

Shattering the Dream - When a Job Offer Becomes a Zero-Day Attack

By matthewsu

Published: 2026-08-11 · Archived: 2026-09-25 02:02:10 UTC

Key Points

- Check Point Research is tracking a long-running campaign called **Operation Dream Job**, targeting organizations worldwide, with a particular focus on the defense sector. The campaign is affiliated to DPRK-linked [Lazarus group](#) and its latest wave focuses on the defense sector in Europe and India.
- In the latest variant of the **Operation Dream Job** campaign, the threat actor distributed **SecurityPDF**, a modified PDF viewer designed to open attacker-crafted PDF documents and execute a new backdoor which we named **Troy**.
- During the intrusion, the threat actor exploited **CVE-2026-68820**, a zero-day vulnerability in the Microsoft **AFD.sys** driver, to deploy a new version of **FudModule**, Lazarus' kernel-mode rootkit. Following Check Point Research responsible disclosure, Microsoft released a patch as part of their August Patch Tuesday updates.
- Lazarus also used **CVE-2025-49113** to exploit vulnerable **Roundcube** webmail servers. The compromised servers were infected with **RelayShell**, a PHP webshell that repurposes compromised web servers as relay nodes within the attacker's command-and-control infrastructure.
- At least in one case, a compromised organization in Western Europe was leveraged to conduct a spear-phishing campaign, allowing the attackers to abuse the organization's reputation and trust to target additional victims.

Introduction

Since early 2026, Check Point Research has tracked a wave of the [Operation Dream Job](#) campaign. This wave primarily targeted the defense sector worldwide, with a particular emphasis on companies operating in the aerospace and aviation industries.

We observed the threat actor distributing modified PDF viewers designed to execute malicious payloads embedded within specially crafted PDF files, opened by the user. In this campaign, the threat actor expanded its delivery method by leveraging impersonation websites and search engine optimization (SEO) techniques to distribute the trojanized applications, increasing its credibility and helping it evade some phishing-based detections.

During the operation, the threat actor deployed a new version of the **FudModule** rootkit, exploiting a zero-day local privilege escalation (LPE) vulnerability in the Windows **AFD.sys** driver, to obtain **SYSTEM** privileges and disable EDR visibility. Following responsible disclosure, Microsoft assigned the vulnerability **CVE-2026-68820** and released a patch on August 11, 2026, as part of their August Patch Tuesday updates.

The attackers' command-and-control infrastructure consists of compromised **Roundcube** and **WordPress** servers hosting **RelayShell**, a new PHP webshell that repurposes compromised web servers as relay nodes.

In this blog, we analyze the latest **Operation Dream Job** campaign, walking through the complete attack chain and providing a technical analysis of the malware and the novel techniques employed throughout the operation, offering new insights into the group's evolving modus operandi.

Infection Chain

The **Operation Dream Job** campaign begins with targeted spear-phishing lures centered on attractive job opportunities at well-known companies in the defense, aerospace, and aviation industries.

The exact method used to approach victims in the current campaign remains unclear. However, based on previously documented **Dream Job** campaigns, we assess that the threat actor likely approached targets through professional networking platforms such as **LinkedIn**, or directly through messaging applications. Posing as recruiters, the attackers present enticing job opportunities and ultimately direct victims to download malicious files.

During our analysis, we identified **two distinct infection chains** used to compromise targets. While the second chain appears to represent a more recent evolution of the campaign, both infection methods remain active in parallel.

Infection Chain 1: DLL Sideloading chain

In this infection chain, the victim is convinced to download an encrypted zip archive containing three files:

- A legitimate, digitally signed PDF viewer executable.
- A malicious DLL that is loaded through DLL sideloading.
- An encrypted payload with a PDF extension.

 Figure 1 - High-level overview of the DLL sideloading infection chain

Figure 1 – High-level overview of the DLL sideloading infection chain.

When the victim launches the executable, the malicious DLL `libmupdf.dll` is loaded via **DLL sideloading**. The DLL extracts a decoy PDF document from the encrypted payload and displays it to the user, while simultaneously extracting, decrypting, and executing an embedded payload directly in memory.

 Figure 2 - PDF decoy impersonating Lockheed Martin job description.

Figure 2 – PDF decoy impersonating Lockheed Martin job description.

The executed payload is **MISTPEN**, a lightweight in-memory downloader that uses Microsoft Graph API to access OneDrive in order to retrieve additional modules and run them in memory.

