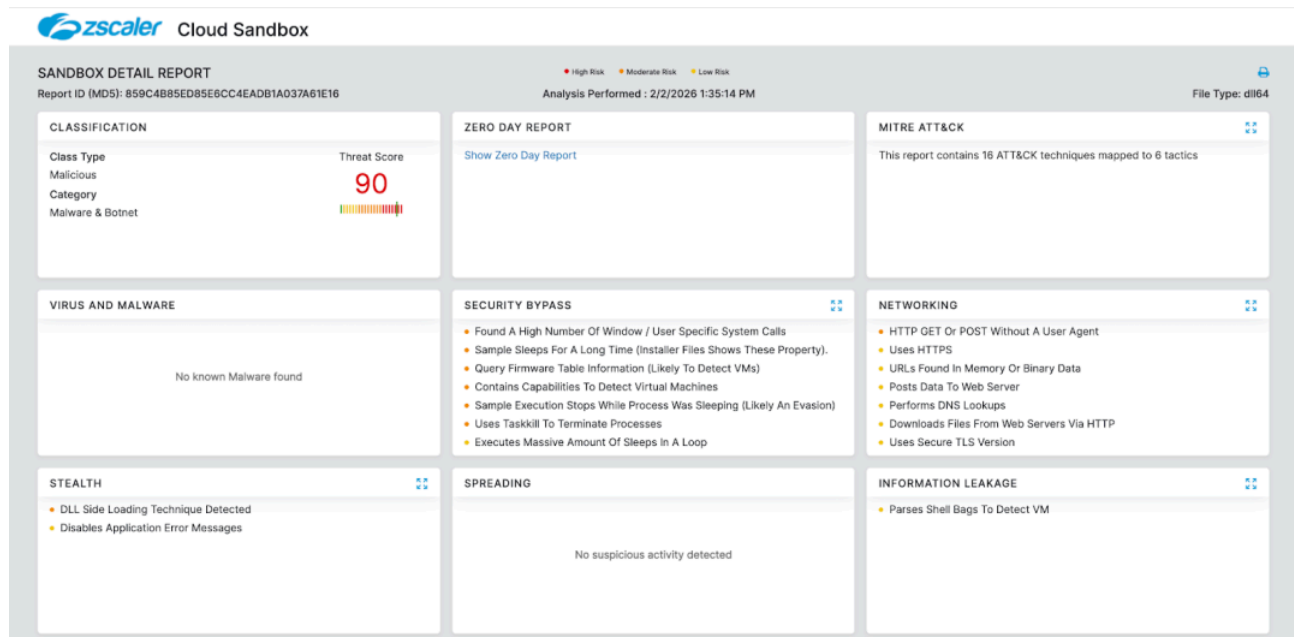


Operation Neusploit: APT28 Uses CVE-2026-21509

zscaler.com/blogs/security-research/apt28-leverages-cve-2026-21509-operation-neusploit

