FULL REPORT · POST-EXPLOITATION · MONOGRAPH 62 / 65 · 20 MIN READ

Linux NSS and Custom CA Toolkits: Three Unlinked Staging Workbenches

Defensive analysis of 95 staged entries across three unlinked Linux workbenches covering libnscd.so loaders, custom Root CA injectors, and application-layer proxy and SSH flood kits

BY
Odin by Brandefense

PUBLISHED
19 November 2026

READING TIME
20 min read

EDITION
Version 1.1 · TLP:CLEAR

SCOPE
Prepared November 2026

1. Executive Summary

IN SHORT

Odin by Brandefense analyzed 95 staged directory entries across three unrelated Linux virtual private servers recorded on 13–20 November 2026. Zero shared SHA-256 digests link the three servers. They hold a Linux Name Service Switch loader script, a custom Root CA injector suite, and application-layer proxy and SSH flood kits. Audit nsswitch.conf and CA trust stores, restrict PROXY headers to edge CIDRs, and rate-limit OpenSSH handshakes.

Linux name-resolution libraries, system certificate stores, and proxied network origins guard core trust across enterprise networks. Modifying Linux Name Service Switch modules lets unauthorized shared libraries load inside local daemons during routine lookups. If an attacker controls a trusted root's private key and intercepts traffic, clients may accept forged certificates without warnings. Direct origin access and unthrottled SSH queues likewise expose perimeter services to connection floods.

Between 13 and 20 November 2026, Odin by Brandefense analyzed exposed directories on three commercial virtual private servers in Hong Kong, Indonesia, and Israel. Together, the three hosts hold 95 deduplicated entries across 90 sized files (275,250,121 bytes, or 275.25 megabytes) and 5 unsized entries. Those files resolve to 78 unique SHA-256 digests.

Zero shared SHA-256 digests or infrastructure indicators link Workbenches A, B, and C. Each server is therefore evaluated strictly as an independent staging workspace. Odin by Brandefense withholds all server IPv4 addresses, unconfirmed command-and-control addresses inside scripts, file SHA-256 digests, and one harvested browser-cookie filename.

1 **Workbench A stages a Python loader referencing libnscd.so alongside an evasive binary.**
Discovered on 20 November 2026 on TCP port 8000 of a Hong Kong server, Workbench A holds two sized root files totaling 1,000,208 bytes. An 82,987-byte Python script references download and callback routes for libnscd.so and kreport beside a 917,221-byte Linux executable named bot-evasive.

On Linux systems, the Name Service Switch configuration file and Name Service Cache Daemon govern account and hostname lookups. Standard Linux distributions install nscd as an executable daemon rather than a shared library named libnscd.so. Referencing libnscd.so aligns with dynamic-linker or NSS library masquerading, though static listings do not confirm downstream target execution.

2 **Workbench B pairs a custom Root CA file with 34 companion files.**
Recorded on 13–20 November 2026 across mirrored ports 80, 443, and 8080 on an Indonesia server, Workbench B holds 35 sized files totaling 349,757 bytes. A 1,318-byte PEM certificate sits beside 34 companion files: Python scripts, Python 3.10 bytecode caches, account JSON data, JavaScript interceptors, and named backups. Backup names trace edits across browser-only, user-ID, device, and multi-user revisions.

3 **Workbench C stages edge-bypass scripts, SSH floods, multi-architecture bots, and one cookie file.**
Recorded on 13–20 November 2026 on TCP port 80 of an Israel cloud server, Workbench C holds 58 entries totaling 273,900,156 sized bytes. Staged tools include HAProxy PROXY header spoofing, HTTP proxy SSH floods, native SSH floods, Cloudflare Worker lists, and 12 cross-compiled bot binaries. The host also holds 210.87 megabytes of local logs and a harvested browser-cookie file (30,642 bytes; filename withheld).

4 **Unlinked staging servers call for host and protocol controls rather than static IP blocks.**
Because zero file digests or network indicators connect Workbenches A, B, and C, address blocklists offer little value. Defenders need file-integrity rules on Linux library and CA paths, origin firewall CIDR restrictions, and OpenSSH pre-authentication limits.

Three Decisions for Infrastructure and Security Leaders

1. **Monitor Linux Name Service Switch and shared-library paths.** Alert whenever any process outside signed OS package managers edits nsswitch.conf, writes libnscd.so or libnss modules, or alters ld.so.preload.
2. **Lock down OS and application Root CA trust stores.** Restrict update-ca-certificates, update-ca-trust, trust anchor commands, and NODE_EXTRA_CA_CERTS overrides to approved PKI hosts. Audit installed PEM digests regularly.
3. **Restrict origin ingress and tune OpenSSH handshake limits.** Accept origin TCP packets and PROXY headers strictly from verified CDN or DDoS-shield CIDRs. Set tight PerSourceMaxStartups and LoginGraceTime limits in sshd_config.

The Study in Numbers

Number	What it is
95	Deduplicated directory entries recorded across three unrelated Linux virtual private servers on `13-20 November 2026`
90	Sized files totaling **275,250,121** bytes (**275.25 MB**), alongside **5** unsized directory entries on Workbench C
78	Unique SHA-256 file digests across Workbenches A (**2**), B (**25**), and C (**51**), all withheld
0	Shared SHA-256 digests or infrastructure indicators linking Workbenches A, B, and C
2	Sized root files on Workbench A (**1,000,208** B): `kp_admin.py` (**82,987** B) and `bot-evasive` (**917,221** B ELF)
35	Sized files on Workbench B (**349,757** B): `kyng-ca.pem` (**1,318** B) and **34** companion Python, JS, `.pyc`, JSON, and backup files
58	Entries on Workbench C (**273,900,156** sized bytes): **57** edge-bypass, SSH flood, bot, C2, and log entries plus **1** withheld cookie file (**30,642** B)

2. Multi-Workbench Inventory, Independence Check, and File Breakdown

Odin by Brandefense recorded the three exposed directories between `13 and 20 November 2026`. Rather than grouping separate hosts into one unproven campaign, we test cross-host overlap first. **Zero shared SHA-256 digests or infrastructure indicators link Workbenches A, B, and C.**

Comparing **2** unique digests on Workbench A, **25** on Workbench B, and **51** on Workbench C yields **78** distinct digests with **0** matches. Hosting regions, port layouts, and programming habits also differ across all three servers. Figure 1 maps the three independent workbenches and their withholding rules.