- **Reconnaissance:** During the initial stages of the infection, the threat actor deploys several reconnaissance modules that collect system and process information, allowing the attacker to verify that the system is a suitable target before proceeding with the next stage of the attack.

- **Persistence:** Once the target has been validated, MISTPEN receives an additional persistence module that installs the malware on disk and ensures that MISTPEN is automatically executed after system reboot.
- **Privilege Escalation:** After persistence is established, MISTPEN loads an in-memory local privilege escalation (LPE) module designed to exploit the zero day vulnerability **CVE-2026-68820** in the Microsoft **AFD.sys** driver. Successful exploitation allows the malware to execute **FudModule**, Lazarus' kernel-mode rootkit, with SYSTEM privileges.
- **Backdoor Deployment:** The final backdoor delivered by MISTPEN is the [ForestTiger](#) backdoor, a well-documented malware family widely attributed to the **Lazarus** threat group. Once deployed, it provides the attackers with long-term remote access to the compromised host.

Infection Chain 2: Trojanized PDF viewer

In July 2026, we observed a new campaign sharing many characteristics with previously documented **Operation Dream Job**, particularly the campaign [described](#) by **ESET** in 2025.

In this infection chain, victims receive fraudulent job offers impersonating **Enveil**, a Privacy Enhancing Technology company, and are instructed to download an encrypted ZIP archive containing two files:

- **SecurityPDF** – a trojanized PDF viewer that has been modified to extract and execute an encrypted payload from specially crafted PDF documents.
- **A malicious PDF file** – an encrypted payload disguised as a PDF document that is decrypted and executed when opened with the modified viewer.

Figure 3 - Crafted PDF opened by SecurityPDF.

Figure 3 – Crafted PDF opened by **SecurityPDF**.

SecurityPDF is a trojanized version of a legitimate open-source PDF viewer built on the **MuPDF** framework. The threat actor modified two code paths responsible for opening PDF documents: the **File** → **Open** dialog and the **drag-and-drop** file handling routine.

As a result, whenever a user opens a PDF document, the application checks whether the file contains the following marker `This document is encrypted with sumatrapdf reader!!!!!!!!!!!!!!`. If the marker is present, the application extracts the embedded payload, decrypts it using a single-byte XOR key (`0x39`), writes the resulting executable to `%TEMP%\new.exe`, and launches it as a child process.

The `new.exe` file is a small executable responsible for reflectively loading an embedded DLL containing the **Troy** backdoor, a previously undocumented backdoor first observed in this campaign.

In addition, we identified at least **three websites** impersonating **Enveil** that distribute the trojanized PDF viewer. Some of these websites rank highly in search engine results, with some even appearing as the **top result** for relevant search queries. It is important to note that the attacker only impersonates Enveil, and there are no indications that the company was targeted or compromised.



Figure 4 – Website appearing as the top search result for “Enveil SecurityPDF”.

Although we did not directly observe how the threat actor incorporated these websites into the phishing campaign, we assess that they were likely used to separate the delivery of the trojanized PDF viewer from the delivery of the crafted PDF document. In this scenario, victims would first receive the malicious PDF file through a phishing message and later be instructed to download the PDF viewer from what appears to be the vendor’s legitimate website. Separating these infection chain stages reduces the likelihood of detection.

MISTPEN

MISTPEN is the first in-memory module executed during the attack chain. First [documented](#) by Mandiant in 2024, it functions as a lightweight downloader that uses the Microsoft Graph API to communicate through attacker-controlled files hosted on OneDrive and retrieve additional payloads

All files exchanged through OneDrive are encrypted with AES, using separate keys for uploads and downloads. MISTPEN’s primary capability is the reflective loading of PE DLL files directly into memory, enabling the deployment of additional payloads without touching disk.

Before delivering the final backdoor, **MISTPEN** often deploys several in-memory modules designed to perform specific tasks. These modules do not implement their own network communication mechanisms; instead, they execute their designated tasks and return the resulting data to **MISTPEN**, which uploads it to the C2.

Below is a description of the modules we observed being loaded by **MISTPEN** during our analysis.

GetInfoPlugin – Host Reconnaissance Module

This module is a 64-bit Windows DLL internally named **Release_GetInfoPlugin_x64.dll**. Its primary purpose is to profile the compromised host and return the collected information as a single wide-character string.

