FULL REPORT · ENTERPRISE EXPLOITATION · MONOGRAPH 63 / 66 · 18 MIN READ

Blind SQL and Tunnel Staging: An Exposed Workbench

Inside a 186-entry China-hosted workbench pairing staged VShell and GOST tunnel plugins with 142 blind-SQL, IIS, and xp_cmdshell JSON files and JNDI callback bytecode

BY
Odin by Brandefense

PUBLISHED
19 November 2026

READING TIME
18 min read

EDITION
Version 1.1 · TLP:CLEAR

SCOPE
Prepared November 2026

1. Executive Summary

IN SHORT

An open TCP 8090 staging workbench on a China-hosted cloud VPS recorded on 13–25 November 2026 holds 186 entries (57.08 MB). Wave 1 (13 November, 36 entries) stages VShell configs and GOST and fscan binaries. Wave 2 (25 November, 150 entries) adds 142 blind-SQL, IIS, and xp_cmdshell JSON files plus five JNDI files. Disable xp_cmdshell, lock down IIS webroots, and block SQL and JVM internet egress.

Enterprise Windows and Java web stacks often place IIS reverse proxies and JSON APIs ahead of Microsoft SQL Server databases and JVM backend services. Over-privileged SQL logins or unrestricted outbound connections let SQL injection bugs escalate into command shells and encrypted proxy tunnels. This exposed workbench brings four distinct intrusion and tunnel-staging tracks together on one server.

Between 13 and 25 November 2026, Odin by Brandefense analyzed an open Python directory (`SimpleHTTP/0.6`) on TCP port 8090 of a China-hosted cloud VPS. On 25 November 2026, that Python `3.14.4` listener reported 186 active files and 21 directories.

Every file recorded on 13 November remained on disk on 25 November. The deduplicated inventory therefore matches all 186 entries across 176 sized files totaling 57,082,695 bytes (57.08 megabytes)

and 10 unsized entries, spanning 166 unique SHA-256 digests.

The workbench exposes an RSA private-key file (location withheld), an operator token file (location withheld), and 142 Wave 2 JSON probe files. Odin by Brandefense therefore publishes zero server IPv4 addresses and zero file SHA-256 digests. Candidate VShell ports **8888** and **7878**, target-output filenames, cloud socket names, and target identifiers are also withheld.

1. **Two dated waves separate persistent tunnel staging from SQL and JNDI probing.**
   Thirty-six entries recorded on `13 November` total **56.97** megabytes (**99.8%** of byte volume) and stage VShell server configs, web bundles, Python helpers, and tunnel plugins. One hundred fifty entries recorded on `25 November` add **117.15** kilobytes across **142** JSON probe files, five JNDI files, and three helper files.

2. **Wave 2 adds 142 JSON files named for blind-SQL, IIS, and `xp_cmdshell` checks.**
   Ninety-seven JSON files have names for database-context, server-role, context-slice, and row checks (paths withheld). Thirty-four JSON files are named for IIS webroot, `TCP 8001`, and VShell client checks, while **11** JSON filenames suggest `xp_cmdshell` checks and cleanup (`target execution unverified`).

3. **One staged JNDI class references HTTP proof and error callbacks on `TCP 8091`.**
   Two files in `jndi/` pair Java source and compiled `Exploit.class` bytecode beside one HTTP log and two LDAP logs. Stored bytecode inspection reveals `/RCE_PROOF` and `/RCE_ERR` callback URLs on `TCP 8091`, while `CVE-2021-44228` serves strictly as architectural context.

4. **Co-hosted VShell ports and 54.13 megabytes of plugins prepare encrypted pivoting.**
   Six files in `vsc/` pair TLS files and settings with Windows and Linux GOST tunnel plugins (**47.91** megabytes) and the `fscan` intranet scanner (**6.22** megabytes). Candidate VShell ports **8888** and **7878** ran on the same server beside Python HTTP ports **8090**, **8899**, and **8000**.

Three Decisions for Enterprise Security Leaders

1. **Keep SQL Server `xp_cmdshell` disabled and strip `sysadmin` from web logins.** Enforce `xp_cmdshell = 0`, run web database pools under least-privilege schema logins, and block direct internet traffic from SQL Server subnets.
2. **Enforce read-only IIS webroots and monitor worker shell creation.** Remove write permissions for IIS application pools inside `inetpubwwwroot` and isolate internal ports like `TCP 8001`. Alert immediately whenever `w3wp.exe` or `sqlservr.exe` spawns a shell.
3. **Patch Java naming libraries, disable remote JNDI class loading, and hunt GOST tunnels.** Keep Log4j2 on `2.17.1` or later, block outbound LDAP, RMI, and unapproved WebSocket streams, and deploy protocol and endpoint detections across backend tiers.

2. Workbench Architecture, Two-Wave Reconciliation, and Co-Hosted Ports

The exposed staging workbench sat on a **China-hosted cloud virtual private server** (address withheld). That server exposed an unauthenticated Python directory (`SimpleHTTP/0.6 Python/3.14.4`) on `TCP 8090`. Engineering teams use external attack surface management to find exposed web interfaces and database gateways before operators enumerate backend paths.

Odin by Brandefense recorded the workbench between `13 and 25 November 2026`. Defenders also apply cyber threat intelligence to link C2 plugin staging with SQL and Java exploitation workflows. Figure 1 maps the four operational tracks present on the server.

