(0xcae)
                IsDeclared("tackle#nowhere#orientation#")
                Chr(0x7de)
                $franchisebeerdidheat = $franchisebeerdidheat + 0xcc476 / 0xcc476
            Case 0x7585
                Chr(0xfeb)
                Cos(0x1b34)
                Exp(0x1602)
                PixelGetColor("Technician#", "Technician#")
                $franchisebeerdidheat = $franchisebeerdidheat + 0x9409e / 0x9409e
            Case 0x7586
                $birth = Call(StringReverse("tilpSgnirtS"), $interference, "E", 0x2)
                ExitLoop
            Case 0x7587
                Exp(0x478)
                ObjGet("Buffalo    Enquiry    Preston    Costumes    ")
                IsDeclared("Equivalent=Upgrades=Cartridge=Contact=")
                IsDeclared("Smoking Default Font ")
                Log(0x1198)
                IsDeclared("MENT*DEPTH*")
                $franchisebeerdidheat = $franchisebeerdidheat + 0x57309 / 0x57309
        EndSwitch
    WEnd
    For $randepend = 0x9fcc + 0xffff6034 To Call("UBound", $birth) + 0xffffffff
```

Figure 11: NAMACCEPT Function

```python
def namaccept(encoded_str: str, key: int) -> str:
    parts = encoded_str.split("E")
    decoded = ""
    for part in parts:
        if not part.strip().isdigit():
            continue
        try:

            code = (int(part) - key) & 0xFFFF
            decoded += chr(code)
        except:
            decoded += "?"
    return decoded

# Example usage
example = "113E107E120E116E107E114E57E56E52E106E114E114"
key = 0x7 + 0xffffffff

decoded = namaccept(example, key)
print("Decoded string:", decoded)
```

Figure 12: Reverse NAMACCEPT.py

This function takes an encoded string in the form of numbers separated by "E", along with a key. It subtracts the key from each number and converts the result into a textual character to recover the original message.

During testing with various values, the following were identified:
dword Size;
dword Usage;
dword ProcessID;
ulong_ptr DefaultHeapID;
dword ModuleID;
dword Threads;
dword ParentProcessID;
long PriClassBase;
dword Flags;
char ExeFile[260];

1. The PROCESSENTRY32 Structure in Shellcode
When a shellcode includes this structure — often encoded or hidden — its purpose is to analyze the processes that are active in the system.
Once the attacker has this list, they can choose a process in which to inject their shellcode stealthily, e.g., into a legitimate process like **explorer.exe** or **svchost.exe**.

This is a common practice before performing a shellcode injection, where the attacker attempts to inject their code into a trusted system process (e.g., **explorer.exe**) to ensure **persistence** or to **evade detection** by security mechanisms.

The presence of the field **ExeFile[260]** supports this hypothesis, as it identifies the name of the executable of the target process for injection.

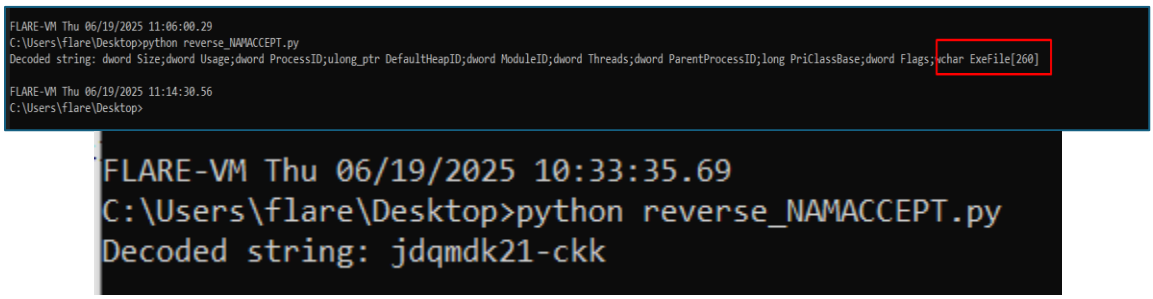


Figure 13: jdqmdk21-ckk

jdqmdk21-ckk is simply an encoded string that hides the name of the standard Windows library: **Windows: kernel32.dll**.
This string has been observed in a segment of Meterpreter shellcode https://blog.restkhz.com/post/glance-at-shellcode-3.

