

# Mustang Panda targets India's government and energy sectors with ZOHOMURK and MINIRECON

---

 [acronis.com/en/tru/posts/mustang-panda-targets-indias-government-and-energy-sectors](https://acronis.com/en/tru/posts/mustang-panda-targets-indias-government-and-energy-sectors)



## Summary

---

Acronis Threat Research Unit (TRU) has been tracking two concurrent campaigns orchestrated by [Mustang Panda](#) targeting Indian government entities, delivering new malware implants and abusing [Zoho WorkDrive](#), a legitimate cloud storage platform commonly used in the Indian government sector.

The two identified campaigns target India's hydropower sector and government entities engaged in cooperation agreements (MOUs) with Taiwanese government institutions, leveraging a newly discovered malware toolkit comprising SHARDLOADER, MINIRECON and ZOHOMURK.

SHARDLOADER variants demonstrate moderate sophistication, leveraging persistence and DLL sideloading to deploy **two newly identified implants**: ZOHOMURK and MINIRECON.

ZOHOMURK is a newly identified implant that leverages Zoho WorkDrive for command-and-control, data exfiltration and remote task execution.

The campaigns showcase new additions to Mustang Panda's malware arsenal while incorporating code overlaps with previously documented tooling, including [TONESHELL](#).

Mustang Panda has been repeatedly [linked in public reporting to China and operations aligned with China's strategic interests](#). Based on targeting patterns, lure themes, operational characteristics and code overlaps, TRU assesses with high confidence that the activity is espionage-motivated and aligned with intelligence collection related to India's hydropower initiatives and defense cooperation with Taiwan.

During the investigation, TRU identified multiple compromised systems within India's government sector and collaborated with CERT-In to support mitigation and victim notification efforts.

## Introduction

---

Acronis TRU has identified two espionage-focused campaigns targeting India's hydropower sector and government entities, using lure documents themed around cooperation agreements between Indian and Taiwanese institutions. Both campaigns delivered previously undocumented DLL-based loaders, which we track as SHARDLOADER, through hydropower- and government-themed lure documents distributed in compressed archives.

Upon execution, one SHARDLOADER variant decrypts and launches MINIRECON, a newly identified implant derived from the Toneshell malware family, while the second variant deploys ZOHOMURK, a novel implant that leverages legitimate cloud services for command-and-control, data exfiltration and remote task execution.

The campaigns shared identical tooling and multiple artefacts, with minor variations between targets, suggesting a moderate retooling effort by the operator while maintaining a consistent focus on victims within India. We also observed that the threat actor appears well versed in the [country's software compliance landscape](#), which likely influenced the development of the new implant.

Through this research, we outline the analytical basis for attributing this activity to Mustang Panda, a group that has been repeatedly publicly linked to China, with high confidence, primarily based on observed deployment patterns, operational characteristics, and overlaps with previously documented campaigns.

## Background and context

---

Mustang Panda is a long-running espionage-oriented state-aligned threat entity, known for aligning its operations with **current geopolitical developments**. The group has consistently leveraged themes tied to [international conferences](#), [bilateral engagements](#) and [region-specific political events](#) to support targeted intrusion activity against government and policy-related entities.

Previous reporting documented the use of [ShadowPad](#) in [campaigns targeting India's power sector in 2021](#), highlighting the longstanding interest of China-aligned threat actors in the country's critical infrastructure. The campaigns examined in this report continue that trend, introducing new malware while maintaining operational characteristics that support attribution despite limited code reuse.

## Attack chain

---



Figure 1.1: Payload execution chain for Campaign I



Figure 1.2: Payload execution chain for Campaign II

## Technical details

---

**Note on structure:** Campaigns are presented chronologically, not in the order of discovery. SHARDLOADER v1.0 corresponds to Campaign I and SHARDLOADER v1.1 to Campaign II; these version numbers are used throughout to tie each variant and artefact to its campaign.

