FULL REPORT · DEVELOPER TOOLING · MONOGRAPH 65 / 62 · 20 MIN READ

Bare Git Repositories and Python Install Hooks: Candidate VShell Ports

Defensive analysis of 328 staged entries across two independent China cloud servers exposing dumb-HTTP bare Git repositories, Python install wrappers, and candidate VShell ports

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 328 staged directory entries across two independent China cloud servers recorded on 13–25 November 2026. Nineteen shared SHA-256 digests are standard Git initialization files, and zero custom script hashes link the servers. Both pair bare Git repositories with Python install wrappers and candidate VShell ports. Enforce wheel-only pip installs, patch GitPython, and block high-port egress.

Developer workstations, build runners, and plugin managers fetch Git repositories and Python packages every day. When an open web directory serves a bare Git repository, a supply chain attack can ride standard Git clones without login checks. Once cloned, source build steps or package install scripts can pull second-stage code over high TCP ports.

Between 13 and 25 November 2026, Odin by Brandefense analyzed open Python HTTP directories on two cloud servers in China. Together, the two servers hold 328 deduplicated entries across 326 sized files (35,738,208 bytes, or 35.74 megabytes) and 2 unsized entries. Those files resolve to 76 distinct SHA-256 digests across both servers.

The 19 shared SHA-256 digests are standard Git initialization files (`hooks/*.sample` , `description` , `HEAD` , `exclude` , `packs` , and empty blob). Zero custom operator script hashes link Workbench A and Workbench B. Each host is therefore assessed consistently as an independent adversary staging server. Odin by Brandefense withholds both server IPv4 addresses, one external command host address, and all file SHA-256 digests.

1. **Workbench A pairs an evil-node Python tree with a bare Git repo and a TCP 9999 route.**
   Recorded on 13–25 November 2026 on TCP port 8888, Workbench A holds 50 entries totaling 53,865 sized bytes. Its working folder stages a 191-byte Python install script that points to a second-stage `/slw` path on TCP port 9999. The same server exposes a candidate VShell listener on TCP port 9999 and nginx returning HTTP 500 on TCP port 80.

2. **Workbench B pairs five bare Git repos with five install wrappers and three root Python stage loaders.**
   Its TCP 8088 directory inventory (13–20 November 2026) holds 278 deduplicated entries totaling 35,684,343 sized bytes. Five working folders and five bare Git repositories sit beside three root Python loaders, a 177-byte command file, and a 35.41-megabyte HTTP log. Separate service observations on 23–24 November recorded candidate VShell listeners on sibling TCP ports 8084 and 8085.

3 **Dumb-HTTP Git manifests let static Python servers host cloneable Git repositories.**
All six bare repositories across both servers carry a 59-byte `info/refs` file and a 105-byte `packed-refs` file. None of the six working trees hold those two files. Running `git update-server-info` generates `info/refs` and `objects/info/packs`, while `git pack-refs` generates `packed-refs` for static HTTP clones.

4 **Unlinked Git staging hosts call for build, Git-client, and egress rules rather than IP blocklists.**
Because zero custom script hashes link Workbench A and Workbench B, static IP lists age out fast. Defenders should enforce wheel-only `pip` installs, upgrade `GitPython` to 3.1.30 or newer, audit Git filter rules, and restrict high-port egress.

Three Decisions for Engineering and Security Leaders

1. **Enforce binary wheel-only, hash-checked Python installs in CI and server builds.** Pass `--only-binary=:all:` and `--require-hashes` to `pip` so automated builds reject raw Git source checkouts and untrusted setup scripts.
2. **Upgrade GitPython and lock down Git client protocol and filter settings.** Patch `GitPython` to version 3.1.30 or newer (`CVE-2022-24439`), restrict `git config protocol.allow`, and audit `clean`, `smudge`, and `fsmonitor` filters.
3. **Block direct outbound connections to raw IPv4 hosts and high TCP ports.** Force developer and build-runner web traffic through proxies that block raw-IP Git clones and high ports (**8084**, **8085**, **8088**, **8888**, and **9999**).

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

Odin by Brandefense recorded the two open Python HTTP directories between `13 and 25 November 2026`. Workbench A's `TCP 8888` directory was recorded on `13–25 November 2026`. Workbench B's `TCP 8088` directory inventory covers `13–20 November 2026`. Separate service observations on `23–24 November 2026` recorded candidate `VShell` ports `TCP 8084` and `TCP 8085`.

Both hosts are assessed consistently as **independent adversary staging servers**. They expose offensive repository names (`/evil-node`, `/evil-node-bare`, and `/exp1.git` through `/exp5.git`), high-port Python callback loaders, and co-hosted candidate `VShell` ports.

Comparing **31** unique digests on Workbench A and **64** on Workbench B (`31 + 64 = 95`) yields **76** distinct digests overall. **The 19 shared SHA-256 digests are standard Git initialization files (`hooks/*.sample`, `description`, `HEAD`, `exclude`, `packs`, and empty blob), and 0 custom operator hashes link both hosts.** Cloud autonomous systems (`AS37963` vs `AS45090`), port layouts, and loader scripts also differ across the two hosts. Figure 1 maps both independent workbenches and their withholding boundaries.