The module collects basic system information, including the machine’s domain or workgroup membership (via `NetGetJoinInformation`), the computer name, the current user name, and the operating system version and build number. The collected data is formatted in the following template and returned to MISTPEN:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

Domain: <domain_or_workgroup>

ComputerName: <hostname>

UserName: <username>

OsInfo: <Windows product name> <build_number>.<UBR>

Domain: <domain_or_workgroup> ComputerName: <hostname> UserName: <username> OsInfo: <Windows product name> <build_number>.<UBR>

```
Domain: <domain_or_workgroup>
ComputerName: <hostname>
UserName: <username>
OsInfo: <Windows product name> <build_number>.<UBR>
```

PvPlugin – Process List Module

This module is a 64-bit Windows DLL internally named **Release_PvPlugin_x64.dll**. It serves as an extended version of the **GetInfoPlugin** module, collecting the same host reconnaissance data while adding detailed information about running processes.

For each running process, the module collects the Process PID, PPID, creation timestamp, associated domain and user, and process name. The collected information is formatted into a tabular process list and returned to MISTPEN.

OneScreenCapture – Screenshot Module

This module is a 64-bit Windows DLL internally named **OneScreenCapture64.dll**, it is responsible for capturing the current desktop (including all monitors) and returns the screenshot to its caller.

The module uses standard Windows **USER32** and **GDI** APIs to capture the virtual desktop into a bitmap. The bitmap is then converted to a JPEG image and Base64-encoded into a single wide-character string before being returned to MISTPEN for exfiltration.

LPE loader

This module is a 64-bit Windows DLL that acts as a loader for a local privilege escalation (LPE) exploit module. It is loaded by an extended version of MISTPEN that provides it with an RPC buffer used for communication between the two components. Messages written to this buffer are forwarded by MISTPEN to the attacker through its existing Microsoft Graph API communication channel, while responses received from the C2 are relayed back to the module through the same interface.

 Figure 5 - Writing and reading data through the shared RPC buffer

Figure 5 – Writing and reading data through the shared RPC buffer.

In addition to MISTPEN's AES-based transport encryption, the module encrypts all exchanged data using **GOST-CBC** with a randomly generated 16-byte session key. The encrypted data is then Base64-encoded, with the session key prepended to each packet.

The module operates in four stages:

1. **Host Fingerprinting** – The module gathers detailed information about the compromised host, including the operating system version, build number, installed security products, and other system characteristics.
2. **Key Exchange** – The module requests a set of four public keys from the C2 server.
3. **Session Key Generation** – Using the received public keys, the module generates new key material using the **Kyber/ML-KEM** algorithm and transmits the resulting encapsulated key material back to the C2.
4. **LPE Deployment** – Finally, the module requests the encrypted LPE payload, decrypts it using the negotiated key, and executes it directly in memory with export **DestroyEnv**. Throughout the process, status messages are sent back to the C2 to indicate whether each stage of the exploitation succeeded.

 Figure 6 - Execution of LPE module with export DestroyEnv

Figure 6 – Execution of LPE module with export DestroyEnv.

The downloaded LPE payload is **FudModule**, Lazarus' kernel-mode exploit module. It exploits a local privilege escalation vulnerability to obtain **SYSTEM** privileges and injects a payload into a **SYSTEM** process. In the observed attack, the injected payload was another instance of MISTPEN, allowing the malware to continue operating with elevated privileges and without EDR visibility.

CVE-2026-68820: Yet another Zero-Day discovered by Lazarus

The file we investigated, `Afd4Eop12_x64.dll`, has a compiler timestamp of `July 7, 2026, 22:07:44 UTC`. Its strings immediately suggest a variant of FudModule, including references such as “`enable_god_mode passed.`” and a main function similar to previous Fud Modules. FudModule is a Lazarus privilege escalation tool, reported and being used since around 2021.

 Figure 7 - Exploitation and post-exploitation function calls of FudModule, similar to the 2024 variant

Figure 7 – Exploitation and post-exploitation function calls of FudModule, similar to the 2024 variant.

The module targets `afd.sys`, the Windows Ancillary Function Driver, a part of the Windows kernel that is in charge of managing and handling sockets in Windows. In 2024, FudModule was reported to use another zero-day, CVE-2024-38193, a use-after-free vulnerability in the same `afd.sys` driver.

At first sight, the vulnerability looked similar to [CVE-2025-60719](#), which is also a use-after-free vulnerability in the AFD.sys driver fixed in November 2025 and not linked to any particular threat actor. In the sample itself, we observed an explicit minimum-version check for [Windows 11 build 26100 \(24H2\)](#), with explicit support also for **build 26200 (25H2)**. However, testing on the latest fully patched Windows 11 system confirmed that the

exploit targets a distinct, previously undocumented vulnerability, actively being used in the wild as a part of Operation ‘Dream Job’ since at least early July 2026.

