FULL REPORT · OPERATIONS · MONOGRAPH 64 / 61 · 18 MIN READ

Operation DockHarvest: SQL CLR, Rogue MySQL, and Logback Socket Risks

Inside a 736-entry United States workbench combining staged SQL Server CLR and OPENQUERY scripts, a rogue MySQL 8888 listener, Logback XML stagers, and /recv callback routes

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

Operation DockHarvest sits on an open TCP 8080 adversary staging workbench on a United States server recorded on 13–20 November 2026. The directory reported 874 active files on 20 November and 736 deduplicated entries totaling 299.31 megabytes. It holds 15 staged SQL Server CLR and OPENQUERY scripts, a rogue MySQL listener on port 8888, Logback XML stagers, and six /recv callback files. Enforce SQL CLR strict security, disable MySQL client local file reads, and patch Logback libraries.

Enterprise supply-chain and backend apps often pair SQL Server and IIS web tiers with Java services, MySQL drivers, and CLI coding agents. Permissive database assemblies, unsafe JDBC flags, or credentials in files can expose secrets across tiers. Operation DockHarvest brings four backend and credential tracks onto one adversary staging server.

Between 13 and 20 November 2026, Odin by Brandefense studied an open HTTP directory on TCP port 8080 of a United States server. That host is assessed consistently as an adversary staging workbench. On 20 November 2026, the live directory header reported 874 active files and 284 directories.

Across the 13 to 20 November window, our deduplicated index captured 736 entries. Those entries span 428 sized files totaling 299,306,613 bytes (299.31 megabytes) and 308 unsized entries across 425 unique digests.

The workbench holds a credential output file, an SSH private key, target-portal login scripts, and MySQL logs with 335 client addresses. Odin by Brandefense therefore publishes zero IPv4 addresses and zero file SHA-256 digests. All target organization names, portal filenames, client addresses, and external open-directory addresses are withheld.

First, eighteen root entries stage SQL Server CLR, OPENQUERY, and logistics table scripts. Fifteen sized `clr_*` scripts (**75,548** bytes) sit beside one **997**-byte credential JSON file (name withheld) and two unsized items. They target SQL CLR settings, `OPENQUERY` links, IIS configs, and an inventory management system (`WMS`), though target execution is unverified.

Second, rogue MySQL logs on port 8888 record 1,779 connections from 335 withheld addresses. Three files (**27.84** megabytes) run and log a rogue MySQL server targeting **11** client files into `/tmp/stolen`. Across **5,461** captured lines of `evil_mysql8888.log`, **1,393** of **1,779** connections dropped during authentication and **28** lines mention `/tmp/stolen` or saves.

Third, three Logback XML and binary files stage Java naming and socket tests. Root holds `logback-evil.xml` (**919** bytes), `logback-socket.xml` (**681** bytes), and unsized `socket_recv.bin`. Legacy flaws `CVE-2021-42550` and `CVE-2017-5929` provide architectural context, while staged XML files do not confirm target JVM execution.

Fourth, six /recv endpoint files and seven Claude agent files expose connector and CLI risks. A `TCP 8899` listener pairs with `http8080.log` (**242.83** megabytes) and six `/recv/` files (`t5.bin`, `sdk`, `mcp`, `sse`, `acp`, `unnamed`). Sixteen more entries hold seven `.claude` files, eight withheld portal-probe items, and one withheld SSH private key.

Three Decisions for Enterprise Security Leaders

1. **Enforce SQL Server CLR strict security and audit linked servers.** Keep CLR execution off unless signed assemblies are required, verify CLR strict security stays enabled, audit OPENQUERY logins, and protect IIS connection strings.
2. **Disable MySQL client local file transfers and upgrade Logback.** Force local infile settings off in MySQL Connector/J and CLI clients, upgrade Logback past legacy vulnerable releases, and block outbound Java logging ports.
3. **Lock down AI coding-agent credentials and outbound callback paths.** Restrict file permissions on developer `.claude` credentials, rotate tokens in backup snapshots, and monitor unexpected HTTP posts to `/recv/` routes.

2. Workbench Architecture, 736-Entry Reconciliation, and Timeline Scope

Operation DockHarvest operated from a **United States commercial virtual private server** (address withheld) assessed as an **adversary staging workbench**. That host exposed an unauthenticated HTTP directory on `TCP 8080` beside a **242,826,779**-byte access log (`http8080.log`). Enterprise defenders use dark web monitoring to spot exposed developer agent credentials and database connection secrets before reuse.

Odin by Brandefense recorded the directory between `13 and 20 November 2026`. Detection engineers also apply cyber threat intelligence to correlate SQL CLR tooling, rogue MySQL listeners, and Logback stagers. Figure 1 shows the four operational tracks on the workbench.