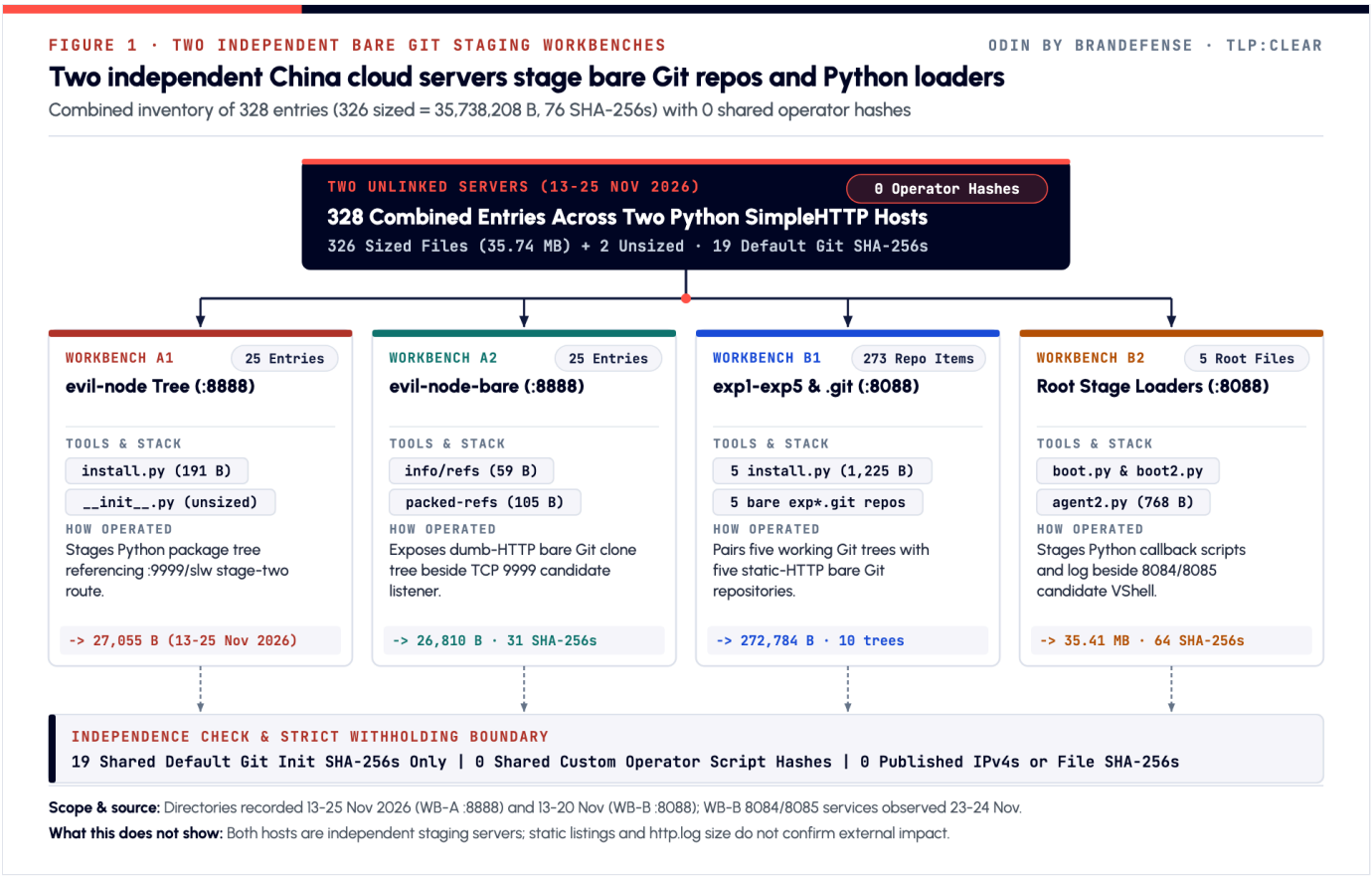

Figure 1: Two independent China cloud servers stage bare Git repos and Python loaders

Dual-Workbench Partition and Snapshot Reconciliation (`n = 328`)

Across both servers, the deduplicated inventory totals **328 directory entries**. Three hundred twenty-six sized files sum to **35,738,208 bytes (35.74 MB)**, while **2** entries carry no byte size in listings.

Defenders can use external attack surface management to find open `SimpleHTTP` repository directories exposed on cloud instances. Pairing those checks with cyber threat intelligence helps detection teams turn staged Git and Python loader patterns into network and host rules. Table 1 reconciles all **328** entries across Workbenches A and B.

Workbench & hosting profile	Top-level buckets	Total entries	Share (%)	Sized files	Unsized entries	Sized bytes	Unique SHA-256 (withheld)	Observed adversary staging focus
Workbench A `( China Cloud · TCP 8888 )`	2 (`evil-node` , `evil-node-bare`)	50	15.24 %	49	1	53,865	31	Recorded `13-25 Nov` (26 dirs); stages `install.py` (191 B) -> `:9999/slw` beside `TCP 9999` candidate `VShell` .
Workbench B `( China Cloud · TCP 8088 )`	11 (`root` , 5 `exp*` , 5 `exp*.git`)	278	84.76 %	277	1	35,684,343	64	Directory inventory `13-20 Nov` (255 live + 23 objects); stages 5 `install.py` , `boot` / `boot2` / `agent2` , `cmd` , and `http.log` ; candidate `VShell` :8084/:8085 observed `23-24 Nov` .

Workbench & hosting profile	Top-level buckets	Total entries	Share (%)	Sized files	Unsized entries	Sized bytes	Unique SHA-256 (withheld)	Observed adversary staging focus
Combined study inventory	13 buckets	328	100.00%	326	2	35,738,208	76 (19 Git init; 0 custom)	Two independent Git-hosted Python loader workbenches (35.74 MB).

Two snapshot details explain how the live directory counts reconcile to **328** entries:

- **Workbench A (50 live files = 50 entries across 26 directories).** On `25 November 2026`, `SimpleHTTP/0.6 Python/3.12.3` on `TCP 8888` exposed **25** sized files in `/evil-node-bare/` (**26,810** B). It also exposed **25** entries in `/evil-node/` (**24** sized = **27,055** B plus **1** unsized `/evil-node/__init__.py`).
- **Workbench B (255 live `20 November` files + 23 earlier Git objects = 278 entries).** On `20 November 2026`, `SimpleHTTP/0.6 Python/3.12.3` on `TCP 8088` listed **255** live files (**254** sized = **35,681,690** B and **1** unsized `/exp3/README.md`) across **133** directories. Adding **23** loose `.git/objects/*` files (**2,653** B) recorded on `13 November 2026` yields **278** deduplicated entries (**35,684,343** B).

Combined File-Class and Five-Cluster Functional Breakdown

Table 2 reconciles the same **328** entries (**326** sized files, **2** unsized entries, **35,738,208** bytes) by file class. Script entries lead by file count (**178** entries, **54.27%**) across both servers. Git creates **14** `.sample` hook templates inside all **12** repositories (**168** `.sample` files = **309,852** B, plus **10** `.py` files = **4,799** B). Text files and logs (**102** entries, **31.10%**) hold **99.10%** of total bytes due to Workbench B's **35.41 MB** `/http.log`.

Artifact class	Total entries	Share (%)	Sized files	Unsized entries	Sized bytes	Cross-workbench composition
Scripts & hook templates (`script`)	178	54.27%	177	1	314,651	30 WB-A (28 `.sample` = 51,642 B, `install.py` 191 B, unsized `__init__.py`) and 148 WB-B (140 `.sample` = 258,210 B, 8 `.py` = 4,608 B).
Git refs, config & logs (`text`)	102	31.10%	101	1	35,417,602	16 WB-A (1,586 B) and 86 WB-B (83 Git/`cmd` files, 2 `README.md` [1 unsized], and `/http.log` 35,407,999 B).
Binary index & objects (`other`)	48	14.63%	48	0	5,955	4 WB-A (1 `.git/index` 225 B + 3 binary Git objects 221 B) and 44 WB-B (5 `.git/index` files 869 B + 39 binary Git objects 4,640 B).
Total	**328**	**100.00%**	**326**	**2**	**35,738,208**	**Exact parity with Table 1 (328 entries = 35.74 MB).**

Table 3 and Figure 5 divide all **328** entries into five mutually exclusive functional clusters (`168 + 93 + 56 + 10 + 1 = 328`). Three Git repository plumbing groups account for **317** entries (**96.65%**). Those Git files surround **10** staged Python install and callback scripts (**3.05%**, **4,799** B) and **1** HTTP access log (**0.30%**, **35,407,999** B).