```
int main() {
    unsigned char shellcode[] = "\xda\xca......blabla";

    char *va = "Uhqst`k@kknb";
    char *ct = "Bqd`sdSgqd`c";
    char *mc = "ldlbox";
    char *k32 = "jdqmdk21-ckk";
    char *msvcrt = "lrubqs-ckk";

    va = shift_string_by_one(va);
    ct = shift_string_by_one(ct);
    mc = shift_string_by_one(mc);
    k32 = shift_string_by_one(k32);
    msvcrt = shift_string_by_one(msvcrt);

    size_t length = sizeof(shellcode);
    increment_hex_string(shellcode, length);

    HMODULE kernel32_dll = LoadLibrary(k32);

    HMODULE msvcrt_dll = LoadLibrary(msvcrt);

    VIRTUALALLOC VIALFunc = (VIRTUALALLOC)GetProcAddress(kernel32_dll, va);

    CREATETHREAD CreateThreadFunc = (CREATETHREAD)GetProcAddress(kernel32_dll, ct);
    if (CreateThreadFunc == NULL) {
        printf("Failed to find CT function\n");
        FreeLibrary(kernel32_dll);
        return 1;
    }
```

Figure 14: shellcode

This code is a typical example of a **shellcode loader**, which uses obfuscation techniques to evade detection by antivirus or security systems.

***Purpose of the Code**

This program:

1. **Decodes the hidden strings for Windows API functions (VirtualAlloc, CreateThread, etc.)**
2. **Decodes the shellcode**
3. **Allocates executable memory via VirtualAlloc**
4. **Copies the shellcode into the allocated memory**
5. **Executes the shellcode using CreateThread**

If we continue extracting other strings from our Python code, we will also observe outputs such as **VirtualAddress**, **user32.dll**, indicating that we are dealing with function and library calls from the Windows system itself.

```
FLARE-VM Thu 06/19/2025 11:02:16.35
C:\Users\flare\Desktop>python reverse_NAMACCEPT.py
Decoded string: VirtualAddress
```

Figure 15: VirtualAddress

```
FLARE-VM Thu 06/19/2025 10:51:08.37
C:\Users\flare\Desktop>python reverse_NAMACCEPT.py
Decoded string: user32.dll
```

Figure 16: user32.dll

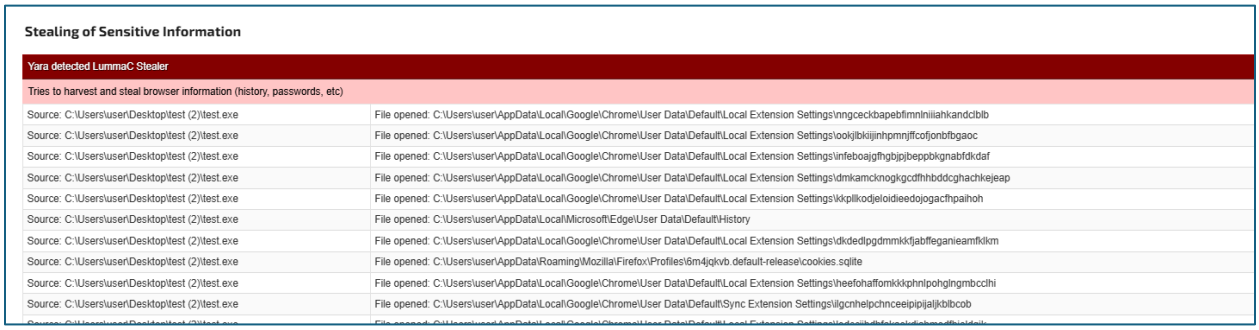
Additionally, in the main part of the decompiled file, we can identify a variable named **$tfjccxwtmwwi**.
This file contains a very large number of character strings, which are concatenated during execution and passed as a parameter to the function **Binary()**. This suggests that we are dealing with a shellcode that is decoded at runtime. The output of the **Binary** function is then passed to several other complex functions, as demonstrated in the following part of the code. **REALLYFAQSSERIALOWNS(REPRESENTEDPERSIAN(GUYANAKGDEFENSIVE(Binary($tfjccxwtmwwi).**

Figure 17: Shellcode Variable