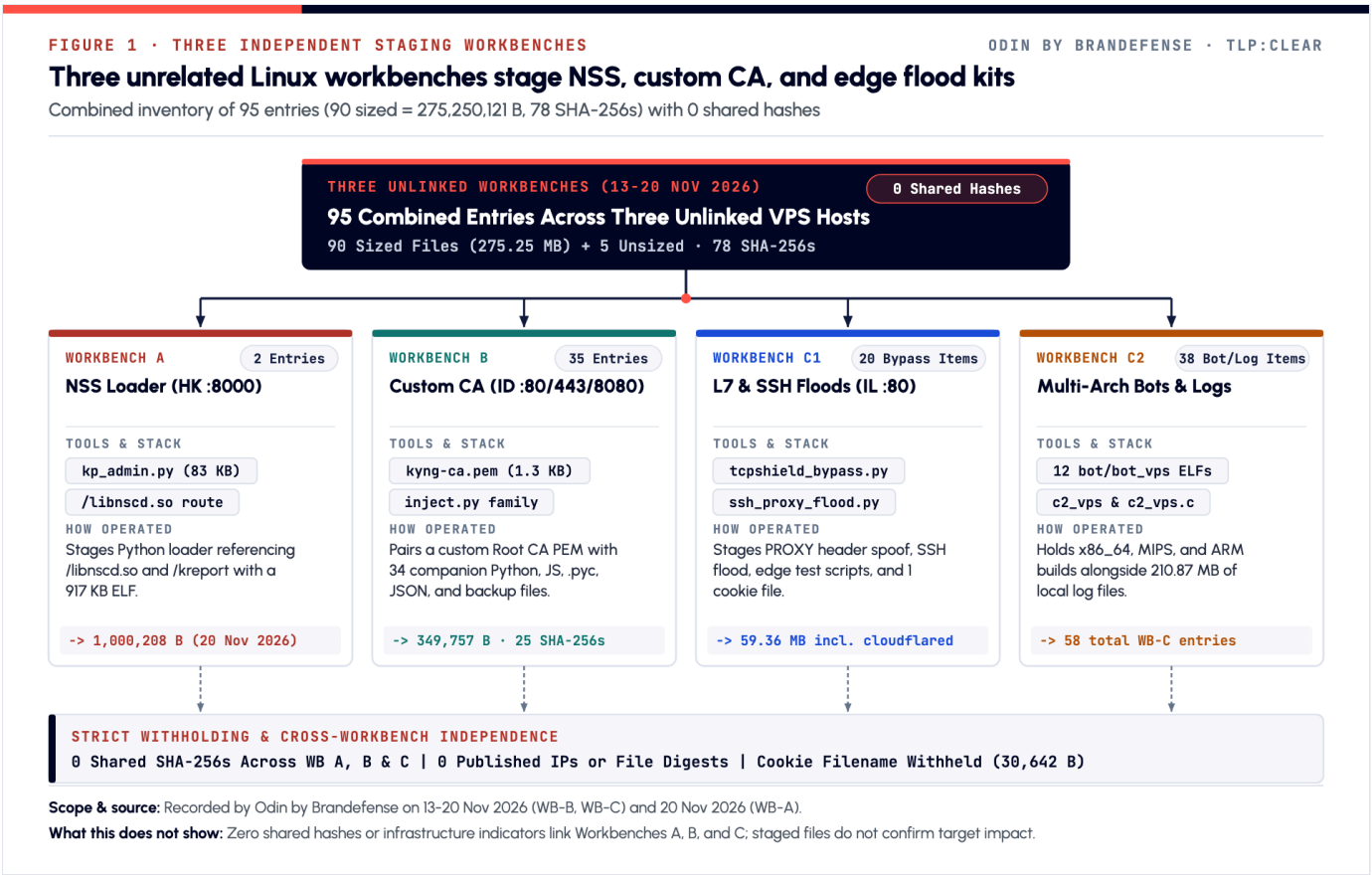

Figure 1: Three unrelated Linux workbenches stage NSS, custom CA, and edge flood kits

Three-Workbench Partition (`n = 95`)

Across all three virtual private servers, the deduplicated inventory totals **95 directory entries**. Ninety entries report byte sizes summing to **275,250,121 bytes (275.25 MB)**. Five entries on Workbench C omit byte sizes in listings.

Security teams can use external attack surface management to spot exposed staging ports on cloud instances. Pairing those checks with cyber threat intelligence helps engineers translate staged tools into host and network detections. Table 1 reconciles all **95** entries across Workbenches A, B, and C.

Workbench & hosting profile	Directory buckets	Total entries	Share (%)	Sized files	Unsized entries	Sized bytes	Unique SHA-256 (withheld)	Observed staging focus
Workbench A (`Hong Kong VPS · TCP 8000`)	1 (`root`)	2	2.11%	2	0	1,000,208	2	Discovered `20 Nov 2026`; stages `kp_admin.py` (82,987 B) referencing `/libnscd.so` and `/kreport` plus `bot-evasive` (917,221 B ELF).
Workbench B (`Indonesia VPS · TCP 80/443/8080`)	2 (`root`, `__pycache__`)	35	36.84%	35	0	349,757	25	Recorded `13–20 Nov 2026`; mirrors three ports staging `kyng-ca.pem` (1,318 B) with 34 companion Python, JS, `.pyc`, backup, and account JSON files.

Workbench C (`Israel Cloud VPS · TCP 80`)	6 (`root` + 5 dirs)	58	61.05%	53	5	273,900,156	51	Recorded `13-20 Nov 2026`; stages **57** L7/SSH flood, bot, C2, and log entries plus **1** harvested cookie file (**30,642** B; name withheld).
Combined study inventory	**9 buckets**	**95**	**100.00%**	**90**	**5**	**275,250,121**	**78** (0 shared)	**Three unlinked VPS workbenches (275.25 MB sized volume).**

Directory depth differs cleanly across the three servers.

- **Workbench A (1 bucket).** Both files sit in the root directory on `TCP 8000` (**1,000,208** bytes).
- **Workbench B (2 buckets).** Thirty-two sized files (**345,437** bytes) sit in the root directory mirrored across `TCP 80`, **443**, and **8080**. Three CPython **3.10** `.pyc` files (**4,320** bytes) sit inside `__pycache__/`. Ten identical-content backup copies explain why **35** files yield **25** unique digests.
- **Workbench C (6 buckets).** Fifty-two entries (**48** sized = **264,119,732** bytes and **4** unsized) sit in root on `TCP 80`. Five subdirectories (`serve/`, `debug/`, `scratch/`, `.ssh/`, and `.cache/`) hold the remaining six entries (**5** sized = **9,780,424** B and **1** unsized).

Combined File-Class and Functional Cluster Breakdown

Table 2 reconciles the same **95** entries (**90** sized files, **5** unsized entries, **275,250,121** bytes) by file class. Text, log, backup, and C source files lead by count (**41** entries, **43.16%**) and volume (**211,080,131** bytes). Compiled Linux ELF binaries (**23** entries, **24.21%**) and scripts (**22** entries, **23.16%**) make up another **47.37%**.

Artifact class	Total entries	Share (%)	Sized files	Unsized entries	Sized bytes	Cross-workbench composition
Text, logs & C source (`text`)	41	43.16%	38	3	211,080,131	18 WB-B backups, `users.json`, and save files (146,229 B) plus 23 WB-C logs, worker lists, `.c` sources, dotfiles, and 1 withheld 30,642 B cookie file (210,933,902 B).
Compiled binaries (`exe`)	23	24.21%	23	0	63,810,159	1 WB-A `bot-evasive` ELF (917,221 B) plus 22 WB-C `x86_64`, `MIPS`, `ARM`, `cloudflared`, and `c2_vps` binaries (62,892,938 B).
Scripts (`script`)	22	23.16%	22	0	320,772	1 WB-A `.py` (82,987 B), 10 WB-B (5 `.py`, 5 `.js`; 164,469 B), and 11 WB-C (5 `.py`, 3 `.js`, 3 `.sh`; 73,316 B).
Bytecode & state (`other`)	5	5.26%	3	2	4,320	3 WB-B CPython 3.10 `.pyc` files (4,320 B) and 2 WB-C unsized host markers (0 B).
Backup snapshots (`data`)	3	3.16%	3	0	33,421	3 WB-B `.bak` files (`kyng_c.js.bak` 30,331 B, `auth.py.bak` 1,631 B, `inject.py.bak` 1,459 B).
Certificates (`secret`)	1	1.05%	1	0	1,318	1 WB-B staged custom Root CA PEM certificate (`kyng-ca.pem`, 1,318 B).
Total	**95**	**100.00%**	**90**	**5**	**275,250,121**	**Exact parity with Table 1 (95 entries = 275.25 MB).**

Figure 5 divides all **95** entries into six mutually exclusive functional clusters (`24 + 22 + 20 + 16 + 11 + 2 = 95`). Four active tooling clusters on Workbenches B and C account for **77** entries (**81.05%**). Local logs and dotfiles on Workbench C (**16** entries, **16.84%**) and Workbench A (**2** entries, **2.11%**) form the rest.

