FULL REPORT • RECON WORKBENCHES • MONOGRAPH 67 / 63 • 21 MIN READ

Chrome V8 Cage Recon, CVE-2022-4262 Archives, and Browser Workbenches

Inside a 325-file browser exploitation workbench on TCP 9000, mapping staged V8 cage reconnaissance to V8 Sandbox boundaries, and hardening enterprise Chromium fleets

BY
Odin by Brandefense

PUBLISHED
19 November 2026

READING TIME
21 min read

EDITION
Version 1.1 · TLP:CLEAR

SCOPE
Prepared November 2026

1. Executive Summary

IN SHORT

Odin by Brandefense analyzed a 325-file browser exploitation workbench on TCP port 9000 of a China-hosted cloud server, recorded on 13–25 November 2026. The open Python HTTP directory pairs packaged CVE-2022-4262 V8 reproducers with 94 in-cage reconnaissance scripts and archives. Modern V8 Sandbox tables block direct pointer overwrites. Keep Chromium browsers on current supported security releases, apply JIT-less browser policies, and restrict high-port web egress.

Modern Chromium browsers isolate untrusted JavaScript and WebAssembly inside strict memory boundaries. V8 pointer compression uses a 4 GB heap region. The separate V8 Sandbox reserves a larger address space and aims to prevent in-cage corruption from writing outside it. Separate out-of-cage tables guard native code pointers and C++ backing stores.

Between 13 and 25 November 2026, Odin by Brandefense studied an open HTTP directory on TCP port 9000 of a China-hosted cloud server. That listener ran `SimpleHTTP/0.6 Python/3.12.3`. On 25 November 2026, the directory listing reported 325 active files and 10 directories.

Deduplicating all paths yields 325 sized files totaling 1,928,392 bytes (1.93 MB) and zero unsized entries. Those 325 files map to 291 unique SHA-256 digests and 34 duplicate file copies across subdirectories.

We assess this host consistently as an adversary browser-exploitation staging and cage-testing workbench. Odin by Brandefense publishes zero IPv4 addresses and zero SHA-256 digests in this report. We withhold the server IP, all 291 file digests, and all folder and file names inside a 197-entry backup test subdirectory.

1 **Standard-renderer CVE-2022-4262 files sit beside seven parallel staging folders.**
Six entries totaling 27,104 bytes hold a reproducer suite for CVE-2022-4262, a V8 type-confusion bug fixed in Chrome 108.0.5359.94 in December 2022. Archive `cveweb.tar.gz` (4,722 bytes) packs HTML and JavaScript files that run without V8 debug flags. Meanwhile, seven short subdirectories each hold an 86-byte HTML loader, an `exp.js` script, and a 9,996-byte `exploit-q.js` copy.

2 **Ninety-four recon scripts and archives test V8 cage and WebAssembly layouts.**
Eighty-seven root HTML and JavaScript files (261,904 bytes) and seven `.tar.gz` bundles (12,269 bytes) probe inside the V8 heap cage. Numbered archives step from typed-array external pointers and WebAssembly tables to JavaScript function code links and `float64` constant pools.

3 **V8 Sandbox tables and non-writable code pages stop direct pointer swaps.**
External Pointer Tables replace raw C++ backing-store addresses with type-tagged handles. The out-of-cage Code Pointer Table removes direct instruction pointers from JavaScript function objects. Write-xor-execute rules and hardware memory protection keys also block direct writes to WebAssembly jump tables and JIT pages.

4 **Local Linux procfs pause code is consistent with a test rig.**
Two archives (`recon5.tar.gz` and `r5.tar.gz`) are written to print candidate addresses to page output and pause for a local `/proc/mem` check. A remote page cannot itself read a visitor's `/proc/mem`. Those staged pauses are consistent with a test rig, and no live drive-by attack is established.

Three Decisions for Security Leaders

1. **Keep Chromium browsers on current security releases with prompt relaunches.** Keep Chrome, Edge, kiosk browsers, and Electron apps on current supported security releases (`108.0.5359.94` is the historical fix for `CVE-2022-4262`) to patch known exploited vulnerabilities. Set enterprise relaunch policies so security updates restart browsers within 48 to 72 hours.
2. **Turn on V8 JIT-less mode for high-risk users and lock Site Isolation.** Block JavaScript JIT compilation by default for executive, admin, and finance profiles while allowlisting vetted web apps. Enforce `SitePerProcess` and alert on `--no-sandbox` or `--allow-natives-syntax` flags.
3. **Block direct browser traffic to high TCP ports and run structural hunts.** Stop workstation browsers from opening direct HTTP sessions to public hosts on ports like TCP 9000. Route web traffic through a proxy watched by Suricata, Sentinel KQL, Sigma, and YARA rules.

The Study in Numbers

Number	What it is
325	Active sized files (1,928,392 bytes; 0 unsized) across 10 reported directories on TCP 9000 on 25 November 2026
291	Unique file SHA-256 digests across the 325 sized files (all withheld; 34 duplicate copies across subdirectories)
197	Backup browser PoC files (1,473,170 bytes; 60.6% of files) inside a withheld test subdirectory
94	In-cage V8 and WebAssembly recon files: 87 root `recon*` files (261,904 B) plus 7 `.tar.gz` bundles (12,269 B)
8	Compressed `.tar.gz` bundles totaling 16,991 bytes (40,582 unpacked bytes across 15 members and 1,096 lines)
6	`CVE-2022-4262` V8 type-confusion reproducer entries (27,104 bytes), fixed in Chrome `108.0.5359.94`

Number	What it is
0	Published IPv4 addresses, file SHA-256 digests, exploit snippets, or backup subdirectory names

2. Workbench Layout, 325-File Inventory, and Cluster Partition

The workbench ran an unauthenticated `SimpleHTTP/0.6 Python/3.12.3` server on `TCP 9000` of a China-hosted cloud VPS (`[REF-09]`). Odin by Brandefense recorded the directory on `13-25 November 2026`, with **325 active files** and **10 directories** reported on `25 November 2026`.

Enterprise defenders can apply vulnerability intelligence to verify browser patch floors across managed endpoints and virtual desktops. Security operations teams can also use cyber threat intelligence to correlate staged archive names and high-port web banners across proxy logs.

We reviewed directory listings, byte counts, and archive structures statically without running any script or printing offensive code. All **325 deduplicated entries carry exact sizes (0 unsized)** totaling **1,928,392 bytes (1.93 MB)**, with **291 unique SHA-256 digests** (all withheld) and **34 duplicate copies**. Figure 1 maps the four workbench tracks.