We will not be disclosing full technical details of the vulnerability in this article, as it was patched on the August 11 Patch Tuesday fix. At a high level, the exploit takes advantage of how `afd.sys` handles a socket is created when it is accessed concurrently by **several threads at once**.

The driver maintains a small piece of information about the state associated with each socket. Under specific concurrent conditions, two of its own code paths can operate on this state at the same simultaneously, without synchronization, creating a **race condition**. If triggered at the right moment, one code path can access memory after it has already been released by another, resulting in a **use-after-free** vulnerability.

From there, the module does what these modules do – it leverages this memory corruption to obtain a **kernel read/write primitive**, which is subsequently used to achieve **local privilege escalation to SYSTEM**.

We disclosed the issue to Microsoft, and Microsoft issued a fix quickly.

Disclosure timeline

- **Jul 28, 2026:** Issue reported to the Microsoft Security Response Center (MSRC).
- **Jul 31, 2026:** Microsoft confirmed the bug
- **Aug 5, 2026:** Microsoft assigned **CVE-2026-68820** to the issue.
- **Aug 11, 2026:** Fixed on Patch Tuesday.

FudModule v3.1

Except for a novel, completely different exploit chain, this FudModule’s post-exploitation behavior is quite similar to FudModule v3, [reported](#) by Gen Digital back in 2024.

Shared with v3

- The entire telemetry teardown suite: process, thread, and image notify callbacks; object and registry callbacks; minifilter removal by altitude band; and the termination of the NT Kernel Logger.
- Crash-dump suppression, executed before everything else.
- The WFP stage, which is activated when Kaspersky is present and Symantec is absent.
- The hardcoded ETW provider kill-list: its 94 GUIDs match the first 94 entries of Gen’s published 95-GUID list, in identical order.
- The driver selection engine, with the same universal preserve list and per-class keep and kill rules.
- Privileged-handle forgery and the same two-hop spawn through `services.exe` into a `SYSTEM` `msiexec.exe` process.
- Logging vocabulary, surviving essentially string-for-string, including: `GetGodMode` failed, `GetSystemHandle` passed., `CreateRemoteProcess` passed., `RemoteDllExecute` passed., and the `ClearVaccine*` family.

Functionality removed from v3

- The dedicated Microsoft Defender stage used to disable monitoring of `MsMpEng.exe`. Only the orphaned string `SuspendDefender` passed. remains, and is no longer referenced by executable code, while Gen's FudModule v3 YARA rule contains the active-stage variant `SuspendDefender` skipped.
- The PPL stripping functionality targeting AhnLab's `asdsvc.exe`.

Microsoft Defender is still blinded here, but only through the generic security-product suppression engine, like any other vendor, rather than through a dedicated Defender-specific stage.

New functionality since v3

- A Smart App Control tampering functionality not documented in publicly analyzed FudModule versions through v3. Within the SYSTEM-level `msiexec.exe` child process, its remote stub sets `VerifiedAndReputablePolicyState` to zero and invokes `NtSetSystemInformation` class `0xA4` with option `0x10000000`, triggering an in-place reload of the code integrity policy.

Targeting

As mentioned before, this version only targets newer Windows builds 26100/26200, unlike the previous version that also targeted older ones.

Troy Backdoor

The **Troy** backdoor is a newly identified modular remote access trojan in **Lazarus'** arsenal. Delivered as a 64-bit DLL, it supports **17 operator commands**, providing a broad range of remote access and post-exploitation capabilities.

The name **Troy** is derived from a PDB path embedded in the sample: `E:\HK\Tool_Module\Troy_Handle\1Troy_Create_Dll_Tool\x64\Release\Test_Dll.pdb`. Notably, the term **Troy** has also appeared in PDB paths associated with previously documented Lazarus samples. For example, an ESET [report](#) published last year documented a sample containing a PDB path `E:\Work\Troy\안정화\...`

The Troy backdoor supports three Command and Control (C2) servers, each configured with a URL and port. At startup, the implant iterates through the configured servers in order, parsing each URL into its host and path components, establishing an HTTP connection, and issuing a connection request. It validates the response against the string `CONNECTED` and uses the first server that responds successfully.