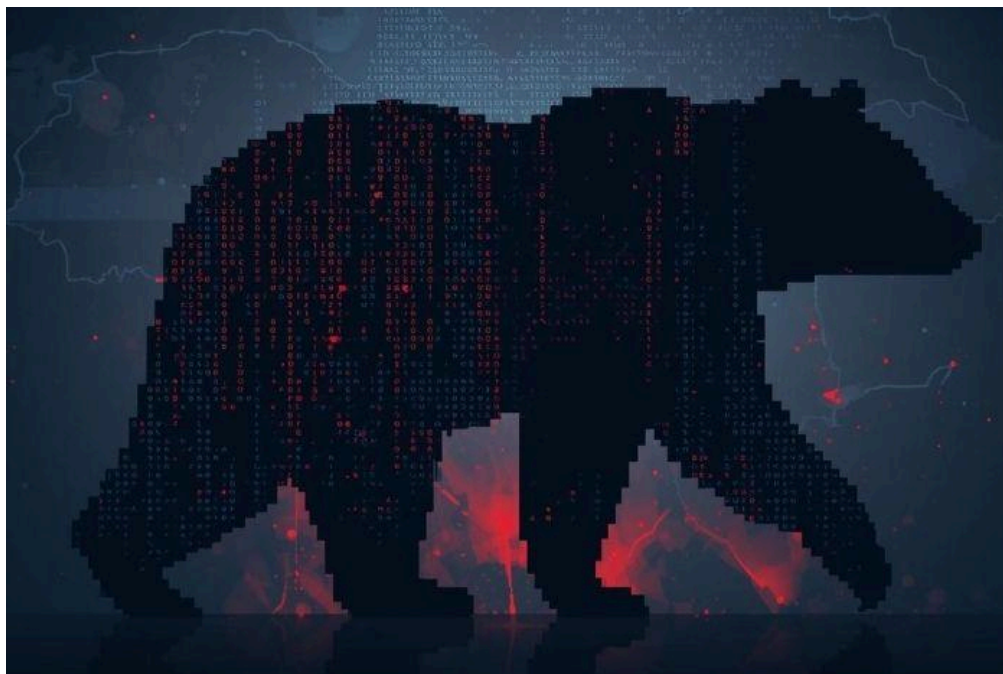
Sudeep Singh, Roy Tay

February 2, 2026



Zscaler Blog

Get the latest Zscaler blog updates in your inbox



[Security Research](#)

Introduction

In January 2026, Zscaler ThreatLabz identified a new campaign in-the-wild, tracked as *Operation Neusplit*, targeting countries in the Central and Eastern European region. In this campaign, the threat actor leveraged specially crafted Microsoft RTF files to exploit CVE-2026-21509 and deliver malicious backdoors in a multi-stage infection chain. Due to significant overlaps in tools, techniques, and procedures (TTPs) between this campaign and those of the Russia-linked advanced persistent threat (APT) group APT28, we attribute this new campaign to APT28 with high confidence. Microsoft released an out-of-band update to address CVE-2026-21509 on January 26, 2026. ThreatLabz observed active in-the-wild exploitation on January 29, 2026. We are actively collaborating with Microsoft as we continue to monitor Operation Neusplit.

In this blog post, ThreatLabz examines the technical details of Operation Neusplit, including the weaponized RTF exploit, staged payload delivery, and the execution chain. We analyze the capabilities of the resulting tools, including *MiniDoor*, *PixyNetLoader*, and a *Covenant Grunt* implant, along with their command-and-control (C2) communications.

Key Takeaways

- In January 2026, ThreatLabz identified [APT28](#) weaponizing [CVE-2026-21509](#) to target users in Central and Eastern Europe, including Ukraine, Slovakia, and Romania.
- Social engineering lures were crafted in both English and localized languages, (Romanian, Slovak and Ukrainian) to target the users in the respective countries.
- The threat actor employed server-side evasion techniques, responding with the malicious DLL only when requests originated from the targeted geographic region and included the correct *User-Agent* HTTP header.
- ThreatLabz discovered two variants of a dropper that led to the deployment of *MiniDoor*, an Outlook macro-based email stealer, and *PixyNetLoader* that led to deployment of a *Covenant Grunt* implant.

Technical Analysis

In the following sections, ThreatLabz discusses the technical details of Operation Neusplit, including how the backdoors and stealers function and how they were deployed. We observed two variants of the attack chain. Both variants begin with a specially crafted RTF file that weaponizes CVE-2026-21509 and, after successful exploitation, downloads a malicious dropper DLL from the threat actor's server. There are two variants of this dropper DLL that deploy different components. We will discuss both the variants in the following sections.

Dropper Variant 1

The first dropper variant DLL is responsible for deploying a malicious Microsoft Outlook Visual Basic for Applications (VBA) project named *MiniDoor*. MiniDoor's primary goal is to steal the user's emails and forward them to the threat actor.

MiniDoor dropper DLL analysis

MiniDoor is a lightweight 64-bit DLL written in C++. The malicious functionality is implemented in the exported function: *UIClassRegister*. The DLL does not use code obfuscation and includes two variants of string decryption:

- Strings decrypted using a hardcoded 1-byte XOR key (0x3a).
- Encrypted strings prefixed with a 1-byte XOR key, which is then used to decrypt the strings.