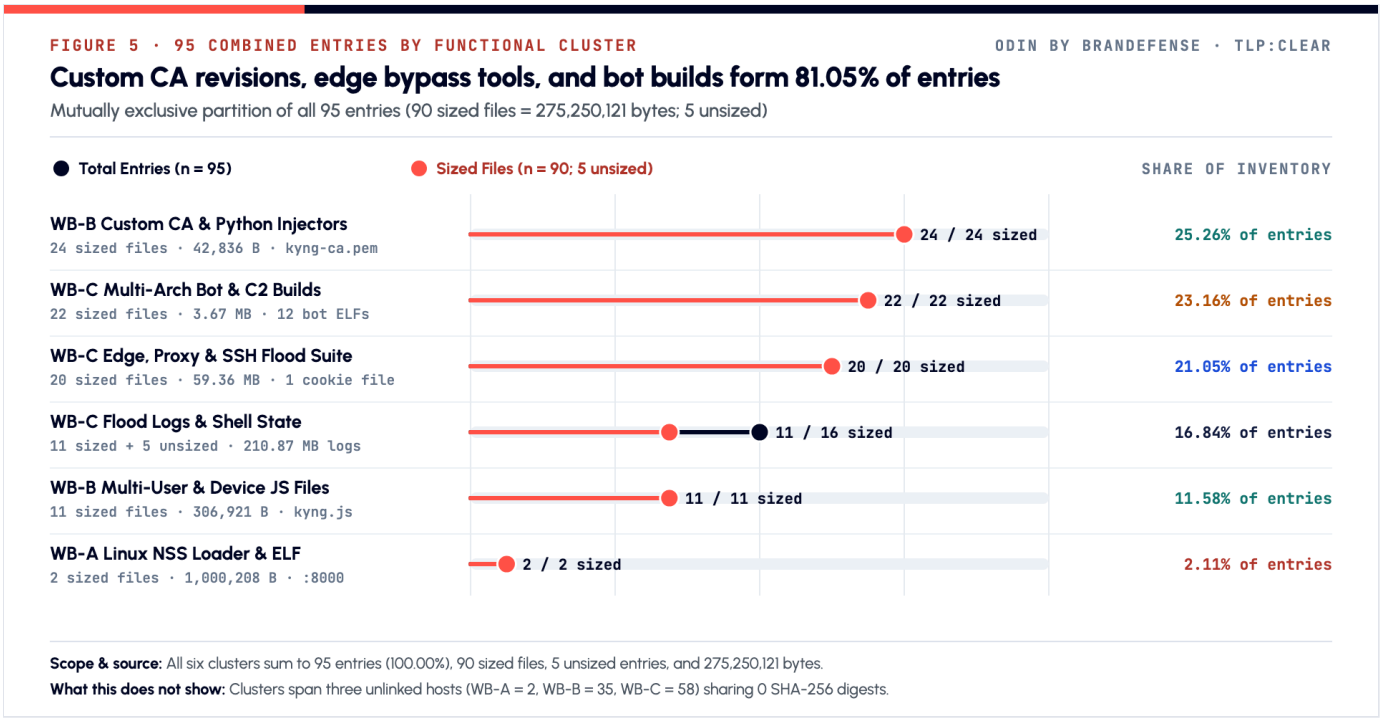

Figure 5: Custom CA revisions, edge bypass tools, and bot builds form 81.05% of entries

3. Linux Name Service Switch Loaders and Custom Root CA Suites

Workbenches A and B focus on host-level trust mechanisms: Linux name-resolution libraries (`Workbench A`) and Root CA trust anchors (`Workbench B`). Neither server shares files with the other. Figure 3 presents a dashed analytical model showing how defenders can spot these staged mechanisms alongside Workbench C's network floods.

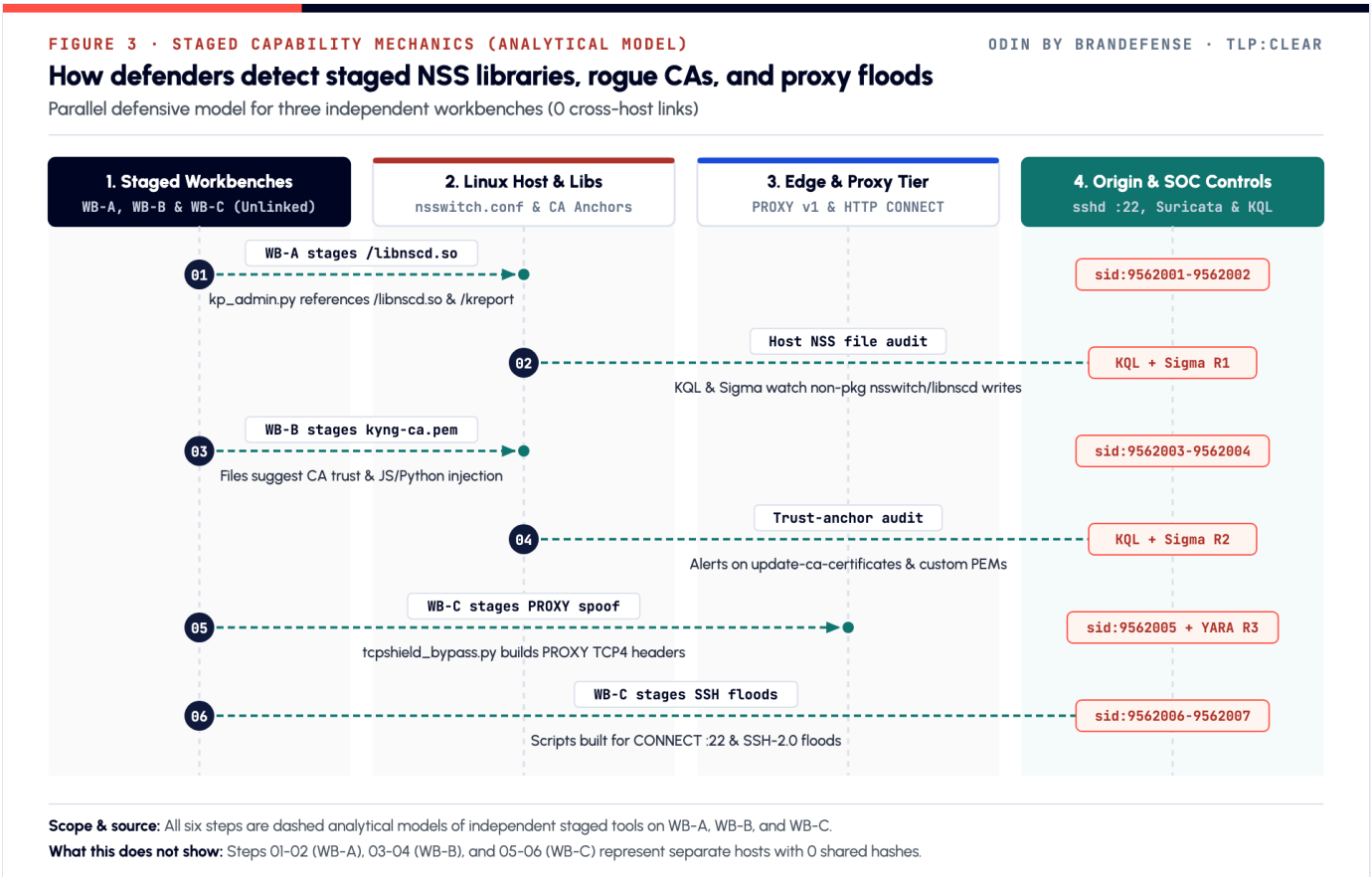

Figure 3: How defenders detect staged NSS libraries, rogue CAs, and proxy floods

Workbench A: Linux `nsswitch.conf`, `nscd`, and Staged `/libnscd.so` Loading (2 Entries)

Discovered on `20 November 2026` on a Hong Kong VPS exposing `TCP 8000` (address withheld), Workbench A holds **2 sized files totaling 1,000,208 bytes**.

- **`/kp_admin.py` (82,987 bytes, `script`).** This Python administration script references two HTTP endpoints: `/libnscd.so` and `/kreport`.
- **`/bot-evasive` (917,221 bytes, `exe`).** This compiled Linux ELF executable sits in the same root directory.

Defenders can assess `/libnscd.so` by reviewing how GNU `glibc` resolves names and accounts on Linux (`[REF-01]`, `[REF-02]`).

1. **Name Service Switch (`nsswitch.conf`) loading (`[REF-01]`).** When a dynamically linked Linux process calls `getpwnam()` or `getaddrinfo()`, `glibc` parses `/etc/nsswitch.conf`. For each listed module name, `glibc` loads `libnss_<module>.so.2` into the calling process with `dlopen()`. Because `sshd`, `sudo`, `cron`, and web servers run these lookups, an added NSS library loads inside privileged processes (`T1556`, `[REF-04]`).
2. **Daemon naming vs shared-object masquerading (`[REF-02]`, `[REF-03]`).** Linux distributions install the Name Service Cache Daemon as `/usr/sbin/nscd`, not as `/libnscd.so` (`[REF-02]`). Naming a rogue `.so` file `libnscd.so` helps it blend into `/lib/` or `/usr/lib/` during manual checks. Attackers can stage such

libraries for `/etc/ld.so.preload`, `LD_PRELOAD`, linker path hijacking (`T1574.006`, `[REF-03]`), or `/etc/nsswitch.conf` edits.

3. **Staged reference boundary (`/kreport`).** Referencing `/libnscd.so` and `/kreport` separates binary fetches from status callbacks. Because `libnscd.so` itself did not sit in the root listing on `20 November 2026`, both paths represent staged script references only.

Workbench B: Staged `kyng-ca.pem` Root CA and Multi-User Injectors (35 Entries)

From `13-20 November 2026`, an Indonesia VPS mirrored identical directories on `TCP 80`, **443**, and **8080** (address withheld). Workbench B holds **35 sized files totaling 349,757 bytes** (**25** unique SHA-256s). Table 3 summarizes all **35** files across its five component groups.