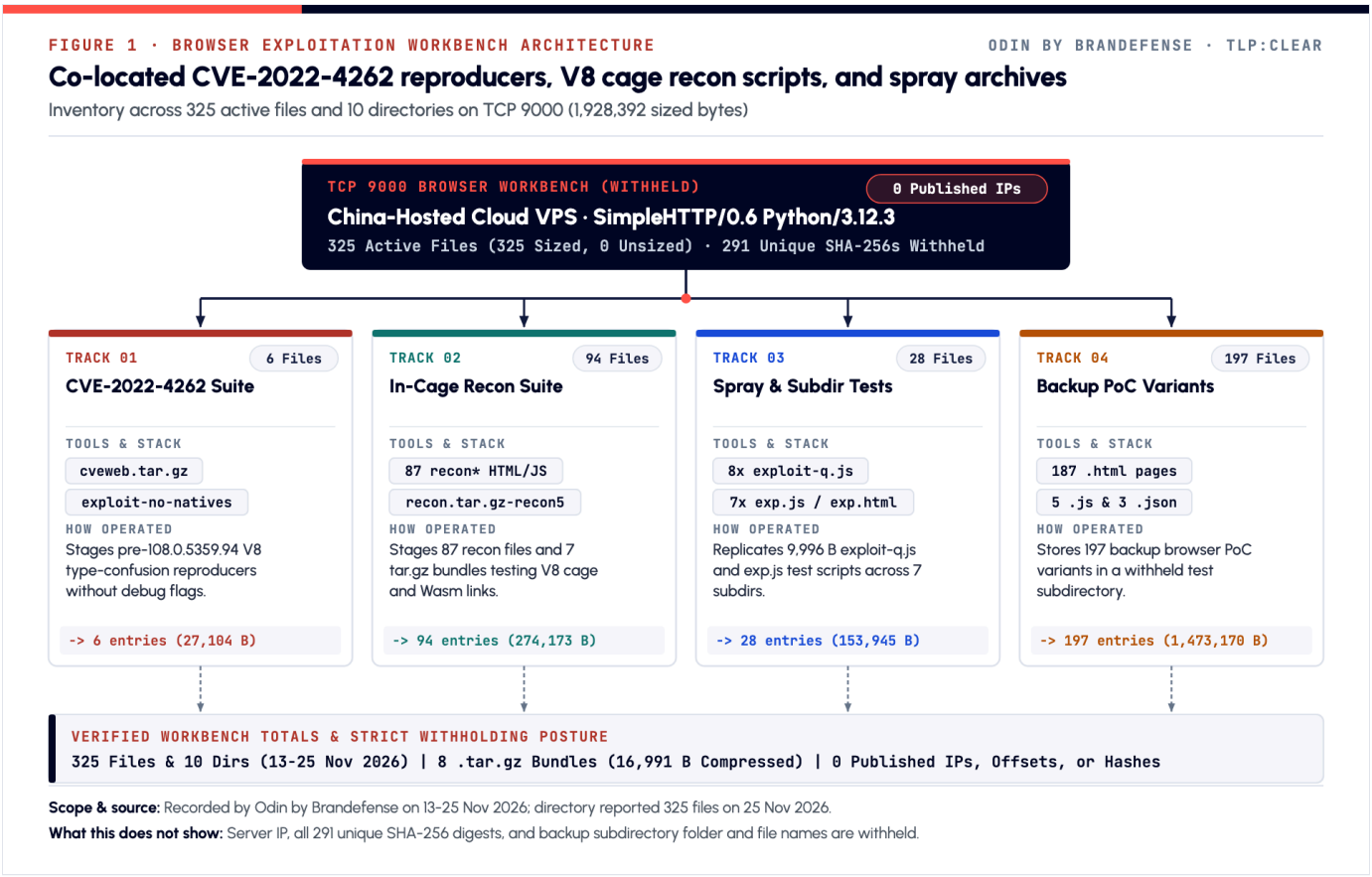

Figure 1: Co-located CVE-2022-4262 reproducers, V8 cage recon scripts, and spray archives

Six Functional Workbench Clusters (`n = 325`)

Sorting all **325** files into six mutually exclusive clusters separates bulk backup test pages from active V8 cage reconnaissance (`197 + 87 + 16 + 12 + 7 + 6 = 325`). Table 1 reconciles file counts and byte totals across all six groups.

Functional cluster	Entries	Share (%)	Sized bytes	Byte share (%)	Observed contents
Backup Browser PoC Variants (withheld dir)	197	60.6%	1,473,170	76.4%	**187** `.html` , **5** `.js` , **3** `.json` , and **2** `.bak` files in a withheld folder.
Iterative Wasm & V8 Recon (`recon*`)	87	26.8%	261,904	13.6%	Root HTML/JS files up to `recon24r.html` / `recon24r.js` (peaking at **13,028** B in `recon24p.js`).
Subdirectory Staging & Aux Loaders	16	4.9%	60,138	3.1%	**7** `exp.js` (7,618–9,076 B), **7** 86-byte `exp.html` , `diag-ladder.html` (2,366 B), `test.txt` (10 B).
Debug, JIT-Spray & Code-Page Scripts	12	3.7%	93,807	4.9%	`exploit-dbg.js` (10,321 B), **8x** `exploit-q.js` (9,996 B each), and **3** `dbg103*.html` pages.
Packaged In-Cage Recon Archives (`*.tar.gz`)	7	2.2%	12,269	0.6%	`r6` , `recon4` , `recon5` , `recon3` , `recon2` , `r5` , and `recon` `.tar.gz` bundles.
`CVE-2022-4262` **Type-Confusion Suite**	6	1.8%	27,104	1.4%	`exploit-no-natives.js` , `exploit-loadable.js` , `cveweb.tar.gz` , HTML loader, and **2** `._*` stubs.
Total workbench (325 **sized,** 0 **unsized)**	**325**	**100.0%**	**1,928,392**	**100.0%**	10 **reported directories;** 291 **unique SHA-256 digests withheld.**

Two clear patterns stand out in Table 1 and Figure 5. First, the withheld backup folder (**197** files) and root `recon*` files (**87** files) hold **284 of the 325 entries (87.4%)** and **1,735,074 bytes (90.0%)**. Folder and file names inside that **197**-entry backup folder reflect copied third-party PoC labels that do not prove who runs the server, so we withhold them all.

Second, the remaining **41 entries (12.6%; 193,318 bytes)** hold the packaged archives, `CVE-2022-4262` suite, debug scripts, and subdirectory harnesses. Adding the **87** root `recon*` files to the **7** recon `.tar.gz` bundles gives **94 in-cage recon files (274,173 bytes)**. Figure 5 shows the six-cluster breakdown.

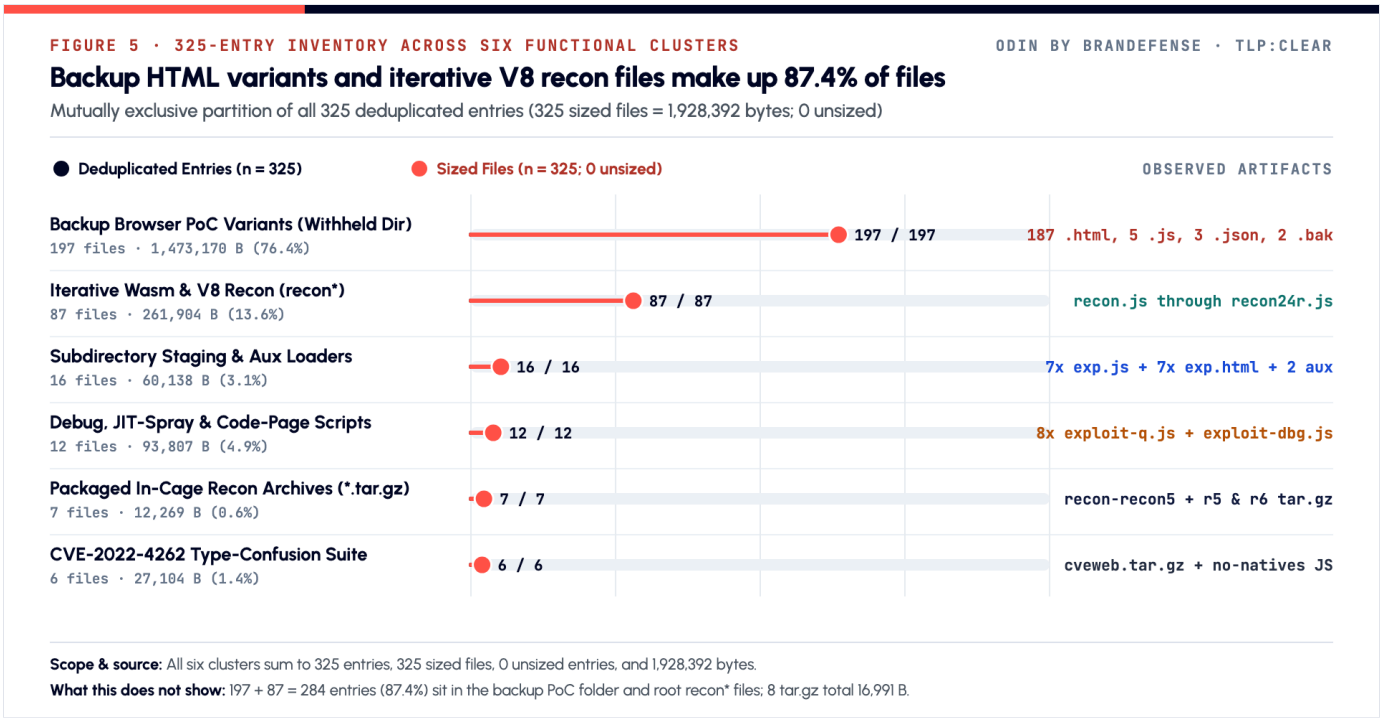

Figure 5: Backup HTML variants and iterative V8 recon files make up 87.4% of files

Four Artifact Classes and Five Directory Zones (1,928,392 Bytes)

Grouping the **325** entries by file type shows that the reviewed directory lists only web pages, JavaScript files, tarballs, and two `.bak` backups. The reviewed directory inventory lists no server binaries, private keys, or identified victim files. Table 2 reconciles all four file classes (`248 + 67 + 8 + 2 = 325`).