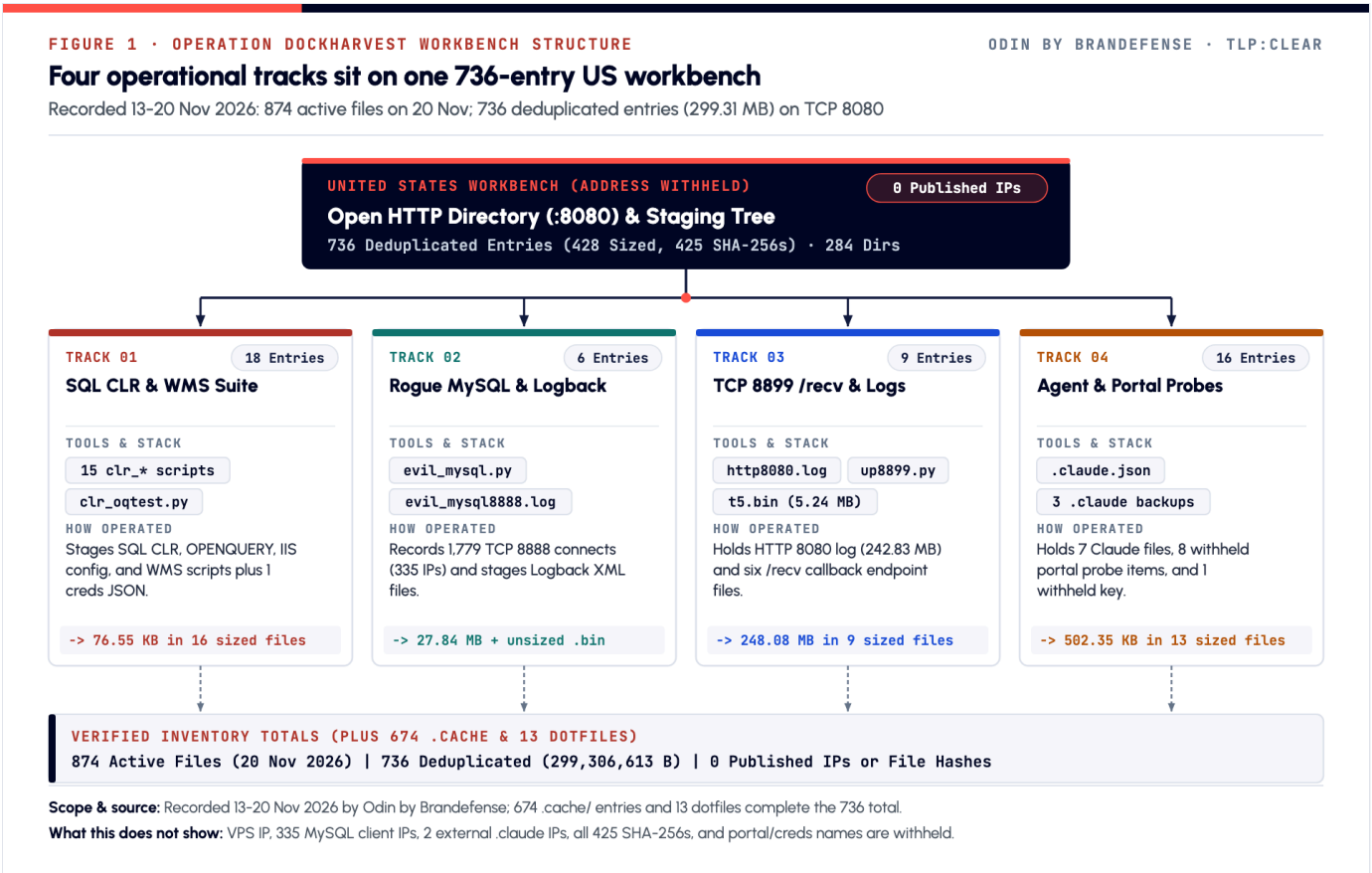

Figure 1: Four operational tracks sit on one 736-entry US workbench

Reconciling the `20 November` Listing (874 Files) and 736 Deduplicated Entries

Two counts describe the workbench across `13-20 November 2026`. On `20 November 2026`, the live open-directory header reported **874 active files** and **284 directories** (**7** top-level folders plus **277** nested subdirectories). Almost all nested subdirectories are two-character hexadecimal shard folders inside `/.cache/go-build/`.

Across `13-20 November 2026`, deduplicated indexing captured **736 total entries**. Those **736** entries span **428 sized files** totaling **299,306,613 bytes (299.31 MB)** and **308 unsized entries**. Bounded shard traversal inside `.cache/go-build/` explains why **674** `.cache/` entries were indexed rather than every leaf shard in the **874**-file header.

All **62** non-cache operational, log, config, and `/recv/` entries across `(root)` and the other six folders are captured in full. The **736** entries span **425 unique SHA-256 digests**, all withheld. The table below reconciles all **736** entries across four workspace zones.

Workspace zone	Top-level dirs	Deduplicated entries	Share (%)	Sized files	Unsized entries	Sized bytes	Operational role
Go Build & Tooling Cache (`.cache/`)	1	674	91.6%	374	300	22,770,504	Go compiler cache shards (`/.cache/go-build/`, 374 sized + 300 unsized) built for Go recon utilities.
Root Operator Workspace (`(root)`)	0 (root)	40	5.4%	37	3	270,785,443	Holds `http8080.log`, 15 `clr_*` scripts + 1 withheld creds JSON + 2 unsized items, `evil_mysql` files, `logback-*.xml`, unsized `socket_recv.bin`, `up8899.py` / `.log`, 3 withheld portal scripts, 1 withheld SSH key, `.claude.json`, and 8 dotfiles.
Agent, Probe & Host Subfolders (5 dirs)	5	16	2.2%	11	5	506,793	Spans `.claude/` (6 sized = 445,245 B), 1 withheld probe folder (5 items: 2 sized = 47,428 B + 3 unsized), `.config/` (3), `.ssh/` (1), and `.local/` (1).
TCP 8899 Receiver Directory (`recv/`)	1	6	0.8%	6	0	5,243,873	Holds 6 sized callback files under `/recv/`: `t5.bin` (5,242,880 B), `sdk` (441 B), `mcp` (164 B), `sse` (164 B), `acp` (159 B), and `unnamed` (65 B).
Combined deduplicated inventory	**7 top dirs + root**	**736**	**100.0%**	**428**	**308**	**299,306,613**	**Spans 425 withheld SHA-256 digests (874 active files & 284 dirs on `20 Nov 2026`).**

A second breakdown reconciles the same **736** entries (**428** sized, **308** unsized, **299,306,613** bytes) by file type. Cache and metadata entries (**400**), text and logs (**251**), and archived cache shards (**62**) make up **96.9%** of entries. By byte size, root log files inside `text` (**271.24 MB**) make up **90.6%** of the server.

Artifact class (`kind`)	Deduplicated entries	Share (%)	Sized files	Unsized entries	Sized bytes	Verified extensions & contents
Cache & metadata (`other`)	400	54.3%	95	305	249,069	395 extensionless cache/probe items, 3 `.hdr`, 1 `.legal-displayed`, and 1 `.token`.

Artifact class (`kind`)	Deduplicated entries	Share (%)	Sized files	Unsized entries	Sized bytes	Verified extensions & contents
Logs, configs & text (`text`)	251	34.1%	249	2	271,239,519	229 extensionless items, 5 `.log`, 5 `.json` (1 withheld 997 B creds file), 3 `.txt`, 2 `.xml`, 1 `.md`, 1 `.body` (47,102 B), 1 `.yaml`, and 2 `.claude.json.backup.*` files.
Archived cache (`archive`)	62	8.4%	62	0	22,483,582	62 extensionless Go build cache objects under `/.cache/go-build/`.
Executable scripts (`script`)	20	2.7%	20	0	89,884	18 `.py` (14 `clr_*.py`, `evil_mysql.py`, `up8899.py`, 2 withheld portal scripts) and 2 `.sh` (`clr_guess_tables.sh`, 1 withheld portal script).
Binary payloads (`exe`)	2	0.3%	1	1	5,242,880	2 `.bin` entries: `/recv/t5.bin` (5,242,880 B) and unsized `/socket_recv.bin`.
Private keys (`secret`)	1	0.1%	1	0	1,679	1 SSH RSA private key file in root (1,679 B; filename withheld).
Total	**736**	**100.0%**	**428**	**308**	**299,306,613**	**Exact byte parity (299.31 MB; shares rounded to 0.1%).**

Figure 5 groups all **736** deduplicated entries into six mutually exclusive clusters (`674 + 18 + 16 + 13 + 9 + 6 = 736`). Go build cache shards represent **91.6%** of entries (**674**), while HTTP/receiver logs (**82.9%**) and rogue MySQL/Logback files (**9.3%**) hold **92.2%** of sized bytes.