Workbench B component group	Entries	Sized bytes	Representative filenames & sizes	Staged architectural role
Custom Root CA (`secret`)	1	1,318	`kyng-ca.pem` (1,318 B)	PEM X.509 certificate staged to anchor local or proxied TLS interception (`T1553.004`).
Python Injectors (`inject*`)	15	24,522	`inject.py` (1,632 B), `inject_b.py` (1,461 B), `inject_c.py` (1,461 B), `inject.py.bak-debug` (3,050 B)	Active Python scripts and 11 backups spanning browser-only, login, user-ID, device, and multi-user edits.
CPython 3.10 Bytecode (`__pycache__`)	3	4,320	`inject.cpython-310.pyc` (1,610 B), `inject_b.cpython-310.pyc` (1,355 B), `inject_c.cpython-310.pyc` (1,355 B)	Bytecode caches created when Python 3.10 imported `inject`, `inject_b`, and `inject_c` locally.
Auth & Accounts (`auth*` + `users.json`)	5	12,797	`auth.py` (3,695 B), `auth.py.backup-sekarang` (3,695 B), `auth.py.bak` (1,631 B), `users.json` (81 B)	Session auth scripts (1,631 B to 3,695 B) paired with an 81-byte JSON account file.
JS Interceptors (`kyng*.js*`)	11	306,921	`kyng.js` (31,733 B), `kyng-before-total-fix.js` (31,446 B), `kyng_b.js` (30,331 B), `kyng_c.js` (30,331 B), `kyng.js.save` (1 B)	Client or proxy JavaScript revisions (30.3 KB to 31.7 KB) plus a 1-byte save stub.
Workbench B Total	**35**	**349,757**	32 root files (345,437 B) + 3 `__pycache__` files (4,320 B)	**25 unique SHA-256 digests (0 unsized entries).**

Three practical lessons stand out on Workbench B.

- **Root CA trust subversion (`T1553.004`, `[REF-05]`, `[REF-06]`).** Under RFC 5280 (`[REF-06]`), a client may trust a server certificate signed by an installed root certificate authority when its validation checks pass. Registering `kyng-ca.pem` with `update-ca-certificates`, `update-ca-trust`, `trust anchor`, or `NODE_EXTRA_CA_CERTS` lets an intercepting proxy with the matching private key inspect or modify HTTPS traffic (`[REF-05]`).
- **Clear feature evolution in backups.** Early Python snapshots (`inject.py.bak-browser-only` and `inject.py.backup-before-multiuser`, **1,459** B each) grow into `inject.py` (**1,632** B). Likewise, `kyng.js.backup-before-device` (**30,392** B) grows into `kyng.js` (**31,733** B). One backup (`inject_c.py.bak-20261111-072238`) carries an `11 November 2026` filename timestamp created two days before collection began.
- **Language clues vs attribution.** Two backups (`auth.py.backup-sekarang` and `inject.py.backup-sebelum-userid`) use Indonesian words ("now" and "before") on an Indonesia-hosted VPS. Hosting geography and filename words describe the workspace only and do not establish operator nationality.

4. Application-Layer Proxy, SSH, and Shield Bypass Kits (`Workbench C`)

Recorded on `13-20 November 2026` on an Israel cloud VPS exposing `TCP 80` (address withheld), Workbench C holds **58 deduplicated entries**. Fifty-three sized files total **273,900,156 bytes**, five entries are unsized, and **51** unique SHA-256 digests appear (**0** shared with A or B). Figure 4 compares port layouts and directory slices across all three hosts.

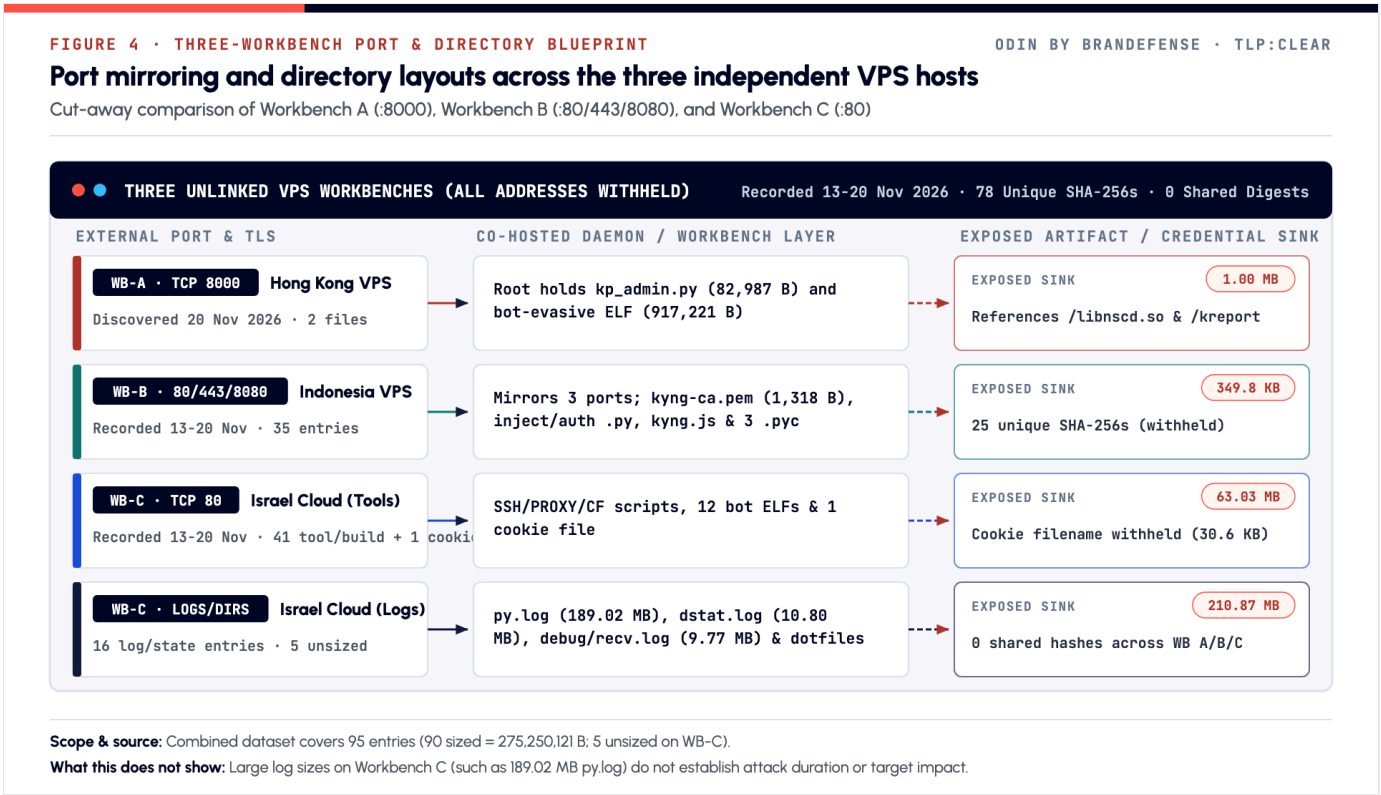

Figure 4: Port mirroring and directory layouts across the three independent VPS hosts

Edge-Shield Bypass, SSH Floods, Multi-Arch Bots, and Logs (58 Entries)

Table 4 reconciles all **58** entries (**53** sized = **273,900,156** bytes and **5** unsized) across Workbench C's three functional clusters (`20 + 22 + 16 = 58`).

Workbench C component group	Entries	Sized bytes	Representative files & sizes	Staged mechanism
Origin & Shield Bypass	5	15,418,214	`tcpshield_bypass.py` (3,541 B), `cf_bypass.x86_64` (10,829,986 B), `mc_bypass.x86_64` (2,314,402 B), `ovh_specter.x86_64` (2,269,449 B), `ovhkill.sh` (836 B)	Direct-to-origin HAProxy `PROXY TCP4` header spoofing and edge flood binaries.
Tunnel, Worker & L7 Scripts	10	39,862,087	`cloudflared` (39,798,028 B), `tornado.js` (35,814 B), 4 CF Worker `.txt` lists (310 B), `c2_roblox.py` , `roblox_join.js` , `snipe` , `snipe.c`	Serverless Worker lists, tunnel utility, and HTTP flood scripts.
SSH Floods, Bandwidth & Cookie	5	4,077,784	`ssh_native.py` (3,872 B), `ssh_proxy_flood.py` (6,278 B), `ssh_killer.x86_64` (4,026,633 B), `dstat.py` (10,359 B), 1 withheld cookie file (30,642 B)	Native and HTTP `CONNECT :22` SSH floods, bandwidth monitor, and withheld cookie file.
Multi-Arch Bots & C2	22	3,668,226	`c2_vps` & `c2_vps_new` (936,280 B each), `c2_vps.c` (25,276 B), 12 `bot` / `bot_vps` ELFs (`x86_64` , `MIPS` , `ARM`), 7 build/test files	Cross-compiled bot client binaries, C2 server builds, and C parser helpers.
Local Logs & Dotfiles	16	210,873,845	`py.log` (189,024,142 B), `dstat.log` (10,795,269 B), `debug/recv.log` (9,774,454 B), `http.log` (1,271,776 B), 5 unsized entries	Local console logs (11 sized, 5 unsized) and shell dotfiles (5,395 B).
Workbench C Total	**58**	**273,900,156**	**53 sized files + 5 unsized entries (51 unique SHA-256s)**	**0 shared digests with Workbenches A or B.**

Four operational takeaways guide edge and SSH defenders (`T1499` , `[REF-07]`).