Functional cluster	Total entries	Share (%)	WB-A	WB-B	Sized / Unsized	Sized bytes	Key staged files & structural role
Git Hook Templates (`/hooks/*.sample`)	168	51.22%	28	140	168 / 0	309,852	14 default `.sample` hook files across 2 WB-A and 10 WB-B repositories (25,821 B per repo).
Git Refs, Config, Index & Metadata	93	28.35%	15	78	92 / 1	9,872	6 bare `info/refs` (59 B each), 6 `packed-refs` (105 B each), 6 `.git/index` , `HEAD` , `config` , 2 `README.md` , and WB-B `/cmd` (177 B).
Compressed Git Objects (`/objects/*`)	56	17.07%	5	51	56 / 0	5,686	Compressed commit, tree, and blob objects plus `objects/info/packs` (375 B on WB-A; 5,311 B on WB-B).
Python Install Wrappers & Agents	10	3.05%	2	8	9 / 1	4,799	WB-A `install.py` (191 B) + unsized `__init__.py` ; WB-B 5 `install.py` (1,225 B) + `agent2.py` (768 B), `boot.py` (1,248 B), `boot2.py` (1,367 B).
HTTP Directory Access Log (`http.log`)	1	0.30%	0	1	1 / 0	35,407,999	WB-B root `/http.log` (35.41 MB), forming 99.08% of combined sized bytes.
Combined inventory (99.99% rounded)	**328**	**100.00 %**	**50**	**278**	**326 / 2**	**35,738,208**	19 shared default Git SHA-256s; 0 shared custom operator SHA-256s.

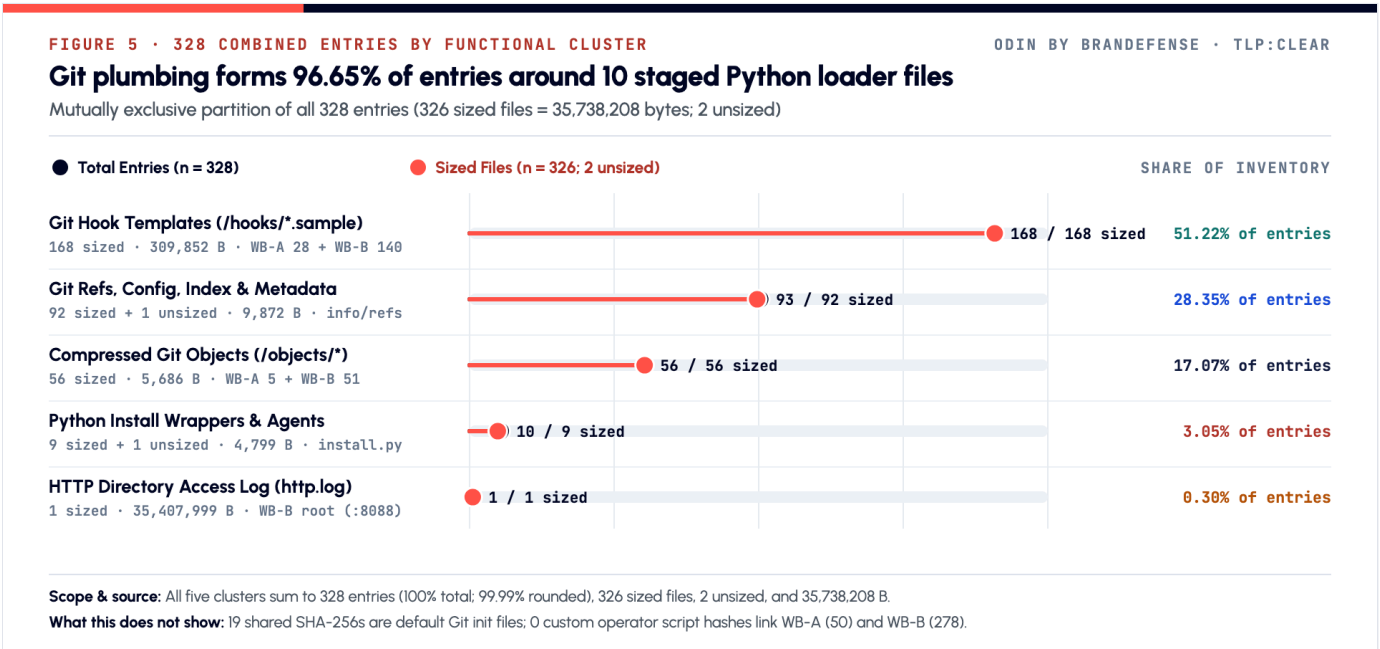

Figure 5: Git plumbing forms 96.65% of entries around 10 staged Python loader files

3. Dumb-HTTP Bare Git Staging and Python Install Wrappers (`Workbench A`)

Both Workbench A and Workbench B solve a practical staging task for an attacker. They serve cloneable Git repositories from a single `python3 -m http.server` command without setting up CGI, Nginx, or SSH keys. Figure 3 presents a dashed analytical model showing how bare Git `info/refs` fetches, client checkouts, `install.py` scripts, and high-port routes meet defender controls.