## Technical details: Campaign I

---

### Initial analysis and delivery mechanism



Figure 2: Launcher executable present with DLL Hidden

Our investigation began after identifying a suspicious archive, Hydropower Cooperation Project Proposal.zip, believed to have been distributed via spear-phishing and subsequently uploaded to VirusTotal in May 2026. Upon examining the archive, we found that the DLL intended for sideloading had been assigned the hidden attribute, a technique commonly observed in [Mustang Panda operations](#).



Figure 3: Signed executable and legitimate binary by Solid PDF Creator

Examining the launcher executable revealed that it is a legitimate, digitally signed component of Solid PDF Creator. Its naming convention, delivery method and execution chain closely align with patterns observed in previous Mustang Panda campaigns, prompting a deeper analysis that led to the identification of a previously undocumented loader we track as **SHARDLOADER v1.0**.

### DLL sideloading



Figure 4: Signature details related to the launcher executable



Figure 5: DLL-Sideloading into the launcher executable with export function visible

When the victim executes **Project Proposal.exe** from the archive, both the launcher executable and the malicious **SolidPDFCreator.dll** are present in the same directory. Because the legitimate, digitally signed executable statically imports functions from SolidPDFCreator.dll, Windows automatically loads the attacker-controlled DLL during startup.

One of these imported functions, **GetSPApp**, is exported by the malicious DLL, allowing SHARDLOADER v1.0 to execute under the context of a trusted application while the launcher continues to operate normally.

## SHARDLOADER – Variant 1.0



Figure 6: Malicious export function found inside SHARDLOADER Variant 1.0

Despite some differences in how the variant of the loader is used across both the campaigns, the exports are nearly identical in both the samples. While in this case the DLL exports multiple functions, execution ultimately passes through the **GetSPApp**, which serves as the loader's primary entry point.



Figure 7: Creating named event-object –1



Figure 8: Creating named event-object –2

The loader starts by initially creating a named object also can be called a named event, **uydgcfteionxcfd**, which acts as a counter against the instances of the malware binary present on the memory using the [CreateEventW](#) Windows API.



Figure 9: Implant staging mechanism – 1



Figure 10: Implant staging mechanism – 2

After creating the named event, the implant creates a hidden staging directory at **C:\ProgramData\IDM\logs\** and copies both the sideloading host (**MediumInstStart.exe**) and the malicious DLL (**SolidPDFCreator.dll**) into it. This hidden location, disguised as legitimate Internet Download Manager data, provides persistence and allows the payload chain to be re-executed following a system reboot.



Figure 11: Shellcode extraction from the .rdata section

Further analysis revealed that the implant stores its shellcode in an obfuscated form within the .rdata section.



Figure 12: Shellcode bytes stored at sequential offsets

A total of 69 functions reconstruct the payload at runtime, each loading 32 hardcoded 16-byte XMM constants from scattered locations into a temporary buffer before copying them into a heap region.



Figure 13: Shellcode decrypting via XOR-Key



Figure 14: Size of MINIRECON shellcode

Once assembled, each 512-byte chunk is passed through a decryption routine that applies a rolling XOR using a 28-byte key, followed by a byte-reordering operation that reverses portions of the buffer around a pivot point at byte 23. Repeating this process across all 69 functions reconstructs a fully decrypted 34,816-byte shellcode payload in memory.



Figure 15: Memory allocation and permission grant for shellcode execution

After decoding the shellcode, the implant allocates memory for the payload, writes it into the allocated buffer, and updates the memory permissions prior to execution.



Figure 16: Executing Shellcode via EnumSystemLocalesA callback functionality

Finally, the implant invokes [EnumSystemLocalesA](#) and abuses its callback mechanism to execute the shellcode in memory, launching the next-stage implant we track as MINIRECON.