Below are the key functionalities of this DLL.

- Creates a mutex with the static name *adjgfenkbe*.
- A 58-byte XOR key is first decrypted using a single-byte XOR key (0x3a). The decrypted string, *savntjkengkvnnblhfbegjbtlnhkwrenvbjnkhejhkwenrjvbejbrbrncbis*, is then used as a rolling XOR key to decrypt the Outlook VBA project stored (encrypted) inside the *.rdata* section of the DLL.
- Creates the directory structure *%appdata%\Microsoft\Outlook* recursively if it does not already exist.
- Writes the decrypted VBA project (*MiniDoor*) to *%appdata%\Microsoft\Outlook\VbaProject.OTM*.
- Sets the relevant Windows registry keys to downgrade Outlook security and allow the malicious project to load automatically each time Microsoft Outlook launches.

The table below shows the registry keys set by the dropper.

Subkey	Value Name	Value	Description
HKCU\Software\Microsoft\Office\16.0\Outlook\Security	Level	1	Enables all macros in Microsoft Outlook.
Software\Microsoft\Office\16.0\Outlook\Options\General	PONT_STRING	0x20	Disables the "Content Download Warning" dialog box.
Software\Microsoft\Office\16.0\Outlook	LoadMacroProviderOnBoot	1	Ensures macro provider loads when the Microsoft Outlook application starts.

Table 1: The registry keys set by the MiniDoor DLL dropper to steal email from Microsoft Outlook.

MiniDoor analysis

ThreatLabz named this VBA-based malware *MiniDoor*, as it appears to be a minimal version of *NotDoor* reported by [Lab52](#). Similar to *NotDoor*, *MiniDoor* collects emails from the infected machine, but does not support the email-based commands implemented in *NotDoor*. Below are key functionalities of the Outlook VBA.

- Monitors the *MAPILogonComplete* event which occurs after the Outlook user has logged on. Once triggered, the macro sleeps for 6 seconds before iterating through four folders in the user's mailbox..
- Two pre-configured email addresses are hardcoded in the VBA macro by the threat actor:
 - ahmeclaw2002@outlook.com
 - ahmeclaw@proton.me
- The *SearchNewMessageAndHandle* method searches the following folders for existing emails.
 - Inbox
 - RssFeeds
 - Junk
 - Drafts
- The stealing functionality is implemented in the *ForwardEmail* method, which iterates over each folder's contents and, for each message that was not already forwarded:
 - Saves a local copy to *%TEMP%\temp_email.msg*.
 - Drafts a new email, attaches temp_email.msg, and sends the email to both configured recipient addresses.
- Sets the *DeleteAfterSubmit* property of the *mailItem* to *true* to ensure that no copy of the message is saved in the Sent folder after it is forwarded to the threat actor.
- For each Outlook message that is forwarded, the macro sets the *AlreadyForwarded* property to *Yes* to prevent the same message from being forwarded twice.
- Handles the *Application_NewMailEx* event (triggered when new emails are received) by forwarding the received email to the above-mentioned email addresses.

The complete *MiniDoor* macro code is available in the [ThreatLabz GitHub repository](#).

Dropper Variant 2

In the second dropper variant, the infection chain is more complex and involves multiple stages. Similar to the first dropper variant, after successful exploitation of CVE-2026-21509, the attack chain downloads a tool that ThreatLabz named *PixyNetLoader*, which drops malicious components on the endpoint and sets up the Windows environment to start the infection chain.

PixyNetLoader analysis

The dropper DLL used in variant 2 of the attack chain is new and previously undocumented.

PixyNetLoader's string decryption mechanism is similar to the *MiniDoor* dropper DLL. Below are the key functionalities.