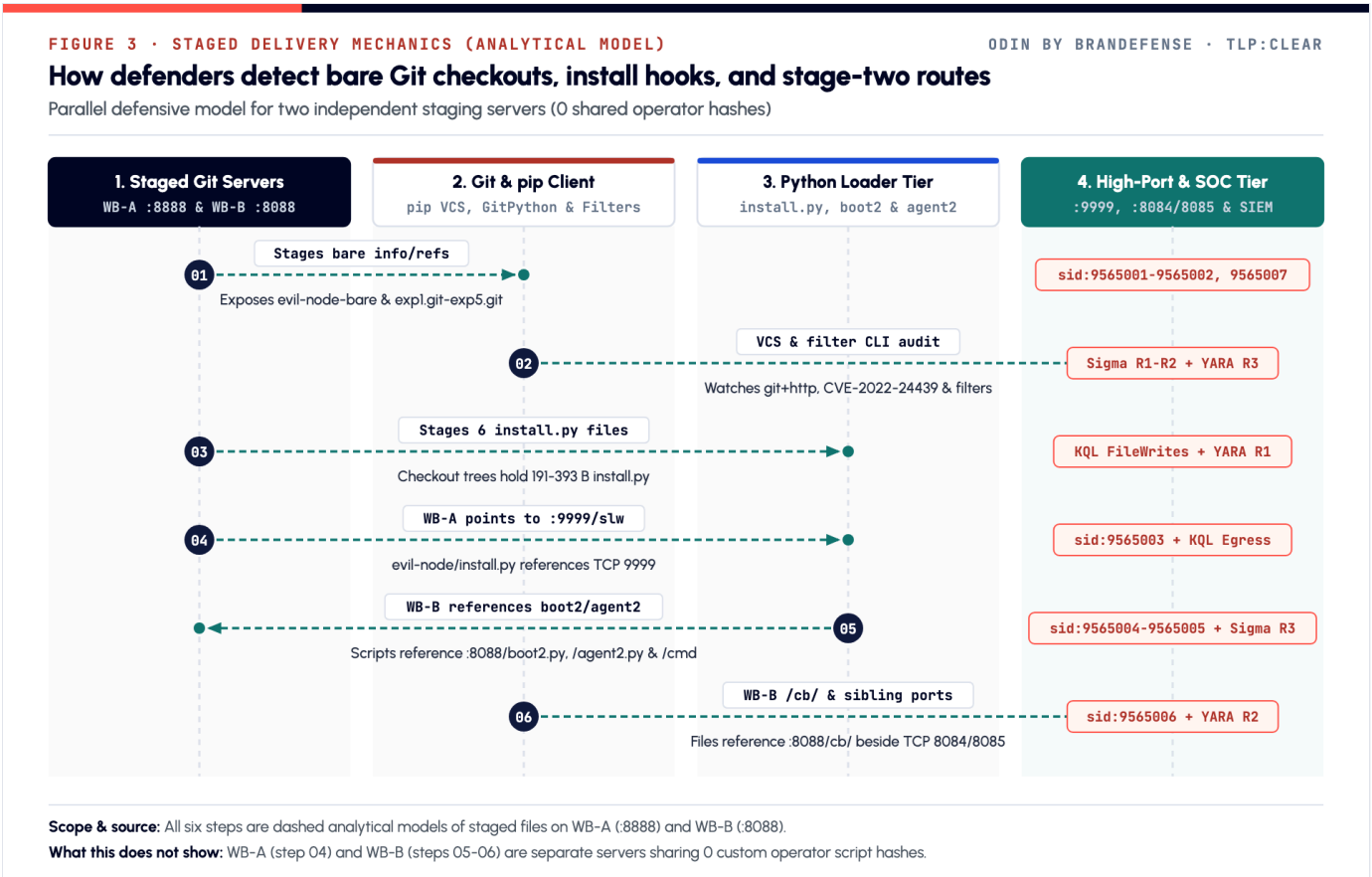

Figure 3: How defenders detect bare Git checkouts, install hooks, and stage-two routes

Workbench A Layout: `/evil-node` and `/evil-node-bare` (50 Entries)

Recorded on `13-25 November 2026` on a China cloud server (`AS37963` , IPv4 withheld), Workbench A runs `SimpleHTTP/0.6 Python/3.12.3` on `TCP 8888` . It holds **50 deduplicated entries (49 sized = 53,865 bytes and 1 unsized; 31 unique SHA-256s)** across two paired folders:

Workbench A directory	Entries	Sized / Unsized	Sized bytes	Key staged files & Git metadata
`/evil-node/` (`working`)	25	24 / 1	27,055	`install.py` (191 B), unsized `__init__.py` , 14 `.sample` hooks (25,821 B), and 9 `.git/` metadata files (1,043 B).
`/evil-node-bare/` (`bare`)	25	25 / 0	26,810	`info/refs` (59 B), `packed-refs` (105 B), 14 `.sample` hooks (25,821 B), 5 `/objects/*` (375 B), and 4 bare metadata files (450 B).
Workbench A Total	50	49 / 1	53,865	31 unique SHA-256s; `install.py` references `:9999/slw` .

How Dumb-HTTP `info/refs` , `pip` VCS Checkouts, and `CVE-2022-24439` Work

Comparing `/evil-node/.git/` against `/evil-node-bare/` highlights four key mechanisms for Python and CI/CD defenders (`[REF-01]` – `[REF-07]`):

1. **Dumb-HTTP Git cloning via `git update-server-info`** . Python's `http.server` (`SimpleHTTP/0.6`) serves static files only and cannot run Git's Smart HTTP `git-upload-pack` handler. Running `git update-server-info` inside a `--bare` repo writes `info/refs` (59 B) and `objects/info/packs` (1 B). When a client clones `/evil-node-bare` over HTTP, Git falls back to its static walker, reads `HEAD` and `info/refs` , and pulls `/objects/*` .

2. **Plugin node installers and `pip` VCS checkouts (`[REF-01]` , `[REF-03]` , `[REF-04]`).** Pairing `__init__.py` with `install.py` inside `/evil-node/` matches Python plugin and AI custom-node managers (`T1195.001` , `T1059.006` , `[REF-08]` , `[REF-09]`). A `pip install` over `git+http` can clone the tree and invoke PEP 517 or `setup.py` hooks. `--require-hashes` rejects VCS URLs; `--only-binary=:all:` alone does not block cloning (`[REF-01]` , `[REF-02]`). In the live **20** and `25 November` trees, `install.py` sits directly in the working roots without a plain `setup.py` or `pyproject.toml` . Compressed commits reside inside `/objects/*` .
3. **Git `clean` / `smudge` filter rules and `GitPython` `CVE-2022-24439` (`[REF-05]` , `[REF-06]` , `[REF-07]`).** Python tools using `GitPython` **prior to** `3.1.30` are vulnerable to `CVE-2022-24439` (`CVSS 9.8` Critical, December 2022, `[REF-05]`). Unvalidated remote URLs let attackers inject `git clone` flags such as `--upload-pack=` or `-c` config overrides. All **168** `/hooks/*.sample` files are inactive, and `git clone` never copies remote server hooks (`[REF-07]`). A tracked `.gitattributes` file (`[REF-06]`) only sets a `filter=<name>` label. Running shell code on checkout requires `filter.<name>.clean` , `filter.<name>.smudge` , or `core.fsmonitor` in Git config (`[REF-07]`).
4. **Sibling `TCP 9999` stage-two path (`:9999/slw`).** Inside `/evil-node/install.py` (**191** B), the script references `:9999/slw` (`T1571` , `[REF-10]`). Workbench A co-hosts a **candidate `VShell` listener on `TCP 9999`** beside `TCP 80` `nginx` (`HTTP 500`). Because `TCP 9999` is a candidate record without a live C2 beacon capture, Workbench A's IPv4 is withheld while `sid:9565003` covers `:9999/slw` .