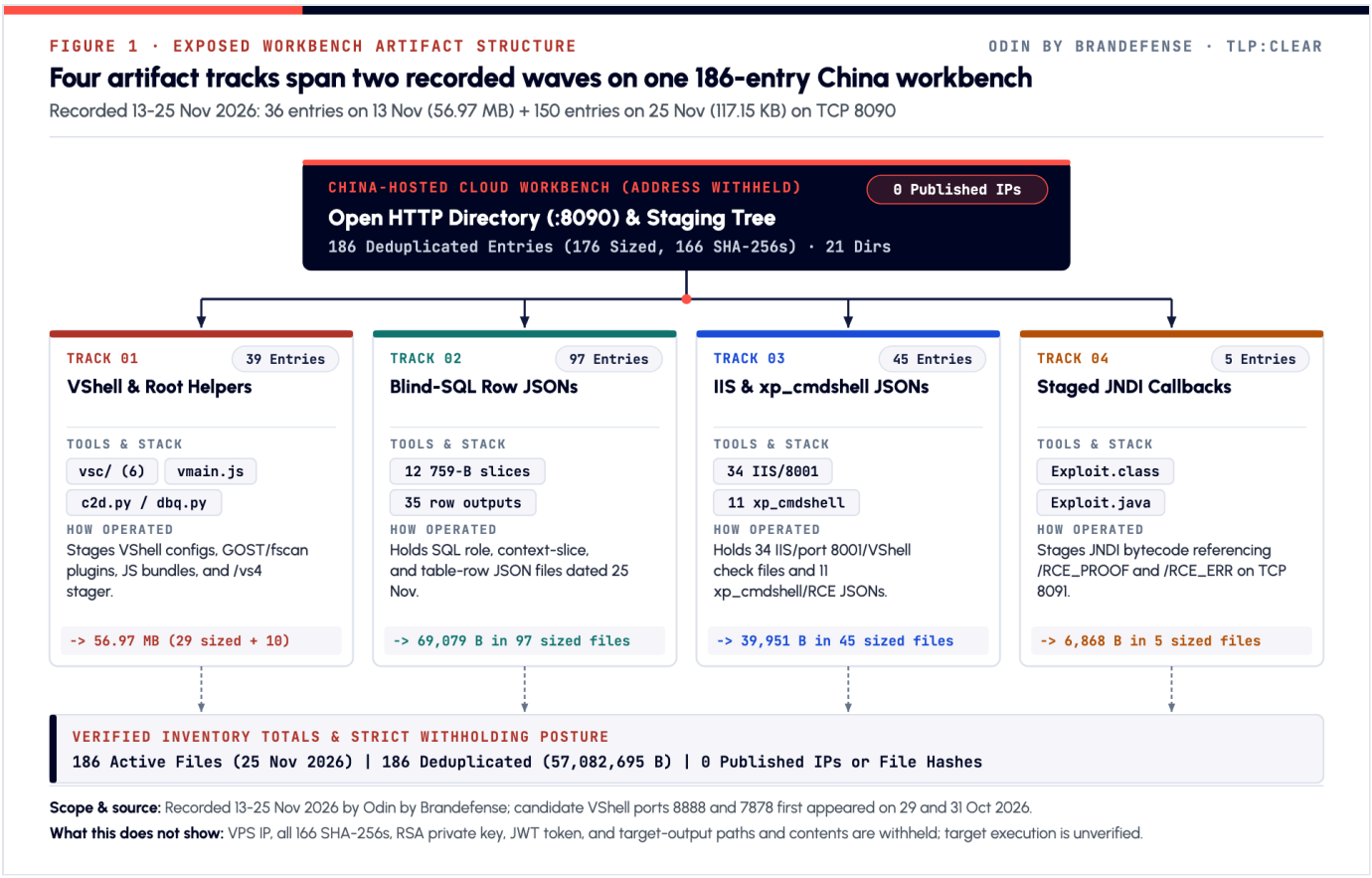

Figure 1: Four artifact tracks span two recorded waves on one 186-entry China workbench

Reconciling the 186 Deduplicated Entries Across Waves and Folders

On `25 November 2026`, the live `TCP 8090` directory reported **186 active files** and **21 directories** (2 populated top-level folders, `vsc/` and `jndi/`, plus `(root)`). All **36** entries recorded on `13 November` remained present when **150** additional entries were recorded on `25 November`, yielding **186 deduplicated entries** (`36 + 150 = 186`).

Those **186** entries span **176 sized files** totaling **57,082,695 bytes (57.08 MB)** and **10 unsized entries** (**9** extensionless or socket items plus **1** unsized `vs4.log`). Content inspection covered **6 stored binaries and bundles** (`Exploit.class`, `vmain.js`, `v2.js`, `bash`, `gost.x64.so`, `fscan.x64.elf`). Five indexed key and callback URL records were also reviewed, while the **144** `.json` files and **7** `.log` files were not retrieved in body form.

The **176** sized files span **166 unique SHA-256 digests** (all withheld). Eight exact-hash duplicate groups account for **9** duplicate copies across `vs1` – `vs3`, `vc.jar` / `vc2.jar`, the two LDAP logs, and five `.json` pairs. One **26.37 MB** binary (`gost.x64.exe`) had no recorded digest, completing the **10**-file difference (`176 - 10 = 166`).

Workspace zone	Top-level dirs	Deduplicated entries	Share (%)	Sized files	Unsized entries	Sized bytes	Operational role
Root Wave 2 Captures (`(root)`, `25 Nov`)	0 (root)	148	79.6%	148	0	114,417	Holds **142** Wave 2 `.json` files (**109,030** B; paths withheld), **3** JNDI logs (**4,134** B), **2** `.txt` markers, and `dbg_client.py`.
Root Wave 1 Helpers & Bundles (`(root)`, `13 Nov`)	0 (root)	30	16.1%	20	10	2,835,305	Holds `bash`, `vmain.js`, `v2.js`, `vh.html`, 4 `.py` helpers, `http8000.log`, `clients.json`, an operator token file (location withheld), and **10** unsized items.
VShell Configs & Plugins (`vsc/`, `13 Nov`)	1	6	3.2%	6	0	54,130,239	Holds an RSA private-key file (**1,679** B; location withheld), `server.pem`, `setting.conf`, and 3 `vsc/plugins/` GOST and `fscan` binaries.
JNDI Callback Directory (`jndi/`, `25 Nov`)	1	2	1.1%	2	0	2,734	Holds staged `jndi/Exploit.class` (**1,739** B, referencing `/RCE_PROOF` and `/RCE_ERR` on `TCP 8091`) and `jndi/Exploit.java` (**995** B).
Combined deduplicated inventory	**2 populated + root**	**186**	**100.0%**	**176**	**10**	**57,082,695**	**Combined `(root)` = 178 (2,949,722 B); Wave 1 = 36 (56,965,544 B) & Wave 2 = 150 (117,151 B).**

A second breakdown reconciles the same **186** entries (**176** sized, **10** unsized, **57,082,695** bytes) across five file types. Text, JSON, and log files represent **87.6%** of entries (**163**), while five binaries represent **97.5%** of total byte volume (**55.67 MB**).

Artifact class (`kind`)	Deduplicated entries	Share (%)	Sized files	Unsized entries	Sized bytes	Verified extensions & contents
JSON outputs, logs & source (`text`)	163	87.6%	162	1	123,656	144 `.json` (2 Wave 1 + 142 Wave 2), 7 `.log` (1 unsized), 5 extensionless, 2 `.txt`, 2 `.jar` stubs, and `.html` / `.conf` / `.java`.
Unsized stagers & socket (`other`)	9	4.8%	0	9	0	8 extensionless Wave 1 stagers (`vc1`, `vc1b`, `vc2`, `vc3`, `vclient`, `vn1` – `vn3`) and 1 unsized `.sock` entry (vendor socket name withheld).
Python helpers & JS bundles (`script`)	7	3.8%	7	0	1,287,543	5 `.py` scripts (7,111 B across `c2d2.py`, `c2d.py`, `lst.py`, `dbq.py`, `dbg_client.py`) and 2 `.js` bundles (1,280,432 B).
Tunnel, scan & bytecode binaries (`exe`)	5	2.7%	5	0	55,668,471	`gost.x64.exe` (26,366,976 B), `gost.x64.so` (21,542,180 B), `fscan.x64.elf` (6,217,056 B), `bash` (1,540,520 B), and `jndi/Exploit.class` (1,739 B).
TLS key & certificate (`secret`)	2	1.1%	2	0	3,025	an RSA private-key file (1,679 B; location withheld) and `server.pem` (1,346 B certificate).
Total	**186**	**100.0%**	**176**	**10**	**57,082,695**	**Exact byte-level parity (57.08 MB; 166 withheld SHA-256 digests).**

Figure 5 partitions all **186** entries into six mutually exclusive clusters (`97 + 34 + 33 + 11 + 6 + 5 = 186`). Blind-SQL context/row JSONs (**97**) and IIS/port **8001**/VShell JSONs (**34**) make up **70.4%** of all indexed items.