Figure 17: Persistence via RunKeys technique

For persistence, SHARDLOADER creates a Run key under **HKCU\Software\Microsoft\Windows\CurrentVersion\Run** named **MediumNetMonIt**, pointing to **C:\ProgramData\IDM\logs\MediumInstStart.exe**. This mechanism ensures the DLL sideloading chain executes at user logon, reloading SolidPDFCreator.dll and reestablishing the infection chain. The next stage of the campaign is MINIRECON.

## MINIRECON

Upon initially looking into the shellcode, we found that it is a variant of [Toneshell8](#), which had been previously reported by researchers at IBM X-Force. While it retains several core characteristics of the malware family, we identified notable changes in its functionality and command-and-control communications, leading us to track it as MINIRECON.



Figure 18: PEB-Walking technique



Figure 19: Identical features with respect to Toneshell variant – 1



Figure 20: Identical features with respect to Toneshell variant – 2



Figure 21: Victim desktop enumeration



Figure 22: Live recon by the operator on victim's machine – I



Figure 23: Live recon by the operator on victim's machine – II

The MINIRECON variant of the Toneshell8 implant stays intact in terms of features such as PEB to locate kernel32.dll at runtime and resolves all subsequent APIs through the same 13131313 hash multiplier that has been a fingerprint of the Toneshell family.

For session key generation, it relies on a **Linear Congruential Generator** using the constants **0xBD828** and **0x4373A**, and the beacon encryption follows the same pattern where a **256-byte XOR** key is generated from the **LCG PRNG each cycle**, prepended in cleartext to the outgoing packet, with the victim GUID and hostname XOR-encrypted after it.

On the command handling side, the implant supports two parallel reverse shells, file upload and download, remote command execution with null-byte validation, and a drop-and-execute chain, all dispatched through a single opcode byte in the range of 1 through 7.



Figure 24: Upgrade in terms of C2 communications



Figure 25: Infrastructural details – I



Figure 26: Infrastructural details – II

The primary difference from earlier Toneshell8 variants lies in its command-and-control communications. Instead of using raw TCP or TLS-like traffic, MINIRECON establishes a WebSocket connection over HTTPS using the native WinHTTP API. During setup, the implant upgrades the connection to WebSocket mode and disables certificate validation through the security flag 0x3300, allowing it to communicate with C2 servers using self-signed certificates. The implant also includes a proxy fallback mechanism. If a direct connection fails, it enumerates locally configured proxy settings and attempts to route traffic through them, likely helping it blend into enterprise network environments. On the server side, we observed the C2 running Python 3.12 with the websockets 11.0.3 library, responding with a standard HTTP 101 Switching Protocols message before transitioning to WebSocket-based beaconing and command-and-control traffic.



Figure 27: Network communications with attackers' infrastructure

During our analysis, we recovered the server's WebSocket upgrade response, further confirming the use of WebSocket-based command-and-control communications. We also observed the operator actively conducting reconnaissance on the infected system. After identifying the environment as an analyst sandbox, the operator attempted to disrupt analysis by deleting tools and corrupting multiple files on the host.

## Technical details: Campaign II

---

### Initial analysis and delivery mechanism



Figure 28: DLL-hidden attribute

While investigating the first campaign, we identified a second ZIP archive, **MOU USI-INDSR TAIWAN.zip**, that used the same launcher executable observed previously. As in the first campaign, the malicious DLL was hidden within the archive, indicating a similar delivery and execution chain.

### SHARDLOADER – Variant 1.1



Figure 29: Malicious export function

Like the previous variant, this version modifies the internal structure of SolidPDFCreator.dll while retaining the malicious export **GetSPApp** as its primary execution entry point.



Figure 30: Dropper logic in SHARDLOADER variant 2

Analysis of the function revealed embedded dropper functionality responsible for extracting and deploying multiple files that form the final stages of the campaign.



Figure 31: Staging mechanism in SHARDLOADER variant 2 – I



Figure 32: Staging mechanism in SHARDLOADER variant 2 – II

The implant first creates a staging directory at **C:\Users\Public\Documents** and uses it to store the next-stage payloads extracted to disk.