Artifact class	Entries	Share (%)	Sized bytes	Byte share (%)	Extension breakdown
`text`	248	76.3%	1,437,381	74.5%	244 `.html` pages, 3 `.json` files, and 1 `.txt` file (`test.txt` , 10 B).
`script`	67	20.6%	452,898	23.5%	67 `.js` V8/Wasm recon, spray, and reproducer scripts.
`archive`	8	2.5%	16,991	0.9%	8 `.tar.gz` bundles (15 members = 40,582 B unpacked).
`data`	2	0.6%	21,122	1.1%	2 `.bak` backup files inside the withheld 197-entry folder.
Total	**325**	**100.0%**	**1,928,392**	**100.0%**	325 **sized files (0 unsized).**

Across the folder tree, **197** files (**1,473,170** B, **60.6%**) sit in the withheld backup subdirectory and **104** files (**307,438** B, **32.0%**) sit in `(root)` . Eight named subdirectories hold the remaining **24** files (**147,784** B, **7.4%**). Folders `/nfb` and `/rs1` – `/rs3` hold **12** files (**70,800** B; **17,700** B per folder). Folders `/k` – `/k3` hold **9** files (**56,934** B), and `/cve-2022-4262` holds **3** files (**20,050** B).

Seven copies of `exploit-q.js` (**9,996** B each) mirror `/exploit-q.js` across `/k` – `/k3`, `/nfb`, and `/rs1` – `/rs3` (**8** copies = **79,968** B). Matching **86**-byte `exp.html` loaders and **7,618**-byte `exp.js` scripts also repeat across `/nfb` and `/rs1` – `/rs3`, explaining the **34** duplicate copies.

3. CVE-2022-4262 Standard-Renderer Archives and Subdirectory Staging

This workbench stages a patched V8 reproducer beside later cage-recon archives. On `TCP 9000`, six entries (**27,104** bytes) center on `CVE-2022-4262`, while seven `.tar.gz` bundles (**12,269** bytes) store follow-on cage checks. Figure 4 shows the `TCP 9000` server blueprint.



Figure 4: One China-hosted VPS exposes 325 browser test files and 8 tar.gz bundles

`CVE-2022-4262` Patch Scope and Why `no-natives` Matters

Per NIST NVD (`[REF-01]`) and Google Chrome Releases (`[REF-02]`), `CVE-2022-4262` is a high-severity type confusion in Google Chrome V8 before `108.0.5359.94`. Google fixed the flaw in Desktop Stable update `108.0.5359.94/.95` on `2 December 2022` (`[REF-02]`). On unpatched builds below `108.0.5359.94`, a crafted HTML page could trigger V8 heap corruption (`[REF-01]`).

Public bug-tracker test cases for V8 bugs often rely on `%` runtime intrinsics gated behind `--allow-natives-syntax`. Normal Chrome browsers reject those debug intrinsics. Two staged files adapt the reproducer for standard pre-`108.0.5359.94` renderers:

- **Standard-renderer `cveweb.tar.gz` (4,722 bytes).** Unpacking `/cveweb.tar.gz` yields **4** members totaling **12,467** bytes (**336** lines). It packs `cve-2022-4262-no-natives.html` (**1,958** B), `cve-2022-4262/exploit-no-natives.js`

(9,917 B), and two macOS AppleDouble `._*` stubs (374 B and 218 B = 592 B). Dropping `--allow-natives-syntax` lets the page run in an unpatched browser build without debug flags.

- **Modular `exploit-loadable.js` (9,915 bytes).** Stored beside `exploit-no-natives.js` inside `/cve-2022-4262/`, this 9,915-byte file lets separate HTML wrappers import the baseline script. The two AppleDouble `._*` stubs show that the archive was packed on macOS before upload to the Linux VPS.

Eight Packaged `.tar.gz` Bundles (16,991 Compressed Bytes)

Across the root directory, **eight `.tar.gz` archives** total **16,991 compressed bytes** and unpack to **40,582 bytes** (1,096 lines) across **15 members**. Table 3 summarizes each bundle without printing exploit code or memory offsets.

Archive path	Compressed bytes	Members	Unpacked bytes	Unpacked lines	Structural role
`/cveweb.tar.gz`	4,722	4	12,467	336	`cve-2022-4262-no-natives.html` (1,958 B), `exploit-no-natives.js` (9,917 B), and 2 `._*` stubs (592 B).
`/r6.tar.gz`	2,472	1	6,277	158	Single-file bundle (`recon6.js`) testing constant spacing and indirect code-entry behavior.
`/recon4.tar.gz`	1,991	2	4,473	126	`recon4.js` + `recon4.html` testing JIT-spray markers in `TurboFan` `float64` constant pools.
`/recon5.tar.gz`	1,893	2	4,344	117	`recon5.js` + `recon5.html` written to print candidate addresses and pause for `/proc/mem` checks.
`/recon3.tar.gz`	1,780	2	3,923	108	`recon3.js` + `recon3.html` walking in-cage `JSFunction` -to- `Code` links before a write check.
`/recon2.tar.gz`	1,647	2	3,708	109	`recon2.js` + `recon2.html` checking WebAssembly instance structures and jump tables.
`/r5.tar.gz`	1,345	1	3,037	79	Single-file snapshot (`recon5.js` , 3,037 B) written to pause for local `/proc/mem` checks.
`/recon.tar.gz`	1,141	1	2,353	63	Baseline script (`recon.js` , 2,353 B) checking `JSTypedArray` `external_pointer` and Wasm fields.
Total (8 bundles)	**16,991**	**15**	**40,582**	**1,096**	4,722 B (`cveweb.tar.gz`) + 12,269 B (7 recon bundles = 28,115 B unpacked).

Alongside those eight tarballs, `(root)` holds **87** unpacked `recon*` files (**261,904** bytes). Those files run from `recon.html` / `recon.js` through `recon24r.html` / `recon24r.js` and peak at **13,028** bytes in `recon24p.js` . Root also holds `diag-ladder.html` (**2,366** B), `exploit-dbg.js` (**10,321** B), and three `dbg103` pages (**469** B, **1,358** B, and **1,691** B).

---

4. In-Cage Reconnaissance Versus Modern V8 Sandbox Boundaries

Checking `recon.tar.gz` through `r6.tar.gz` against V8 Sandbox design docs (`[REF-03]` , `[REF-04]`) shows why older V8 chains stall on current Chrome. Figure 3 presents an analytical model of those staged checks against modern Chromium boundaries.