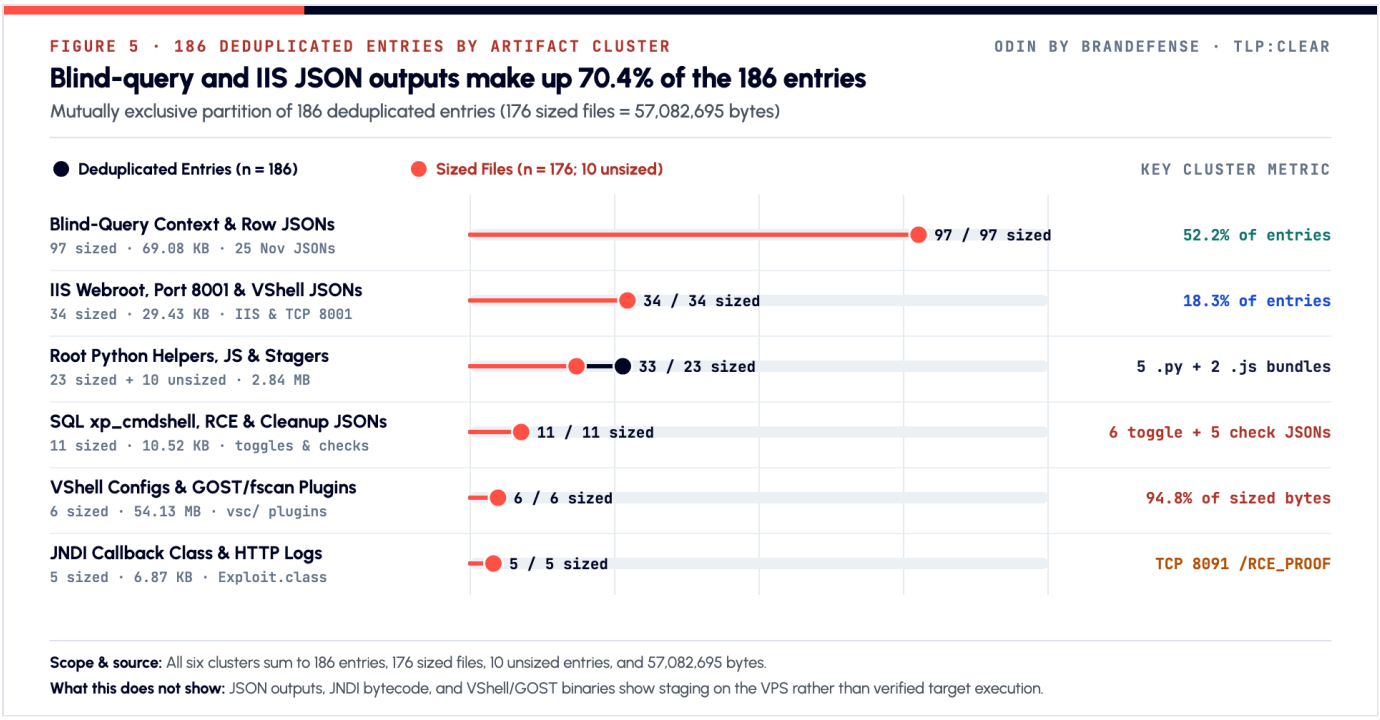

Figure 5: Blind-query and IIS JSON outputs make up 70.4% of the 186 entries

Co-Hosted Listener Ports on the China Cloud VPS

Service banners and staged callback URLs connect several TCP ports on the same withheld VPS. Each listener aligns with a specific stage of the workbench:

- **Candidate VShell listeners (`TCP 8888` and `TCP 7878`).** `TCP 8888` was first observed on `29 October 2026` and stayed active on `25 November 2026`, serving an `HTTP 200` `Welcome to nginx` decoy title. `TCP 7878` was first observed on `31 October 2026` and stayed active through `23 November 2026`. Both remain candidate C2 listeners rather than blocklist indicators without raw implant handshakes.
- **Python `3.14.4` HTTP listeners (`TCP 8090`, `TCP 8899`, and `TCP 8000`).** Ports **8090** and **8899** served `SimpleHTTP/0.6` directory listings. Meanwhile, `TCP 8000` ran `BaseHTTP/0.6` (`HTTP 404` on `/`), matching `http8000.log` (**2,060** B) and the indexed stager URI `/vs4` on `TCP 8000`.
- **JNDI callback and auxiliary ports (`TCP 8091` and `TCP 8082`).** `TCP 8091` is referenced inside `jndi/Exploit.class` (`/RCE_PROOF` and `/RCE_ERR`), while `TCP 8082` returned `HTTP 401` beside `TCP 80`, **443**, and **22**.

3. Blind SQL Enumeration, `xp_cmdshell` Activation, and IIS Webroot Probing

Wave 2 (`25 November 2026`) added **142** `.json` files (**109,030** bytes) to `(root)`. Those files sit beside `dbq.py` (**623** B, `13 Nov`) and `dbg_client.py` (**196** B, `25 Nov`). Figure 3 presents an analytical model separating verified `13 November` workbench staging (solid) from unverified target-side steps inferred from filenames (dashed).

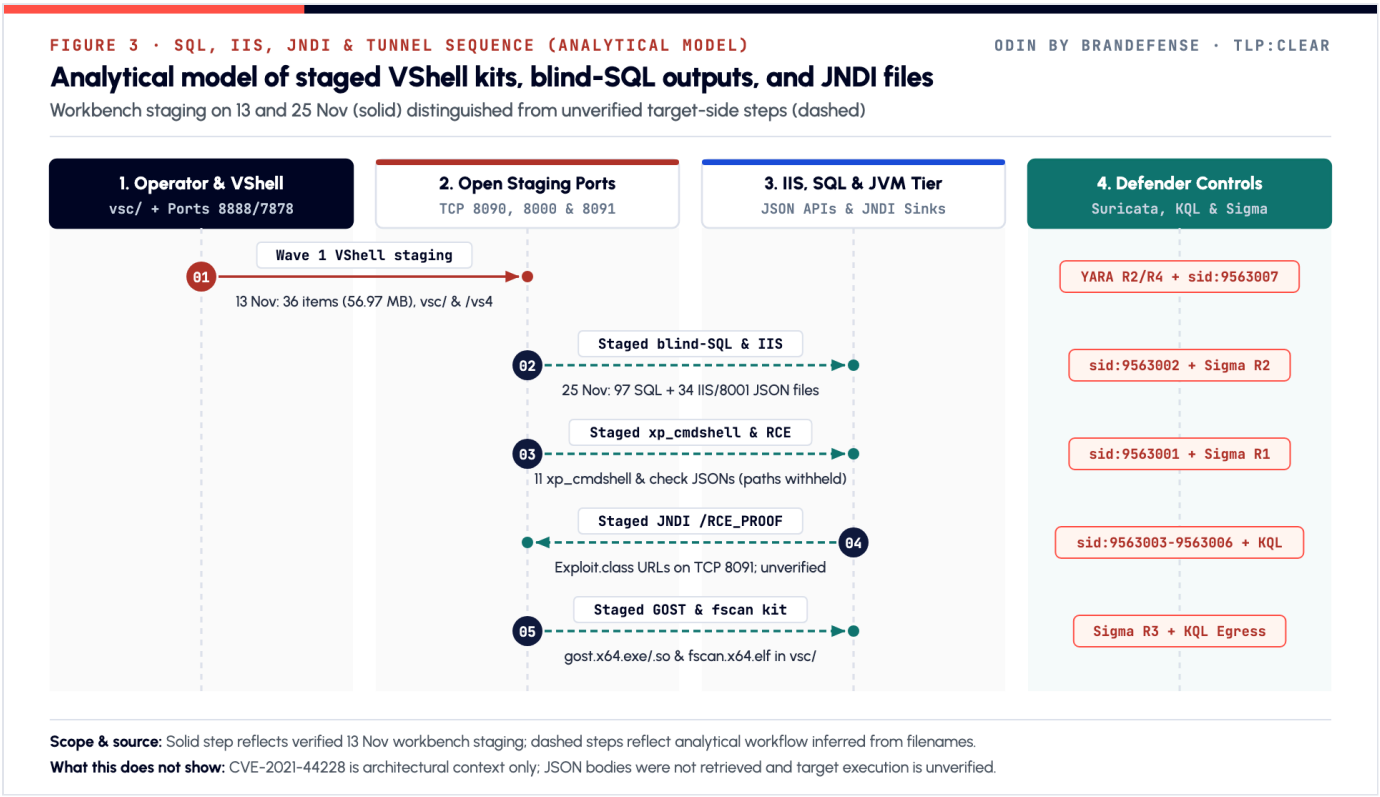

Figure 3: Analytical model of staged VShell kits, blind-SQL outputs, and JNDI files

Blind SQL Context Slices and Table-Row Enumeration (97 Entries)

Ninety-seven Wave 2 `.json` files (**69,079** bytes, **52.2%** of entries; paths withheld) have names consistent with a blind-SQL workflow. Their bodies do not verify target requests or returned data. Root pairs baseline (**62** B) and injected (**129** B) request templates with four calibration captures (**345** B to **759** B). Those four files are named for boolean, newline, response-length, and stacked-query checks.