Figure 33: Decrypting final stager payloads – I



Figure 34: Encoded blob related to the launcher executable



Figure 35: Encoded blob related to the ZOHOMURK implant

After creating the staging directory, the implant extracts two embedded files from the .rdata section: **pcl2bmp.exe**, a legitimate signed Citrix Receiver binary, and **ctxmui.dll**, a malicious DLL that serves as the **ZOHOMURK** implant and is sideloaded by the executable.



Figure 36: RC4 decryption mechanism

The implant uses RC4 to decrypt an encrypted blob embedded within the binary using the hardcoded key **'urt!@#ghsiet63540(mk)?78Xdesr\*%rt\$36.'**



Figure 36: Using COM for persistence – I



Figure 37: Using COM for persistence – II



Figure 38: Using COM for persistence – III

For persistence, this variant takes a different approach compared to v1.0. Instead of relying on a simple Run key, it uses the COM-based Task Scheduler API to register for a scheduled task named SolidPDFPcl2Bmp. The implant begins by calling CoCreateInstance to instantiate the Task Scheduler service, connects to it, and retrieves the root task folder via `ITaskService::GetFolder("\\")`. It then creates a new task definition and sets up a daily trigger identified as Pcl2BmpDailyTrigger, configured to repeat every five minutes (PT5M) over a one-day duration (P1D), with the start time pulled from `GetLocalTime` now of execution.



Figure 39: Using COM for persistence – IV

On the settings side, it disables battery-related restrictions by setting both `StopIfGoingOnBatteries` and `DisallowStartIfOnBatteries` to false, ensuring the task runs regardless of power state, and sets the multiple instances policy to ignore new instances if one is already running. For the execution time limit, it first attempts to set PT0S (no limit), and if that fails, falls back to PT24H. Finally, the task is registered under the root folder with the name SolidPDFPcl2Bmp, using `TASK_CREATE_OR_UPDATE` (flag 6) and `TASK_LOGON_INTERACTIVE_TOKEN` (logon type 3), after which it is immediately triggered via `IRegisteredTask::Run`.

## ZOHOMURK



Figure 40: Malicious export function inside ZOHOMURK

The ZOHOMURK DLL exposes a single exported function, **CTXMUI\_ParseArgvA**, which acts as the primary entry point for the implant's execution flow.



Figure 41: Anti-analysis features

The implant employs a timing-based anti-analysis technique using **QueryPerformanceCounter**. By measuring the execution time of a 64-iteration dummy loop, it can detect debugging activity and terminate execution when the delay exceeds an expected threshold. We observed this check before registry writes and the initial C2 beacon, among other locations throughout the binary.



Figure 42: Victim enumeration

Once running, the implant generates a unique victim identifier by combining the hostname, obtained via **GetComputerNameA**, with the system's public IP address retrieved from IPInfo. If the lookup fails, **the implant falls back to hostname|UNKONW, preserving a consistent misspelling of "UNKNOWN."** It then XOR-encrypts the resulting string with a 29-byte hardcoded key and hex-encodes the output to create the folder name used on the **Zoho WorkDrive C2 infrastructure**.



Figure 43: Victim registration



Figure 44: ZOHO folder root ID

After constructing the victim identifier, the implant checks if a folder with the encoded name already exists under the root WorkDrive directory. If the victim is connecting for the first time, it creates two folders on the operator's Zoho WorkDrive account, one as the main victim folder and a subfolder prefixed with c which serves as the outbox for exfiltrated data. A registration beacon is then sent to notify the operator of the new victim. If the victim has already been registered from a previous execution, the implant skips folder creation and instead cleans up any leftover files before resuming the polling loop.