The initial connection is followed by a challenge-response handshake used to authorize the implant against the server. Once authenticated, Troy collects host information and registers the victim by sending a client identifier and a system profile containing the user profile directory, account name, Windows version, local IPv4 address, and current working directory.

Following registration, Troy enters its command-processing loop. Tasks received from the C2 server are Base64-encoded; the implant decodes them and identifies commands using plaintext prefix matching. Command results are returned through the send channel in a compact JSON envelope: `{ "to": "<channel>", "msg": "<base64>" }`. Responses that exceed the maximum message size are divided into numbered chunks and reassembled on the C2 side.

The Troy backdoor provides a notably broad feature set for a single-DLL implant, and a cohesive design. Its seventeen supported commands span the capabilities required for each stage of post-compromise operations, from initial reconnaissance and file operations, to command execution and in-memory code delivery, while following a consistent tasking and result-framing model throughout.

 Figure 8 - Troy's reflective DLL injection flow, showing remote RWX allocation, loader and payload writes, and execution through RtlCreateUserThread.

Figure 8 – Troy's reflective DLL injection flow, showing remote RWX allocation, loader and payload writes, and execution through RtlCreateUserThread.

Troy Backdoor Supported C2 Commands

Command	Capability	What it does
WAIT	Keepalive	Server-side no-op that keeps the session alive and feeds the idle back-off counter.
DRIVES	Drive enumeration	Reports every mounted volume letter present on the host.
LIST <path>	Directory listing	Enumerates a directory with names, sizes and timestamps, sending the listing length first and the listing itself second.
OPEN <exe> [args]	Process creation	Launches an executable with arguments in a hidden window with no console.
DELETE <path>	File and folder deletion	Removes a single file, or an entire directory tree through a silent shell file operation.
ZIPDOWNLOAD <src> <dst>	Archive and exfiltrate	Compresses a path with PowerShell <code>Compress-Archive</code> into a temporary archive, uploads it, then removes the archive.
DOWNLOAD <victim-source> <client-destination>	File exfiltration	Streams a file from the victim to the operator in chunks.
UPLOAD <client-source> <victim-destination>	File drop	Writes an operator-supplied file to disk, appending the filename when the destination is a directory.
CMD <commandline>	Interactive shell	Runs a command and captures its output, tracking <code>cd /d</code> so the working directory persists between commands, with a 10 second execution watchdog.

Command	Capability	What it does
<code>mem <dllpath> <pid></code>	In-memory DLL injection	Maps a DLL into a remote process using an embedded reflective loader, matching architecture before injecting.
<code>pk <pid></code>	Process termination	Terminates a process by identifier and reports the outcome.
<code>sleep <N></code>	One-shot delay	Pauses the implant for N minutes without changing the stored interval.
<code>DEFAULTSLEEP</code>	Configured delay	Acknowledges, then pauses for the currently configured beacon interval.
<code>GET_CONFIG</code>	Configuration read	Returns the stored configuration as eight fields covering the client ID, the sleep interval, and the three server and port pairs. The stored values may differ from the connection actually in use.
<code>SET_CONFIG </code>	Configuration update	Writes eight replacement fields into stored configuration state. Only the idle interval takes effect at runtime, because the connection loop does not read the stored servers and the port remains hardcoded to 80.
<code>pvd</code>	Process listing with command lines	Enumerates processes with session, owner and start time, enriched with full command lines retrieved over WMI.
<code>pv</code>	Process listing	The same enumeration without the command line column.

Compromised Infrastructure Used as ForestTiger C2

As previously reported, ForestTiger's C2 infrastructure has historically relied primarily on compromised servers mainly running WordPress and SharePoint. In more recent campaigns, the threat actor appears to have **shifted** toward using **compromised Roundcube webmail servers** as C2 infrastructure.

The majority of the Roundcube servers we analyzed were running versions vulnerable to **CVE-2025-49113**, a critical PHP Object Deserialization vulnerability that can lead to remote code execution (RCE). Exploitation of this vulnerability requires authentication with valid Roundcube credentials. During our investigation, we identified several credential leaks that are available in the Darkweb, and contain usernames and passwords associated with accounts on the compromised webmail servers. We assess that the threat actor likely leveraged these credentials to authenticate to the affected Roundcube instances before exploiting **CVE-2025-49113** to deploy **RelayShell** web shells, which subsequently serve as a C2 relay mechanism.

In addition, we observed the threat actor compromise **PrestaShop** websites and deploy the same **RelayShell** web shell.