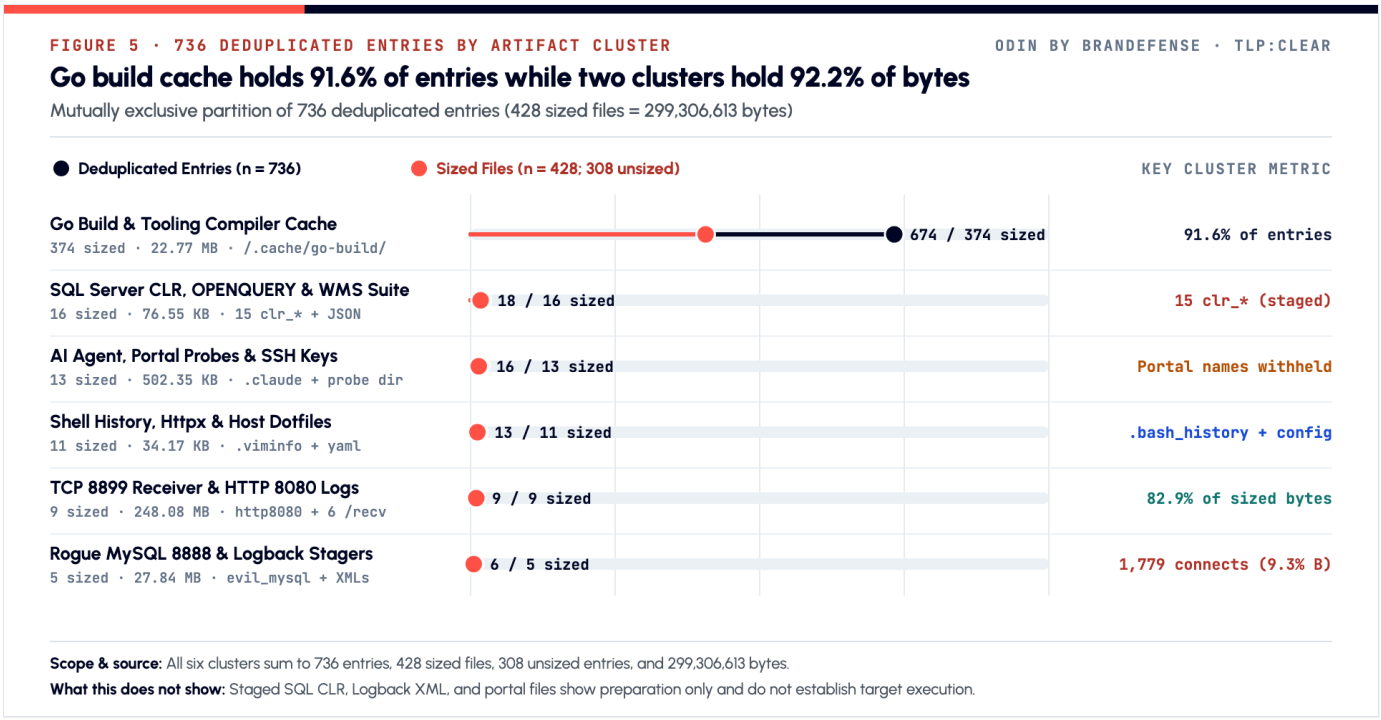

Figure 5: Go build cache holds 91.6% of entries while two clusters hold 92.2% of bytes

Observation Dates Versus Host Provisioning Files

Every date in the dataset carries a distinct boundary. Odin by Brandefense observed the open `TCP 8080` directory between `13 and 20 November 2026`, with **874** active files and **284** directories reported on `20 November 2026`.

Root holds `virt-sysprep-firstboot.log` (**547** B) from virtual machine image setup. Inside `.config/go/`, one **2,516**-byte local Go configuration JSON file carries a **2026-06-30** filename date beside `.config/httpx/config.yaml` (**11,028** B).

That June filename marks when the base Go configuration file was initialized, not how long the workbench operated. Likewise, `http8080.log` (**242.83 MB**) and `evil_mysql.log` (**26.66 MB**) reflect internet scan volume rather than continuous targeted testing.

3. SQL Server CLR Integration, `OPENQUERY`, and Rogue MySQL 8888 Mechanics

Supply-chain and inventory platforms frequently run IIS web apps backed by Microsoft SQL Server beside Java services using MySQL drivers. Figure 3 shows an analytical model separating logged `TCP 8888` MySQL connections (solid) from staged SQL CLR, Logback, portal, and `/recv/` files (dashed).

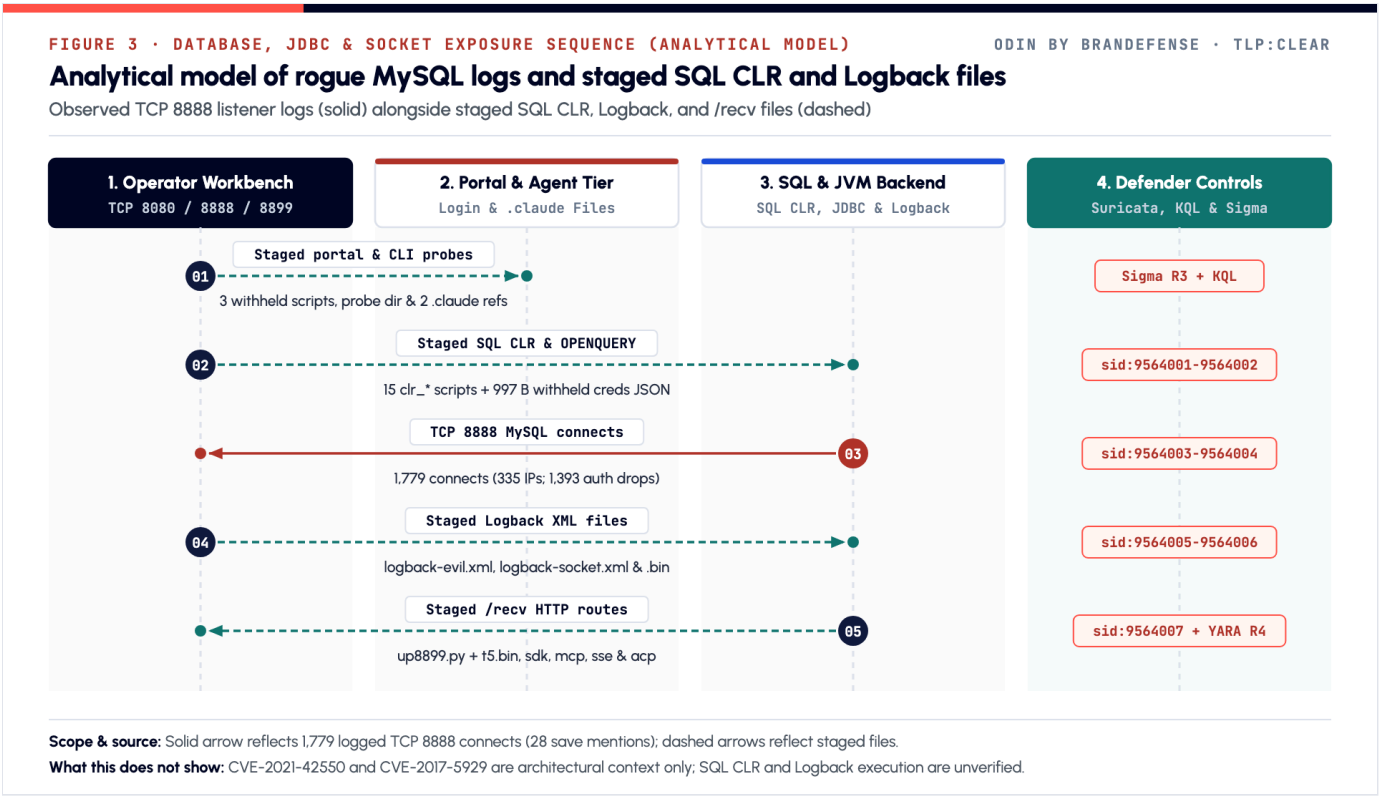

Figure 3: Analytical model of rogue MySQL logs and staged SQL CLR and Logback files

Hardening SQL Server CLR Assemblies, `OPENQUERY`, and IIS Connection Strings (18 Entries)

The SQL Server and supply-chain logistics system (`WMS`) cluster holds **18** entries (**16** sized files totaling **76,545** bytes plus **2** unsized root items). Fifteen sized `clr_*` scripts (**75,548** bytes) sit beside one **997**-byte credential output JSON file (name withheld). While numbered revisions show structured preparation, directory files alone do not confirm external SQL execution.

Script role in `CL-02`	Count	Sized bytes	Staged workbench artifacts
Instance & `OPENQUERY` checks	4	20,927	`clr_fingerprint.py` (8,291 B), `clr_jump_probe.py` (4,901 B), `clr_multi.py` (4,875 B), `clr_oqtest.py` (2,860 B).
IIS config & credential queries	5	27,250	`clr_iis_find.py` (9,545 B), `clr_get_creds3.py` (5,251 B), `clr_get_creds.py` (4,784 B), `clr_get_creds2.py` (4,482 B), `clr_get_connstr.py` (3,186 B).
Schema & WMS table extraction	6	27,373	`clr_dump_wms.py` (9,197 B), `clr_fast_dump.py` (6,274 B), `clr_dump_final.py` (4,327 B), `clr_tablescan.py` (3,312 B), `clr_scan_tables.py` (2,322 B), `clr_guess_tables.sh` (1,941 B).
Credential JSON & unsized items	3	997	1 withheld credential output JSON (997 B) plus 2 unsized root entries (0 B).