- Creates a mutex with the name *asagdugughi41*.
- Checks for the presence of *EhStoreShell.dll* at *%programdata%\USOPublic\Data\User\EhStoreShell.dll*.
- If *EhStoreShell.dll* is not found at location above, then the main dropper logic is invoked.
- All the embedded payloads are encrypted using a 0x47 byte long rolling XOR key: *shfioehh243t3dcwehortjbo6k7pjl8lop7ku45ht3u4grywefdyehriobjoko5k65iyh*. They are decrypted and dropped to the file system locations in the table below:

Location	Size (in bytes)
%programdata%\Microsoft OneDrive\setup\Cache\SplashScreen.png	0x39649
%programdata%\USOPublic\Data\User\EhStoreShell.dll	0x36200
%temp%\Diagnostics\office.xml	0xDE4

Table 2: Decrypted embedded payloads, including their file system drop locations and corresponding sizes.

Uses *COM object hijacking* to establish persistence. *EhStorShell.dll* is the legitimate name for the Enhanced Storage Shell Extension DLL. By setting the Windows registry keys listed in the table below, PixyNetLoader ensures that the next-stage malicious shellcode loader DLL is loaded each time the *explorer.exe* process starts.

subKey	ValueName	Value
Software\Classes\CLSID\{D9144DCD-E998-4ECA-AB6A-DCD83CCBA16D}\InProcServer32	Null	%programdata%\USOPublic\Data\User\EhStoreShell.dll
Software\Classes\CLSID\{D9144DCD-E998-4ECA-AB6A-DCD83CCBA16D}\InProcServer32	ThreadingModel	Apartment

Table 3: Windows registry keys set by PixyNetLoader to ensure persistence.

Executes the following command using the *CreateProcess* Windows API to set up a Windows scheduled task. This command leverages the previously dropped *office.xml* file to configure the scheduled task as shown below.

```
schtasks.exe /Create /tn "OneDriveHealth" /XML "%temp%\Diagnostics\office.xml"
```

The Windows scheduled task is named *OneDriveHealth* and configured to launch the command below exactly one minute after the task is registered. The OneDriveHealth scheduled task launches the following command:

```
%windir%\system32\cmd.exe
/c (taskkill /f /IM explorer.exe >nul 2>&1) & (start explorer >nul 2>&1) & (schtasks /delete /f /tn OneDriveHealth)
```

The complete *office.xml* Windows scheduled task configuration file is available in the [ThreatLabz GitHub repository](#).

Shellcode loader analysis

The dropped DLL *EhStoreShell.dll* is loaded in the *explorer.exe* process. Its key functionality is to extract shellcode embedded using steganography in the file named *SplashScreen.png* (that was previously dropped) and execute it.

The string decryption in the *EhStoreShell.dll* is similar to the *MiniDoor* dropper DLL. In addition, all the API names are resolved at runtime using the DJB2 API hashing algorithm.

Below are the key functionalities:

- Loads the legitimate version of *EhStorShell.dll*.
- Resolves addresses for the following exports from the legitimate DLL:
 - *DllCanUnloadNow*
 - *DllGetClassObject*
 - *DllRegisterServer*
 - *DllUnregisterServer*
- Overwrites the export addresses in the malicious *EhStoreShell.dll* with the API addresses above to proxy the execution to the legitimate version of *EhStorShell.dll*. This is done to preserve the functionality of the COM service.

Conditional execution of malicious functionality

The *EhStoreShell.dll* executes its malicious logic only when both of the following conditions are met:

- Checks the host process that loaded the DLL. The malicious functionality is invoked only when the host process is *explorer.exe*. If the host process is not *explorer.exe*, then the code remains dormant.
- Checks whether the *Sleep()* API is short circuited (a common implementation used by several sandboxes) to detect the analysis environment. This check is implemented as shown below.
 - Calculates current timestamp by calling *std::chrono::steady_clock::now()*.
 - Calls *Sleep()* to delay execution by 3 seconds.
 - Calculates current timestamp again by calling *std::chrono::steady_clock::now()*.
 - If the difference between the current timestamp and the previous timestamp is greater than 2.9 seconds, only then it continues with the malicious activity. If the difference is less than 2.9 seconds, then the code assumes that the *Sleep()* API call has been tampered with.