```
4485    WEnd
4486    Func REALLYFAQSSERIALOWNS($hellopatriciaericsson, $explainsunderstandingrepairs, $nottinghamprogrammerscommonorder = NAMACCEPT("107E126E118E114E11
4487        While 0x1a4
4488            $fisheritalia = 0xa0bf
4489            Switch $fisheritalia
4490                Case 0xa0bd
4491                    ProgressOff()
4492                    IsDeclared(NAMACCEPT("73E110E102E108E115E116E120E121E110E104E52E87E106E120E117E116E115E105E106E115E121E120E52E85E119E110E115E121E1
4493                    ProgressOff()
4494                    MemGetStats()
4495                    $fisheritalia = $fisheritalia + 0xcbeb7 / 0xcbeb7
4496                Case 0xa0be
4497                    IsDeclared(NAMACCEPT("89E118E103E120E107E38E82E103E123E116E105E110E107E121E38E89E127E115E118E110E117E116E127E38", 0x6 + 0x0))
4498                    ObjGet(NAMACCEPT("85E114E127E115E118E123E121E38E89E107E103E120E105E110E107E121E38", 0x9 + 0xfffffffd))
4499                    DirGetSize(NAMACCEPT("75E105E118E105E116E69E74E116E69E90E122E120E69", 0x9 + 0xffffffff))
4500                    Chr(0x1af8)
4501                    ProgressOff()
4502                    Exp(0x1276)
4503                    Log(0x37f)
4504                    Exp(0x16da)
4505                    $fisheritalia = $fisheritalia + 0x7ba29 / 0x7ba29
4506                Case 0xa0bf
4507                    $gayvotingvocabularyass = DllStructCreate(NAMACCEPT("99E122E117E102E92", 0x1 + 0x0) & Call(NAMACCEPT("73E112E117E104E121E128E83E10
4508                    ExitLoop
4509            EndSwitch
4510        WEnd
```

Figure 18: REALLYFAQSSERIALOWNS Function

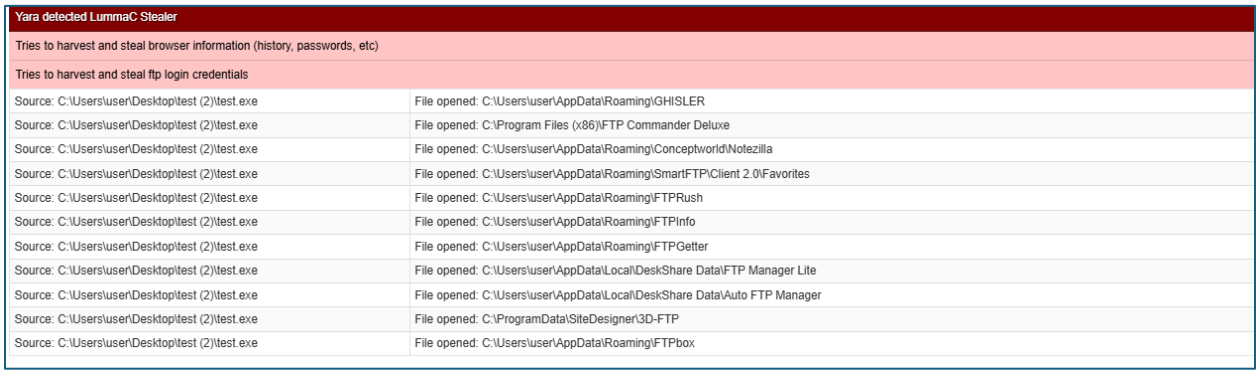
Each function has its own implementation, and the payload is transformed and decoded until it reaches the final stage.
This indicates the highly **polymorphic nature** of this malicious file, which is why a more **automated analysis in a Sandbox** is required to observe what happens in the final phase.
During sandbox analysis, it was observed that there is communication with the domain **drafxc[.]xyz**, which serves as the **Command and Control (C2) server**.
The injection of **Lumma-Stealer** was detected within the legitimate **chrome.exe** process.



Stealing of Sensitive Information

Yara detected LummaC Stealer	
Tries to harvest and steal browser information (history, passwords, etc)	
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\nngceckbapebfimnlniiiahkandclblb
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\ookjlbkiijinhpmnjffcofjonbfbgaoc
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\infeboajgfhgbjpjbeppbkgnabfdkdaf
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\dmkamcknogkgcdfhhbddcghachkejeap
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\kkpllkodjeloidieedojogacfhpaihoh
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Microsoft\Edge\User Data\Default\History
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\dkdedlpgdmmkkfjabffeganieamfklkm
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Mozilla\Firefox\Profiles\6m4jqkvb.default-release\cookies.sqlite
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\heefohaffomkkkphnlpohglngmbcclhi
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Sync Extension Settings\ilgcnhelpchnceeipipijaljkblbcob

Figure 19: Extraction of Browser Credentials



Yara detected LummaC Stealer	
Tries to harvest and steal browser information (history, passwords, etc)	
Tries to harvest and steal ftp login credentials	
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\GHISLER
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Program Files (x86)\FTP Commander Deluxe