Three database-context files (**513** B, **645** B, and **627** B; paths withheld) are named for SQL Server database-identity and role checks. Their naming aligns with SQL Server `DB_NAME()` and `IS_SRVROLEMEMBER('sysadmin')` checks used to test administrative privileges (`[REF-02]`).

Repeated file sizes are consistent with scripted captures, but the uninspected bodies do not confirm extraction. Twelve context-slice files measure **759 bytes each**. Thirty-five row-output files comprise nine 717-byte and 26 723-byte entries; their paths and contents are withheld.

Remaining captures include two aggregate output files (**1,221** B and **1,839** B), six secondary row slices (**717** B each), and three table- and view-count files. Thirty-three column-slot and row-check entries plus one **213**-byte summary file complete the cluster (paths withheld).

SQL Server `xp_cmdshell` and Companion Check Files (11 Entries)

Eleven Wave 2 `.json` files (**10,521** bytes; paths withheld) follow the SQL stored-procedure escalation pattern in MITRE ATT&CK `T1505.001` (`[REF-02]` , `[REF-07]`). Six filenames suggest `xp_cmdshell` existence, status, and activation checks (**591** B to **1,149** B across six captures).

Microsoft SQL Server disables `xp_cmdshell` by default (`[REF-02]`). However, a `sysadmin` login can re-enable it via `sp_configure` , causing `sqlservr.exe` to spawn Windows `cmd.exe` processes (`[REF-02]` , `[REF-07]`).

Four companion filenames suggest command-output checks, and a fifth suggests cleanup. Their bodies were not retrieved, so table writes, restoration, and target execution remain unverified.

IIS Webroot Discovery, Port 8001 Checks, and VShell Verification (34 Entries)

Thirty-four Wave 2 `.json` files (**29,430** bytes; paths withheld) are named for IIS webroot, internal-port, and VShell client checks. Eight files are named for IIS paths and sites (including **1,635**-byte and **1,149**-byte entries alongside count and row-output files). Three **849**-byte files are named for standard web folders.

Twenty-three additional files are named for host-recon, `TCP 8001` , and VShell checks (paths withheld). Recon files include **2,043**-byte and **1,545**-byte entries, two **699**-byte count files, and **11** row-output slices (**741** to **747** B). Port and implant checks span two **861**-byte `TCP 8001` files and six VShell check files (**675** B to **1,161** B). Exact-hash matches on two **675**-byte entries fit repeated empty responses.

4. Staged JNDI Callback Bytecode and the VShell / GOST Tunnel Suite

Alongside the **142** JSON files, the server pairs five Wave 2 JNDI callback files with **39** VShell, GOST, `fscan`, and Python helper files. Figure 4 shows the directory layout and co-hosted ports on the China VPS.



Figure 4: One China VPS pairs open TCP 8090 staging with co-hosted listener ports

Staged JNDI Bytecode and `/RCE_PROOF` Callback URLs (5 Entries)

Five Wave 2 files (**6,868** bytes) stage a Java JNDI callback harness (`[REF-01]` , `[REF-03]`). Inside `jndi/` , `Exploit.java` (**995** B) compiles to `Exploit.class` (**1,739** B). Stored inspection of `Exploit.class` reveals hardcoded HTTP callback paths `/RCE_PROOF` and `/RCE_ERR` directed to `TCP 8091` on the same VPS.

Root also holds `jndi_http.log` (**3,024** B) and two identical **555**-byte LDAP logs (`jndi_ldap.log` and `jndi_ldap_stdout.log`). In Apache Log4j2 `2.0-beta9` **through** `2.15.0` (excluding `2.12.2` , `2.12.3` , and `2.3.1`), `CVE-2021-44228` allowed JNDI remote code execution (`[REF-01]`).

Apache fixed that flaw in `2.16.0` and `2.17.0`+ (`[REF-01]`), while Oracle JDK `8u191+` , `11.0.1+` , and **17+** set `trustURLCodebase=false` by default (`[REF-03]`). We cite `CVE-2021-44228` **strictly as architectural context**, and staged JNDI files do not confirm target JVM execution.

VShell C2 Configs, GOST Tunnel Plugins, and Root Helpers (39 Entries)

Six Wave 1 files in `vsc/` (**54,130,239** bytes, **94.8%** of sized bytes) pair TLS and configuration files with three compiled plugins. Specifically, that set includes an RSA private-key file (**1,679** B; location withheld), `server.pem` (**1,346** B), and `setting.conf` (**1,002** B). Meanwhile, `vsc/plugins/` stages `gost.x64.exe` (**26.37 MB**) and `gost.x64.so` (**21.54 MB**) for GOST proxy tunneling (`[REF-04]` , `[REF-08]`). It also holds `fscan.x64.elf` (**6.22 MB**; `[REF-05]`).

Thirty-three root entries (**2,836,558** B across **23** sized + **10** unsized) complete the inventory. Wave 1 (**30** entries) holds `bash` (**1.54 MB**) and VShell web bundles (`vmain.js` **1.17 MB**, `v2.js` **114.80 KB**, `vh.html`). It also holds an operator token file (**217** B; location withheld), `clients.json` , and `tun.json` .

Root Wave 1 also holds four Python helpers (`c2d2.py` , `c2d.py` , `lst.py` , `dbq.py` ; `[REF-09]`) with logs and `http8000.log` (**2,060** B). It stages URI `/vs4` on `TCP 8000` , two `.jar` stubs, `vs1` – `vs3` , `.del` , and **10** unsized items. Wave 2 added `dbg_client.py` (**196** B) and two `.txt` markers (**1,057** B; one name withheld), and Figure 2 maps all **186** entries to detections.