PNG steganography and shellcode loader

Once all the checks pass, *EhStoreShell.dll* creates a new thread using *beginthreadex*. The thread start function performs the following actions:

- Decrypts the PNG path, *%programdata%\Microsoft OneDrive\setup\Cache\SplashScreen.png*, then expands environment variables to obtain the full file path.
- Uses steganography to extract the malicious shellcode from the PNG file.
 - Each pixel of the PNG image is represented by 4 bytes (1 byte per channel) for the red, green, blue, and alpha channels.
 - The Least Significant Bit (LSB) of each byte represents an encoded data bit, hence each byte of encoded data is stored within 8 bytes of image data (or 2 pixels)
 - The first 4 bytes of encoded data represents the payload size in little endian byte order and is followed by the cleartext payload itself.
- Creates a mutex named *dvyubgbqfusdv32*.

The complete code to extract the shellcode from the PNG file is available in the [ThreatLabz GitHub repository](#).

The shellcode is executed by the *EhStoreShell.dll* via the following actions:

- Allocates executable memory using the native Windows API *NtAllocateVirtualMemory*.
- Copies the extracted shellcode into the newly allocated memory region.
- Transfers execution to the shellcode.

Shellcode analysis

The main purpose of this 64-bit shellcode is to load a .NET assembly embedded inside it. In order to load a managed assembly from native code, the shellcode uses the *CLR hosting* technique. Below are the key steps used to achieve managed code execution in-memory from unmanaged code.

- Loads the *mscorlib.dll* and *oleaut32.dll* libraries.
- Initializes the .NET runtime by calling *CLRCreateInstance* (exported by *mscorlib.dll*).
- Requests the *ICLRMetaHost* interface, selects the .NET version v4.0.30319, and initializes *ICorRuntimeHost* interface.
- Retrieves the application domain by calling *ICorRuntimeHost::GetDefaultDomain*, then queries this object to obtain the *_AppDomain* interface.
- Uses *SafeArrayCreate* and *SafeArrayAccessData* methods to copy 0xfc00 bytes of the embedded .NET assembly into the array.
- Calls *_AppDomain::Load_3* to load the .NET assembly passed via *SafeArray*, enabling in-memory execution of the .NET assembly.
- Retrieves the entrypoint of the .NET assembly and invokes it using *MethodInfo::Invoke_3*.

Covenant Grunt analysis

The embedded .NET assembly is a *Grunt* implant associated with the open source .NET *Covenant* C2 framework. In this sample, the implant uses the *Filen* API as a [C2Bridge](#) to communicate and receive tasks from the threat actor. This abuse of legitimate APIs was previously observed in other *Covenant Grunt* implants linked to APT28 by ThreatLabz and other researchers.

Strings in this sample are XOR-encoded with the hardcoded string *EIZ4EG2K8R* and then Base64-encoded. These include the domains for querying the *Filen* API, the Authorization Bearer Token, and *Filen* parent folder UUID (*fe644d8c-2601-46ea-bf7d-3db110aa08d4*).

Threat Attribution

ThreatLabz attributes this campaign to the Russia-linked threat actor APT28 with high confidence, based on the following factors:

- **Victimology:** The use of Romanian, Ukrainian, and English language content in the exploit RTFs suggest potential targets within Europe. European countries, especially those in Central and Eastern Europe, have been targeted previously by APT28.
- **Tooling:** *MiniDoor* is a stripped down variant of *NotDoor*, which was reported by [Lab52](#) in September 2025 and attributed to APT28. This variant replaces the backdoor functionality with a simple email stealing capability.
- **Infrastructure:** Abuse of the Filen API for C2 communication by *Covenant Grunt* samples was previously reported by Sekoia in [Operation Phantom Net Voxel](#) (also attributed to APT28) in September 2025.
- **Techniques:** The *PixyNetLoader* infection chain shares notable overlap with Operation Phantom Net Voxel. Although the earlier campaign used a VBA macro, this activity replaces it with a DLL while retaining similar techniques, including:
 - COM hijacking for execution.
 - DLL proxying.
 - XOR string encryption techniques.
 - *Covenant Grunt* and its shellcode loader embedded in a PNG via steganography

Conclusion

This campaign by the Russia-linked group APT28 targeted countries in Central Europe and Eastern Europe with specially crafted RTF files that exploit CVE-2026-21509, resulting in the deployment of *MiniDoor* and *PixyNetLoader*. ThreatLabz research highlights that APT28 continues to evolve its TTPs by weaponizing the latest vulnerabilities in popular and widely used applications such as Microsoft Office.