Script role in `CL-02`	Count	Sized bytes	Staged workbench artifacts
Cluster `CL-02` total	18	76,545	16 sized files + 2 unsized entries in `(root)`.

SQL Server leaves `clr enabled` off (**0**) by default (`[REF-01]`). Since SQL Server **2017**, `clr strict security` also defaults to **1**, requiring cryptographic signatures even on `SAFE` and `EXTERNAL_ACCESS` `.NET` assemblies (`[REF-01]`).

If admins disable `clr strict security`, set `TRUSTWORTHY ON`, or run apps as `sysadmin`, operators can load `UNSAFE` CLR procedures (`T1505.001` ; `[REF-08]`). Those procedures can read IIS `web.config` files and probe internal hosts, while `OPENQUERY` (`[REF-02]`) relays queries across linked servers (`sys.servers`).

Blocking Rogue MySQL `LOCAL INFILE` Client File Reads (3 of 6 Cluster Entries)

Three sized files (**27,837,567** bytes) run and log a rogue MySQL server on the host. Root holds `evil_mysql.py` (**6,382** B), `evil_mysql.log` (**26,658,404** B), and `evil_mysql8888.log` (**1,172,781** B on disk). Under the MySQL wire protocol, a server can answer an initial query with byte `0xfb` and a file path (`[REF-06]`).

The server chooses which file to request regardless of the client's SQL query (`[REF-06]`). Clients using `allowLoadLocalInfile=true` or Connector/J builds before `8.0.15` (`[REF-07]`) upload that local file back to the server.

A **196,357-character (5,461-line) captured excerpt** of `evil_mysql8888.log` provides exact listener metrics. Chinese startup headers log `Evil MySQL` on `0.0.0.0:8888`, output folder `/tmp/stolen`, and **11 configured target files**.

Across those **5,461** lines, **1,779 client connections** arrived from **335 withheld IPv4 addresses**, and **1,393 (78.3%)** dropped during authentication. Twenty-eight lines mention `/tmp/stolen` or save events, though scanner noise and missing `/tmp/stolen` files prevent confirming third-party file reads.

4. Staged Logback Socket XMLs, `/recv/` Agent Endpoints, and Credential Hygiene

Beside the SQL CLR suite and rogue MySQL listener, the server holds staged Logback XML files, a `TCP 8899` upload receiver, `.claude` configs, and withheld portal probes. Figure 4 shows the directory layout across `(root)` and the seven top-level folders.



Figure 4: One open TCP 8080 directory spans root scripts and seven top-level folders

Staged Logback XML Configurations and Socket Deserialization Context (3 Entries)

Three remaining items in the MySQL and Logback cluster target QOS[.]ch Logback: `logback-evil.xml` (**919** B), `logback-socket.xml` (**681** B), and unsized `socket_recv.bin`. Two NVD records explain why operators stage these files (`[REF-03]` , `[REF-04]` , `[REF-05]`).

In Logback **1.2.x before** `1.2.9` (plus `1.3.0-alpha0` to `1.3.0-alpha10`), `CVE-2021-42550` allowed LDAP code loading from modified, reloaded XML configurations (`[REF-03]`). QOS[.]ch fixed that flaw in **Logback** `1.2.9` **and** `1.3.0-alpha11` (`[REF-03]`).

Earlier, `CVE-2017-5929` affected Logback **prior to** `1.2.0` (fixed in `1.2.0`), where `SocketServer` and `ServerSocketReceiver` deserialized untrusted Java `ObjectInputStream` streams (`[REF-04]`). Logback's release notes also explain that `SocketReceiver` opens outbound TCP sockets to read serialized events (`[REF-05]`). Both CVEs are cited **strictly as architectural context**, and staged files do not confirm target JVM execution.

Monitoring `TCP 8899` `/recv/` Callback Endpoints and `TCP 8080` Logs (9 Entries)

Nine sized files (**248,083,871** bytes, **82.9%** of sized bytes) cover HTTP directory logs and a callback receiver. In `(root)` , `http8080.log` (**242,826,779** B) sits beside `up8899.py` (**1,015** B) and `up8899.log` (**12,204** B).

Six sized files under `/recv/` total **5,243,873** bytes, led by `t5.bin` (**5,242,880** B) and `sdk` (**441** B). Four smaller files cover `mcp` (**164** B), `sse` (**164** B), `acp` (**159** B), and `unnamed` (**65** B). Names matching Model Context Protocol (`mcp`),

Server-Sent Events (`sse`), and Agent Communication Protocol (`acp`) highlight outbound connector paths (`T1071.001` ; `[REF-10]`).

Protecting AI-Agent Credentials, Portal Probes, and Go Recon Configs (703 Entries)

Sixteen entries (**502,353** sized bytes across **13** sized files plus **3** unsized) cover AI-agent configs, portal probes, and SSH keys. Seven Claude CLI files (**446,307** B) include `.claude/cache/changelog.md` (**443,393** B), `.claude.json` (**1,062** B), and **3** `.claude.json.backup.*` snapshots. Two cross-directory references point to credential files on two external open directories; their paths and addresses are withheld (`[REF-09]`).

Eight withheld target-portal entries total **54,367** bytes across **3** root login scripts (**6,939** B) and **5** probe-folder items (**47,428** B, including a **47,102**-byte `.body` file). One withheld **1,679**-byte SSH RSA private key completes that **16**-entry cluster.

Finally, **674** `.cache/` shards (**22.77 MB**) pair with **13** host dotfiles (**34,173** B across **11** sized and **2** unsized items). Sized dotfiles include `.viminfo` (**11,054** B), `.config/httpx/config.yaml` (**11,028** B), `.bash_history` (**4,524** B), and `.bashrc` (**3,359** B). Smaller items hold a **2,516**-byte `.config/go/` JSON file, `.ssh/authorized_keys` (**576** B), `virt-sysprep-firstboot.log` (**547** B), and `.profile` (**377** B).

Figure 2 maps the five operational artifact groups to network, host, and file detections.