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Conceptworld\Notezilla
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\SmartFTP\Client 2.0\Favorites
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\FTPRush
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\FTPInfo
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\FTPGetter
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\DeskShare Data\FTP Manager Lite
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\DeskShare Data\Auto FTP Manager
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\ProgramData\SiteDesigner\3D-FTP
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\FTPbox

Figure 20 Reading of FTP Login Credentials

Tries to steal Crypto Currency Wallets	
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Exodus\exodus.wallet
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Exodus\exodus.wallet
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Ledger Live
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\atomic\Local Storage\leveldb
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Armory
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Coinomi\Coinomi\wallets
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Local\Coinomi\Coinomi\wallets
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Bitcoin\wallets
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Binance
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\com.liberty.jaxx\IndexedDB
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Electrum\wallets
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Electrum-LTC\wallets
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Guarda\IndexedDB
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\DashCore\wallets
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\WalletWasabi\Client\Wallets
Source: C:\Users\user\Desktop\test (2)\test.exe	File opened: C:\Users\user\AppData\Roaming\Daedalus Mainnet\wallets

Figure 21: Reading of Crypto Wallets

Tries to steal Mail credentials (via file / registry access)	
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000001
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000002
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000003
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000004
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_CURRENT_USER\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000001
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_CURRENT_USER\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000002
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_CURRENT_USER\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000003
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_CURRENT_USER\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000004
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000001
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000002
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000003
Source: C:\Users\user\Desktop\test (2)\test.exe	Key opened: HKEY_USERS.DEFAULT\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000004

Figure 22: Reading of Mail Credentials

Communication was identified through a Telegram channel, specifically: **https://t.me/njkwevnfv32v432132**. Afterall the sensitive information of the compromised user is read, it is sent to this channel.

MITRE ATT&CK

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 2 Windows Management Instrumentation	1 DLL Side-Loading	1 1 Process Injection	1 Masquerading	3 OS Credential Dumping	1 Query Registry	Remote Services	1 Email Collection	2 Encrypted Channel	Exfiltration Over Other Network Medium	Abuse Accessibility Features