- **Direct-to-origin `PROXY` header spoofing.** Game servers behind TCPShield and proxied web backends parse HAProxy `PROXY TCP4` headers (`[REF-10]`) to read client IPs. When origin firewalls fail to drop non-shield packets, `tcpshield_bypass.py` (**3,541** B) can hit the origin directly and spoof client addresses (`[REF-10]`).

- **Native and HTTP-proxied SSH handshake floods.** Every TCP connection to `sshd` triggers an `SSH-2.0-` banner and key exchange before login (`[REF-09]`). OpenSSH limits unauthenticated queues through `MaxStartups` (`[REF-08]`). Staged tools `ssh_native.py` , `ssh_killer.x86_64` , and `ssh_proxy_flood.py` (via HTTP `CONNECT` to :**22**) aim to fill those queues.
- **Withheld cookie file (30,642 bytes).** Workbench C holds **a harvested browser-cookie file (30,642 bytes; filename withheld)**. Operators often load session or clearance cookies into HTTP flooders. Odin by Brandefense withholds its filename, digest, and contents to protect affected users.
- **Local log sizes vs external impact.** Six sized logs (`py.log` at **189.02 MB**, `dstat.log` at **10.80 MB**, and `debug/recv.log` at **9.77 MB**) hold nearly all bytes in the log group. Verbose socket errors grow rapidly on disk and never establish run duration or remote service impact.

Figure 2 maps all five staged capability groups across Workbenches A, B, and C to the detections in Chapter 8.

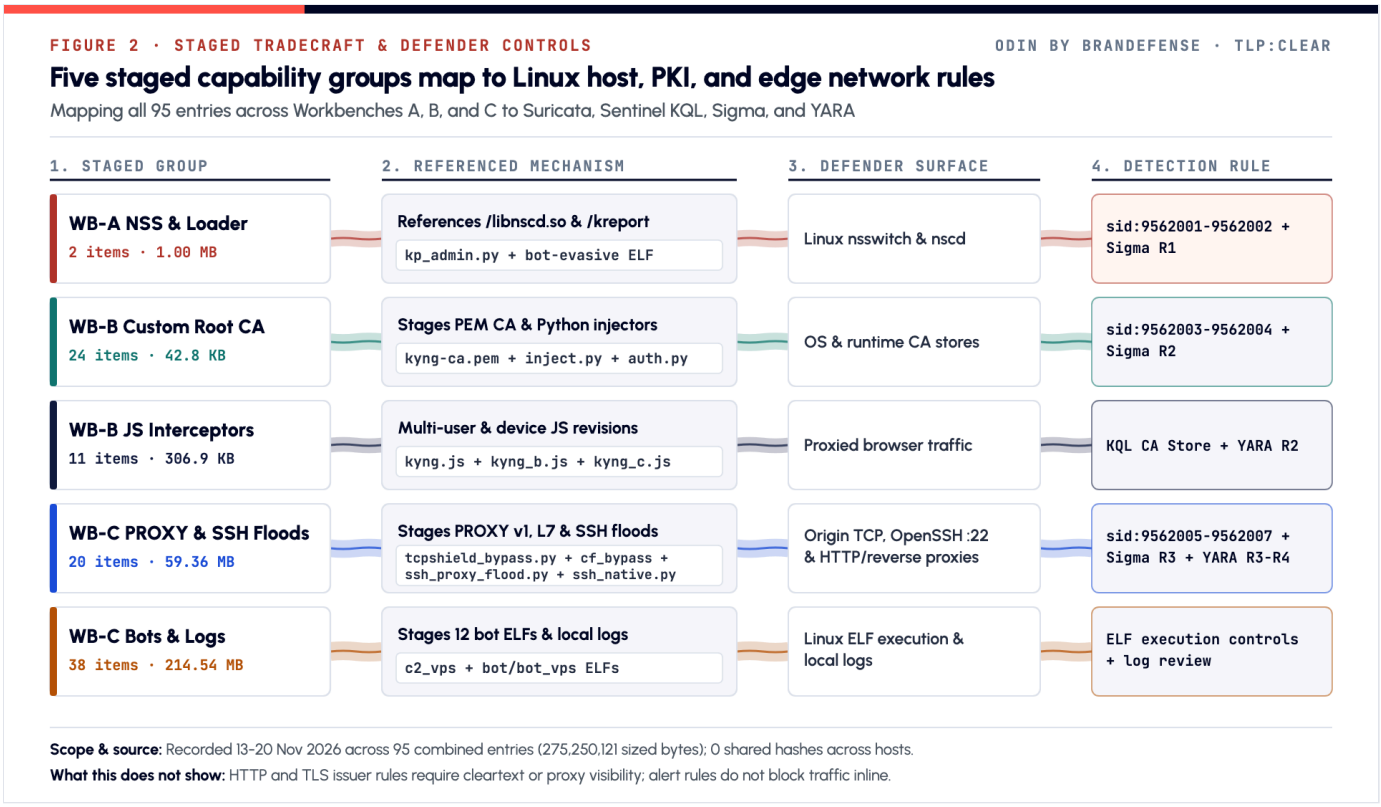

Figure 2: Five staged capability groups map to Linux host, PKI, and edge network rules

5. Linux Identity, TLS Trust, and Edge Exposure Analysis

Defenders should verify four concrete controls across Linux servers, PKI trust stores, and proxied network perimeters before treating any configuration gap as an established root cause.

Exposure surface	Observed workbench evidence	Control to verify	Operational risk
Linux NSS & library paths	WB-A `kp_admin.py` (82,987 B) referencing `/libnscd.so` and `/kreport` plus `bot-evasive` (917,221 B).	Verify whether `/etc/nsswitch.conf`, `/etc/ld.so.preload`, `/lib*`, and `/usr/lib*` alert on non-package writes (`[REF-01]`, `[REF-03]`).	Rogue NSS or preloaded `.so` files load into `sshd` and `sudo` during lookups (`[REF-04]`).
OS & runtime Root CA stores	WB-B `kyng-ca.pem` (1,318 B) plus 34 companion Python, JS, `.pyc`, backup, and account JSON files (348,439 B).	Verify whether `update-ca-certificates`, `update-ca-trust`, `trust anchor`, and `NODE_EXTRA_CA_CERTS` are restricted (`[REF-05]`, `[REF-06]`).	With a matching private key and traffic path, a rogue Root CA lets an intercepting proxy forge certificates without client warnings.
Origin firewall & `PROXY` ACLs	WB-C `tcpshield_bypass.py` (3,541 B), `cf_bypass.x86_64`, `mc_bypass.x86_64`, `ovh_specter.x86_64`, 4 CF lists.	Verify whether origin firewalls drop non-shield packets and restrict `PROXY TCP4` parsing to upstream CIDRs (`[REF-10]`).	Direct origin connections bypass DDoS scrubbing edges and spoof source IPs.
OpenSSH queues & HTTP proxies	WB-C `ssh_native.py` (3,872 B), `ssh_proxy_flood.py` (6,278 B), `ssh_killer.x86_64`, 12 bot ELFs.	Verify whether `sshd` sets `PerSourceMaxStartups` and `LoginGraceTime` (`[REF-08]`) and proxies block `CONNECT :22`.	Unauthenticated `SSH-2.0-` floods fill pre-auth slots and block admin SSH access (`[REF-07]`, `[REF-09]`).

6. Alternative Hypotheses

Hypothesis	Evidence for	Evidence against	Assessment
Single operator across Workbenches A, B, and C	All three Linux VPS hosts were recorded during `13-20 November 2026`.	Comparing `2 + 25 + 51 = 78` unique SHA-256s shows **0 shared hashes**; regions (`HK`, `ID`, `IL`), ports, and code differ.	**Refuted.** Treat A, B, and C strictly as three independent hosts.