RelayShell

Following the post-exploitation of a web server, the threat actor deployed a previously undocumented PHP web shell that we named **RelayShell**. Unlike a traditional web shell that provides direct command execution, RelayShell primarily acts as a communication relay between the threat actor and an infected endpoint.

RelayShell operates in two distinct modes, selected by the password supplied in the HTTP POST request. For clarity, we refer to these as **Victim mode** and **Operator mode**.

Victim Mode

When accessed using the victim password, RelayShell creates a new PHP session that is subsequently used for communication with the infected endpoint.

The webshell then decrypts a hidden configuration stored in an external file using a custom substitution cipher. The configuration contains two values:

- A backbone URL
- A unique identifier (PID) assigned to the compromised server

RelayShell then immediately sends an HTTP POST request to the configured backbone URL using the unique identifier and authentication password.

 Figure 9 - WebShell contacting the backbone compromised server on new session creation.

Figure 9 – WebShell contacting the backbone compromised server on new session creation.

Based on our analysis, the backbone URL appears to point to another RelayShell instance acting as an upstream relay or notification server. This request signals that a new victim session has been established, allowing the operator to subsequently connect using the second password.

Operator Mode

When accessed using the operator password, RelayShell enters **operator mode**, providing a set of commands for interacting with the compromised server. These commands support session management, connectivity checks, file upload and deletion, and retrieval of activity logs.

Command Type	Description
Session auth / selection	Scans existing <code>.ses</code> files, picks the latest session, and returns its data.
Check & cleanup	Updates configuration, deletes old session/log/temp files, and checks connectivity to the backbone URL.

Command Type	Description
Download log	Sends back the encoded log file containing activity records.
File upload	Writes an arbitrary file to disk, using Base64-encoded filename and content.
Self-delete / file removal	Self-delete Deletes a specified file (provided as Base64-encoded path).

File-Based Communication Channel

After both the victim and operator sessions are established, RelayShell provides two commands, **send** and **receive**, which implement a lightweight file-based communication channel using temporary files stored on the compromised server.

Messages are exchanged through files following the naming convention `<session_id><object>.log` where `object` identifies the side of the communication channel: **1** for the victim and **2** for the operator.

When sending data, RelayShell writes the supplied content to the session file corresponding to the sender. When receiving data, RelayShell reads and returns the contents of the file corresponding to the opposite side, creating a bidirectional communication between the victim and the operator.

 Figure 10 - Obfuscated command switch for requesting and sending data

Figure 10 – Obfuscated command switch for requesting and sending data.

This mechanism effectively turns the compromised web server into a relay node. The victim-side implant establishes the session and notifies the backbone server that is monitored by the threat actor, after which the actor connects to the RelayShell instance and exchanges commands and responses through the file-based messaging channel.

During our investigation, we observed the threat actor accessing RelayShell through shared VPN services, including **ExpressVPN**, further obscuring the origin of their infrastructure.

We also identified **17 unique identifiers**, suggesting that at least 17 compromised servers were likely used as relay nodes during the campaign. However, we were unable to identify all of the affected servers.

Victimology

This new **Operation Dream Job** campaign focused heavily on the defense sector, particularly organizations involved in military technologies such as surveillance sensors, drones, and robotics. The campaign had a global reach, with activity extending into South America, including Brazil, and successful targeting observed in Western Europe, including France and Germany.

During the campaign, a compromised organization headquartered in France was later leveraged by the threat actor to conduct spear-phishing attacks against targets worldwide, likely to increase the perceived campaign's authenticity and credibility.

Another notable target was India, which has a substantial and rapidly growing defense and aerospace industry, with expanding domestic production and technology exports.

 Figure 11 - Lazarus Operation Dream Job Global Campaign Targets.

Figure 11 – Lazarus Operation Dream Job Global Campaign Target Distribution.

Conclusion

The latest **Operation Dream Job** campaign demonstrates that **Lazarus** continues to evolve both its malware capabilities and operational tradecraft. Beyond deploying a new version of **FudModule** that exploits the **CVE-2026-68820** zero-day vulnerability, the threat actor also refined its initial access techniques by combining targeted spear-phishing with impersonation websites and search engine optimization (SEO) to distribute trojanized software.

The threat actor's decision to rely on compromised Roundcube instances and content management system (CMS) servers for C2 reflects an operational approach well suited to highly monitored defense-sector environments, where network activity may be closely inspected by organizational security teams as well as government and national cybersecurity authorities. By abusing legitimate web infrastructure, the threat actor can better blend malicious communications within normal network traffic.