Credentials	Domains	Default Accounts	Scheduled Task/Job	Boot or Logon Initialization Scripts	1 DLL Side-Loading	2 1 Virtualization/ Sandbox Evasion	LSASS Memory	3 1 Security Software Discovery	Remote Desktop Protocol	4 1 Data from Local System	1 Non-Application Layer Protocol	Exfiltration Over Bluetooth	Network Denial of Service
Email Addresses	DNS Server	Domain Accounts	At	Logon Script (Windows)	1 Extra Window Memory Injection	1 1 Process Injection	Security Account Manager	2 1 Virtualization/ Sandbox Evasion	SMB/Windows Admin Shares	Data from Network Shared Drive	2 Application Layer Protocol	Automated Exfiltration	Data Encrypted for Impact
Employee Names	Virtual Private Server	Local Accounts	Cron	Login Hook	Login Hook	1 Rundll32	NTDS	1 Process Discovery	Distributed Component Object Model	Input Capture	Protocol Impersonation	Traffic Duplication	Data Destruction
Gather Victim Network Information	Server	Cloud Accounts	Launchd	Network Logon Script	Network Logon Script	1 DLL Side-Loading	LSA Secrets	1 1 File and Directory Discovery	SSH	Keylogging	Fallback Channels	Scheduled Transfer	Data Encrypted for Impact
Domain Properties	Botnet	Replication Through Removable Media	Scheduled Task	RC Scripts	RC Scripts	1 Extra Window Memory Injection	Cached Domain Credentials	2 2 System Information Discovery	VNC	GUI Input Capture	Multiband Communication	Data Transfer Size Limits	Service Stop

Indicators of Compromise

BB68002A0DD100649BFB77AEAE875CD084B7EFDCDCA7C5AA2CF7F4C4A6A73C04	**AeKGyRcAMwUmEbxrKMkrfnYhKy**
AA855EB28018AC7AECCB992AF417D7FAD057D19AE43CD1322C6D8A15A99A01B0	**Sign.midi**
C5353C06CED7B539ED6393E0A23CFD13942A3FFA1499BC4CED78EE8FEB18C252	**Neighbor.midi**
B27ECEEDEC33FC3AFF9875CE47132400BB22A8667C66648D57054A65E4BD64D6	**Mike.midi**
A5949E03D197D70506FE25D9BF7D534E54C04424D111BFE0E813714354DE9B22	**Metro.midi**
06200CE96FDD63CD859BEA1A9BCED664195F023BF387E9E2CDC554CCF287A43E	**Extra.midi**
4F0EA7AF73EA52C654329D17805F11BDC83B752A56A73F34C8DCC6D999C7E698	**Stage2.exe**
drafxc[.]xyz	**C2**

Recommendations

The National Cybersecurity Authority recommends:

Immediate blocking of the above-mentioned **Indicators of Compromise** on your protective devices.
• Continuous analysis of logs coming from **SIEM** (Security Information and Event Management) systems.
• Training of non-technical staff regarding **phishing attacks** and methods to avoid infection.
• Installation of **network perimeter devices** that perform deep traffic inspection, relying not only on access control lists but also on traffic behavior (NextGen Firewalls).
• Segmentation of identified systems into separate **VLANs**, applying **access control lists** across the entire network perimeter. Web services must be separated from their databases, and **Active Directory** should reside in its own VLAN.
• Application and use of the **LAPS** technique for Microsoft systems, for managing **Local Administrator Passwords**.
• Application of **traffic filters** for remote access to hosts (employees/third parties/clients).
• Implementation of solutions that filter, monitor, and block malicious traffic between web applications and the internet, such as a **Web Application Firewall (WAF)**.
• Conducting **behavior-based traffic analysis** for endpoint devices, through the use of **EDR/XDR solutions**. This allows for the detection of malicious files not only by signature but also by behavior.
• Designing and implementing an **Identity Access Management (IAM)** solution to control user identities and privileges in real-time, following the **zero-trust** principle.