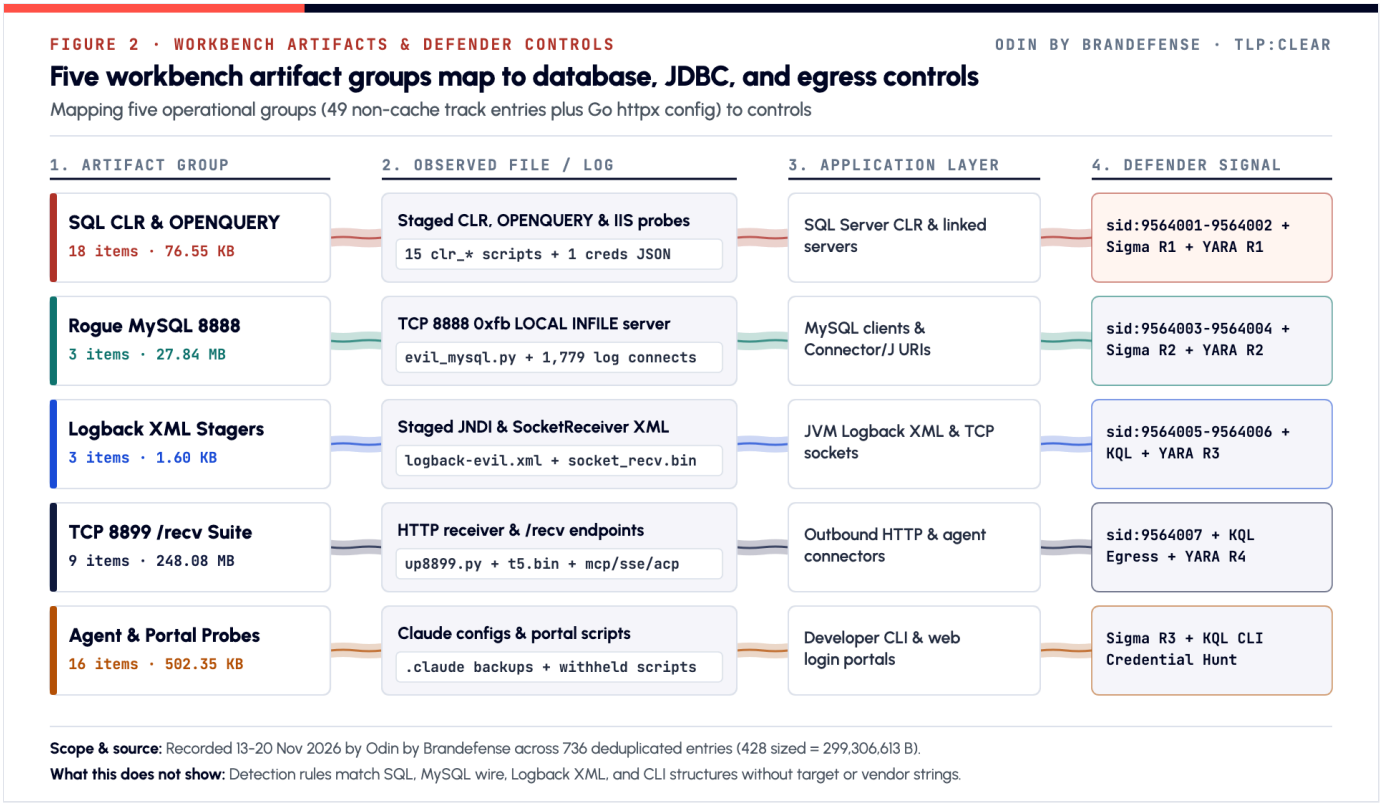

Figure 2: Five workbench artifact groups map to database, JDBC, and egress controls

5. Enterprise Database, JDBC, and Logging Exposure Analysis

Database, Java, and platform teams should verify five defensive controls mapped to the observed DockHarvest artifacts before assuming any single flaw occurred on an external target:

Exposure layer	Observed DockHarvest evidence	Control to verify	Operational risk
SQL Server CLR & `OPENQUERY`	15 `clr_*` scripts (75,548 B), 1 withheld 997 B creds JSON, 2 unsized items; execution unverified.	Verify whether `clr enabled = 0`, `clr strict security = 1`, `TRUSTWORTHY = OFF`, and `OPENQUERY` logins enforce least privilege (`[REF-01]` , `[REF-02]`).	`UNSAFE` CLR procedures run OS commands and read IIS `web.config` secrets (`T1505.001` ; `[REF-08]`).
MySQL / JDBC `LOCAL INFILE`	`evil_mysql.py` , `evil_mysql.log` (26.66 MB), `evil_mysql8888.log` (1,779 connects across 335 withheld IPs; 11 target files).	Verify whether JDBC datasources block user URLs, set `allowLoadLocalInfile=false` , and block outbound `TCP 8888` (`[REF-06]` , `[REF-07]`).	Connecting to a rogue MySQL server lets `0xfb` packets read `/etc/passwd` , keys, and app configs.
JVM Logback XML & sockets	`logback-evil.xml` (919 B), `logback-socket.xml` (681 B), unsized `socket_recv.bin` ; execution unverified.	Verify whether Logback is `>= 1.2.9` , XML config paths are read-only, and `SocketReceiver` is absent (`[REF-03]` , `[REF-04]` , `[REF-05]`).	Writable, reloaded XML can enable `CVE-2021-42550` . Separately, `CVE-2017-5929` concerns untrusted streams reaching `SocketServer` or `ServerSocketReceiver` before Logback 1.2.0.
Outbound `/recv/` connectors	`up8899.py` (1,015 B), `up8899.log` (12,204 B), and 6 `/recv/` files (`t5.bin` 5.24 MB, `sdk` , `mcp` , `sse` , `acp` , `unnamed`).	Verify whether backend services and MCP/SSE agent workers enforce outbound proxy allowlists (`[REF-10]`).	Open egress allows SSRF callbacks or payload uploads to external ports like `TCP 8899` .
Developer `.claude` credentials	7 `.claude` files (446,307 B, 3 backups) and 2 external credential-file references (paths withheld).	Verify whether developer-agent credential files and `.claude.json.backup.*` enforce 0600 permissions outside web roots (`[REF-09]`).	Exposed CLI agent tokens leak API access and developer session histories (`T1552.001`).

6. Alternative Hypotheses

Hypothesis	Evidence for	Evidence against	Assessment
All 1,779 `evil_mysql8888.log` connects (335 IPs) are breached clients	`evil_mysql8888.log` logs 1,779 connects across 335 IPs and 28 `/tmp/stolen` or save mentions.	1,393 connects (78.3%) dropped at the auth stage like internet port scanners, and `/tmp/stolen` is not exposed to separate self-tests.	**Refuted for total counts.** Withhold all 335 connecting client IPs as unconfirmed traffic.
Staged `clr_*` scripts and Logback XMLs achieved target RCE	Scripts span `clr_get_creds` `v1` – `v3` and `clr_dump_final.py` beside a 997 B withheld creds JSON and `logback-*.xml` files.	Target SQL Server and Logback versions are unknown, and directory files alone do not confirm remote code execution.	**Unverified.** Cite `CVE-2021-42550` and `CVE-2017-5929` as context only and treat both suites as staged.
Benign developer host or authorized test server	Holds 674 Go build cache items, `.claude/` files, and standard shell dotfiles.	Runs `evil_mysql.py` targeting 11 files into `/tmp/stolen`, stages `clr_dump_wms.py` and `logback-evil.xml`, and exposes credentials on `TCP 8080`.	**Contradicted.** Rogue MySQL file theft, WMS scripts, and open credential exposure confirm an adversary staging workbench.
Multi-track adversary workbench across separate targets	Co-hosts SQL CLR/WMS scripts, `evil_mysql.py`, Logback XMLs, `/recv/` routes, Go `httpx` config, and portal scripts on one US VPS.	Separate tracks may target distinct systems rather than one single organization.	**Supported.** Assess as one adversary workbench hosting four distinct tracks without claiming a single chained victim.

7. Validated Detection Rules

All IPv4 addresses and all **425** file SHA-256 digests are withheld. Defenders should deploy **protocol, process, and file-structure detections** across SQL, JDBC, JVM, and CLI tiers. Static syntax checks passed in the Odin Suricata linter (**0** problems) and offline structural checks across KQL, **3** Sigma rules, and **4** `filesize`-bounded YARA rules. Because live `suricata -T`, `Kusto.Language`, `pySigma`, and `YARA` binaries were not run locally, teams should test each rule in staging first.