Hypothesis	Evidence for	Evidence against	Assessment
Authorized security or load testing	`dstat.py`, `test_parse.c`, and dated backups resemble developer test habits.	WB-C holds 12 bot ELFs, `ssh_killer.x86_64`, and a withheld 30,642 B cookie file on an open port; A and B expose offensive kits openly.	**Authorized use is unsupported by the recorded files.** The staged loader, CA injector suite, and flood kit support an attacker-workbench assessment. Withhold all VPS IPs.
Large WB-C logs (189.02 MB `py.log`) show prolonged target outages	`py.log` (189.02 MB), `dstat.log` (10.80 MB), and `recv.log` (9.77 MB) hold large console outputs.	Per-socket Python error loops can write hundreds of megabytes in minutes without affecting any remote target.	**Unsupported.** File size alone never establishes run duration or target impact.
Indonesian backup suffixes establish operator nationality	`auth.py.backup-sekarang` and `inject.py.backup-sebelum-userid` sit on an Indonesia VPS.	Reused kits and outsourced scripts often keep earlier filename words regardless of operator location.	**Insufficient for attribution.** Treat backup names as revision labels only.

7. What Defenders Should Do

Infrastructure and security teams can close these host, PKI, and perimeter gaps in five steps.

1. **Protect `/etc/nsswitch.conf` and system library paths (`0-48 Hours`).** Add `auditd` write watches on `/etc/nsswitch.conf` and `/etc/ld.so.preload` (`[REF-01]`, `[REF-03]`). Run `dpkg -V` or `rpm -Va` to verify shared libraries against package digests. Disable `nscd` (`[REF-02]`) where local caching is unnecessary.
2. **Baseline OS and runtime Root CA stores (`0-72 Hours`).** Hash all PEM and CRT anchors under `/etc/ssl/certs/`, `/usr/local/share/ca-certificates/`, and `/etc/pki/ca-trust/source/anchors/` against an approved baseline (`[REF-05]`, `[REF-06]`). Audit service files and shell profiles for unauthorized `NODE_EXTRA_CA_CERTS` or `SSL_CERT_FILE` variables.
3. **Close direct-to-origin leaks and scope `PROXY` headers (`Days 1-7`).** Configure security groups or `nftables` so origin TCP ports accept packets strictly from verified CDN or TCPShield CIDRs (`[REF-10]`). Scope Nginx `set_real_ip_from` and HAProxy `accept-proxy` ACLs to those same CIDRs. Rotate origin public IPs if historical DNS records exposed them.

---

4. **Throttle OpenSSH pre-auth queues and forward proxies (`Days 7-14`).** Set `LoginGraceTime 20`, `MaxStartups 30:30:100`, and `PerSourceMaxStartups 5` in `/etc/ssh/sshd_config` (`[REF-08]`, `[REF-09]`). Deny HTTP `CONNECT` requests to :**22** on forward proxies (`sid:9562007`).
5. **Enforce outbound egress rules on Linux workloads (`Days 14-30`).** Route server egress through proxies or strict VPC allowlists so workloads cannot fetch staged `.so` binaries (`sid:9562001`) or join outbound flood traffic.

8. Detection Rules

All VPS IPv4 addresses and all **78** workbench SHA-256 digests are withheld. Defenders should deploy **protocol, file-integrity, process, and file-structure rules** that work across any infrastructure.

Each rule was checked statically before release. Suricata **7.x** sticky-buffer, flow, and semicolon syntax passed the static Suricata linter with **0** problems (`sid:9562001` -**9562007**). Sentinel KQL table schemas, `in~` allowlists, Sigma YAML fields, and `filesize` -bounded YARA conditions were verified structurally. Live Suricata, Sentinel, Sigma, and YARA engines were not installed in this build environment, so validate in staging before rollout.

8.1 Suricata 7.x Network Signatures (`sid:9562001` -9562007)

These seven `alert` signatures detect staged `/libnscd.so` and `/kreport` HTTP routes (`sid:9562001` -**9562002**) and `/kyng-ca.pem` fetches (`sid:9562003`). Rule `sid:9562004` matches TLS server certificates whose `tls.cert_issuer` contains `kyng`. Rules `sid:9562005` -**9562007** match inbound `PROXY TCP4` spoofing, high-rate `SSH-2.0-` floods (`>=40` in **10s**), and HTTP `CONNECT` requests to :**22**.

HTTP rules (`sid:9562001` -**9562003**, **9562007**) need cleartext HTTP or a TLS-inspecting proxy. Rule `sid:9562004` sees `tls.cert_issuer` in cleartext on TLS 1.2, whereas TLS 1.3 encrypts server certificates and requires TLS inspection. For `sid:9562005`, exclude verified upstream CDN or TCPShield proxy CIDRs from `$EXTERNAL_NET`. All seven rules use `alert` mode and do not drop packets inline.

```
SURICATA

# Odin by Brandefense - Monograph 62 Suricata 7.x Ruleset
# Covers staged libnscd.so / kreport routes, custom kyng Root CA fetches/issuers, PROXY v1 spoofing,
and SSH floods.

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-A Outbound HTTP Request for Staged l
ibnscd.so Shared Object"; flow:established,to_server; http.uri; content:"/libnscd.so"; endswith; fas
t_pattern; classtype:trojan-activity; sid:9562001; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-A Outbound HTTP POST Callback to Sta
ged kreport Endpoint"; flow:established,to_server; http.method; content:"POST"; bsize:4; http.uri; c
ontent:"/kreport"; startswith; fast_pattern; classtype:command-and-control; sid:9562002; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-B Outbound HTTP Fetch of Custom kyng
-ca.pem Root Certificate"; flow:established,to_server; http.uri; content:"/kyng-ca.pem"; endswith; f
ast_pattern; classtype:policy-violation; sid:9562003; rev:1;)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R62 WB-B Inbound TLS Server Certificate Issu
```

```
ed by Custom kyng CA"; flow:established,to_client; tls.cert_issuer; content:"kyng"; nocase; fast_pat
tern; classtype:bad-unknown; sid:9562004; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R62 WB-C Inbound Direct-to-Origin HAProxy PR
OXY v1 Header Spoof Attempt"; flow:established,to_server; content:"PROXY TCP4 "; startswith; fast_pa
ttern; classtype:attempted-recon; sid:9562005; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"ODIN-R62 WB-C Inbound High-Rate SSH-2.0 Handshake
Flood Against TCP 22"; flow:established,to_server; content:"SSH-2.0-"; startswith; fast_pattern; thr
eshold:type both, track by_src, count 40, seconds 10; classtype:attempted-dos; sid:9562006; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-C Outbound HTTP CONNECT Proxy Tunnel
Targeting SSH Port 22"; flow:established,to_server; http.method; content:"CONNECT"; bsize:7; http.ur
i.raw; content:":22"; endswith; fast_pattern; classtype:policy-violation; sid:9562007; rev:1;)
```