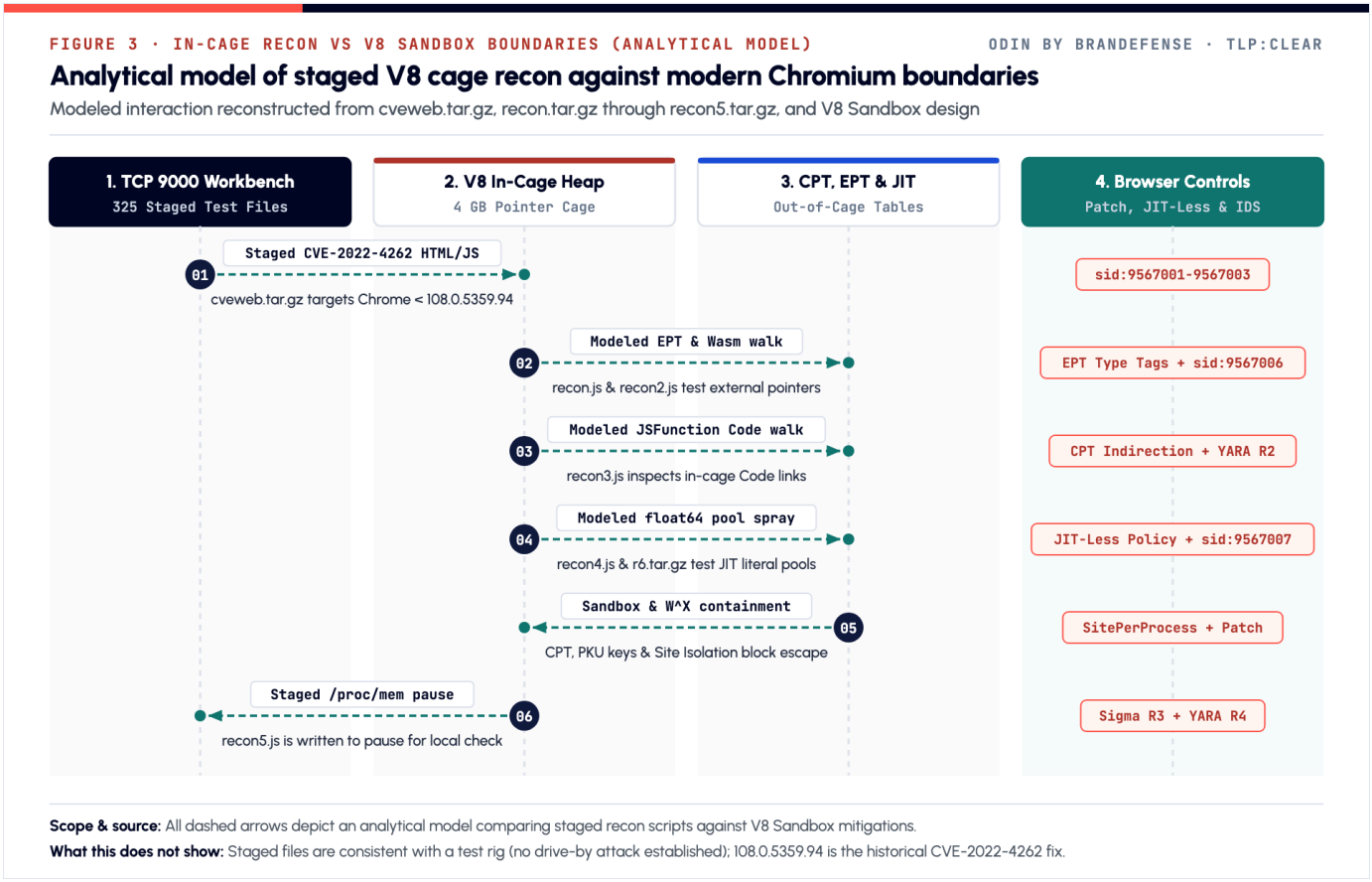

Figure 3: Analytical model of staged V8 cage recon against modern Chromium boundaries

Four V8 Sandbox Barriers Tested by the `recon*` Scripts

On 64-bit systems, V8 Pointer Compression keeps JavaScript objects inside a **4 GB** virtual cage using 32-bit offsets (`[REF-03]`). Older V8 exploits turned heap bugs into code execution by overwriting raw 64-bit pointers stored inside heap objects. Current Chromium releases enforce four barriers that block those paths:

1. **External Pointer Table (`EPT`) handles (`recon.tar.gz` , 2,353 B unpacked).** Corrupting a `JSTypedArray` backing-store pointer once gave arbitrary 64-bit memory read/write. The **V8 Sandbox** (`[REF-03]` , `[REF-04]`) replaces raw C++ pointers with bounds-checked, type-tagged indices into an out-of-cage `EPT` . When `recon.js` checks `external_pointer` fields on modern Chrome, it meets table handles instead of raw addresses.
2. **WebAssembly trusted data and `W^X` / `PKU` pages (`recon2.tar.gz` , 3,708 B unpacked).** WebAssembly `RWX` jump tables were once a primary code-execution target. Modern V8 moves trusted Wasm state outside the cage and guards JIT pages with `W^X` or hardware `PKU` keys (`[REF-03]` , `[REF-04]`). That blocks direct jump-table writes tested in `recon2.js` .

3. **Code Pointer Table (`CPT`) indirection (`recon3.tar.gz` 3,923 B & `r6.tar.gz` 6,277 B unpacked).** Corrupting a `JSFunction` once let an attacker point `code_entry_point` at shellcode. Chromium's **Code Pointer Table (`CPT`)** (`[REF-04]`) moves `Code` metadata and native entry pointers outside the cage behind validated table handles.
4. **`TurboFan` `float64` pools and local `/proc/mem` pauses (`recon4.tar.gz` 4,473 B & `recon5.tar.gz` 4,344 B unpacked).** Because `EPT` , `CPT` , and `W^X` block raw pointer swaps, `recon4.js` , `r6.tar.gz` , and `exploit-q.js` test constant spacing inside `TurboFan` `float64` pools. Script `recon5.js` (in `recon5.tar.gz` and `r5.tar.gz`) is written to print `addr_of` , `v8_read64` , and `v8_write64` values to the page. It is written to pause for a local Linux `/proc/<pid>/mem` check, consistent with a test rig rather than establishing a live drive-by attack.

Figure 2 links all five workbench artifact groups to browser policy and detection controls.

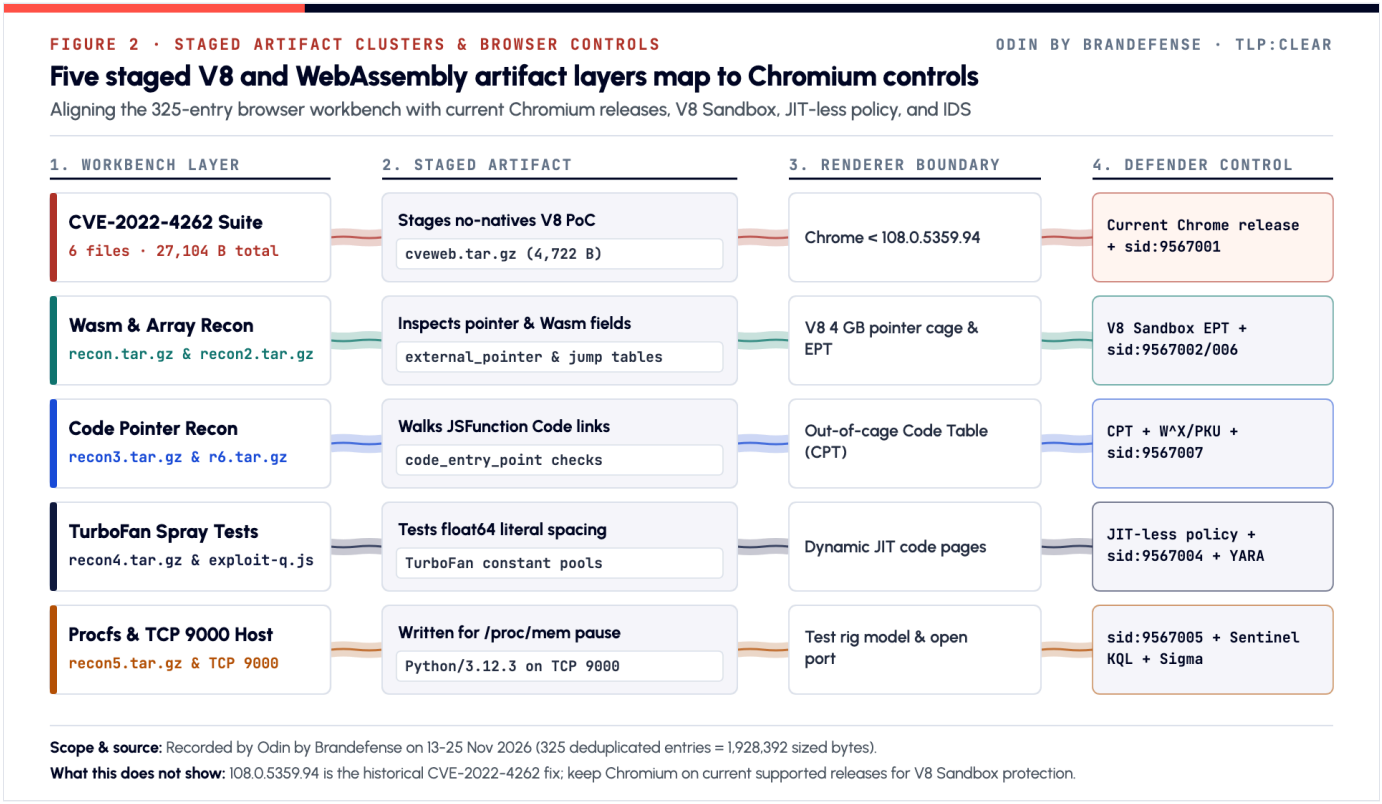

Figure 2: Five staged V8 and WebAssembly artifact layers map to Chromium controls