7.1 Suricata 7.x Network Signatures (`sid:9564001`–9564007)

Rules `sid:9564001`–**9564003** inspect inbound SQL CLR `CREATE ASSEMBLY`, `sp_configure`, and `jdbc:mysql://` `allowLoadLocalInfile` requests. Rule `sid:9564004` matches rogue MySQL `0xfb` `LOCAL INFILE` responses asking for `/etc/passwd`. When an internal client initiates outbound TCP to a rogue MySQL server, the server reply flows `$EXTERNAL_NET -> $HOME_NET` (`to_client`).

Rules `sid:9564005`–**9564006** detect inbound Logback XML configs with `<insertFromJNDI` or `SocketReceiver` (matching `ServerSocketReceiver` too). Rule `sid:9564007` alerts on outbound HTTP `POST` calls to `/recv/`, which teams should tune if approved webhooks use `/recv/`. None of these `alert` rules block traffic inline; HTTP rules need cleartext or proxy decryption, and `sid:9564004` needs cleartext MySQL traffic.

```
SURICATA

# Odin by Brandefense - Monograph 64 (Operation DockHarvest) Suricata 7.x Ruleset
# 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-R64 DockHarvest Inbound SQL CLR CREATE ASSEMBLY
Payload"; flow:established,to_server; http.request_body; content:"CREATE ASSEMBLY"; nocase; fast_patter
n; content:"PERMISSION_SET"; nocase; classtype:web-application-attack; sid:9564001; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound SQL sp_configure CLR To
ggle Attempt"; flow:established,to_server; http.request_body; content:"sp_configure"; nocase; fast_patte
rn; content:"clr "; nocase; classtype:web-application-attack; sid:9564002; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound JDBC MySQL allowLoadLoc
alInfile URI"; flow:established,to_server; http.request_body; content:"jdbc:mysql://"; nocase; fast_patt
ern; content:"allowLoadLocalInfile"; nocase; classtype:web-application-attack; sid:9564003; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Rogue MySQL Server LOCAL INFILE
/etc/passwd Request"; flow:established,to_client; content:"|0c 00 00 01 fb|/etc/passwd"; depth:16; fast_
pattern; classtype:credential-theft; sid:9564004; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound Logback insertFromJNDI
XML Config"; flow:established,to_server; http.request_body; content:"<insertFromJNDI"; nocase; fast_patt
ern; content:"env-entry-name"; nocase; classtype:web-application-attack; sid:9564005; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound Logback SocketReceiver
XML Config"; flow:established,to_server; http.request_body; content:"ch.qos.logback."; fast_pattern; con
tent:"SocketReceiver"; nocase; classtype:web-application-attack; sid:9564006; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R64 DockHarvest Outbound Callback POST to /rec
v/ Agent Route"; flow:established,to_server; http.method; content:"POST"; bsize:4; http.uri; content:"/r
ecv/"; startswith; fast_pattern; classtype:trojan-activity; sid:9564007; rev:1;)
```