4. Multi-Repository Python Stage Loaders and Callback Routes (`Workbench B`)

On a second China cloud server (`AS45090` , IPv4 withheld), Workbench B's `TCP 8088` directory inventory (`13-20 November 2026`) runs `SimpleHTTP/0.6 Python/3.12.3` . Separate service observations on `23-24 November 2026` recorded sibling candidate `VShell` listeners on `TCP 8084` and `TCP 8085` . Across `TCP 8088` , Workbench B holds **278 deduplicated entries (277 sized = 35,684,343 bytes and 1 unsized; 64 unique SHA-256s)**. Figure 4 compares the port and directory layouts across both servers.

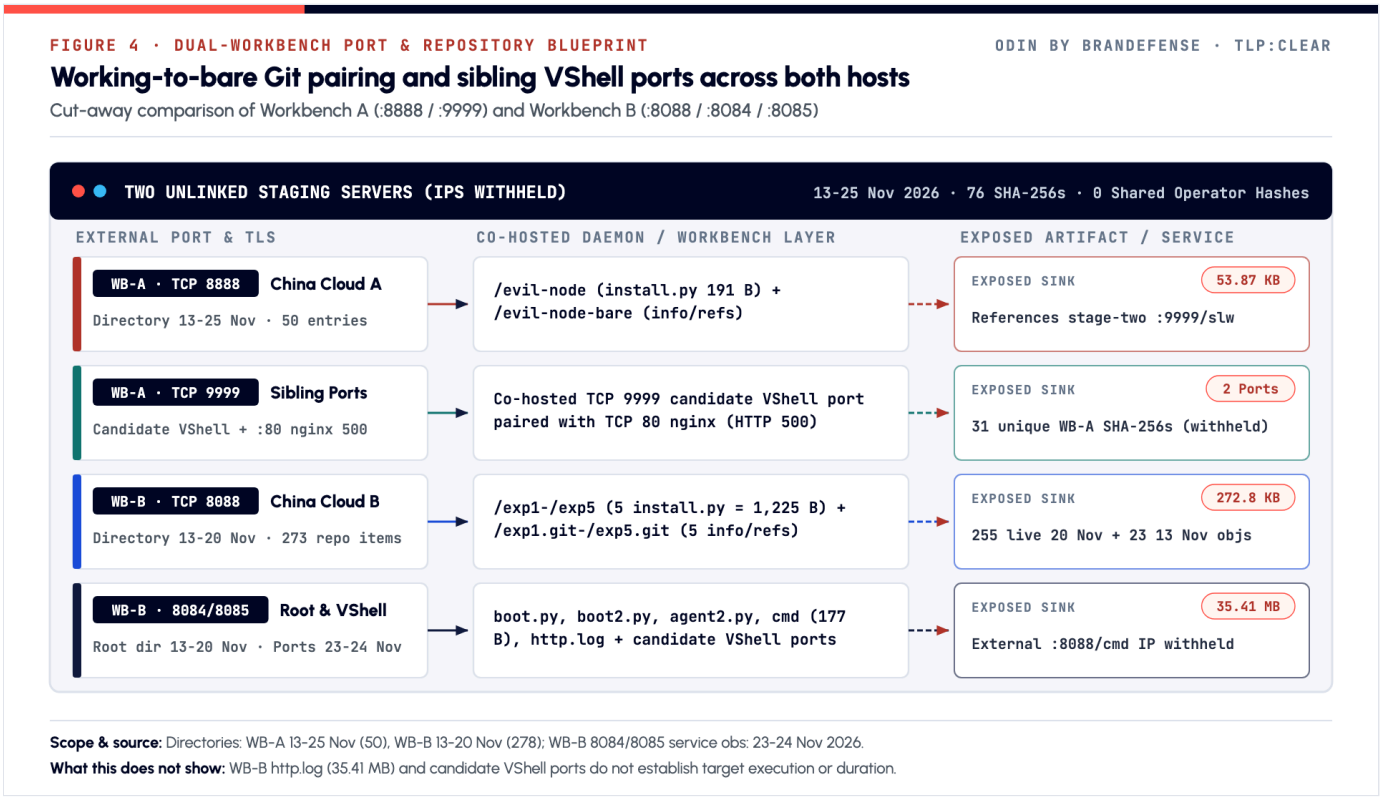

Figure 4: Working-to-bare Git pairing and sibling VShell ports across both hosts

Ten-Repository Matrix (`/exp1` – `/exp5` and `/exp1.git` – `/exp5.git`) and Root Loaders

Table 4 reconciles all **11** top-level directory buckets on Workbench B (**278** entries = **277** sized files totaling **35,684,343** bytes + **1** unsized entry). Notice two clean repo templates: `/exp1` and `/exp3` (**32** working + **28** bare entries each), versus `/exp2`, `/exp4`, and `/exp5` (**27** working + **24** bare entries each).

Workbench B directory bucket	Total entries	Sized / Unsized	Sized bytes	Key staged files & repository role
`/exp1 ( working tree )`	32	32 / 0	28,036	`/exp1/install.py` (208 B), `/exp1/README.md` (7 B), 14 `.sample` hooks (25,821 B), 7 `13 Nov` objects (815 B), and 9 `.git/` files (1,185 B).
`/exp3 ( working tree )`	32	31 / 1	28,226	`/exp3/install.py` (393 B), unsized `/exp3/README.md`, 14 `.sample` hooks (25,821 B), 7 `13 Nov` objects (830 B), and 9 `.git/` files (1,182 B).
`/exp1.git ( bare repo )`	28	28 / 0	27,242	`info/refs` (59 B), `packed-refs` (105 B), 14 `.sample` hooks (25,821 B), 8 `/objects/*` files (816 B), and 4 bare metadata files (441 B).
`/exp3.git ( bare repo )`	28	28 / 0	27,259	`info/refs` (59 B), `packed-refs` (105 B), 14 `.sample` hooks (25,821 B), 8 `/objects/*` files (831 B), and 4 bare metadata files (443 B).
`/exp2`, `/exp4`, `/exp5` `( 3 working )`	81 (27 × 3)	81 / 0	81,732 (27,244 × 3)	Each holds `install.py` (208 B), 14 `.sample` hooks (25,821 B), 3 `13 Nov` objects (336 B), and 9 `.git/` files (879 B).
`/exp2.git`, `/exp4.git`, `/exp5.git ( 3 bare )`	72 (24 × 3)	72 / 0	80,289 (26,763 × 3)	Each holds `info/refs` (59 B), `packed-refs` (105 B), 14 `.sample` hooks (25,821 B), 4 `/objects/*` files (337 B), and 4 bare metadata files (441 B).
HTTP root (`/` , 5 files)	5	5 / 0	35,411,559	`/boot.py` (1,248 B), `/boot2.py` (1,367 B), `/agent2.py` (768 B), `/cmd` (177 B), and `/http.log` (35,407,999 B).
Workbench B Total	**278**	**277 / 1**	**35,684,343**	**64 unique SHA-256s (255 live `20 Nov` files + 23 `13 Nov` loose objects).**