8.2 Microsoft Sentinel KQL Hunt

This query joins `DeviceFileEvents` (non-package NSS and CA store writes), `DeviceProcessEvents` (trust-anchor and flood commands), and `DeviceNetworkEvents` (staged `/libnscd.so`, `/kreport`, and `/kyng-ca.pem` fetches). Replace the placeholder FQDNs in `ApprovedAdminHosts` with your configuration and PKI management hosts before scheduling.

```KQL
// Odin by Brandefense - Monograph 62
// Hunts Linux /etc/nsswitch.conf edits, non-package libnscd.so writes, Root CA trust additions, and
scripted SSH/proxy floods.
let Lookback = 7d;
let ApprovedAdminHosts = dynamic(["cfg-mgmt-01.corp.example", "pki-admin-01.corp.example"]);
let ApprovedPackageManagers = dynamic(["dpkg", "apt", "apt-get", "rpm", "yum", "dnf", "apk", "pacma
n", "zypper"]);
let NssAndCaFileWrites =
    DeviceFileEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminHosts))
    | where ActionType in ("FileCreated", "FileModified", "FileRenamed")
    | where not(InitiatingProcessFileName in~ (ApprovedPackageManagers))
    | where (
        FolderPath == "/etc" and FileName == "nsswitch.conf"
        or FileName == "libnscd.so"
        or (FileName startswith "libnss_" and FileName has ".so")
        or FolderPath startswith "/usr/local/share/ca-certificates"
        or FolderPath startswith "/etc/pki/ca-trust/source/anchors"
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "Linux
NssOrCaStoreWrite", Detail = strcat(InitiatingProcessFileName, " -> ", FolderPath, "/", FileName);
let TrustAndFloodProcesses =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminHosts))
    | where (
        ProcessCommandLine has_any ("update-ca-certificates", "update-ca-trust", "trust anchor")
        or (ProcessCommandLine has "/etc/nsswitch.conf" and not(FileName in~ (ApprovedPackageManager
s)))
```

```
        or ProcessCommandLine has_any ("libnscd.so", "tcpshield_bypass", "ssh_proxy_flood", "ssh_nat
ive", "cf_bypass")
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "TrustStoreOrBypassProcess", Detail =
ProcessCommandLine;
let StagedLoaderEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminHosts))
    | where InitiatingProcessFileName in~ ("python", "python3", "curl", "wget")
    | where RemoteUrl has_any ("/libnscd.so", "/kreport", "/kyng-ca.pem")
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "Stage
dLoaderOrCaFetch", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteUrl, " (", RemoteIP,
":", tostring(RemotePort), ")");
union NssAndCaFileWrites, TrustAndFloodProcesses, StagedLoaderEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = m
ake_set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

8.3 Portable Sigma and YARA Rules

- **Sigma (`sigma_linux_nss_hijack_root_ca_and_ssh_flood.yml`).** Rule 1 detects Linux commands referencing `/etc/nsswitch.conf`, `/libnscd.so`, `kp_admin.py`, or `/kreport` outside `cfg-mgmt-01.corp.example`. Rule 2 catches `update-ca-certificates`, `update-ca-trust`, `trust anchor`, or `NODE_EXTRA_CA_CERTS=` paired with `.pem`, `.crt`, `ca-certificates`, or `kyng-ca` outside `pki-admin-01.corp.example`. Rule 3 detects `tcpshield_bypass`, `ssh_proxy_flood`, `ssh_native.py`, `ssh_killer`, `cf_bypass`, or `ovh_specter` outside `net-test-01.corp.example`.
- **YARA (`odin_nss_loader_custom_ca_and_l7_bypass.yar`).** Four `filesize`-bounded rules match `/libnscd.so` + `/kreport` loader scripts (**< 256KB**) and custom CA trust-injection scripts (**< 128KB**). Two further rules match raw `PROXY TCP4` spoofing scripts (**< 64KB**) and HTTP `CONNECT` **:22** SSH flood scripts (**< 64KB**).

9. Indicators

No VPS IPv4 addresses, referenced C2 addresses, file SHA-256 digests, or victim cookie filenames appear in the indicator feeds (**0** published indicators in the Indicators tab). Defenders should hunt using the 10 protocol, CLI, and file-integrity markers below.

Marker category	Pattern / artifact marker	Layer	Defensive hunt & control
Staged library URI	`GET /libnscd.so`	HTTP egress	Alert on visible HTTP requests with `sid:9562001` and KQL `StagedLoaderEgress`. Block unauthorized binary fetches at egress proxies.

Marker category	Pattern / artifact marker	Layer	Defensive hunt & control
Loader callback URI	`POST /kreport`	HTTP egress	Alert via `sid:9562002` behind egress proxies and inspect the initiating Linux process in EDR.
Linux NSS & `.so` FIM	`/etc/nsswitch.conf` & `libnscd.so`	Linux FIM	Alert on writes outside `dpkg`, `apt`, `rpm`, or `dnf` via Sentinel `NssAndCaFileWrites` and Sigma Rule 1.
Custom Root CA URI	`GET /kyng-ca.pem` (1,318 B)	HTTP egress	Alert via `sid:9562003` and inspect endpoints fetching `.pem` files from untrusted servers.
Custom TLS CA issuer	`tls.cert_issuer` containing `kyng`	TLS / proxy	Alert via `sid:9562004` on TLS 1.2 or TLS-inspected traffic; verify against approved enterprise CAs.
CA trust registration	`update-ca-certificates` / `trust anchor`	Linux CLI	Alert via Sentinel `TrustAndFloodProcesses` and Sigma Rule 2 outside approved PKI hosts.
Spoofed PROXY header	`PROXY TCP4` from untrusted WAN IPs	TCP ingress	Alert via `sid:9562005` and drop direct non-shield packets at the origin firewall (`[REF-10]`).
SSH banner flood	`SSH-2.0-` (`>=40` / 10s per source)	TCP 22	Alert via `sid:9562006` and enforce `PerSourceMaxStartups` in `sshd_config` (`[REF-08]`).
Proxied SSH flood	`CONNECT <host>:22`	HTTP proxy	Alert via `sid:9562007` and restrict forward proxies to `CONNECT` port 443 only.
Flood script CLI	`tcpshield_bypass` / `ssh_proxy_flood`	Linux EDR	Alert via Sigma Rule 3 and stop unauthorized flood processes on Linux hosts.

10. Scope and Limits

Odin by Brandefense analyzed exposed directories on three commercial virtual private servers recorded on `13-20 November 2026`. Workbench A was recorded on `20 November 2026`; Workbenches B and C on `13-20`

`November 2026`. Counts cover all **95** deduplicated entries (**90** sized files = **275,250,121** bytes and **5** unsized entries; **78** unique SHA-256 digests).

Directory listings and local log files (**189,024,142**-byte `py.log`) document staged files and local console output, not confirmed compromises or outages. All VPS IPv4 addresses, referenced C2 addresses, all **78** file digests, and the **30,642**-byte cookie filename on Workbench C are withheld. **Zero shared SHA-256 digests or infrastructure indicators link Workbenches A, B, and C.**

11. Sources

1. `[REF-01]` Linux Manual Pages (`man7.org`), *nsswitch.conf(5) - Name Service Switch Configuration File* (verified on 24 November 2026).
2. `[REF-02]` Linux Manual Pages (`man7.org`), *nscd(8) - Name Service Cache Daemon* (verified on 24 November 2026).
3. `[REF-03]` MITRE ATT&CK, *Hijack Execution Flow: Dynamic Linker Hijacking (T1574.006)* (verified on 24 November 2026).
4. `[REF-04]` MITRE ATT&CK, *Modify Authentication Process (T1556)* (verified on 24 November 2026).
5. `[REF-05]` MITRE ATT&CK, *Subvert Trust Controls: Install Root Certificate (T1553.004)* (verified on 24 November 2026).
6. `[REF-06]` IETF, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC 5280)* (verified on 24 November 2026).
7. `[REF-07]` MITRE ATT&CK, *Endpoint Denial of Service (T1499)* (verified on 24 November 2026).
8. `[REF-08]` OpenBSD Manual Pages, *sshd_config(5) - OpenSSH Daemon Configuration File* (verified on 24 November 2026).
9. `[REF-09]` IETF, *The Secure Shell (SSH) Transport Layer Protocol (RFC 4253)* (verified on 24 November 2026).
10. `[REF-10]` HAProxy Technologies, *The PROXY Protocol Versions 1 & 2 Specification* (verified on 24 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.

Threat Search

Pivot on an indicator across the Brandefense dataset.

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_nss_loader_custom_ca_and_l7_bypass.yar** 72 lines

```
/*
   Odin by Brandefense - Monograph 62 YARA Suite
   4 filesize-bounded structural rules covering staged Linux NSS / libnscd.so loaders,
   custom Root CA trust injectors, HAProxy PROXY v1 origin-bypass scripts, and SSH proxy floods.
*/

rule ODIN_WB62_Linux_NSS_Libnscd_Loader_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "62"
        description = "Detects staged administrative loader scripts referencing /libnscd.so, /krepor
t, and Linux NSS or dynamic linker paths"
        date = "2026-11-24"
    strings:
        $u1 = "/libnscd.so" ascii
        $u2 = "/kreport" ascii
        $n1 = "nsswitch.conf" ascii
        $n2 = "LD_PRELOAD" ascii
        $n3 = "nscd" ascii
    condition:
        filesize < 256KB and $u1 and $u2 and ($n1 or $n2 or $n3)
}

rule ODIN_WB62_Custom_CA_Interceptor_Injector
{
    meta:
        author = "Odin by Brandefense"
        monograph = "62"
        description = "Detects Python or JavaScript TLS trust-injection scripts pairing custom PEM C
A files with trust-store or TLS context overrides"
        date = "2026-11-24"
    strings:
        $c1 = "kyng-ca.pem" ascii
        $c2 = "BEGIN CERTIFICATE" ascii
        $t1 = "update-ca-certificates" ascii
        $t2 = "NODE_EXTRA_CA_CERTS" ascii
        $t3 = "load_verify_locations" ascii
    condition:
        filesize < 128KB and ($c1 or $c2) and ($t1 or $t2 or $t3)
}

rule ODIN_WB62_HAProxy_TCPShield_Origin_Bypass_Script
{
    meta:
        author = "Odin by Brandefense"
```

```
        monograph = "62"
        description = "Detects Python scripts building spoofed HAProxy PROXY v1 TCP4 headers to test
direct-to-origin shield bypasses"
        date = "2026-11-24"
    strings:
        $p1 = "PROXY TCP4 " ascii
        $s1 = "socket.socket" ascii
        $s2 = "SOCK_STREAM" ascii
        $h1 = "\\r\\n" ascii
    condition:
        filesize < 64KB and $p1 and ($s1 or $s2) and $h1
}

rule ODIN_WB62_SSH_Proxy_CONNECT_Flood_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "62"
        description = "Detects Python scripts tunneling high-rate SSH-2.0 handshake floods through H
TTP CONNECT proxies to port 22"
        date = "2026-11-24"
    strings:
        $m1 = "CONNECT " ascii
        $m2 = ":22 HTTP/1." ascii
        $b1 = "SSH-2.0-" ascii
        $s1 = "socket" ascii
    condition:
        filesize < 64KB and $m1 and $m2 and $b1 and $s1
}
```

Suricata `odin_nss_loader_custom_ca_and_l7_bypass_hunt.rules` 11 lines

```
# Odin by Brandefense - Monograph 62 Suricata 7.x Ruleset
# Covers staged libnscd.so / kreport routes, custom kyng Root CA fetches/issuers, PROXY v1 spoofing,
and SSH floods.