Our findings highlight **Lazarus**'s continued evolution toward stealthier and more resilient operations, combining new delivery techniques, modular malware, zero-day exploitation, and compromised web infrastructure. We believe the technical details presented in this research will help defenders identify, detect, and disrupt future Operation Dream Job campaigns.

IOCs

DLL Loader\ Dropper

```
2b4987c07a3d9a9a5d1a9bf4efa3d1903e775090b611710edafdc92874265ca8
3a02d0d798e8d35555776886d92b20ff38a101c9ef7e0eebc8ce5d259516525a
92106b0c62a0a42678232f8273f030b2d3c8e92efce81b98b9eec70cfe98afa1
396192d92d17ace1a521f1351eeeba2825e60badd0d799cc5c338e4934b3c82c
f7e620134ca935067797ab957317b346ce0df84a4e9b9ca54a6acc9b75afda4d
75b93a7103b0562f6497d30052c0c5cf7aa58c1bf0e9297022b74469a7f096f1
a45144d22cac70a45d71cf4dfffa4efbc373658779a56cf1300d6ac863d6cc7e2
1de949c71efcfb0ffc41f33d38833dbc4b082075b1a540fc68c18c535d7ad86c
4c9b804d6155b29f1e27a9ffe531e10bc42a7bdab42f905b50146bf2026768d9
29e24c007549e51319ff3aee011da6f9f93568e8c85a5ad69c9e53bd3f4533a2
4ebdce2f47c23ff8c9e8e80c8b5239c7a5764da31cd3ab8f0505926890adc105
c2aa28bb5e2a749c693712008276f311edd912f689371ef9e8a1ee5fb4167461
```

MISTPEN

2db25ac41a66aa523c79e23e00443573530dd7bd82b8371bcc87bd7232e141eb
5278ee922838352f1480a73e971161017d643a80b7ec22bf725897dfd088696d
b4082d21070d9ddf53fde4ea22524d09e41ec9826ce63cef3c6235e458d21afb
fb3fc5626f68677fb1269a2fefbe70e719211b4065e836ab92e06a8210139a2d
ea7056f2bf36c66a61ff787ff5be975a85f534c3c5ca178791dac2504db2c619
13d10bc99f7f7abe7ee0902be87920b73b2ea41bd9683dbfcad340dacbcdef79
4fd32432341dfcf54d0517a6bbc38e5d265be70933493e4183c2a340cdde9a2d
4dd792c9f672bbdcc8d363d745994efe90f4ffc5fdc2c059c8e379a48ad6a68a
ba96c603e44046de703c67b2c3b7e4ca974afef7b437a0244418bc4edc781bb7

ForestTiger

72dcae85e062f541fecad9ec7a18a3123e7ae5ac5d53c91709b53a46dbbd289
231b1ef8b95bf77887d5377e2a60f649035e78f543af1b82877db36a5759d858
6da9b1e6f3315ceb77dd14a937a26cc3602bf6a7e2c2ecafb3c65ce5319837be
a0578a2b7821d7e2c573530648f26d7a0d98b373ab24fb7f0c792736761e542d
82268052f94df6f4870d02e57b18d4c54136cc7a8c8d80ad162631f99462c943

FudModule

3b6378df8442e63a6ed7317075913e4720847a510d95022d4a8347b2637c245d

PDF Payload

a673ae661593c0de9bbb815593b816a6853dad6d55ad5042d2ef1875cd13d6e7
8ce6c29f92dc45b1474417cbddf4ed0c18e58fa63e3a071ee9f85aa9d2aac07c
acb97cec84e08b89f41967a24e965d1fd2c51751cef158f7aa35bb4306b87b97
3601060c62edeeaa49def6a13be6e126e1024ce011faad4e2d9f585ccf6bd5a6
fecf12088843801215898442bd1ff3e266f29d14e29a94780e857f69c4915d6b
d578c28c9afe7457a0d81f6701332ef8197e8f7468de654935fb29a50ea66459

SecurityPDF.exe

743172aab606974b054a64561534ae66baa3a840657f79d7c6fa18350e8d45d1
db3d69b7eeda2e35e23006bf4b7e206281fce809584207214fc213f9bc30376d

Troy Backdoor

590fb6ae19480d694e08ee85859cad8066f2f87e7e5abba2960c6d115e1615d6
68d4fba7b1300a59cd6212c08910a260cd71b40cd9f51cac933030a68faac0bb
a738059ce07c951c31ab2da3d93d8f69bfff32f9b7d933dbf5943441b9cc99075