Figure 45: OAuth ID hardcoded inside the implant

Looking further into the binary, we came across a function responsible for authenticating with Zoho's OAuth API. The implant sends a POST request to **accounts.zoho.com/oauth/v2/token** with a complete set of hardcoded OAuth credentials including the refresh token, client ID and client secret, all sitting in plaintext as part of the request body. The user agent is set to **Mozilla/5.0 (Windows NT 10.0; Win64; x64)** to blend in with regular browser traffic. If the token refresh is successful, the implant parses the access token from the JSON response and caches it for subsequent API calls to Zoho WorkDrive.



Figure 46: Folder structure creation

After registering the victim, the implant creates the required folder structure within the operator's Zoho WorkDrive account. This is handled by a function that sends a POST request to **www[.]zohoapis[.]com/workdrive/api/v1/files** with the user agent **Zoho API C-Client/1.0**.

The request body is a JSON payload containing the folder name and the parent folder ID, formatted as:

```
{\"data\": {\"attributes\":
```

```
{\"name\": \"%s\", \"parent_id\": \"%s\"}, \"type\": \"files\"}}.
```

The authorization header carries the previously obtained OAuth token. After the folder is created, the implant parses the JSON response to extract the newly created folder ID, which it stores for use in subsequent API calls. This function is called twice during victim registration, first to create the main victim folder and then again to create the outbox subfolder where command responses will be uploaded.



Figure 47: Communicating with Zoho WorkDrive - I



Figure 48 Communicating with Zoho WorkDrive – II



Figure 49: Communicating with Zoho Workdrive – III

We also found that for listing the contents of the operator's WorkDrive, the implant sends a GET request to `workdrive[.]zoho[.]com/api/v1/files/{folder_id}/files` with the user agent set to Zoho Client/1.0. The response is parsed by manually scanning the JSON for `\"id\":` and `\"name\":` fields, with each entry stored in a 384-byte record containing the resource ID and its name. Two nearly identical functions handle this, one filtering for `\"type\": \"folder\"` to enumerate victim directories and the other filtering for `\"type\": \"files\"` to check for incoming command files in the victim's inbox.

Once a command file has been downloaded and processed, the implant removes it from WorkDrive by sending a PATCH request to `/api/v1/files/{file_id}` with the body

```
{\"data\": {\"attributes\": {\"status\": \"51\"}, \"type\": \"files\"}},
```

which moves the file to trash. This cleanup step ensures that each command is only consumed once and reduces the amount of evidence left on the operator's account.



Figure 50: ZOHOMURK features – I

When a new command file is found in the victim's inbox, the implant downloads it to a temporary file named `readata.dat` on disk. It then refreshes the OAuth token

and immediately trashes the command file from WorkDrive to remove any trace of the tasking. The contents of **readata.dat** are read into memory and decrypted using the same 29-byte XOR key **qwuietyoquihwidgqiuwhediqwiegw**, after which the first byte is extracted as the opcode and the remaining bytes as the payload. These are passed to the command dispatcher which handles three opcodes.



Figure 51: ZOHOMURK features – II



Figure 52: ZOHOMURK features – III

The commands dispatcher reads the first byte of the decrypted payload as the opcode and switches on three values. **Opcode 1** handles file operations and uploads the result back to the operator's WorkDrive through the outbox folder. **Opcode 8** is used for interactive shell access; it checks if a command shell pipe is already active and if not, creates one. The payload is then written to the pipe with a `\r\n` appended at the end. If the write fails, the shell is terminated and restarted. **Opcode 10** simply kills the active shell session.

Here is the table of command dispatch routine, which we found along with the description.

Opcode

Function

Description

1

C2\_UploadResponse

Processes the payload and uploads the result to the victim's outbox folder on WorkDrive.

8

Shell\_WriteCommand

Creates a shell if not active, appends `\r\n` to the payload and writes it to the stdin pipe. Tears down the shell if the write fails.

10

## Shell\_Teardown

Terminates the shell process and closes all associated handles.