Four practical takeaways stand out on Workbench B:

- **Small repo wrappers (208 B and 393 B) paired with root loaders.** Four folders (`/exp1`, `/exp2`, `/exp4`, `/exp5`) stage **208**-byte `install.py` scripts, while `/exp3/install.py` grows to **393** B. Hosting `/boot.py` (**1,248** B), `/boot2.py` (**1,367** B), `/agent2.py` (**768** B), and `/cmd` (**177** B) in root lets operators update stage-two code cleanly (`T1059.006`, `[REF-09]`).
- **Local and external :8088 callback paths.** Workbench B scripts reference `:8088/boot2.py`, `:8088/agent2.py`, and `:8088/cb/` alongside an external `:8088/cmd` host (IPv4 withheld).
- **Sibling candidate `VShell` ports (`TCP 8084` and `TCP 8085`).** Separate service observations recorded candidate `VShell` listeners on `TCP 8084` and `TCP 8085` on `23-24 November 2026`; the directory inventory covers `13-20 November` (`T1571`, `[REF-10]`).
- **Reading `/http.log` (35.41 MB) carefully.** Workbench B's **35,407,999**-byte `/http.log` forms **99.08%** of all bytes in this study. Open cloud ports log constant internet scanner traffic and local tests, so file size never establishes victim compromises or campaign length.

Figure 2 maps all five functional clusters across Workbenches A and B to the detections in Chapter 7.

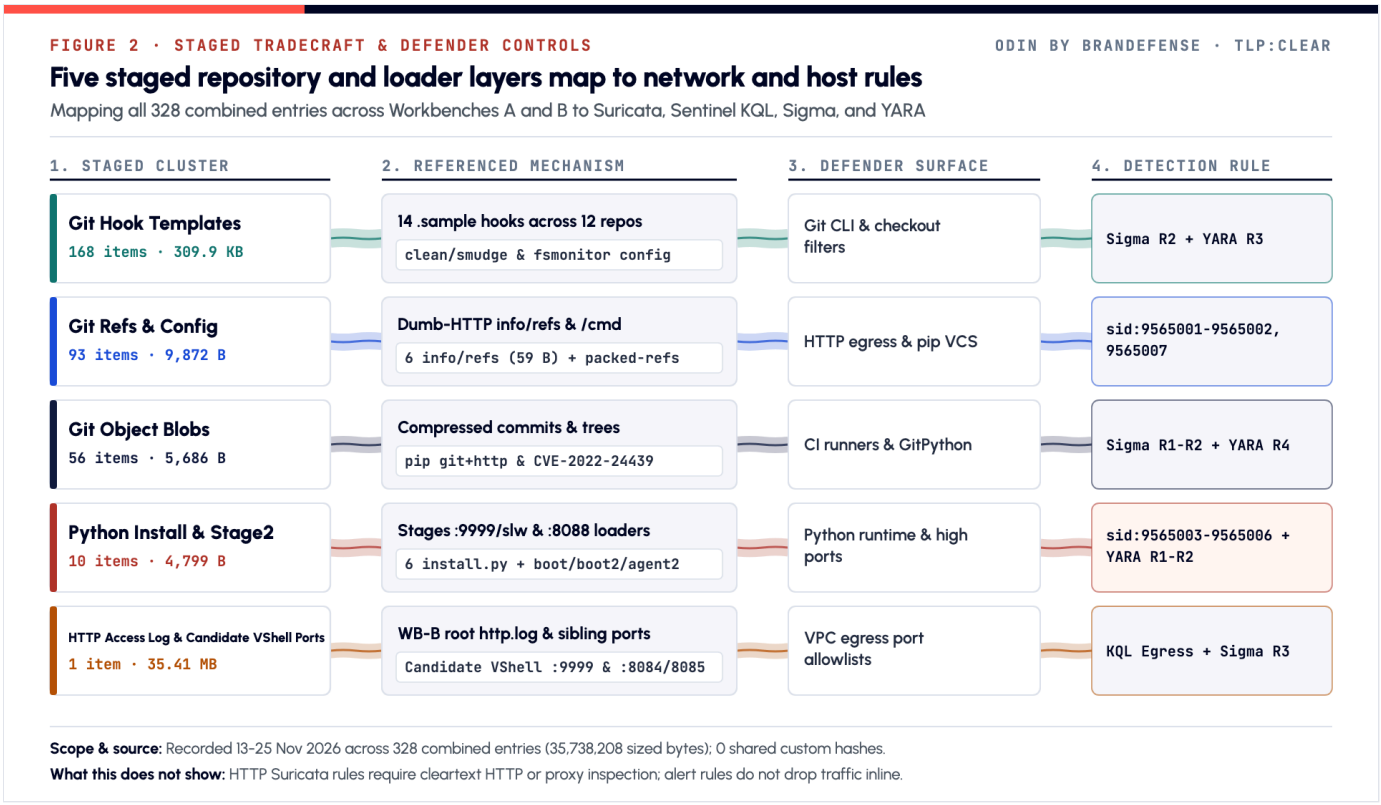

Figure 2: Five staged repository and loader layers map to network and host rules

5. Python VCS, Git Filter, and High-Port Egress Exposure Analysis

Defenders should verify four concrete controls across Python build pipelines, Git client wrappers, and workload egress perimeters before treating any gap as an active exposure:

Exposure surface	Observed workbench evidence	Control to verify	Operational risk
`pip` VCS installs & plugin cloners	6 bare HTTP Git repos (`/evil-node-bare` on :8888; `/exp1.git` – `/exp5.git` on :8088) plus 6 `install.py` scripts (191–393 B) and `__init__.py` .	Verify whether CI/CD and servers enforce `pip install --only-binary=:all: --require-hashes` (`[REF-02]`) and block `git+http://` URLs (`[REF-01]`).	Source builds (`sdist` / PEP 517, `[REF-03]` , `[REF-04]`) or plugin cloners can run untrusted Python scripts during installation (`[REF-08]`).
`GitPython` URL validation (`CVE-2022-24439`)	Dumb-HTTP `info/refs` (59 B across all 6 bare repos) served by `SimpleHTTP/0.6 Python/3.12.3` .	Verify whether Python services run `GitPython >= 3.1.30` (`[REF-05]`) and set `git config protocol.allow` allowlists (`[REF-07]`).	Unvalidated clone URLs in `GitPython` < `3.1.30` allow `--upload-pack=` or `-c` option injection (`CVSS 9.8` , `[REF-05]`).