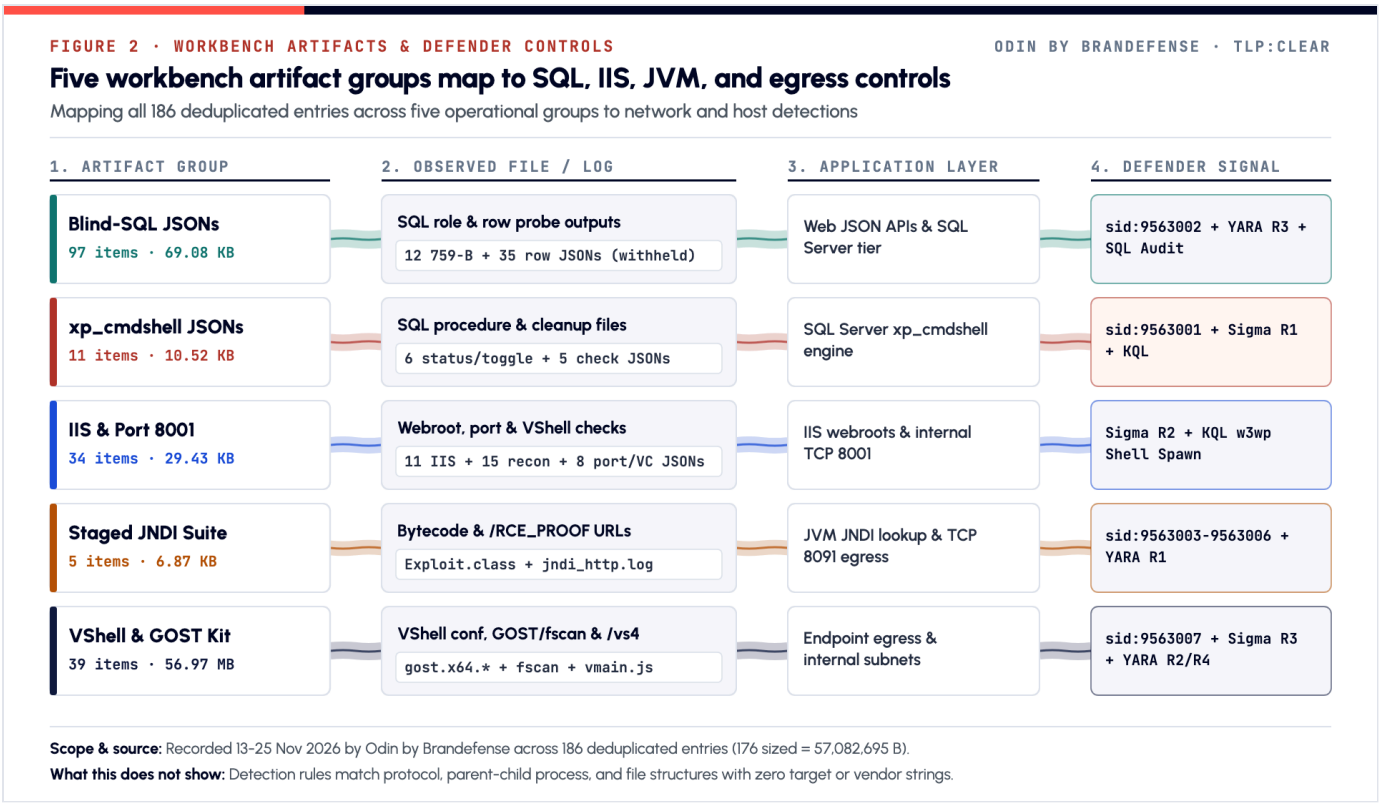

Figure 2: Five workbench artifact groups map to SQL, IIS, JVM, and egress controls

5. Enterprise SQL Server, IIS, JVM, and Egress Exposure Analysis

Platform and security teams should verify four controls mapped to the exposed workbench. Reviewing each layer directly avoids treating unverified staging files as an established breach root cause:

Exposure layer	Observed workbench evidence	Control to verify	Operational risk
JSON APIs & SQL Server roles	97 blind-SQL `.json` files (2 request templates, 3 role checks, 12 759-B context slices, 35 row-output files; paths withheld) + `dbq.py`.	Verify whether JSON endpoints use parameterized SQL queries and whether web DB logins lack `sysadmin` rights (`[REF-02]`, `[REF-10]`).	Blind SQL injection under `sysadmin` exposes database catalogs and server configuration toggles.

Exposure layer	Observed workbench evidence	Control to verify	Operational risk
SQL `xp_cmdshell` procedure	11 `.json` files (6 `xp_cmdshell` status/toggle captures, 4 command-output check files, 1 cleanup file; paths withheld and execution unverified).	Verify whether `xp_cmdshell` stays 0, SQL audit logs `sp_configure`, and `sqlservr.exe` uses a restricted service account (`[REF-02]`, `[REF-07]`).	Enabling `xp_cmdshell` lets SQL injection spawn OS shells (`cmd.exe` / `powershell.exe`) on database hosts.
IIS webroots & internal ports	34 `.json` files (11 IIS path/site/folder files, 15 host-recon files, 2 `TCP 8001` checks, 6 VShell check files; paths withheld).	Verify whether `IIS AppPool` accounts have read-only access on `inetpubwwwroot` and whether `TCP 8001` is segmented (`[REF-06]`, `[REF-10]`).	Writable webroots allow persistent web shells (`w3wp.exe` child shells) and internal port probing.
JVM JNDI & C2 tunnel egress	`jndi/Exploit.class` (`/RCE_PROOF` on :8091), 3 JNDI logs, `vsc/` (`gost.x64.*`, `fscan.x64.elf`), and `/vs4` on :8000.	Verify Log4j2 (`>= 2.17.1`) and JDK defaults (`trustURLCodebase=false`), plus strict firewall blocks on tier internet egress (`[REF-01]`, `[REF-03]`, `[REF-08]`).	Open outbound HTTP, LDAP, or WebSocket egress allows `.class` fetches, `/vs4` staging, and GOST tunnels.

6. Alternative Hypotheses

Hypothesis	Evidence for	Evidence against	Assessment
Wave 2 `xp_cmdshell`, VShell, and JNDI files confirm target RCE	Directory holds 11 `xp_cmdshell` and check `.json` files, 6 VShell check `.json` files (paths withheld), `Exploit.class` (`/RCE_PROOF`), and `jndi_http.log`.	Raw bodies of the 144 `.json` files and 7 `.log` files were not retrieved, and duplicate 675-byte hashes fit empty or negative check responses.	**Unverified for target execution.** Treat JSON and JNDI files as unverified check files and staged tooling.

Hypothesis	Evidence for	Evidence against	Assessment
Wave 2 JNDI files confirm `CVE-2021-44228` on the SQL/IIS target	`Exploit.class` and 3 JNDI logs were recorded on `25 November` alongside the 142 SQL and IIS JSON files.	Current JDKs block remote LDAP `.class` loading (`[REF-03]`), and operators often test JNDI against separate Java gateways or `TCP 8001` services.	**Unverified.** Cite `CVE-2021-44228` strictly as architectural context without merging unlinked target tiers.
Attacker staging workbench classification	Co-locates 54.13 MB of VShell, GOST, and `fscan` binaries, VShell console assets, an `nginx` -decoy :8888 port, 142 SQL and IIS probe JSON files, and compiled JNDI callback bytecode.	Passive directory listings and unretrieved JSON and log bodies do not identify the operator or verify target outcomes.	**Assessed attacker staging.** Co-located VShell server files, tunnel plugins, console assets, and probe files support this workbench classification. The operator's identity and target outcomes remain unknown.
Single exposed workbench reused across 13 and `25 November`	Six Wave 2 VShell-check filenames align with the VShell client components (`vc.jar` , `vc1` – `vc3` , `vclient`) staged on `13 November` .	Separate targets or operators may have used the directory across the two dates, and JSON bodies were not retrieved.	**Partially supported for directory reuse.** Shared `vc*` stager naming ties the `25 November` checks to the `13 November` kit on the same host without establishing one operator across targeting and output.

7. Validated Detection Rules

The China VPS IPv4 address and all **166** file SHA-256 digests are withheld. Defenders should deploy **protocol, process, and file-structure rules** that catch this tradecraft on any network. Static checks passed in the Odin Suricata linter (**0** problems) and offline structural checks across KQL, **3**

Sigma rules, and **4** YARA rules. Because live `suricata -T`, `Kusto.Language`, `pySigma`, and `YARA` binaries were not run locally, test each rule in staging first.

7.1 Suricata 7.x Network Signatures (`sid:9563001` –9563007)

Rules `sid:9563001`–**9563003** inspect inbound HTTP request bodies for `xp_cmdshell`, `IS_SRVROLEMEMBER` + `sysadmin`, and `${jndi:` expressions. Rules `sid:9563004`–**9563005** detect outbound `GET /RCE_` callbacks (`/RCE_PROOF`, `/RCE_ERR`) and `GET /jndi/Exploit.class` fetches. Rule `sid:9563006` alerts on inbound Java `.class` magic bytes (`0xCAFEBABE`), while `sid:9563007` matches exact 4-byte `GET /vs4` stager requests.