ThreatLabz urges readers to install the latest security updates from the official Microsoft website to patch critical vulnerabilities such as CVE-2026-21509.

Zscaler Coverage

Zscaler's multilayered cloud security platform detects indicators related to this campaign at various levels. The figure below depicts the Zscaler Cloud Sandbox, showing detection details for *PixyNetLoader*.

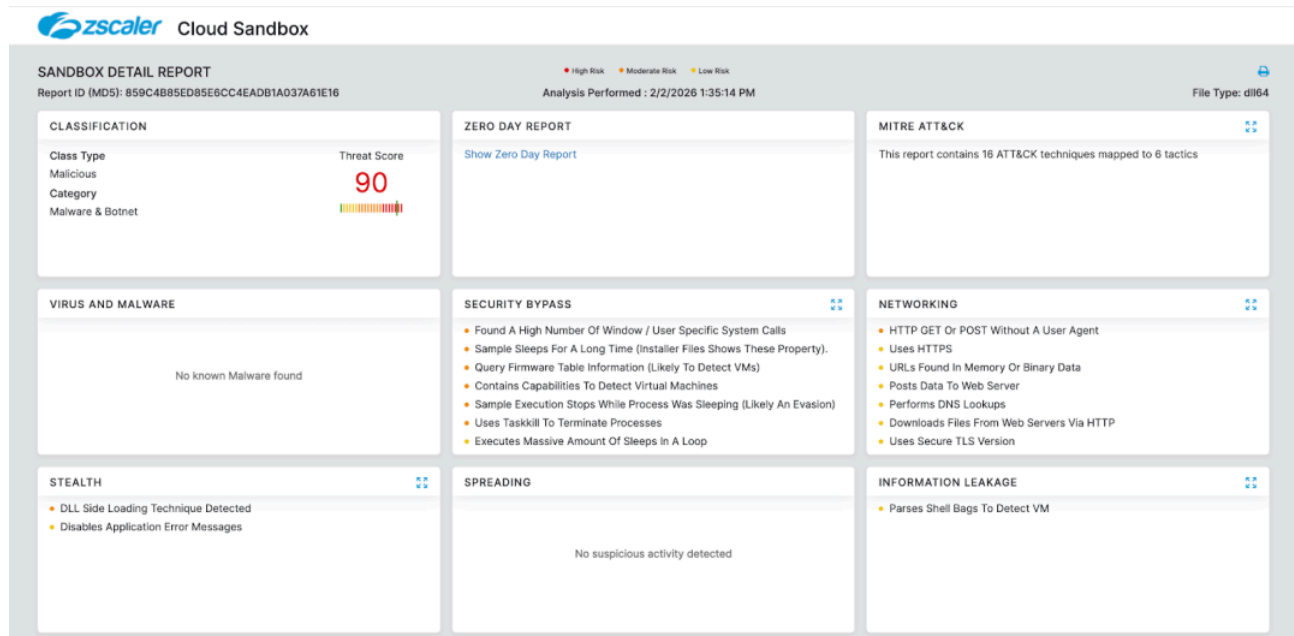


Figure 1: Zscaler Cloud Sandbox report for *PixyNetLoader*.

- [Win32.Backdoor.Covenant](#)
- [Win32.Spyware.MiniDoor](#)
- [RTF.Exploit.CVE-2026-21509](#)
- [Win64.Dropper.PixyNetLoader](#)

Indicators Of Compromise (IOCs)