Exposure surface	Observed workbench evidence	Control to verify	Operational risk
Git `clean` / `smudge` & `fsmonitor` config	12 Git repositories holding 168 `/hooks/*.sample` templates (309,852 B) and `.git/config` files across WB-A and WB-B.	Verify whether CI wrappers block `-c` config overrides in `git clone` calls and audit `.git/config` for `filter.<name>.clean` / `.smudge` (`[REF-06]` , `[REF-07]`).	Pairing `.gitattributes` filter labels with injected `smudge` or `core.fsmonitor` config entries can run shell code on checkout (`[REF-06]`).
Raw-IP & high-port TCP egress	WB-A `:9999/slw` + `TCP 9999` candidate `VShell` ; WB-B `:8088/boot2.py` , `/agent2.py` , `/cb/` , `/cmd` + `TCP 8084` /8085 candidate `VShell` .	Verify whether build runners and servers block direct TCP egress to raw IPv4 hosts and ports 8084, 8085, 8088, 8888, and 9999 (`[REF-10]`).	If run, the staged wrappers could fetch second-stage files; no captured execution or connection to the candidate `VShell` ports is shown.

6. Alternative Hypotheses

Hypothesis	Evidence for	Evidence against	Assessment
Single operator runs both Workbench A and Workbench B	Both China cloud hosts ran `SimpleHTTP/0.6 Python/3.12.3` , paired working and bare Git repos with `install.py` , co-hosted candidate `VShell` , and shared 19 SHA-256s.	All 19 shared SHA-256s are default Git init files (14 `.sample` hooks, `HEAD` , `description` , `exclude` , `packs` , empty blob). 0 **custom hashes** match across `AS37963` and `AS45090` .	**Unproven; treated as independent.** Analyze both servers as separate instances of one tradecraft pattern.
Benign developer Python mirrors or internal package caches	Both hosts use standard `python3 -m http.server` and Git layouts with `README.md` and `__init__.py` .	WB-A exposes `/evil-node` (`:9999/slw`) beside candidate `VShell` . WB-B pairs `/exp1.git` – `/exp5.git` with `/agent2.py` , `/boot2.py` , `/cb/` , `/cmd` , and candidate `VShell` :8084/:8085.	**Refuted.** Both hosts are adversary staging servers; IPs are withheld because candidate `VShell` ports lack live beacon captures.
The 168 `/hooks/*.sample` files run automatically on `git clone`	`hooks/*.sample` files form 51.22% of all 328 entries (309,852 B).	Git adds `.sample` to keep default hooks inactive, and `git clone` never copies server `/hooks/` to client clones (`[REF-07]`).	**Refuted technically.** Execution requires `install.py` , `pip` hooks, `CVE-2022-24439` , or config-registered `clean` / `smudge` filters.
Workbench B's 35.41 MB `/http.log` shows widespread victim infections	`/http.log` (35,407,999 B) on `TCP 8088` grew large by `20 November 2026` .	Open HTTP ports on public cloud IPs record constant internet scan noise without requiring any victim `git clone` .	**Unsupported.** Access log byte size never establishes victim compromises or campaign duration.

7. Detection Rules for Git Checkouts, Python Wrappers, and High Ports

Both server IPv4 addresses, the external `:8088/cmd` IPv4, and all **76** file SHA-256 digests are withheld. Defenders should deploy **protocol, process, file-event, and structural YARA rules** that catch this pattern on any host.

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:9565001` -**9565007**). Sentinel KQL schemas, `in~` / `in` operators, Sigma `ComputerName` allowlists, 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 production rollout.

7.1 Suricata 7.x Network Signatures (`sid:9565001` -9565007)

Rule `sid:9565001` detects outbound `GET` requests for `/info/refs` from Python HTTP clients (`python` User-Agent). Rule `sid:9565002` detects native `git/` clients (`git-remote-http` , including `pip` `git+http` clones) fetching `/info/refs` from raw IPv4 `Host` headers. Rule `sid:9565007` detects inbound `Server: SimpleHTTP/0.6 Python/` responses serving bare Git `HEAD` (`ref: refs/heads/`). Rules `sid:9565003` -**9565006** detect outbound requests to `:9999/slw` and :**8088** paths (`/boot.py` , `/boot2.py` , `/agent2.py` , and `/cb/`).

All seven HTTP rules require cleartext HTTP traffic or a TLS-terminating forward proxy. Every rule uses `alert` mode and logs detections without dropping packets inline.

```
SURICATA

# Odin by Brandefense - Monograph 65 Suricata 7.x Ruleset
# Covers dumb-HTTP bare Git info/refs fetches, raw-IP git clones, WB-A :9999/slw, WB-B :8088 boot/agent2/cb routes, and
SimpleHTTP Git HEAD exposure.

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R65 Outbound Python HTTP Client Fetch of Bare Git info/refs"; f
low:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/info/refs"; fast_pattern; http.user_
agent; content:"python"; nocase; classtype:trojan-activity; sid:9565001; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R65 Outbound Git Client Clone from Raw IPv4 HTTP Endpoint"; flo
w:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/info/refs"; fast_pattern; http.user_ag
ent; content:"git/"; startswith; http.host; pcre:"/^d{1,3}(?:.d{1,3}){3}(?::d+)?$/"; classtype:policy-violation; sid:956
5002; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 9999 (msg:"ODIN-R65 WB-A Outbound HTTP Fetch of Staged slw Payload on TCP 999
9"; flow:established,to_server; http.uri; content:"/slw"; startswith; fast_pattern; classtype:trojan-activity; sid:95650
03; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 8088 (msg:"ODIN-R65 WB-B Outbound HTTP Fetch of Staged boot.py or boot2.py on
TCP 8088"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/boot"; startswith; fast_
pattern; pcre:"/^/boot2?.py$/"; classtype:trojan-activity; sid:9565004; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 8088 (msg:"ODIN-R65 WB-B Outbound HTTP Fetch of Staged agent2.py Loader on TCP
8088"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/agent2.py"; bsize:10; fast_p
attern; classtype:trojan-activity; sid:9565005; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 8088 (msg:"ODIN-R65 WB-B Outbound HTTP Callback to Staged cb Prefix on TCP 808
8"; flow:established,to_server; http.uri; content:"/cb/"; startswith; fast_pattern; classtype:command-and-control; sid:9
565006; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R65 Inbound Python SimpleHTTP Response Serving Bare Git HEAD Re
ference"; flow:established,to_client; http.header; content:"Server: SimpleHTTP/0.6 Python/"; fast_pattern; http.response
_body; content:"ref: refs/heads/"; startswith; classtype:bad-unknown; sid:9565007; rev:1;)
```