RelayShell

21c3ad4838c4324bc5f081021da5fb2e9073d0c9304087811c21eb47c9e22762
cc4e06aa378a190f71384c03023bb3d18a6d66e297d46701220e132963d2e222

```
SecurityPDF Website & Troy C2  
envell[.]xyz  
enveil[.]online  
uxtramine[.]org  
135.181.67[.]203  
135.181.185[.]158
```

YARA – RelayShell Webshell

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

rule lazarus_relayshell

{

meta:

author = "@_CPRResearch_"

description = "Lazarus RelayShell Webshell"

target_entity = "file"

hash = "21c3ad4838c4324bc5f081021da5fb2e9073d0c9304087811c21eb47c9e22762"

strings:

\$str1 = "PqCWom"

\$str2 = "a84038"

\$str3 = "biwbih"

\$str4 = "ddf7acea"

\$str5 = "enRU904U"

\$str6 = "fou2rm"

\$str7 = "kurhiW"

\$str8 = "qcrgl"

\$str9 = "rlzbiw"

```

$Str10 = "tmmvr1"

$Str11 = "win386"

$Str12 = "\"biwbih\"

$Str13 = "\"PqCWom\"

$Str14 = "\"a84038\"

$Str15 = "\"ddf7acea\"

$Str16 = "\"enRU904U\"

$Str17 = "\"fou2rm\"

$Str18 = "\"kurhiW\"

$Str19 = "\"qcrgl\"

$Str20 = "\"rlzbiw\"

$Str21 = "\"tmmvr1\"

$Str22 = "\"win386\"

$Str23 = "D9hWnVEqdgzJ67/B8euS0yKCIMrw5jc:fGUX3AakLH2oYQRp"

```

condition:

3 of (\$Str*)

}

```

rule lazarus_relayshell { meta: author = "@_CPResearch_" description = "Lazarus RelayShell Webshell"
target_entity = "file" hash = "21c3ad4838c4324bc5f081021da5fb2e9073d0c9304087811c21eb47c9e22762"
strings: $Str1 = "PqCWom" $Str2 = "a84038" $Str3 = "biwbih" $Str4 = "ddf7acea" $Str5 = "enRU904U" $Str6
= "fou2rm" $Str7 = "kurhiW" $Str8 = "qcrgl" $Str9 = "rlzbiw" $Str10 = "tmmvr1" $Str11 = "win386" $Str12 =
 "\"biwbih\" $Str13 = "\"PqCWom\" $Str14 = "\"a84038\" $Str15 = "\"ddf7acea\" $Str16 = "\"enRU904U\"
$Str17 = "\"fou2rm\" $Str18 = "\"kurhiW\" $Str19 = "\"qcrgl\" $Str20 = "\"rlzbiw\" $Str21 = "\"tmmvr1\"
$Str22 = "\"win386\" $Str23 = "D9hWnVEqdgzJ67/B8euS0yKCIMrw5jc:fGUX3AakLH2oYQRp" condition: 3
of ($Str*) }

```

```

rule lazarus_relayshell
{
  meta:
    author = "@_CPResearch_"
    description = "Lazarus RelayShell Webshell"
    target_entity = "file"

```

```
hash = "21c3ad4838c4324bc5f081021da5fb2e9073d0c9304087811c21eb47c9e22762"
strings:
$str1 = "'PqCWom'"
$str2 = "'a84038'"
$str3 = "'biwbih'"
$str4 = "'ddf7acea'"
$str5 = "'enRU904U'"
$str6 = "'fou2rm'"
$str7 = "'kurhiW'"
$str8 = "'qcrgl'"
$str9 = "'rlzbiw'"
$str10 = "'tmvr1'"
$str11 = "'win386'"
$str12 = "\"biwbih\""
$str13 = "\"PqCWom\""
$str14 = "\"a84038\""
$str15 = "\"ddf7acea\""
$str16 = "\"enRU904U\""
$str17 = "\"fou2rm\""
$str18 = "\"kurhiW\""
$str19 = "\"qcrgl\""
$str20 = "\"rlzbiw\""
$str21 = "\"tmvr1\""
$str22 = "\"win386\""
$str23 = "D9hWnVEqdgzJ67/B8euS0yKCIrw5jc:fGUX3AakLH2oYQRp"
condition:
  3 of ($str*)
}
```

Source: <https://research.checkpoint.com/2026/shattering-the-dream-when-a-job-offer-becomes-a-zero-day-attack/>