5. Enterprise Browser Exposure and Control Verification

Desktop and security teams should verify five concrete controls across corporate endpoints, developer hosts, and VDI pools rather than assuming unverified root causes:

Exposure layer	Observed workbench evidence	Control to verify	Operational risk
Legacy Chromium `< 108.0.5359.94`	`cveweb.tar.gz` (4,722 B), `cve-2022-4262-no-natives.html`, and `exploit-no-natives.js`.	Verify that browsers, kiosks, PDF containers, and Electron apps run current supported Chromium releases (`108.0.5359.94` is the historical fix for `CVE-2022-4262`; `[REF-01]`, `[REF-05]`, `[REF-10]`).	Pre-`108.0.5359.94` V8 builds remain exposed to `CVE-2022-4262`, and outdated releases lack current V8 Sandbox hardening (`[REF-03]`, `[REF-07]`).
Unrestricted `TurboFan` & Wasm JIT	`recon4.tar.gz` (`float64` pools), `r6.tar.gz`, 8x `exploit-q.js`, and `recon2.tar.gz`.	Verify whether high-risk profiles enforce `DefaultJavaScriptJitSetting` (`Block JavaScript JIT`) with allowlists (`[REF-06]`).	Active `TurboFan` and Wasm JIT compilers expose dynamic code-generation surfaces (`[REF-03]`, `[REF-06]`).
Disabled sandbox or Site Isolation	`no-natives` vs `--allow-natives-syntax` tests and `dbg103*.html` debug pages.	Verify whether endpoint controls enforce `SitePerProcess` and block `--no-sandbox`, `--allow-natives-syntax`, and `--disable-site-isolation-trials` (`[REF-08]`, `[REF-12]`).	Weakened flags strip OS sandbox and `SitePerProcess` barriers around renderer bugs (`[REF-12]`).
Direct high-port browser egress	Open `SimpleHTTP/0.6 Python/3.12.3` directory on `TCP 9000` serving 325 files (`[REF-09]`).	Verify whether firewalls and web gateways block direct browser calls to public hosts on `TCP 9000` and peer high ports.	Unfiltered high-port egress lets browsers pull staged test pages and tarballs from ad-hoc VPS servers.
Linux `/proc/mem` & `http.server`	`recon5.tar.gz` & `r5.tar.gz` (`/proc/mem` pauses) and `Python/3.12.3` on `TCP 9000`.	Verify `kernel.yama.ptrace_scope >= 1` on Linux hosts and alert on unapproved `python3 -m http.server` processes.	Same-UID processes can inspect browser memory or expose local security test folders online.

6. Alternative Hypotheses

Hypothesis	Evidence for	Evidence against	Assessment
Adversary browser-exploitation workbench	Pairs `cveweb.tar.gz` (`CVE-2022-4262` `no-natives`), `exploit-dbg.js`, 8x `exploit-q.js`, and 94 V8 recon files on a VPS.	None; `no-natives` archives and `/k*`, `/nfb`, `/rs*` paths fit offensive preparation.	**Supported.** Treated consistently as an adversary workbench; IP, 291 hashes, and backup names stay withheld.
Benign web developer or QA server	Runs `SimpleHTTP/0.6 Python/3.12.3` and holds `test.txt` (10 B) plus HTML pages.	Normal QA hosts never stage `CVE-2022-4262` `no-natives` archives and `float64` JIT-spray scripts.	**Contradicted.** Python's `http.server` here serves V8 exploitation test suites.
Confirmed active drive-by watering hole	Serves 244 `.html` and 67 `.js` files on `TCP 9000`, plus 7 `exp.html` loaders.	`recon5.js` is written to pause for a local `/proc/mem` check, `CVE-2022-4262` dates to Dec 2022, and the reviewed directory lists no identified visitor logs or victim artifacts.	**Not established by staged files.** Staged files are consistent with a test rig; no live drive-by attack is established.
Backup folder names prove operator identity	The 197-entry backup folder (1,473,170 B) carries third-party research labels.	Operators routinely copy public PoC repos and conference writeups onto cloud servers.	**Unsupported for attribution.** Folder labels do not prove authorship; all 197 backup names stay withheld.

7. Validated Detection Rules

The `TCP 9000` server IP, all **291** file SHA-256 digests, and all **197** backup folder names are withheld. Each rule file therefore targets HTTP patterns, browser or Python CLI flags, or structural script strings.

Check scope (`Publication Rules §6`): all seven Suricata rules passed the Odin Suricata linter (**0** problems). YAML checks verified all three Sigma rules, and structural checks verified the Sentinel KQL and YARA files. Live engine binaries (`suricata -T`, `Kusto.Language`, `pySigma`, and `yara`) were not installed on the build host. Rules were not replayed on live traffic, so test and tune them in staging before rollout.

7.1 Suricata 7.x Network Signatures (`sid:9567001` –9567007)

Rules `sid:9567001` and `sid:9567002` match outbound `GET` requests for `/cveweb.tar.gz` and `/recon*.tar.gz` (`recon.tar.gz` through `recon5.tar.gz`). Rules `sid:9567003` and `sid:9567004` match `GET` URIs pairing `cve-2022-4262` with `no-natives` or ending in `/exploit-q.js` . Rules `sid:9567005` –9567007 match inbound HTTP 200 `SimpleHTTP/` listings and V8 recon response bodies (`external_pointer` or `code_entry_point` with `TurboFan`).

All seven `http` rules require cleartext HTTP traffic (such as direct `TCP 9000` sessions) or an IDS tap behind a TLS-terminating forward proxy. Because `alert` rules record events without blocking packets inline, enforce blocks in your web proxy or firewall.

```
SURICATA

# Odin by Brandefense - Monograph 67 Suricata 7.x Ruleset
# Syntax-checked with Odin Suricata linter (0 problems); test with suricata -T prior to deployment.

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound CVE-2022-4262 cveweb Archive Fetc
h"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/cveweb.tar.gz";
endswith; fast_pattern; classtype:attempted-user; sid:9567001; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound V8 Cage Recon Tarball Fetch"; flow:
established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/recon"; fast_pattern; con
tent:".tar.gz"; endswith; classtype:attempted-user; sid:9567002; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound CVE-2022-4262 No-Natives Script Fet
ch"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"cve-2022-4262";
nocase; fast_pattern; content:"no-natives"; nocase; classtype:attempted-user; sid:9567003; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound Staged V8 exploit-q Script Fetch";
flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/exploit-q.js"; endsw
ith; fast_pattern; classtype:attempted-user; sid:9567004; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R67 Inbound SimpleHTTP Browser Workbench Listin
g"; flow:established,to_client; http.stat_code; content:"200"; bsize:3; http.header; content:"Server: Sim
pleHTTP/"; fast_pattern; http.response_body; content:"<title>Directory listing for /"; content:"cve-2022-
4262"; nocase; classtype:web-application-activity; sid:9567005; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R67 Inbound V8 External Pointer Recon Script"; f
low:established,to_client; http.stat_code; content:"200"; bsize:3; http.response_body; content:"external_
pointer"; fast_pattern; classtype:attempted-user; sid:9567006; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R67 Inbound V8 Code Entry and TurboFan Recon Scr
ipt"; flow:established,to_client; http.stat_code; content:"200"; bsize:3; http.response_body; content:"co
de_entry_point"; fast_pattern; content:"TurboFan"; classtype:attempted-user; sid:9567007; rev:1;)
```