All seven HTTP `alert` rules generate alerts without blocking inline. They require cleartext HTTP visibility or deployment behind a TLS-terminating proxy, and `sid:9563001`–**9563002** and `sid:9563006` should be tuned around internal database admin tools or build servers.

```
SURICATA

# Odin by Brandefense - Monograph 63 (Blind SQL and Tunnel Staging) Suricata 7.x Rules
et
# Static syntax checked with Odin Suricata linter (0 problems); test with suricata -T
before deployment.

alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound SQL Server xp_cmd
shell Body Attempt"; flow:established,to_server; http.request_body; content:"xp_cmdshe
ll"; nocase; fast_pattern; classtype:web-application-attack; sid:9563001; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound SQL IS_SRVROLEMEM
BER sysadmin Probe"; flow:established,to_server; http.request_body; content:"IS_SRVROL
EMEMBER"; nocase; fast_pattern; content:"sysadmin"; nocase; classtype:attempted-recon;
sid:9563002; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound JNDI Lookup Expre
ssion in HTTP Body"; flow:established,to_server; http.request_body; content:"${jndi:";
nocase; fast_pattern; classtype:web-application-attack; sid:9563003; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R63 Outbound JNDI RCE_PROOF o
r RCE_ERR HTTP Callback"; flow:established,to_server; http.method; content:"GET"; bsiz
e:3; http.uri; content:"/RCE_"; startswith; fast_pattern; classtype:trojan-activity; s
id:9563004; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R63 Outbound JVM Exploit.clas
s HTTP Fetch"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.u
ri; content:"/jndi/Exploit.class"; fast_pattern; classtype:trojan-activity; sid:956300
5; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound Compiled Java Cla
ss Bytecode Stream"; flow:established,to_client; http.response_body; content:"|CA FE B
A BE|"; startswith; fast_pattern; classtype:trojan-activity; sid:9563006; rev:1;)
```

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R63 Outbound VShell /vs4 HTTP
Stager Request"; flow:established,to_server; http.method; content:"GET"; bsize:3; htt
p.uri; content:"/vs4"; bsize:4; fast_pattern; classtype:trojan-activity; sid:9563007;
rev:1;)
```

7.2 Microsoft Sentinel KQL Hunt

This 7-day Sentinel KQL query correlates SQL and IIS child shells, GOST and `fscan` switches, and unexpected public egress from backend tiers. Replace the two placeholder FQDNs (`dba-jump-01.corp[.]example` and `ci-java-01.corp[.]example`) in `ApprovedAdminAndBuildHosts` before scheduling.

```KQL
// Odin by Brandefense - Monograph 63 (Blind SQL and Tunnel Staging)
// Hunts SQL Server xp_cmdshell and IIS worker shell spawns, GOST/fscan CLI activity,
and unexpected SQL/IIS/Java public egress.
let Lookback = 7d;
let ApprovedAdminAndBuildHosts = dynamic(["dba-jump-01.corp.example", "ci-java-01.cor
p.example"]);
let SqlOrIisShellSpawns =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where (
        (InitiatingProcessFileName in~ ("sqlservr.exe", "w3wp.exe") and FileName in~
("cmd.exe", "powershell.exe", "pwsh.exe", "whoami.exe", "curl.exe", "certutil.exe"))
        or FileName in~ ("gost", "gost.exe", "gost.x64.exe", "fscan", "fscan.exe", "fs
can.x64.elf")
        or (ProcessCommandLine has "-L=" and ProcessCommandLine has "-F=")
        or ProcessCommandLine has_any ("mwss://", "gost.x64.so", "-nopoc")
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "HostSqlIisShellOrTunne
lCli", Detail = strcat(InitiatingProcessFileName, " -> ", FileName, ": ", ProcessComma
ndLine);
let UnexpectedTierPublicEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where RemoteIPType == "Public"
    | where (
        InitiatingProcessFileName in~ ("sqlservr.exe", "w3wp.exe")
        or (InitiatingProcessFileName in~ ("java", "java.exe") and RemotePort in (389,
636, 1099, 1389, 8091))
```