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-A Outbound HTTP Request for Staged l
ibnscd.so Shared Object"; flow:established,to_server; http.uri; content:"/libnscd.so"; endswith; fas
t_pattern; classtype:trojan-activity; sid:9562001; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-A Outbound HTTP POST Callback to Sta
ged kreport Endpoint"; flow:established,to_server; http.method; content:"POST"; bsize:4; http.uri; c
ontent:"/kreport"; startswith; fast_pattern; classtype:command-and-control; sid:9562002; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-B Outbound HTTP Fetch of Custom kyng
-ca.pem Root Certificate"; flow:established,to_server; http.uri; content:"/kyng-ca.pem"; endswith; f
ast_pattern; classtype:policy-violation; sid:9562003; rev:1;)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R62 WB-B Inbound TLS Server Certificate Issu
ed by Custom kyng CA"; flow:established,to_client; tls.cert_issuer; content:"kyng"; nocase; fast_pat
tern; classtype:bad-unknown; sid:9562004; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R62 WB-C Inbound Direct-to-Origin HAProxy PR
OXY v1 Header Spoof Attempt"; flow:established,to_server; content:"PROXY TCP4 "; startswith; fast_pa
ttern; classtype:attempted-recon; sid:9562005; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"ODIN-R62 WB-C Inbound High-Rate SSH-2.0 Handshake
Flood Against TCP 22"; flow:established,to_server; content:"SSH-2.0-"; startswith; fast_pattern; thr
```

```
eshold:type both, track by_src, count 40, seconds 10; classtype:attempted-dos; sid:9562006; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R62 WB-C Outbound HTTP CONNECT Proxy Tunnel
Targeting SSH Port 22"; flow:established,to_server; http.method; content:"CONNECT"; bsize:7; http.ur
i.raw; content:":22"; endswith; fast_pattern; classtype:policy-violation; sid:9562007; rev:1;)
```

Show full rule

Sentinel KQL **sentinel_linux_nss_custom_ca_and_ssh_flood_hunt.kql** 40 lines

```
// Odin by Brandefense - Monograph 62
// Hunts Linux /etc/nsswitch.conf edits, non-package libnscd.so writes, Root CA trust additions, and
scripted SSH/proxy floods.
let Lookback = 7d;
let ApprovedAdminHosts = dynamic(["cfg-mgmt-01.corp.example", "pki-admin-01.corp.example"]);
let ApprovedPackageManagers = dynamic(["dpkg", "apt", "apt-get", "rpm", "yum", "dnf", "apk", "pacma
n", "zypper"]);
let NssAndCaFileWrites =
    DeviceFileEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminHosts))
    | where ActionType in ("FileCreated", "FileModified", "FileRenamed")
    | where not(InitiatingProcessFileName in~ (ApprovedPackageManagers))
    | where (
        FolderPath == "/etc" and FileName == "nsswitch.conf"
        or FileName == "libnscd.so"
        or (FileName startswith "libnss_" and FileName has ".so")
        or FolderPath startswith "/usr/local/share/ca-certificates"
        or FolderPath startswith "/etc/pki/ca-trust/source/anchors"
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "Linux
NssOrCaStoreWrite", Detail = strcat(InitiatingProcessFileName, " -> ", FolderPath, "/", FileName);
let TrustAndFloodProcesses =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminHosts))
    | where (
        ProcessCommandLine has_any ("update-ca-certificates", "update-ca-trust", "trust anchor")
        or (ProcessCommandLine has "/etc/nsswitch.conf" and not(FileName in~ (ApprovedPackageManager
s)))
        or ProcessCommandLine has_any ("libnscd.so", "tcpshield_bypass", "ssh_proxy_flood", "ssh_nat
ive", "cf_bypass")
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "TrustStoreOrBypassProcess", Detail =
ProcessCommandLine;
let StagedLoaderEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedAdminHosts))
    | where InitiatingProcessFileName in~ ("python", "python3", "curl", "wget")
    | where RemoteUrl has_any ("/libnscd.so", "/kreport", "/kyng-ca.pem")
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "Stage
dLoaderOrCaFetch", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteUrl, " (", RemoteIP,
```

```
":", tostring(RemotePort), ")");
union NssAndCaFileWrites, TrustAndFloodProcesses, StagedLoaderEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = m
ake_set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

Sigma `sigma_linux_nss_hijack_root_ca_and_ssh_flood.yml` 92 lines

```
title: Linux NSS Configuration Edit or Staged libnscd Shared Library Command
id: 62a9d410-3b18-4f21-9c01-62b10e48a101
status: experimental
description: Detects Linux command lines referencing /etc/nsswitch.conf modifications or staging a n
on-standard libnscd.so shared object and /kreport callback route.
author: Odin by Brandefense
date: 2026-11-24
logsource:
  category: process_creation
  product: linux
detection:
  selection_nss_or_loader:
    CommandLine|contains:
      - '/etc/nsswitch.conf'
      - '/libnscd.so'
      - 'kp_admin.py'
      - '/kreport'
  filter_cfg_hosts:
    ComputerName:
      - 'cfg-mgmt-01.corp.example'
  condition: selection_nss_or_loader and not filter_cfg_hosts
falsepositives:
  - Designated configuration management hosts applying approved identity-provider templates
level: high
tags:
  - attack.persistence
  - attack.privilege-escalation
  - attack.credential-access
  - attack.t1574.006
  - attack.t1556
---
title: Linux Custom Root CA Trust Store Registration Command
id: 62a9d410-3b18-4f21-9c01-62b10e48a102
status: experimental
description: Detects Linux trust-store update utilities invoked alongside custom PEM or CRT certific
ate arguments or NODE_EXTRA_CA_CERTS overrides.
author: Odin by Brandefense
date: 2026-11-24
logsource:
  category: process_creation
  product: linux
detection:
  selection_trust_cmd:
    CommandLine|contains:
      - 'update-ca-certificates'
```

```
      - 'update-ca-trust'
      - 'trust anchor'
      - 'NODE_EXTRA_CA_CERTS='
  selection_cert_indicator:
    CommandLine|contains:
      - '.pem'
      - '.crt'
      - 'ca-certificates'
      - 'kyng-ca'
  filter_pki_hosts:
    ComputerName:
      - 'pki-admin-01.corp.example'
  condition: (selection_trust_cmd and selection_cert_indicator) and not filter_pki_hosts
falsepositives:
  - Approved enterprise PKI provisioning servers deploying internal root certificates
level: high
tags:
  - attack.defense-evasion
  - attack.t1553.004
---
title: Linux Application-Layer Edge Bypass and SSH Handshake Flood Execution
id: 62a9d410-3b18-4f21-9c01-62b10e48a103
status: experimental
description: Detects execution of staged edge-protection bypass, HAProxy PROXY header spoofing, and
SSH proxy or native handshake flood tools on Linux hosts.
author: Odin by Brandefense
date: 2026-11-24
logsource:
  category: process_creation
  product: linux
detection:
  selection_bypass_tools:
    CommandLine|contains:
      - 'tcpshield_bypass'
      - 'ssh_proxy_flood'
      - 'ssh_native.py'
      - 'ssh_killer'
      - 'cf_bypass'
      - 'ovh_specter'
  filter_lab_hosts:
    ComputerName:
      - 'net-test-01.corp.example'
  condition: selection_bypass_tools and not filter_lab_hosts
falsepositives:
  - Authorized network resilience testing hosts inside isolated lab segments
level: high
tags:
  - attack.impact
  - attack.t1499
```