File indicators

Hashes	Filename	Description
95e59536455a089ced64f5af2539a449 4592e6173a643699dc526778aa0a30330d16fe08 b2ba51b4491da8604ff9410d6e004971e3cd9a321390d0258e294ac42010b546	Consultation_Topics_Ukraine(Final).doc	RTF file exploiting CVE-2026-21509.
2f7b4dca1c79e525aef8da537294a6c4 c4799d17a4343bd353e0edb0a4de248b99295d4d 1ed863a32372160b3a25549aad25d48d5352d9b4f58d4339408c4eea69807f50	Courses.doc	RTF file exploiting CVE-2026-21509.
4727582023cd8071a6f388ea3ba2feaa d788d85335e20bb1f173d4d0494629d36083dddc 5a17cfaea0cc3a82242fdd11b53140c0b56256d769b07c33757d61e0a0a6ec02	N/A	RTF file exploiting CVE-2026-21509.
d47261e52335b516a777da368208ee91 c8c84bf33c05fb3a69bc5e2d6377b73649b93dce fd3f13db41cd5b442fa26ba8bc0e9703ed243b3516374e3ef89be71cbf07436b	1291.doc	RTF file exploiting CVE-2026-21509.
7c396677848776f9824ebe408bbba943 D577c4a264fee27084ddf717441eb89f714972a5 c91183175ce77360006f964841eb4048cf37cb82103f2573e262927be4c7607f	BULLETEN_H.doc	RTF file exploiting CVE-2026-21509.
f3b869a8d5ad243e35963ba6d7f89855 c1b272067491258ea4a2b1d2789d82d157aaf90a a944a09783023a2c6c62d3601cbd5392a03d808a6a51728e07a3270861c2a8ee	2_2.d	Dropper DLL (Variant 1) for MiniDoor.
f05d0b13c633ad889334781cf4091d3e 7bbb530eb77c6416f02813cd2764e49bd084465c bb23545380fde9f48ad070f88fe0afd695da5fcae8c5274814858c5a681d8c4e	VbaProject.OTM	MiniDoor
859c4b85ed85e6cc4eadb1a037a61e16 da1c3e92f69e6ca0e4f4823525905cb6969a44ad 0bb0d54033767f081cae775e3cf9ede7ae6bea75f35fbfb748ccba9325e28e5e	table.d	PixyNetLoader dropper DLL (Variant 2).
e4a5c4b205e1b80dc20d9a2fb4126d06 e52a9f004f4359ea0f8f9c6eb91731ed78e5c4d3 a876f648991711e44a8dcf888a271880c6c930e5138f284cd6ca6128eca56ba1	EhStoreShell.dll	Shellcode loader
154ff6774294e0e6a46581c8452a77de 22da6a104149cad87d5ec5da4c3153bebf68c411 2822c72a59b58c00fc088aa551cdeeb92ca10fd23e23745610ff207f53118db9	SplashScreen.png	PNG file containing shellcode embedded using steganography.

Hashes	Filename	Description
ee0b44346db028a621d1dec99f429823 cea7e9323d79054f92634f4032c26d30c1cedd7e 9f4672c1374034ac4556264f0d4bf96ee242c0b5a9edaa4715b5e61fe8d55cc8	office.xml	Windows scheduled task configuration file.
ea6615942f2c23dba7810a6f7d69e2da 23b6f9c00b9d5475212173ec3cbbcff34c4400a7 3f446d316efe2514efd70c975d0c87e12357db9fca54a25834d60b28192c6a69	N/A	Covenant Grunt implant using Filen API as C2Bridge.

Network indicators

Type	Indicator
Malicious domain	freefoodaid[.]com
Malicious domain	wellnesscaredmed[.]com
URL hosting MiniDoor dropper DLL	hxxps://freefoodaid[.]com/documents/2_2.d
URL hosting PixyNetLoader	hxxps://freefoodaid[.]com/tables/tables.d
URL hosting LNK	hxxps://freefoodaid[.]com/documents/2_2.lnk

MITRE ATT&CK Framework

ID	Tactic, Technique	Description
T1566.001	Initial Access, Phishing: Spearphishing Attachment	Exploit RTFs were observed delivered as email attachments.
T1203	Execution, Exploitation for Client Execution	CVE-2026-21509 was exploited to initiate the infection chain.
T1106	Execution, Native API	Native APIs were used to execute the shellcode for Variant 2.
T1053.005	Execution, Scheduled Task/Job: Scheduled Task	A scheduled task was used for triggering the COM hijacking that runs the shellcode loader DLL.
T1204.002	Execution, User Execution: Malicious File	Users must execute the exploit RTF to start the infection chain.
T1546.015	Persistence, Event Triggered Execution: Component Object Model Hijacking	COM hijacking is used for executing the Variant 2 shellcode loader DLL.
T1137.006	Persistence, Office Application Startup: Add-ins	A malicious Outlook VBA project is executed on Outlook startup.