```
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, Signa
lType = "UnexpectedDatabaseWebOrJndiEgress", Detail = strcat(InitiatingProcessFileNam
e, " -> ", RemoteIP, ":", tostring(RemotePort), " (", RemoteUrl, ")");
union SqlOrIisShellSpawns, UnexpectedTierPublicEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count
(), Signals = make_set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceNa
me, AccountName
| order by LastSeen desc
```

7.3 Portable Sigma and YARA Rules

- **Sigma process-creation rules.** Rule 1 catches Windows `sqlservr.exe` spawning shells or transfer tools outside `sql-maint-01.corp[.]example`. Rule 2 catches Windows `w3wp.exe` spawning shells or `whoami.exe` / `netstat.exe` outside `iis-deploy-01.corp[.]example`. Rule 3 catches Linux `/gost` or `/fscan` binaries or `mwss://`, `gost.x64.so`, and `-nopoc` flags outside `net-relay-01.corp[.]example`.
- **YARA file-structure rules.** Four `filesize`-bounded rules detect `/RCE_PROOF` Java callback classes and VShell server configuration files. They also match Python `xp_cmdshell` harnesses and exposed VShell frontend JS bundles.

8. Behavioral and Protocol Hunt Markers (0 Published IPs or Workbench SHA-256s)

Zero server IPv4 addresses or workbench-file SHA-256 digests appear in the indicator feeds. The workbench holds an exposed RSA key, an operator JWT, and **142** probe JSON files, while ports **8888** and **7878** remain candidate VShell listeners. Defenders can hunt using the 10 behavioral markers below.

Marker category	Pattern / protocol artifact	Layer	Defensive hunt & control
SQL `xp_cmdshell` body	`xp_cmdshell` in HTTP request body	HTTP / WAF ingress	Keep `xp_cmdshell = 0` (`[REF-02]`), parameterize queries, and alert via `sid:9563001`.

Marker category	Pattern / protocol artifact	Layer	Defensive hunt & control
SQL role enumeration	`IS_SRVROLEMEMBER` + `sysadmin` in HTTP body	HTTP / SQL Audit	Remove `sysadmin` from web DB pools (`[REF-02]`) and alert via `sid:9563002`.
Inbound JNDI lookup	`${jndi:` in HTTP request body	HTTP / WAF ingress	Patch Log4j2 (`>= 2.17.1`, `[REF-01]`) and alert via `sid:9563003`.
JNDI HTTP proof callback	`GET /RCE_PROOF` or `GET /RCE_ERR` (`GET /RCE_`)	HTTP egress (:8091)	Block unapproved app-tier HTTP egress and alert via `sid:9563004` and YARA Rule 1.
Remote `.class` fetch	`GET /jndi/Exploit.class` & `0xCAFEBABE` stream	HTTP egress (:8090)	Keep JDK `trustURLCodebase=false` (`[REF-03]`) and alert via `sid:9563005` –9563006.
VShell stager URI	`GET /vs4` (exact 4-byte URI)	HTTP egress (:8000)	Block direct server HTTP downloads and alert via `sid:9563007`.
SQL child shell spawn	`sqlservr.exe` -> `cmd.exe` / `powershell.exe`	Windows EDR	Alert immediately on `sqlservr.exe` child shells (Sigma Rule 1; KQL; `[REF-07]`).

Marker category	Pattern / protocol artifact	Layer	Defensive hunt & control
IIS worker shell spawn	`w3wp.exe` -> `cmd.exe` / `powershell.exe` / `whoami.exe`	Windows EDR	Enforce read-only IIS webroots and alert via Sigma Rule 2 and KQL (`[REF-06]`).
GOST & `fscan` CLI	`gost` (`-L=` / `-F=` , `mwss://`) & `fscan` (`-nopoc`)	Windows / Linux EDR	Hunt GOST proxy plugins and `fscan` switches (Sigma Rule 3; KQL; `[REF-04]` , `[REF-05]`).
Tier public egress	`sqlservr.exe` , `w3wp.exe` , or `java` (389/636/1099/1389/8091)	Firewall / EDR	Block direct public internet egress from SQL, IIS, and JVM subnets (Sentinel KQL).

9. Hardening Priorities for SQL Server, IIS, and JVM Tiers

Database, Windows, Java, and network defenders can close each exposure path systematically. Apply the five engineering controls below in priority order:

1. **Keep `xp_cmdshell` disabled and enforce least-privilege SQL logins (`0-48 Hours`).** Confirm `xp_cmdshell` stays **0** in `sys.configurations` (`[REF-02]`) and alert on `sp_configure` (**15457** event ID). Migrate web connection strings off `sa` or `sysadmin` accounts to schema-scoped logins (`[REF-02]` , `[REF-10]`), and run `sqlservr.exe` under an isolated virtual service account (`[REF-07]`).
2. **Parameterize JSON API database queries (`0-72 Hours`).** Audit JSON POST endpoints to enforce prepared statements on every filter and offset (`[REF-06]` , `[REF-10]`). Suppress verbose SQL error traces to disrupt blind length-based probing.
3. **Lock down IIS webroot ACLs and internal ports (`Days 1-7`).** Grant `IIS AppPool` identities read-only access on `inetpubwwwroot` and virtual folders so file-write bugs cannot plant web shells (`[REF-10]`). Isolate internal ports like `TCP 8001` inside authenticated backend VLANs.
4. **Patch Log4j2 and harden JVM JNDI defaults (`Days 7-14`).** Keep Apache Log4j2 on `2.17.1` **or later** (`CVE-2021-44228` , `[REF-01]`) and run current LTS JDKs where

`com.sun.jndi.ldap.object.trustURLCodebase` defaults to `false` (`[REF-03]`). Ensure untrusted strings never reach `InitialContext.lookup()` .

5. **Block direct internet egress from SQL, IIS, and JVM subnets (`Days 14-30`).** Block outbound `HTTP` , `HTTPS` , `LDAP 389/636` , `RMI 1099/1389` , and high ports (**7878**, **8000**, **8090**, **8091**, **8888**) from backend tiers (`[REF-04]` , `[REF-08]`). Force approved outbound calls through an authenticated TLS-inspecting proxy to block `/vs4` , `/RCE_PROOF` , and GOST WebSocket tunnels.

10. Scope, Limits, and External References

Odin by Brandefense analyzed an exposed staging workbench on a China-hosted cloud VPS (`TCP 8090` , `SimpleHTTP/0.6`) recorded between `13 and 25 November 2026` . On `25 November 2026` , the Python `3.14.4` directory reported **186** active files and **21** directories.

That count matches all **186** deduplicated entries across **176** sized files (**57,082,695** bytes) and **10** unsized items (**166** SHA-256 digests). Wave 1 (`13 November`) contributed **36** entries (**56,965,544** B), and Wave 2 (`25 November`) added **150** entries (**117,151** B).

Content inspection covered **6** stored binaries and bundles (`Exploit.class` , `vmain.js` , `v2.js` , `bash` , `gost.x64.so` , `fscan.x64.elf`). Five indexed key and URL records were also reviewed. The **144** `.json` files and **7** `.log` files were not retrieved in body form. `CVE-2021-44228` is architectural context only.

Candidate VShell ports **8888** and **7878**, the VPS IPv4, all **166** SHA-256 digests, the RSA private-key file, the operator JWT file, target-output filenames, socket names, and target identifiers are withheld.

1. `[REF-01]` NIST National Vulnerability Database, *CVE-2021-44228 Detail: Apache Log4j2 JNDI Lookup Remote Code Execution* (verified on 25 November 2026).
2. `[REF-02]` Microsoft Learn, *xp_cmdshell Server Configuration Option (SQL Server Security Hardening)* (verified on 25 November 2026).
3. `[REF-03]` Oracle Java SE Documentation, *Java Naming and Directory Interface (JNDI) Architecture and Remote Class Loading Controls* (verified on 25 November 2026).
4. `[REF-04]` GOST (`GO Simple Tunnel`), *Official Source Repository and TLS/WebSocket Tunneling Documentation* (verified on 25 November 2026).

5. [REF-05] `fscan`, *Intranet Scanning Tool Source Repository* (verified on 25 November 2026).
6. [REF-06] MITRE ATT&CK, *Exploit Public-Facing Application (T1190)* (verified on 25 November 2026).
7. [REF-07] MITRE ATT&CK, *Server Software Component: SQL Stored Procedures (T1505.001)* (verified on 25 November 2026).
8. [REF-08] MITRE ATT&CK, *Protocol Tunneling (T1572)* (verified on 25 November 2026).
9. [REF-09] MITRE ATT&CK, *Command and Scripting Interpreter: Python (T1059.006)* (verified on 25 November 2026).
10. [REF-10] CISA, *Advisory AA23-347A: Stopping Adversaries Before They Exploit Internet-Facing Applications and Services* (verified on 25 November 2026).

BRANDEFENSE CAPABILITIES RELATED TO THIS RESEARCH

Cyber Threat Intelligence

Track the actors, infrastructure and campaigns behind threats like these.

Vulnerability Intel

Prioritise the flaws that attackers actually stage.

External Attack Surface

Find exposed middleware before it is targeted.

DEFENDER KIT

Indicators of compromise

Verified adversary infrastructure and tool hashes from this study. Every value below is also discussed in the report.

No network or file indicators are published for this study

The hosts in this report are exposed or unattributed systems rather than confirmed adversary infrastructure, so they stay anonymous. Use the detection rules below to find the same exposure in your own estate.

10 records from this study are not published in this table: candidate, suspected and dual-use hosts, victim-related values, and hashes of exposed secrets or files.

DEFENDER KIT

Detection rules

The rule files exactly as printed in the report. Test them in your own engine before deployment.

YARA **odin_jndi_vshell_and_sqli_artifacts.yar** 77 lines

```
/*
   Odin by Brandefense - Monograph 63 (Blind SQL and Tunnel Staging) YARA Suite
   4 filesize-bounded structural rules covering JNDI HTTP proof callback classes,
   VShell server configuration files, Python blind-SQL/xp_cmdshell harnesses, and
   exposed VShell web UI bundles (zero victim, organization, or vendor strings).
*/

rule ODIN_R63_JNDI_HTTP_Proof_Callback_Class
{
    meta:
        author = "Odin by Brandefense"
        monograph = "63"
        description = "Detects compiled Java .class files or Java sources embedding /R
CE_PROOF and /RCE_ERR HTTP callback paths"
        date = "2026-11-25"
    strings:
        $u1 = "/RCE_PROOF" ascii
        $u2 = "/RCE_ERR" ascii
        $j1 = "java/net/URL" ascii
        $j2 = "java/lang/Runtime" ascii
        $j3 = "ProcessBuilder" ascii
        $j4 = "HttpURLConnection" ascii
    condition:
        filesize < 64KB and $u1 and $u2 and 1 of ($j*)
}

rule ODIN_R63_VShell_Server_Configuration
{
    meta:
        author = "Odin by Brandefense"
        monograph = "63"
        description = "Detects staged VShell C2 server configuration files referencing
TLS certificates and WebSocket listener settings"
        date = "2026-11-25"
    strings:
        $c1 = "server.pem" ascii
        $k1 = "vshell" ascii nocase
        $k2 = "plugins" ascii
        $k3 = "websocket" ascii nocase
```