7.2 Microsoft Sentinel KQL Hunt

Replace the two placeholder hostnames in `ApprovedAdminAndBuildHosts` with your approved SQL administration and CI build FQDNs before scheduling.

```
KQL

// Odin by Brandefense - Monograph 64 (Operation DockHarvest)
// Hunts SQL Server CLR child-process spawning, rogue MySQL LOCAL INFILE / Logback stagers, and unexpect
ed SQL/Java public egress.
let Lookback = 7d;
let ApprovedAdminAndBuildHosts = dynamic(["sql-admin-01.corp.example", "ci-go-01.corp.example"]);
let HostSqlClrAndStagingAnomalies =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where (
        (InitiatingProcessFileName =~ "sqlservr.exe" and FileName in~ ("cmd.exe", "powershell.exe", "pws
h.exe", "whoami.exe", "ipconfig.exe", "net.exe"))
        or (ProcessCommandLine has_any ("CREATE ASSEMBLY", "clr strict security", "allowLoadLocalInfil
```

```
e", "evil_mysql", "logback-evil", "logback-socket"))
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "HostSqlClrOrStagingTool", Detail = Proce
ssCommandLine;
let UnexpectedSqlAndJavaPublicEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where RemoteIPType == "Public"
    | where (
        InitiatingProcessFileName =~ "sqlservr.exe"
        or (InitiatingProcessFileName in~ ("java", "java.exe", "mysql", "mysql.exe") and RemotePort in
(8888, 8899, 1389, 4560))
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "Unexpecte
dSqlOrJavaPublicEgress", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteIP, ":", tostring(Remo
tePort), " (", RemoteUrl, ")");
union HostSqlClrAndStagingAnomalies, UnexpectedSqlAndJavaPublicEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = make_
set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

7.3 Portable Sigma and YARA Rules

- **Sigma (`sigma_dockharvest_sql_clr_mysql_and_claude_creds.yml`).** Rule 1 catches Windows `sqlservr.exe` spawning `cmd.exe`, `powershell.exe`, `pwsh.exe`, `whoami.exe`, `ipconfig.exe`, or `net.exe` outside `sql-admin-01[.]corp[.]example`. Rule 2 detects Linux `/python`, `/python3`, `/mysql`, or `/java` passing `evil_mysql`, `allowLoadLocalInfile`, `--local-infile=1`, `logback-evil`, or `logback-socket` outside `db-audit-01[.]corp[.]example`. Rule 3 detects `/curl`, `/wget`, `/python3`, or `/httpx` reading `.claude.json.backup` outside `dev-ws-01[.]corp[.]example`.
- **YARA (`odin_dockharvest_clr_mysql_and_logback_artifacts.yar`).** Four `filesize`-bounded rules detect SQL CLR and `OPENQUERY` scripts, Python `0xfb` rogue MySQL servers, Logback `insertFromJNDI` or `SocketReceiver` XML stagers, and Python `/recv` HTTP upload listeners.

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

Zero IPv4 addresses or file SHA-256 digests appear in the indicator feeds. The server holds withheld credential, private-key, and portal files alongside **335** unconfirmed connecting IPs. Defenders can hunt using the 10 behavioral markers below.

Marker category	Pattern / protocol artifact	Layer	Defensive hunt & control
SQL CLR registration	`CREATE ASSEMBLY ... WITH PERMISSION_SET = UNSAFE`	`TCP 1433` / HTTP	Audit `sys.assemblies`, keep `clr strict security = 1`, and alert via `sid:9564001`.
SQL config toggle	`sp_configure 'clr enabled'` / `'clr strict security'`	`TCP 1433` / HTTP	Alert on `RECONFIGURE` changes in SQL Server Audit and `sid:9564002`.
Linked-server query	`OPENQUERY()` pass-through (`clr_oqtest.py`)	`TCP 1433`	Audit `sys.servers` linked logins (`[REF-02]`) and scan scripts with YARA Rule 1.
JDBC MySQL file read	`jdbc:mysql://...allowLoadLocalInfile=true`	HTTP / JDBC	Force `allowLoadLocalInfile=false` (`[REF-07]`) and alert via `sid:9564003` and Sigma Rule 2.
Rogue MySQL `0xfb` packet	`0x0c000001fb` + `/etc/passwd` (`evil_mysql.py`)	`TCP 8888` response	Block untrusted MySQL egress and alert on cleartext `0xfb` packets (`sid:9564004`).
Logback JNDI XML	`<insertFromJNDI env-entry-name=...>`	HTTP / `TCP 1389`	Upgrade Logback `>= 1.2.9` (`CVE-2021-42550`; `[REF-03]`) and alert via `sid:9564005`.
Logback Socket Receiver	`ch.qos.logback...SocketReceiver` / `ServerSocketReceiver`	HTTP / `TCP 4560`	Upgrade Logback `>= 1.2.0` (`CVE-2017-5929`; `[REF-04]`), drop receivers, and alert via `sid:9564006`.
`/recv/` HTTP callback	`POST /recv/` (`mcp`, `sse`, `acp`, `sdk`, `t5.bin`)	`TCP 8899` egress	Restrict app and agent connector egress (`[REF-10]`) and alert via `sid:9564007`.
SQL child OS shell	`sqlservr.exe` spawning `cmd.exe` / `powershell.exe`	Windows EDR	Alert immediately on `sqlservr.exe` child shells or public egress (Sigma Rule 1; KQL).
AI-agent credential file	`.claude.json.backup`	Linux / macOS EDR	Enforce 0600 permissions (`[REF-09]`) and hunt backup-file CLI access via Sigma Rule 3.

9. Hardening Priorities for SQL Server, MySQL/JDBC, and JVM Logging Tiers

Database, Java, and platform defenders should execute five controls in priority order:

1. **Lock down SQL Server CLR and `OPENQUERY` linked servers (`0-48 Hours`).** Keep `clr enabled = 0` unless signed `.NET` routines are required, and verify `clr strict security = 1` and `TRUSTWORTHY = OFF` (`[REF-01]`). Audit `sys.assemblies` (`[REF-08]`) and `sys.servers` linked logins (`[REF-02]`), and isolate IIS `web.config` secrets from SQL accounts.
2. **Disable MySQL and JDBC `LOCAL INFILE` transfers (`0-72 Hours`).** Set `allowLoadLocalInfile=false` , `allowUrlInLocalInfile=false` , and `allowLoadLocalInfileInPath=""` on MySQL Connector/J datasources (`[REF-07]`). Pass `--local-infile=0` on MySQL/MariaDB clients (`[REF-06]`), block user-supplied JDBC hostnames, and block outbound `TCP 8888` .
3. **Upgrade Logback and make XML configs read-only (`Days 1-7`).** Upgrade QOS[.]ch Logback to a supported **1.6.x** release where compatible, past **1.2.x** before `1.2.9` (`CVE-2021-42550` , fixed in `1.2.9` and `1.3.0-alpha11` ; `[REF-03]`) and pre- `1.2.0` branches (`CVE-2017-5929` , fixed in `1.2.0` ; `[REF-04]`). Mount `logback.xml` read-only, remove `SocketReceiver` components (`[REF-05]`), and block outbound `TCP 1389` and **4560**.
4. **Restrict outbound HTTP and agent connector callbacks (`Days 7-14`).** Route application and MCP/SSE/SDK worker traffic through strict allowlist proxies so arbitrary `POST` callbacks to external ports (`TCP 8899` , `/recv/*` ; `[REF-10]`) fail closed.
5. **Protect developer `.claude` credentials and workstation directories (`Days 14-30`).** Enforce **0600** permissions on developer-agent credential files, `.claude.json` , `.claude/backups/*` , and `.ssh/` keys (`[REF-09]`). Prohibit unauthenticated HTTP file servers inside home folders and rotate keys on shared VMs.

10. Scope, Limits, and External References

Odin by Brandefense analyzed workbench artifacts recorded between `13 and 20 November 2026` on a United States VPS (`TCP 8080`) assessed as an adversary staging workbench. On `20 November 2026` , the open directory reported **874** active files and **284** directories. Across `13-20 November 2026` , deduplicated indexing captured **736** entries (**428** sized files totaling **299,306,613** bytes and **308** unsized entries; **425** unique SHA-256 digests).

In `evil_mysql8888.log` (**1,172,781** B on disk), the **196,357**-character (**5,461**-line) excerpt logged **1,779** connects across **335** IPs. Of those connects, **1,393** dropped at auth and **28** lines mention saves or `/tmp/stolen` .

Staged `clr_*` scripts (**15**), Logback XMLs (**3** items), portal probes (**8** items), and `/recv/` routes do not establish external execution. Both `CVE-2021-42550` and `CVE-2017-5929` serve as design context only, and all IPs, SHA-256s, portal names, and credential or key filenames are withheld.

1. `[REF-01]` Microsoft Learn, *CLR Integration Security in SQL Server* (verified on 25 November 2026).
2. `[REF-02]` Microsoft Learn, *OPENQUERY (Transact-SQL) Ad Hoc Distributed Query Controls* (verified on 25 November 2026).

3. [REF-03] NIST National Vulnerability Database, *CVE-2021-42550 Detail: QOS.ch Logback before 1.2.9 JNDI Configuration Code Execution* (verified on 25 November 2026).
4. [REF-04] NIST National Vulnerability Database, *CVE-2017-5929 Detail: QOS.ch Logback < 1.2.0 SocketServer Deserialization* (verified on 25 November 2026).
5. [REF-05] QOS.ch Logback Documentation, Logback release notes on network receivers (verified on 25 November 2026).
6. [REF-06] MariaDB Knowledge Base, *LOAD DATA INFILE and LOCAL INFILE Client Security* (verified on 25 November 2026).
7. [REF-07] MySQL Connector/J Official Source Repository, *allowLoadLocalInfile Driver Controls* (verified on 25 November 2026).
8. [REF-08] MITRE ATT&CK, *Server Software Component: SQL Stored Procedures (T1505.001)* (verified on 25 November 2026).
9. [REF-09] MITRE ATT&CK, *Unsecured Credentials: Credentials In Files (T1552.001)* (verified on 25 November 2026).
10. [REF-10] MITRE ATT&CK, *Application Layer Protocol: Web Protocols (T1071.001)* (verified on 25 November 2026).

BRANDEFENSE CAPABILITIES RELATED TO THIS RESEARCH

Cyber Threat Intelligence

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

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_dockharvest_clr_mysql_and_logback_artifacts.yar** 79 lines

```
/*
   Odin by Brandefense - Monograph 64 (Operation DockHarvest) YARA Suite
   4 filesize-bounded structural rules covering SQL Server CLR & OPENQUERY scripts,
   rogue MySQL LOCAL INFILE listeners, Logback JNDI/SocketReceiver XML stagers,
   and Python /recv HTTP callback receivers (zero victim or vendor identifiers).
*/

rule ODIN_DockHarvest_SQL_CLR_and_OPENQUERY_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "64"
        description = "Detects scripts staging SQL Server CLR assemblies, OPENQUERY pass-through probes,
and IIS connection-string queries"
        date = "2026-11-25"
    strings:
        $c1 = "CREATE ASSEMBLY" ascii nocase
        $c2 = "clr strict security" ascii nocase
        $c3 = "OPENQUERY(" ascii nocase
        $m1 = "sys.assemblies" ascii nocase
        $m2 = "sp_configure" ascii nocase
        $m3 = "connectionStrings" ascii nocase
        $m4 = "INFORMATION_SCHEMA.TABLES" ascii nocase
    condition:
        filesize < 64KB and 1 of ($c*) and 2 of ($m*)
}

rule ODIN_DockHarvest_Rogue_MySQL_Local_Infile_Server
{
    meta:
        author = "Odin by Brandefense"
        monograph = "64"
        description = "Detects Python rogue MySQL listeners serving 0xfb LOCAL INFILE file-read packets"
        date = "2026-11-25"
    strings:
        $s1 = "socket.socket" ascii
        $s2 = "0.0.0.0" ascii
        $f1 = "\\xfb" ascii
        $f2 = "0xfb" ascii
        $t1 = "/tmp/stolen" ascii
        $t2 = "/etc/passwd" ascii
        $t3 = "mysql_native_password" ascii
    condition:
        filesize < 64KB and $s1 and $s2 and ($f1 or $f2) and 1 of ($t*)
}