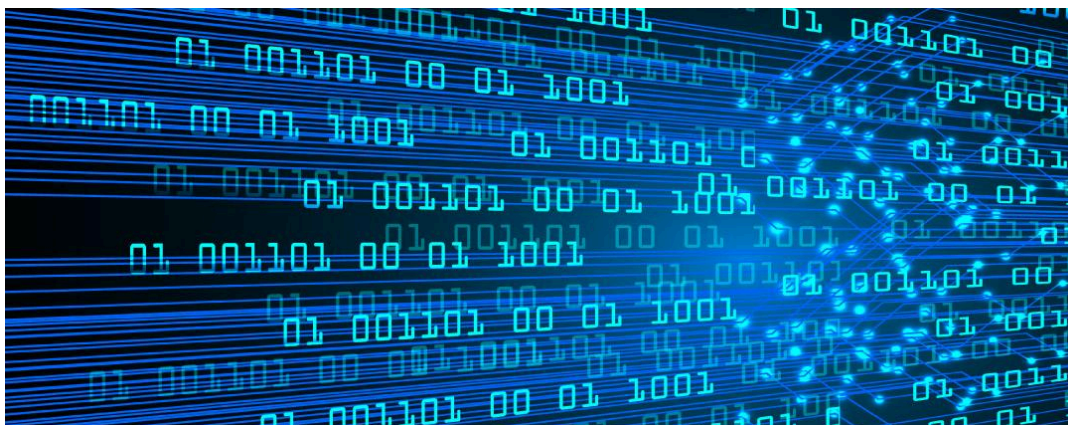
ID	Tactic, Technique	Description
T1140	Defense Evasion, Deobfuscate/Decode Files or Information	Shellcode is encoded within PNG with steganography.
T1480.002	Defense Evasion, Execution Guardrails: Mutual Exclusion	Mutexes are used to prevent multiple instances of the malware from executing at the same time.
T1027.007	Defense Evasion, Obfuscated Files or Information: Dynamic API Resolution	DJB2 hashing is used by the Variant 2 shellcode loader for API resolution.
T1027.003	Defense Evasion, Obfuscated Files or Information: Steganography	Covenant and its loader shellcode is encoded in the PNG with LSB steganography.
T1497.003	Defense Evasion, Virtualization/Sandbox Evasion: Time Based Checks	The Variant 2 shellcode loader checks that Sleep API is not short-circuited as an anti-analysis/sandbox feature.
T1114	Collection, Email Collection	A malicious Outlook VBA project sends newly received emails to hardcoded email addresses controlled by the threat actor.
T1071.001	Command and Control, Application Layer Protocol: Web Protocols	Covenant Grunt uses HTTPS for C2 communication.
T1102.002	Command and Control, Web Service: Bidirectional Communication	The Filen API service is abused to bridge communications between Covenant Grunt implant and the actual Covenant C2 server-side listener.

Was this post useful?

[Yes, very!](#)[Not really.](#)

Disclaimer: This blog post has been created by Zscaler for informational purposes only and is provided "as is" without any guarantees of accuracy, completeness or reliability. Zscaler assumes no responsibility for any errors or omissions or for any actions taken based on the information provided. Any third-party websites or resources linked in this blog post are provided for convenience only, and Zscaler is not responsible for their content or practices. All content is subject to change without notice. By accessing this blog, you agree to these terms and acknowledge your sole responsibility to verify and use the information as appropriate for your needs.

Explore more Zscaler blogs



Steal-It Campaign

[Read post](#)



European diplomats targeted by APT29 (Cozy Bear) with WINELOADER

[Read post](#)



COLDRIVER Updates Arsenal with BAITSWITCH and SIMPLEFIX

[Read post](#)

Get the latest Zscaler blog updates in your inbox

By submitting the form, you are agreeing to our [privacy policy](#).