7.2 Microsoft Sentinel KQL Hunt

This 7-day Sentinel query checks `DeviceProcessEvents` for Chrome, Edge, or `d8` flags weakening renderer isolation (`--no-sandbox` , `--allow-natives-syntax` , `--disable-site-isolation-trials`) and Python `http.server` listeners on port 9000. It also checks `DeviceNetworkEvents` for browsers connecting to `Public` IPs on `RemotePort in (WorkbenchHttpPorts)` (9000). Replace the two placeholder hostnames in `ApprovedBrowserTestHosts` with your own QA runners before scheduling.

```kql
// Odin by Brandefense - Monograph 67
// Hunts Chromium/d8 sandbox-weakening flags, unapproved Python SimpleHTTP staging on TCP 9000, and brows
er egress to public TCP 9000 hosts.
let Lookback = 7d;
let ApprovedBrowserTestHosts = dynamic(["qa-runner-01.corp.example", "browser-ci-01.corp.example"]);
let WorkbenchHttpPorts = dynamic([9000]);
let UnsafeChromiumOrV8Cli =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBrowserTestHosts))
    | where FileName in~ ("chrome.exe", "chrome", "chromium", "msedge.exe", "msedge", "d8", "d8.exe")
    | where ProcessCommandLine contains "--no-sandbox" or ProcessCommandLine contains "--allow-natives-sy
ntax" or ProcessCommandLine contains "--disable-site-isolation-trials"
    | project Timestamp, DeviceName, AccountName, SignalType = "UnsafeBrowserSandboxOrV8Flag", Detail = P
rocessCommandLine;
let PythonSimpleHttpPort9000 =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBrowserTestHosts))
    | where FileName in~ ("python", "python3", "python.exe")
    | where ProcessCommandLine contains "http.server" and ProcessCommandLine contains "9000"
    | project Timestamp, DeviceName, AccountName, SignalType = "PythonSimpleHttpPort9000", Detail = Proce
ssCommandLine;
let BrowserHighPortPublicEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBrowserTestHosts))
    | where InitiatingProcessFileName in~ ("chrome.exe", "chrome", "chromium", "msedge.exe", "msedge")
    | where RemoteIPType == "Public" and RemotePort in (WorkbenchHttpPorts)
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "BrowserPub
licTcp9000Egress", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteIP, ":", tostring(RemotePor
t));
union UnsafeChromiumOrV8Cli, PythonSimpleHttpPort9000, BrowserHighPortPublicEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = make_s
et(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

7.3 Portable Sigma Process-Creation Rules (3 Rules)

Rule 1 (`T1203`) flags `/chrome` , `/chromium` , `/msedge` , or `/d8` launched with `--no-sandbox` , `--allow-natives-syntax` , or `--disable-site-isolation-trials` . Rule 2 (`T1608`) catches Python `http.server` bound to port **9000** (`[REF-09]`). Rule 3 (`T1203`) detects `python` , `python3` , `dd` , or `gdb` command lines containing `/proc/` and `/mem` , using exact `ComputerName` allowlists.

```yaml
title: Chromium or V8 Shell Launched With Disabled Sandbox or Natives Syntax
id: 67b2d4f1-9c12-4a83-8e25-67a19c83d201
status: experimental
description: Detects Chromium, Edge, or V8 d8 processes launched with flags that disable OS renderer sand
boxing, Site Isolation, or V8 native-syntax restrictions.
```

```
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_bin:
    Image|endswith:
      - '/chrome'
      - '/chromium'
      - '/msedge'
      - '/d8'
  selection_flags:
    CommandLine|contains:
      - '--no-sandbox'
      - '--allow-natives-syntax'
      - '--disable-site-isolation-trials'
  filter_qa_hosts:
    ComputerName:
      - 'qa-runner-01.corp.example'
      - 'browser-ci-01.corp.example'
  condition: selection_bin and selection_flags and not filter_qa_hosts
falsepositives:
  - Dedicated browser QA automation runners explicitly listed in the host allowlist
level: high
tags:
  - attack.execution
  - attack.t1203
---
title: Unauthenticated Python SimpleHTTP Directory Staging on TCP 9000
id: 67b2d4f1-9c12-4a83-8e25-67a19c83d202
status: experimental
description: Detects Python http.server directory listeners bound to port 9000 matching the exposed brows
er-exploitation workbench profile.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_py:
    Image|endswith:
      - '/python'
      - '/python3'
  selection_mod:
    CommandLine|contains: 'http.server'
  selection_port:
    CommandLine|contains:
      - 'http.server 9000'
      - '0.0.0.0 9000'
  filter_dev_hosts:
    ComputerName:
      - 'browser-ci-01.corp.example'
  condition: selection_py and selection_mod and selection_port and not filter_dev_hosts
falsepositives:
  - Approved local development servers running on isolated build hosts
level: medium
```

```
tags:
  - attack.resource-development
  - attack.t1608
---
title: Linux Process Inspecting Target Virtual Memory via Procfs Mem
id: 67b2d4f1-9c12-4a83-8e25-67a19c83d203
status: experimental
description: Detects command-line tools reading /proc/<pid>/mem directly to inspect live renderer cage ad
dresses as referenced in staged V8 recon5 test scripts.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_tool:
    Image|endswith:
      - '/python'
      - '/python3'
      - '/dd'
      - '/gdb'
  selection_proc:
    CommandLine|contains: '/proc/'
  selection_mem:
    CommandLine|contains: '/mem'
  filter_sec_lab:
    ComputerName:
      - 'sec-lab-01.corp.example'
  condition: selection_tool and selection_proc and selection_mem and not filter_sec_lab
falsepositives:
  - Authorized crash-triage or debugging workflows on designated lab hosts
level: high
tags:
  - attack.execution
  - attack.t1203
```

7.4 Structural YARA Rules (4 `filesize`-Bounded Rules)

Rules 1 and 2 (`filesize < 512KB`) match unpacked `CVE-2022-4262` `no-natives` scripts and in-cage `JSFunction` pointer scripts. Rules 3 and 4 match `TurboFan` `float64` constant-pool spray tests and V8 address-inspection harnesses (`addr_of` , `v8_read64` , and `v8_write64`).

```
YARA

/*
   Odin by Brandefense - Monograph 67 YARA Suite
   4 filesize-bounded structural rules covering CVE-2022-4262 no-natives reproducers,
   V8 in-cage JSFunction/Code pointer recon scripts, TurboFan float64 constant-pool spray tests,
   and V8 renderer address-inspection harnesses on disk.
*/

rule ODIN_V8_CVE_2022_4262_NoNatives_Reproducer
{
    meta:
```

```
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects unpacked CVE-2022-4262 standard-renderer V8 type-confusion test files"
        date = "2026-11-25"
    strings:
        $c1 = "CVE-2022-4262" ascii nocase
        $n1 = "exploit-no-natives" ascii nocase
        $n2 = "allow-natives-syntax" ascii nocase
    condition:
        filesize < 512KB and $c1 and ($n1 or $n2)
}

rule ODIN_V8_InCage_CodePointer_Recon_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects V8 pointer-compression cage reconnaissance scripts walking JSFunction and
Code object fields"
        date = "2026-11-25"
    strings:
        $s1 = "JSFunction" ascii
        $s2 = "code_entry_point" ascii
        $a1 = "external_pointer" ascii
        $a2 = "in-cage" ascii nocase
    condition:
        filesize < 512KB and $s1 and ($s2 or $a1) and $a2
}

rule ODIN_V8_TurboFan_Float64_JITSpray_Test
{
    meta:
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects V8 JIT-spray diagnostic scripts testing TurboFan float64 constant-pool spa
cing"
        date = "2026-11-25"
    strings:
        $t1 = "TurboFan" ascii
        $t2 = "float64" ascii
        $p1 = "constant pool" ascii nocase
        $p2 = "JIT-spray" ascii nocase
    condition:
        filesize < 512KB and $t1 and $t2 and ($p1 or $p2)
}

rule ODIN_V8_ProcMem_Renderer_Inspection_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects V8 address-inspection harnesses containing addr_of, v8_read64 and v8_write
64."
        date = "2026-11-25"
    strings:
        $v1 = "addr_of" ascii
        $v2 = "v8_read64" ascii
```