rule ODIN_DockHarvest_Logback_JNDI_or_Socket_XML_Stager
{
```

```
    meta:
        author = "Odin by Brandefense"
        monograph = "64"
        description = "Detects Logback XML configurations staging insertFromJNDI lookups or SocketReceiv
er connections"
        date = "2026-11-25"
    strings:
        $x1 = "<configuration" ascii nocase
        $l1 = "ch.qos.logback." ascii
        $l2 = "logback" ascii nocase
        $g1 = "<insertFromJNDI" ascii nocase
        $g2 = "SocketReceiver" ascii nocase
        $g3 = "ServerSocketReceiver" ascii nocase
    condition:
        filesize < 32KB and $x1 and ($l1 or $l2) and 1 of ($g*)
}

rule ODIN_DockHarvest_HTTP_Callback_Receiver_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "64"
        description = "Detects lightweight Python HTTP upload and callback receivers writing under /recv
routes"
        date = "2026-11-25"
    strings:
        $h1 = "BaseHTTPRequestHandler" ascii
        $h2 = "http.server" ascii
        $p1 = "do_POST" ascii
        $r1 = "/recv" ascii
    condition:
        filesize < 32KB and ($h1 or $h2) and $p1 and $r1
}
```

Suricata `odin_dockharvest_sql_clr_mysql_and_logback_hunt.rules` 11 lines

```
# Odin by Brandefense - Monograph 64 (Operation DockHarvest) Suricata 7.x Ruleset
# 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-R64 DockHarvest Inbound SQL CLR CREATE ASSEMBLY
Payload"; flow:established,to_server; http.request_body; content:"CREATE ASSEMBLY"; nocase; fast_patter
n; content:"PERMISSION_SET"; nocase; classtype:web-application-attack; sid:9564001; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound SQL sp_configure CLR To
ggle Attempt"; flow:established,to_server; http.request_body; content:"sp_configure"; nocase; fast_patte
rn; content:"clr "; nocase; classtype:web-application-attack; sid:9564002; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound JDBC MySQL allowLoadLoc
alInfile URI"; flow:established,to_server; http.request_body; content:"jdbc:mysql://"; nocase; fast_patt
ern; content:"allowLoadLocalInfile"; nocase; classtype:web-application-attack; sid:9564003; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Rogue MySQL Server LOCAL INFILE
/etc/passwd Request"; flow:established,to_client; content:"|0c 00 00 01 fb|/etc/passwd"; depth:16; fast_
pattern; classtype:credential-theft; sid:9564004; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound Logback insertFromJNDI
XML Config"; flow:established,to_server; http.request_body; content:"<insertFromJNDI"; nocase; fast_patt
ern; content:"env-entry-name"; nocase; classtype:web-application-attack; sid:9564005; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R64 DockHarvest Inbound Logback SocketReceiver
```

```
XML Config"; flow:established,to_server; http.request_body; content:"ch.qos.logback."; fast_pattern; con
tent:"SocketReceiver"; nocase; classtype:web-application-attack; sid:9564006; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R64 DockHarvest Outbound Callback POST to /rec
v/ Agent Route"; flow:established,to_server; http.method; content:"POST"; bsize:4; http.uri; content:"/r
ecv/"; startswith; fast_pattern; classtype:trojan-activity; sid:9564007; rev:1;)
```

Show full rule

Sentinel KQL `sentinel_dockharvest_sql_clr_mysql_and_logback_hunt.kql` 27 lines

```
// Odin by Brandefense - Monograph 64 (Operation DockHarvest)
// Hunts SQL Server CLR child-process spawning, rogue MySQL LOCAL INFILE / Logback stagers, and unexpect
ed SQL/Java public egress.
let Lookback = 7d;
let ApprovedAdminAndBuildHosts = dynamic(["sql-admin-01.corp.example", "ci-go-01.corp.example"]);
let HostSqlClrAndStagingAnomalies =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where (
        (InitiatingProcessFileName =~ "sqlservr.exe" and FileName in~ ("cmd.exe", "powershell.exe", "pws
h.exe", "whoami.exe", "ipconfig.exe", "net.exe"))
        or (ProcessCommandLine has_any ("CREATE ASSEMBLY", "clr strict security", "allowLoadLocalInfil
e", "evil_mysql", "logback-evil", "logback-socket"))
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "HostSqlClrOrStagingTool", Detail = Proce
ssCommandLine;
let UnexpectedSqlAndJavaPublicEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminAndBuildHosts))
    | where RemoteIPType == "Public"
    | where (
        InitiatingProcessFileName =~ "sqlservr.exe"
        or (InitiatingProcessFileName in~ ("java", "java.exe", "mysql", "mysql.exe") and RemotePort in
(8888, 8899, 1389, 4560))
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "Unexpecte
dSqlOrJavaPublicEgress", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteIP, ":", tostring(Remo
tePort), " (", RemoteUrl, ")");
union HostSqlClrAndStagingAnomalies, UnexpectedSqlAndJavaPublicEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = make_
set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

Sigma `sigma_dockharvest_sql_clr_mysql_and_claude_creds.yml` 95 lines

```
title: DockHarvest SQL Server Engine Spawning OS Shell or Recon Utility
id: 64a1c8e9-4b21-4d83-9a12-64b20e81f401
status: experimental
description: Detects Microsoft SQL Server engine processes spawning Windows command shells or reconnaiss
ance binaries indicative of UNSAFE CLR assemblies or 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'
      - '\whoami.exe'
      - '\ipconfig.exe'
      - '\net.exe'
  filter_sql_admin:
    ComputerName:
      - 'sql-admin-01.corp.example'
  condition: selection_parent and selection_child and not filter_sql_admin
falsepositives:
  - Legacy SQL Server Agent maintenance jobs invoking cmd.exe on approved database administration hosts
level: high
tags:
  - attack.persistence
  - attack.t1505.001
---
title: DockHarvest Rogue MySQL Local Infile or Logback Stager CLI
id: 64a1c8e9-4b21-4d83-9a12-64b20e81f402
status: experimental
description: Detects command lines invoking rogue MySQL LOCAL INFILE listeners, JDBC allowLoadLocalInfil
e flags, or Logback socket/JNDI XML stagers outside approved audit hosts.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_bin:
    Image|endswith:
      - '/python'
      - '/python3'
      - '/mysql'
      - '/java'
  selection_args:
    CommandLine|contains:
      - 'evil_mysql'
      - 'allowLoadLocalInfile'
      - '--local-infile=1'
      - 'logback-evil'
      - 'logback-socket'
  filter_audit_hosts:
    ComputerName:
      - 'db-audit-01.corp.example'
  condition: selection_bin and selection_args and not filter_audit_hosts
falsepositives:
  - Authorized database driver security regression testing on isolated audit hosts
```

```
level: high
tags:
  - attack.credential-access
  - attack.t1552.001
---
title: DockHarvest AI Coding Agent Credential File Harvesting
id: 64a1c8e9-4b21-4d83-9a12-64b20e81f403
status: experimental
description: Detects HTTP or scripting utilities fetching or inspecting .claude.json.backup files outsid
e designated developer workstations.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_bin:
    Image|endswith:
      - '/curl'
      - '/wget'
      - '/python3'
      - '/httpx'
  selection_cred_path:
    CommandLine|contains:
      - '.claude.json.backup'
  filter_dev_hosts:
    ComputerName:
      - 'dev-ws-01.corp.example'
  condition: selection_bin and selection_cred_path and not filter_dev_hosts
falsepositives:
  - Approved developer workstation backup scripts running on allowlisted workstations
level: high
tags:
  - attack.credential-access
  - attack.t1552.001
```