Figure 53: ZOHOMURK's heartbeat feature

Alongside the main loop, the implant runs a heartbeat thread that sleeps for 50 seconds between each cycle. On each iteration it calls **ListFolders** on the root WorkDrive directory to check if the victim's folder is still there. If the listing comes back empty, it means the folder has been removed, so the thread re-registers the victim by recreating the folder's structure. This way the implant can recover its C2 channel on its own even if the operator resets the WorkDrive account, or the folders get cleaned up.



Figure 54: Persistence technique using Runkeys method

For persistence, the implant writes a Run key under **HKCU\Software\Microsoft\Windows\CurrentVersion\Run** with the value name **MicrosoftEdgeUpdateBrokerTask**, pointing to the sideloading executable dropped in **%LOCALAPPDATA%\Microsoft\VaultCache**. Before writing the key, it runs through two environment checks and the timing-based anti-debug gate we mentioned earlier, so if it detects it is being analyzed, it skips the persistence entirely to avoid leaving registry artifacts for the analyst.



Figure 55: ZOHOMURK features – IV

The final piece of the C2 cycle is the response upload function. After a command has been executed, the implant takes the output and splits it using a pipe delimiter to separate the destination folder's ID from the response data. It then refreshes the OAuth token and uploads the result to the victim's outbox folder on WorkDrive using the Zoho API. This completes the full communication loop where the operator drops a command file into the victim's inbox; the implant picks it up, decrypts and executes it, and then uploads the output back to the outbox for the operator to retrieve.

## Additional findings

---



Figure 56: Discovery of suspected POC Document file

During our analysis of ZOHOMURK, we identified an additional variant by pivoting on code overlaps and shared functionality across samples uploaded to public sandboxes from multiple geographic locations. Among these samples, we discovered a malicious document named test.doc that drops an earlier variant of ZOHOMURK. Based on the observed functionality and code similarities, we assess that this sample likely represents an early version of the implant observed in the wild.



Figure 57: Embedded lure – I



Figure 58: Embedded lure – II

The document contained two embedded decoys themed around government affairs in Myanmar and India, including references to the IndiaAI Mission, the Atmanirbhar Bharat initiative, and the Union Budget 2026-27, suggesting the sample may have been used for testing or development purposes. Analysis of the earlier ZOHOMURK variant revealed only minor differences from the newer version, primarily in persistence mechanisms and OPSEC improvements, while the core command-and-control functionality remained largely unchanged. The key differences are summarized in the table below.

#### Feature

ZOHOMURK v1 (4th June 2026)

ZOHOMURK v2(12th June 2026)

Victim ID format

hostname

hostname|public\_IP

Persistence path

LOCALAPPDATA%\ZohoUsing\

LOCALAPPDATA%\Microsoft\VaultCache\

Run Key value

ZohoUsingUpdataAnyssAll\_RunOnece

MicrosoftEdgeUpdateBrokerTask

Mutex name

ZohoUsingUpdataAnyssAll\_event

Local\MS\_Edge\_Update\_Task\_Service\_Sync

## Infrastructural analysis

---



Figure 59: Infrastructural details – I



Figure 60: Infrastructural details – II

During the investigation of the hydropower-themed campaign, we identified the MINIRECON implant communicating with **couldinstallup[.]com** over WebSocket on port 443. The domain, registered in May 2026, resolved to 188.208.141.177, an IP address hosted by Leapswitch Networks (AS132335) in India.



Figure 61: Infrastructural details – III

Notably, 188.208.141.177 resides within the same /24 subnet as 188.208.141.196, an IP address [previously identified by IBM X-Force as a Pubload C2 server associated with Hive0154 \(Mustang Panda\)](#). The shared subnet, hosting provider and autonomous system suggest a recurring infrastructure procurement pattern across campaigns.