```
        $v3 = "v8_write64" ascii
    condition:
        filesize < 512KB and all of ($v*)
}
```

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

The Indicators tab holds **0** published IPv4 addresses and **0** file SHA-256 digests. The `TCP 9000` server IP, all **291** unique SHA-256 digests, and all **197** backup folder filenames are withheld. Defenders can hunt with the 10 protocol, CLI, and host markers below.

Marker category	Pattern / protocol artifact	Layer	Defensive hunt & control
`cveweb` archive URI	`/cveweb.tar.gz` (4,722 B)	Outbound HTTP `GET`	Alert at web proxies with `sid:9567001` and keep Chromium on current supported releases (`108.0.5359.94` fixed `CVE-2022-4262`).
Recon tarball URIs	`/recon*.tar.gz` (`recon.tar.gz` – `recon5.tar.gz`, 8,452 B)	Outbound HTTP `GET`	Alert with `sid:9567002` (`r5.tar.gz` and `r6.tar.gz` add 3,817 B on disk).
`no-natives` reproducer URI	`cve-2022-4262` paired with `no-natives` (11,875 B)	Outbound HTTP `GET`	Alert on visible HTTP with `sid:9567003` and YARA Rule 1.
Staged spray script URI	`/exploit-q.js` (8 copies × 9,996 B = 79,968 B)	Outbound HTTP `GET`	Alert with `sid:9567004` across root and `/k*`, `/nfb`, `/rs*` paths.
Workbench directory banner	`Server: SimpleHTTP/0.6 Python/3.12.3` + `cve-2022-4262`	Inbound HTTP 200	Alert with `sid:9567005` and block direct browser `TCP 9000` egress.
External pointer script body	`external_pointer` in the `recon.js` baseline (2,353 B)	Inbound HTTP 200	Alert on decrypted proxy traffic with `sid:9567006` and YARA Rule 2.
Code-entry & JIT script body	`code_entry_point` + `TurboFan`	Inbound HTTP 200	Alert with `sid:9567007`, YARA Rules 2–3, and JIT-less policy.

Marker category	Pattern / protocol artifact	Layer	Defensive hunt & control
Sandbox-weakening CLI flags	`--no-sandbox` / `--allow-natives-syntax` / `--disable-site-isolation-trials`	Browser / `d8` CLI	Hunt with Sentinel `UnsafeChromiumOrV8Cli` and Sigma Rule 1.
Ad-hoc Python HTTP staging	`python3 -m http.server 9000`	Endpoint process CLI	Hunt unapproved port-9000 staging servers with Sentinel KQL and Sigma Rule 2.
Renderer `/proc/mem` pause	`/proc/<pid>/mem` read + `recon5.js` (`addr_of`, `v8_read64`, `v8_write64`)	Linux host / disk	Enforce `yama.ptrace_scope >= 1`, hunt procfs reads with Sigma Rule 3, and scan scripts with YARA Rule 4.

9. Hardening Priorities for Enterprise Browser Fleets

Desktop engineering and security operations teams can apply five practical steps in order:

1. **Keep Chrome and other Chromium-based browsers on current supported security releases, with prompt relaunches (`0-48 Hours`).** Keep Chrome, Edge, kiosk images, headless browser containers, and Electron apps on current supported security releases; `108.0.5359.94` is the historical fix for `CVE-2022-4262` (`[REF-01]`, `[REF-02]`, `[REF-03]`). Set `RelaunchNotification` to Required and `RelaunchNotificationPeriod` to 48–72 hours; use `RelaunchWindow` only to schedule the restart time (`[REF-05]`, `[REF-10]`, `[REF-11]`).
2. **Apply V8 JIT-less policies (`DefaultJavaScriptJitSetting`) to high-risk users (`0-72 Hours`).** Block JavaScript JIT by default (`DefaultJavaScriptJitSetting = 2`) for executives, admins, finance staff, and SOC analysts. Use `JavaScriptJitAllowedForSites` to allowlist vetted internal apps that need WebAssembly or `TurboFan` speed (`[REF-03]`, `[REF-06]`).
3. **Lock `SitePerProcess` and alert on unsafe browser flags (`Days 1-7`).** Enforce `SitePerProcess = true` in browser policy so cross-site pages stay isolated behind V8 Sandbox (`EPT` and `CPT`) boundaries (`[REF-03]`, `[REF-04]`, `[REF-12]`). Schedule Sentinel KQL and Sigma Rule 1 to catch `--no-sandbox` or `--allow-natives-syntax` outside QA hosts (`[REF-08]`).
4. **Block direct workstation browser egress to `TCP 9000` and peer high ports (`Days 7-14`).** Stop user browsers from opening direct HTTP connections to uncategorized public servers on non-standard ports (`[REF-07]`, `[REF-09]`). Route web traffic through a TLS-inspecting forward proxy watched by Suricata `sid:9567001`–9567007.
5. **Lock Linux `yama.ptrace_scope` and audit `http.server` listeners (`Days 14-21`).** Verify `sysctl kernel.yama.ptrace_scope = 1` (or **2** on non-dev systems) so peer processes cannot read `/proc/<pid>/mem` (`recon5.tar.gz`). Use Sigma Rules 2 and 3 to catch unapproved `python3 -m http.server 9000` staging and `/proc/*/mem` reads.

10. Scope, Limits, and Sources

Odin by Brandefense studied open-directory listings and archive metadata on `TCP 9000` (`SimpleHTTP/0.6 Python/3.12.3`) of a China-hosted cloud server from `13 to 25 November 2026`. On `25 November 2026`, the open directory reported **325** active files and **10** directories (**325** sized files = **1,928,392** bytes, **0** unsized entries, and **291** unique SHA-256 digests).

All findings come from static file and archive metadata review; zero scripts were run against targets and zero exploit snippets or hex offsets are published. Staged `CVE-2022-4262` files (`cveweb.tar.gz`) and `recon*` scripts (`recon5.js` is written to pause for a local `/proc/mem` check) are consistent with a test rig. No live drive-by attack is established, and the reviewed directory contained no identified visitor logs or victim artifacts. The server IP, all **291** file SHA-256 digests, and all folder and file names inside the **197**-entry backup subdirectory (**1,473,170** bytes) are withheld.

1. `[REF-01]` NIST NVD, *CVE-2022-4262 Detail (V8 < 108.0.5359.94)* (verified on 25 November 2026).
2. `[REF-02]` Chrome Releases, *Desktop Update 108.0.5359.94/.95* (verified on 25 November 2026).
3. `[REF-03]` V8 Blog, *The V8 Sandbox: High-Level Design* (verified on 25 November 2026).
4. `[REF-04]` Chromium V8 Docs, *V8 Sandbox README and CPT Design* (verified on 25 November 2026).
5. `[REF-05]` Google Security, *An Update on Chrome Security Updates* (verified on 25 November 2026).
6. `[REF-06]` Chrome Enterprise, *DefaultJavaScriptJitSetting Policy* (verified on 25 November 2026).
7. `[REF-07]` MITRE ATT&CK, *Drive-by Compromise (T1189)* (verified on 25 November 2026).
8. `[REF-08]` MITRE ATT&CK, *Exploitation for Client Execution (T1203)* (verified on 25 November 2026).
9. `[REF-09]` MITRE ATT&CK, *Stage Capabilities (T1608)* (verified on 25 November 2026).
10. `[REF-10]` Chrome Enterprise, *RelaunchNotificationPeriod Policy* (verified on 25 November 2026).
11. `[REF-11]` Chrome Enterprise, *RelaunchWindow Policy* (verified on 25 November 2026).
12. `[REF-12]` Chrome Enterprise, *SitePerProcess Policy* (verified on 25 November 2026).

BRANDEFENSE CAPABILITIES RELATED TO THIS RESEARCH

Cyber Threat Intelligence

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

External Attack Surface

See what reconnaissance tools see first.

Threat Actor Landscape

Profiles of the groups that run recon at scale.

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_chrome_v8_wasm_cage_workbench.yar** 69 lines