```
        $k4 = "secret" ascii nocase
    condition:
        filesize < 64KB and $c1 and 2 of ($k*)
}

rule ODIN_R63_Python_BlindSQL_and_XpCmdshell_Harness
{
    meta:
        author = "Odin by Brandefense"
        monograph = "63"
        description = "Detects Python scripts orchestrating SQL Server role checks, sp
_configure, and xp_cmdshell JSON capture workflows"
        date = "2026-11-25"
    strings:
        $s1 = "xp_cmdshell" ascii nocase
        $s2 = "sp_configure" ascii nocase
        $p1 = "IS_SRVROLEMEMBER" ascii nocase
        $p2 = "requests" ascii
        $p3 = "urllib" ascii
        $p4 = ".json" ascii
    condition:
        filesize < 128KB and $s1 and $s2 and 2 of ($p*)
}

rule ODIN_R63_VShell_WebUI_Bundle_Artifact
{
    meta:
        author = "Odin by Brandefense"
        monograph = "63"
        description = "Detects exposed VShell operator frontend JavaScript bundles ref
erencing GOST tunneling and fscan plugin modules"
        date = "2026-11-25"
    strings:
        $v1 = "vshell" ascii nocase
        $m1 = "gost" ascii nocase
        $m2 = "fscan" ascii nocase
        $m3 = "socks5" ascii nocase
        $m4 = "ws://" ascii
    condition:
        filesize < 3MB and $v1 and 3 of ($m*)
}
```

Suricata `odin_sqli_xpcmdshell_jndi_and_vshell_hunt.rules` 11 lines

```
# Odin by Brandefense - Monograph 63 (Blind SQL and Tunnel Staging) Suricata 7.x Rules
et
```

```
# Static syntax checked with Odin Suricata linter (0 problems); test with suricata -T
before deployment.

alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound SQL Server xp_cmd
shell Body Attempt"; flow:established,to_server; http.request_body; content:"xp_cmdshe
ll"; nocase; fast_pattern; classtype:web-application-attack; sid:9563001; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound SQL IS_SRVROLEMEM
BER sysadmin Probe"; flow:established,to_server; http.request_body; content:"IS_SRVROL
EMEMBER"; nocase; fast_pattern; content:"sysadmin"; nocase; classtype:attempted-recon;
sid:9563002; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound JNDI Lookup Expre
ssion in HTTP Body"; flow:established,to_server; http.request_body; content:"${jndi:";
nocase; fast_pattern; classtype:web-application-attack; sid:9563003; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R63 Outbound JNDI RCE_PROOF o
r RCE_ERR HTTP Callback"; flow:established,to_server; http.method; content:"GET"; bsiz
e:3; http.uri; content:"/RCE_"; startswith; fast_pattern; classtype:trojan-activity; s
id:9563004; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R63 Outbound JVM Exploit.clas
s HTTP Fetch"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.u
ri; content:"/jndi/Exploit.class"; fast_pattern; classtype:trojan-activity; sid:956300
5; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R63 Inbound Compiled Java Cla
ss Bytecode Stream"; flow:established,to_client; http.response_body; content:"|CA FE B
A BE|"; startswith; fast_pattern; classtype:trojan-activity; sid:9563006; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R63 Outbound VShell /vs4 HTTP
Stager Request"; flow:established,to_server; http.method; content:"GET"; bsize:3; htt
p.uri; content:"/vs4"; bsize:4; fast_pattern; classtype:trojan-activity; sid:9563007;
rev:1;)
```

Show full rule

Sentinel KQL **sentinel_sql_xpcmdshell_iis_jndi_and_gost_hunt.kql** 29 lines

```
// Odin by Brandefense - Monograph 63 (Blind SQL and Tunnel Staging)
// Hunts SQL Server xp_cmdshell and IIS worker shell spawns, GOST/fscan CLI activity,
and unexpected SQL/IIS/Java public egress.
let Lookback = 7d;
let ApprovedAdminAndBuildHosts = dynamic(["dba-jump-01.corp.example", "ci-java-01.cor
p.example"]);
let SqlOrIisShellSpawns =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where (
        (InitiatingProcessFileName in~ ("sqlservr.exe", "w3wp.exe") and FileName in~
```

```
("cmd.exe", "powershell.exe", "pwsh.exe", "whoami.exe", "curl.exe", "certutil.exe"))
        or FileName in~ ("gost", "gost.exe", "gost.x64.exe", "fscan", "fscan.exe", "fs
can.x64.elf")
        or (ProcessCommandLine has "-L=" and ProcessCommandLine has "-F=")
        or ProcessCommandLine has_any ("mwss://", "gost.x64.so", "-nopoc")
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "HostSqlIisShellOrTunne
lCli", Detail = strcat(InitiatingProcessFileName, " -> ", FileName, ": ", ProcessComma
ndLine);
let UnexpectedTierPublicEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where RemoteIPType == "Public"
    | where (
        InitiatingProcessFileName in~ ("sqlservr.exe", "w3wp.exe")
        or (InitiatingProcessFileName in~ ("java", "java.exe") and RemotePort in (389,
636, 1099, 1389, 8091))
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, Signa
lType = "UnexpectedDatabaseWebOrJndiEgress", Detail = strcat(InitiatingProcessFileNam
e, " -> ", RemoteIP, ":", tostring(RemotePort), " (", RemoteUrl, ")");
union SqlOrIisShellSpawns, UnexpectedTierPublicEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count
(), Signals = make_set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceNa
me, AccountName
| order by LastSeen desc
```

Sigma `sigma_xpcmdshell_iis_and_gost_fscan.yml` 91 lines

```
title: SQL Server sqlservr.exe Spawning Command or Transfer Utility
id: 6c92d4a1-7e14-4b83-9c21-63a10d84f401
status: experimental
description: Detects Microsoft SQL Server database engine processes spawning Windows c
ommand shells or file-transfer utilities consistent with xp_cmdshell execution.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: windows
detection:
  selection_parent:
    ParentImage|endswith: '\sqlservr.exe'
  selection_child:
    Image|endswith:
      - '\cmd.exe'
```

```
      - '\powershell.exe'
      - '\pwsh.exe'
      - '\curl.exe'
      - '\certutil.exe'
  filter_sql_maint:
    ComputerName:
      - 'sql-maint-01.corp.example'
  condition: selection_parent and selection_child and not filter_sql_maint
falsepositives:
  - Legacy SQL Server Agent or maintenance hosts explicitly authorized to run OS scrip
ts
level: high
tags:
  - attack.persistence
  - attack.t1505.001
---
title: IIS Worker Process w3wp.exe Spawning Shell or Recon Utility
id: 6c92d4a1-7e14-4b83-9c21-63a10d84f402
status: experimental
description: Detects IIS worker processes spawning interactive shells or host reconnai
ssance binaries outside approved deployment hosts.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: windows
detection:
  selection_parent:
    ParentImage|endswith: '\w3wp.exe'
  selection_child:
    Image|endswith:
      - '\cmd.exe'
      - '\powershell.exe'
      - '\pwsh.exe'
      - '\whoami.exe'
      - '\netstat.exe'
  filter_iis_deploy:
    ComputerName:
      - 'iis-deploy-01.corp.example'
  condition: selection_parent and selection_child and not filter_iis_deploy
falsepositives:
  - Self-hosted web deployment or administrative CGI portals on isolated management no
des
level: high
tags:
  - attack.initial-access
  - attack.t1190
```

```yaml
---
title: GOST Tunnel or fscan Intranet Scanner Execution
id: 6c92d4a1-7e14-4b83-9c21-63a10d84f403
status: experimental
description: Detects execution of GOST tunnel plugins or the fscan intranet scanner vi
a binary names or characteristic WebSocket/scan command-line switches.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_bin:
    Image|endswith:
      - '/gost'
      - '/fscan'
      - '/fscan.x64.elf'
  selection_cli:
    CommandLine|contains:
      - 'mwss://'
      - 'gost.x64.so'
      - '-nopoc'
  filter_net_relay:
    ComputerName:
      - 'net-relay-01.corp.example'
  condition: (selection_bin or selection_cli) and not filter_net_relay
falsepositives:
  - Approved internal proxy relay or authorized red-team assessment appliances
level: high
tags:
  - attack.command-and-control
  - attack.t1572
```