Analysis of the second campaign revealed that the operators used multiple usernames and email addresses to create Zoho WorkDrive accounts used for command-and-control and data exfiltration. To facilitate information sharing with CERT-In while protecting potentially sensitive victim information, TRU redacted or removed victim-identifying details from the collected data before providing the relevant findings and indicators for incident response and remediation activities.

During our investigation, we identified active beaconing from multiple compromised systems belonging to Indian government entities, including devices associated with senior administrative personnel. The activity was observed between June 12 and June 22, 2026, during which the infrastructure remained operational and actively tasked by the operators. TRU collaborated with [CERT-In](#) and shared relevant indicators, threat actor infrastructure details, and technical findings to support victim notification and remediation efforts.

## Attribution and confidence

Attribution in threat intelligence often relies on a combination of technical, operational, and infrastructural overlaps rather than any single indicator. Based on observed tradecraft, malware similarities, infrastructure characteristics, and historical research into Mustang Panda activity, we attribute these campaigns to Mustang Panda with high confidence.

The key factors supporting this assessment are outlined below:

1. **Usage of similar launcher executable across campaigns from 2025-2026:** Both campaigns leveraged the [Solid PDF Creator](#) executable for DLL sideloading, a technique previously observed in multiple campaigns attributed to Mustang Panda by independent researchers.
2. **Borrowed code:** MINIRECON shares multiple architectural and functional similarities with [Toneshell](#), including API hashing, command handling, encryption routines, and anti-analysis techniques previously associated with Mustang Panda tooling.
3. **Hidden attributes and staging:** The use of hidden DLLs within spear-phishing archives, coupled with similar staging and persistence mechanisms, closely aligns with tradecraft documented in previous Mustang Panda operations.
4. **Consistent Spelling mistakes by threat actor:** We observed the repeated use of the misspelled string RunOnece across both Toneshell and ZOHOMURK variants, suggesting a shared development lineage and codebase.



Figure 63: Consistent spelling mistake across implants attributed to Mustang Panda

## Hunting tips

---

To assist defenders in identifying potential compromises related to this campaign, we have compiled the following indicators and artifacts that can be used for threat hunting across endpoints and network logs.

### 1. Registry persistence

Check HKCU\Software\Microsoft\Windows\CurrentVersion\Run for:

MicrosoftEdgeUpdateBrokerTask (ZOHOMURK v2)

ZohoUsingUpdataAnyssAll\_RunOnece (ZOHOMURK v1)

MediumNetMonIt (SHARDLOADER v1.0)

The RunOnece typo is unique to this actor across Toneshell and ZOHOMURK families

## 2. Scheduled tasks

Task name: SolidPDFPcl2Bmp with trigger Pcl2BmpDailyTrigger

Repetition interval PT5M (every 5 minutes)

Action pointing to pcl2bmp.exe in C:\Users\Public\Documents\

## 3. Mutex / named events

Local\MS\_Edge\_Update\_Task\_Service\_Sync (ZOHOMURK v2)

ZohoUsingUpdataAnyssAll\_event (ZOHOMURK v1)

uydgcfteionxcfd (SHARDLOADER v1.0)

## 4. File system artifacts

C:\ProgramData\IDM\logs\ (hidden, SHARDLOADER staging)

C:\ProgramData\Citrix\ (ZOHOMURK execution path)

C:\Users\Public\Documents\pcl2bmp.exe + ctxmui.dll

%LOCALAPPDATA%\Microsoft\VaultCache\ (ZOHOMURK v2 persistence)

%LOCALAPPDATA%\ZohoUsing\ (ZOHOMURK v1 persistence)

C:\ProgramData\DexpotMouseEventsPart.dat (MINIRECON 16-byte session key)

readata.dat in temp directories (ZOHOMURK command staging)

## 5. Network indicators

DNS/connections to couldinstallup.com (MINIRECON WebSocket C2)

HTTP GET to http[:]//ipinfo[.]io/ip with UA IPFetcher/1.0 (ZOHOMURK v2)