```
/*
   Odin by Brandefense - Monograph 67 YARA Suite
   4 filesize-bounded structural rules covering CVE-2022-4262 no-natives reproducers,
   V8 in-cage JSFunction/Code pointer recon scripts, TurboFan float64 constant-pool spray tests,
   and V8 renderer address-inspection harnesses on disk.
*/

rule ODIN_V8_CVE_2022_4262_NoNatives_Reproducer
{
    meta:
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects unpacked CVE-2022-4262 standard-renderer V8 type-confusion test files"
        date = "2026-11-25"
    strings:
        $c1 = "CVE-2022-4262" ascii nocase
        $n1 = "exploit-no-natives" ascii nocase
        $n2 = "allow-natives-syntax" ascii nocase
    condition:
        filesize < 512KB and $c1 and ($n1 or $n2)
}

rule ODIN_V8_InCage_CodePointer_Recon_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects V8 pointer-compression cage reconnaissance scripts walking JSFunction and
Code object fields"
        date = "2026-11-25"
    strings:
        $s1 = "JSFunction" ascii
        $s2 = "code_entry_point" ascii
        $a1 = "external_pointer" ascii
        $a2 = "in-cage" ascii nocase
    condition:
        filesize < 512KB and $s1 and ($s2 or $a1) and $a2
}

rule ODIN_V8_TurboFan_Float64_JITSpray_Test
{
    meta:
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects V8 JIT-spray diagnostic scripts testing TurboFan float64 constant-pool spa
cing"
        date = "2026-11-25"
    strings:
```

```
        $t1 = "TurboFan" ascii
        $t2 = "float64" ascii
        $p1 = "constant pool" ascii nocase
        $p2 = "JIT-spray" ascii nocase
    condition:
        filesize < 512KB and $t1 and $t2 and ($p1 or $p2)
}

rule ODIN_V8_ProcMem_Renderer_Inspection_Script
{
    meta:
        author = "Odin by Brandefense"
        monograph = "67"
        description = "Detects V8 address-inspection harnesses containing addr_of, v8_read64 and v8_write
64."
        date = "2026-11-25"
    strings:
        $v1 = "addr_of" ascii
        $v2 = "v8_read64" ascii
        $v3 = "v8_write64" ascii
    condition:
        filesize < 512KB and all of ($v*)
}
```

Suricata `odin_chrome_v8_wasm_workbench_hunt.rules` 11 lines

```
# Odin by Brandefense - Monograph 67 Suricata 7.x Ruleset
# Syntax-checked with Odin Suricata linter (0 problems); test with suricata -T prior to deployment.

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound CVE-2022-4262 cveweb Archive Fetc
h"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/cveweb.tar.gz";
endswith; fast_pattern; classtype:attempted-user; sid:9567001; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound V8 Cage Recon Tarball Fetch"; flow:
established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/recon"; fast_pattern; con
tent:".tar.gz"; endswith; classtype:attempted-user; sid:9567002; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound CVE-2022-4262 No-Natives Script Fet
ch"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"cve-2022-4262";
nocase; fast_pattern; content:"no-natives"; nocase; classtype:attempted-user; sid:9567003; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R67 Outbound Staged V8 exploit-q Script Fetch";
flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/exploit-q.js"; endsw
ith; fast_pattern; classtype:attempted-user; sid:9567004; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R67 Inbound SimpleHTTP Browser Workbench Listin
g"; flow:established,to_client; http.stat_code; content:"200"; bsize:3; http.header; content:"Server: Sim
pleHTTP/"; fast_pattern; http.response_body; content:"<title>Directory listing for /"; content:"cve-2022-
4262"; nocase; classtype:web-application-activity; sid:9567005; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R67 Inbound V8 External Pointer Recon Script"; f
low:established,to_client; http.stat_code; content:"200"; bsize:3; http.response_body; content:"external_
pointer"; fast_pattern; classtype:attempted-user; sid:9567006; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R67 Inbound V8 Code Entry and TurboFan Recon Scr
ipt"; flow:established,to_client; http.stat_code; content:"200"; bsize:3; http.response_body; content:"co
de_entry_point"; fast_pattern; content:"TurboFan"; classtype:attempted-user; sid:9567007; rev:1;)
```

Show full rule

Sentinel KQL **sentinel_chrome_v8_sandbox_and_workbench_hunt.kql** 30 lines

```
// Odin by Brandefense - Monograph 67
// Hunts Chromium/d8 sandbox-weakening flags, unapproved Python SimpleHTTP staging on TCP 9000, and brows
er egress to public TCP 9000 hosts.
let Lookback = 7d;
let ApprovedBrowserTestHosts = dynamic(["qa-runner-01.corp.example", "browser-ci-01.corp.example"]);
let WorkbenchHttpPorts = dynamic([9000]);
let UnsafeChromiumOrV8Cli =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBrowserTestHosts))
    | where FileName in~ ("chrome.exe", "chrome", "chromium", "msedge.exe", "msedge", "d8", "d8.exe")
    | where ProcessCommandLine contains "--no-sandbox" or ProcessCommandLine contains "--allow-natives-sy
ntax" or ProcessCommandLine contains "--disable-site-isolation-trials"
    | project Timestamp, DeviceName, AccountName, SignalType = "UnsafeBrowserSandboxOrV8Flag", Detail = P
rocessCommandLine;
let PythonSimpleHttpPort9000 =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBrowserTestHosts))
    | where FileName in~ ("python", "python3", "python.exe")
    | where ProcessCommandLine contains "http.server" and ProcessCommandLine contains "9000"
    | project Timestamp, DeviceName, AccountName, SignalType = "PythonSimpleHttpPort9000", Detail = Proce
ssCommandLine;
let BrowserHighPortPublicEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBrowserTestHosts))
    | where InitiatingProcessFileName in~ ("chrome.exe", "chrome", "chromium", "msedge.exe", "msedge")
    | where RemoteIPType == "Public" and RemotePort in (WorkbenchHttpPorts)
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "BrowserPub
licTcp9000Egress", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteIP, ":", tostring(RemotePor
t));
union UnsafeChromiumOrV8Cli, PythonSimpleHttpPort9000, BrowserHighPortPublicEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = make_s
et(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

Sigma **sigma_chromium_v8_flags_and_workbench_staging.yml** 95 lines

```
title: Chromium or V8 Shell Launched With Disabled Sandbox or Natives Syntax
id: 67b2d4f1-9c12-4a83-8e25-67a19c83d201
status: experimental
description: Detects Chromium, Edge, or V8 d8 processes launched with flags that disable OS renderer sand
boxing, Site Isolation, or V8 native-syntax restrictions.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_bin:
    Image|endswith:
```

```
      - '/chrome'
      - '/chromium'
      - '/msedge'
      - '/d8'
  selection_flags:
    CommandLine|contains:
      - '--no-sandbox'
      - '--allow-natives-syntax'
      - '--disable-site-isolation-trials'
  filter_qa_hosts:
    ComputerName:
      - 'qa-runner-01.corp.example'
      - 'browser-ci-01.corp.example'
  condition: selection_bin and selection_flags and not filter_qa_hosts
falsepositives:
  - Dedicated browser QA automation runners explicitly listed in the host allowlist
level: high
tags:
  - attack.execution
  - attack.t1203
---
title: Unauthenticated Python SimpleHTTP Directory Staging on TCP 9000
id: 67b2d4f1-9c12-4a83-8e25-67a19c83d202
status: experimental
description: Detects Python http.server directory listeners bound to port 9000 matching the exposed brows
er-exploitation workbench profile.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_py:
    Image|endswith:
      - '/python'
      - '/python3'
  selection_mod:
    CommandLine|contains: 'http.server'
  selection_port:
    CommandLine|contains:
      - 'http.server 9000'
      - '0.0.0.0 9000'
  filter_dev_hosts:
    ComputerName:
      - 'browser-ci-01.corp.example'
  condition: selection_py and selection_mod and selection_port and not filter_dev_hosts
falsepositives:
  - Approved local development servers running on isolated build hosts
level: medium
tags:
  - attack.resource-development
  - attack.t1608
---
title: Linux Process Inspecting Target Virtual Memory via Procfs Mem
id: 67b2d4f1-9c12-4a83-8e25-67a19c83d203
status: experimental
description: Detects command-line tools reading /proc/<pid>/mem directly to inspect live renderer cage ad
```

```
dresses as referenced in staged V8 recon5 test scripts.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_tool:
    Image|endswith:
      - '/python'
      - '/python3'
      - '/dd'
      - '/gdb'
  selection_proc:
    CommandLine|contains: '/proc/'
  selection_mem:
    CommandLine|contains: '/mem'
  filter_sec_lab:
    ComputerName:
      - 'sec-lab-01.corp.example'
  condition: selection_tool and selection_proc and selection_mem and not filter_sec_lab
falsepositives:
  - Authorized crash-triage or debugging workflows on designated lab hosts
level: high
tags:
  - attack.execution
  - attack.t1203
```