HTTPS POST to accounts[.]zoho[.]com/oauth/v2/token from non-browser processes

HTTPS traffic to workdrive.zoho.com from non-browser processes

Look for these specific User-Agents in proxy logs from non-browser executables:

Zoho Client/1.0

Zoho API Client/1.0

Zoho API C-Client/1.0

Zoho-C-Uploader/2.0

IPFetcher/1.0

Multipart POST to workdrive.zoho.com/api/v1/upload with boundary ----  
WebKitFormBoundary7MA4YWxkTrZu0gW

## 6. Process indicators

pcl2bmp.exe loading ctxmui.dll (Citrix sideloading)

MediumInstStart.exe loading SolidPDFCreator.dll from C:\ProgramData\IDM\logs\

Space-padded executable names in C:\ProgramData\Citrix\ (e.g., Meeting Record .exe)

Any executable calling EnumSystemLocalesA shortly after VirtualAlloc with RWX permissions

## 7. Sideloading pairs

Project Proposal.exe + SolidPDFCreator.dll (Campaign I)

pcl2bmp.exe + ctxmui.dll (Campaign II)

PluginsSetup.exe + libcef.dll (Extension.zip variant)

All share export name GetSPApp

## Conclusion: Advanced evasion, limited operational security

---

The campaigns examined in this report demonstrate Mustang Panda's continued investment in expanding its malware arsenal and operational infrastructure while targeting sectors aligned with China's strategic interests. The introduction of ZOHOMURK, which leverages Zoho WorkDrive for command-and-control and data exfiltration, alongside the WebSocket-enabled MINIRECON implant, reflects an effort to blend malicious activity with legitimate services commonly used across enterprise environments.

Despite these developments, the operators exposed several indicators that aided analysis and attribution, including hardcoded OAuth credentials, recurring development artefacts, plaintext identifiers, and reusable infrastructure. These findings enabled TRU to identify active compromises affecting Indian government entities and support remediation efforts in coordination with relevant authorities. During the investigation, victim-related data and technical findings were reviewed by CERT-In, which acknowledged the research and used the information to support its incident response and analysis activities.

Organizations in the government and energy sectors, particularly those involved in cross-border cooperation initiatives, should remain vigilant against spear-phishing campaigns using geopolitically themed lures and maintain visibility into endpoint-driven cloud service activity, as threat actors increasingly abuse legitimate platforms for command-and-control operations.

## Detection by Acronis

---

This threat has been detected and blocked by Acronis EDR / XDR:



## Indicators of compromise

---

### SHA256:

cd9397797216fd4c08df324937509124e57258328c8e4c6d795c6a2cd25b69b0 [ Hydropower Cooperation Project Proposal.zip]

Ebd533de7ca16daa70093b0b1084fb6136b6ba091d6ee0e4199762581e1b2e5a [ SolidPDFCreator.dll]

Fcf4efa82d477c924d42cc6b71aa672ab2381ca256769925ae34dabe2e77e025 [ Hydropower Cooperation Project Proposal.exe] 390148f5157c0f6b337ff19d162c3c2ee3e6d782fdfbe11fb1e411c0684fd33b [ test.doc]

5f22ec5c14dfd47c92850a5fb3bd8e3754d538b8021b6238238e4020336cfb5c [ ctxmui.dll - ZOHOMURK Variant 1]

F53fd0626404a129dcddb8ee7589387dd7bda7999814e0df46c670af6b3da5f5 [ MOU USI-INDSR TAIWAN.zip]

A43084f5af861f44c75c5273c779cb26d506cab6b51c33746626da504148a4ec  
[SolidPDFCreator.dll]

F2bed071676feb831ed460489643fd57f6c6c1e0d024a1ea447820276fb13828 [ ctxmui.dll - ZOHOMURK Variant 2]

**C2 domains:**

couldinstallup[.]com

© 2003–2026 Acronis International GmbH.