7.2 Microsoft Sentinel KQL Hunt

This query unions process, file, and network events, then summarizes signals by device and account. Its network branch hunts high-port connections; inspect HTTP proxy logs for exact request paths. Replace the placeholder hosts in `ApprovedBuildHosts` with verified build mirrors before scheduling.

```kql
// Odin by Brandefense - Monograph 65
// Hunts pip git+http VCS installs, Git clean/smudge or CVE-2022-24439 option injection, staged install.py wrappers, and high-port connection candidates.
let Lookback = 7d;
let ApprovedBuildHosts = dynamic(["app-ci-01.corp.example", "pkg-mirror-01.corp.example"]);
let VcsInstallAndGitFilterProcesses =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBuildHosts))
    | where (
        (FileName in~ ("pip", "pip3", "python", "python3") and ProcessCommandLine has_any ("git+http://", "evil-node", "exp1.git", "exp2.git", "exp3.git", "exp4.git", "exp5.git"))
        or (FileName in~ ("git", "git-remote-http") and ProcessCommandLine has_any ("--upload-pack=", "protocol.ext.allow", "core.fsmonitor", ".smudge", ".clean"))
        or (FileName in~ ("python", "python3") and ProcessCommandLine has_any ("install.py", "boot.py", "boot2.py", "agent2.py") and ProcessCommandLine has_any ("evil-node", "/exp1", "/exp2", "/exp3", "/exp4", "/exp5", ":8088", ":9999"))
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "GitVcsOrPythonInstallExec", Detail = ProcessCommandLine;
let SuspiciousGitAttributesAndWrapperWrites =
    DeviceFileEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBuildHosts))
    | where ActionType in ("FileCreated", "FileModified", "FileRenamed")
    | where InitiatingProcessFileName in~ ("git", "git-remote-http", "python", "python3", "pip", "pip3")
    | where (
        FileName in~ ("install.py", "boot.py", "boot2.py", "agent2.py", ".gitattributes")
        and (FolderPath startswith "/tmp/" or FolderPath startswith "/var/tmp/" or FolderPath startswith "/dev/shm/" or FolderPath has "evil-node")
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "TempGitCheckoutOrLoaderWrite", Detail = strcat(InitiatingProcessFileName, " -> ", FolderPath, "/", FileName);
let BareGitAndStageTwoEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBuildHosts))
    | where InitiatingProcessFileName in~ ("python", "python3", "pip", "pip3", "git", "git-remote-http", "curl", "wget")
    | where RemotePort in (8084, 8085, 8088, 8888, 9999)
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "HighPortGitOrLoaderEgress", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteUrl, " (", RemoteIP, ":", tostring(RemotePort), ")");
union VcsInstallAndGitFilterProcesses, SuspiciousGitAttributesAndWrapperWrites, BareGitAndStageTwoEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = make_set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

7.3 Portable Sigma and YARA Rules

- **Sigma (`sigma_git_vcs_install_hooks_and_python_loaders.yml`).** Rule 1 catches `pip` or `pip3` commands containing `git+http`, `evil-node`, or `exp1.git` - `exp5.git` outside `app-ci-01.corp[.]example`. Rule 2 detects `/git` commands carrying `--upload-pack=`, `protocol.ext.allow`, `core.fsmonitor`, `.smudge`, or `.clean` outside `pkg-mirror-01.corp[.]example`. Rule 3 detects Python runs of staged `install.py`, `/boot2.py`, `/agent2.py`, `:9999/slw`, or `:8088/cb/` paths outside `sec-lab-01.corp[.]example`.
- **YARA (`odin_bare_git_python_install_and_vshell_loaders.yar`).** Four `filesize`-bounded rules match compact Python `install.py` wrappers (**< 16KB**) and multi-stage `/cb/` + `boot2.py` / `agent2.py` loaders (**< 32KB**). Rule 3 matches Git `smudge` / `clean` filter config abuse (**< 64KB**). Rule 4 matches small bare Git `config` files (**< 8KB**) for analyst triage; this structural match does not establish malicious use.

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

No server IPv4 addresses, external `:8088/cmd` addresses, or file SHA-256 digests appear in the indicator feeds. Defenders should hunt using the 10 behavioral, CLI, and protocol markers below.

Marker category	Pattern / behavioral marker	Layer	Defensive hunt & control
`pip` VCS HTTP install	`pip install git+http://`	Host CLI	Alert via Sentinel `VcsInstallAndGitFilterProcesses` and Sigma Rule 1. Enforce `--require-hashes` and `--only-binary=:all:` (`[REF-02]`).
Bare Git ref fetch	`GET /*/info/refs` (`Python` or raw-IP `git/` UA)	HTTP egress	Alert via `sid:9565001`-9565002 and block raw-IP Git clones at egress web proxies.
WB-A stage-two URI	`GET /slw` on `TCP 9999`	HTTP :9999	Alert via `sid:9565003`, HTTP proxy logs, and KQL :9999 connection candidates. Block direct `TCP 9999` egress.
WB-B Python loaders	`GET /boot.py`, `/boot2.py`, `/agent2.py`	HTTP :8088	Alert via `sid:9565004`-9565005, HTTP proxy logs, YARA Rules 1-2, and KQL :8088 connection candidates.
WB-B callback & cmd	`/cb/` & `/cmd` on `TCP 8088`	HTTP :8088	Alert via `sid:9565006`, HTTP proxy logs, and KQL :8088 connection candidates. Inspect initiating processes.
Post-clone wrapper	`python install.py` in `/evil-node` or `/exp1` - `/exp5`	Host EDR	Alert via Sigma Rule 3 and KQL `SuspiciousGitAttributesAndWrapperWrites`. Disable auto-install hooks.
Git filter abuse	`filter.<name>.clean` / `filter.<name>.smudge`	Git config	Alert via Sigma Rule 2 and YARA Rule 3. Audit `.git/config` files in build workspaces (`[REF-06]`).
`GitPython` RCE flags	`--upload-pack=` / `-c protocol.ext.allow=always`	Git CLI	Alert via Sigma Rule 2 and upgrade `GitPython` to `>= 3.1.30` (`CVE-2022-24439`, `[REF-05]`).
SimpleHTTP Git HEAD	`Server: SimpleHTTP/0.6 Python/` + `ref: refs/heads/`	HTTP ingress	Alert via `sid:9565007` and close accidental `python3 -m http.server` ports on cloud hosts.
Sibling `VShell` ports	`TCP 9999` (:8888) & `TCP 8084`/8085 (:8088)	Network / VPC	Restrict workload egress to `TCP 443` via proxies and hunt high-port connection candidates in KQL (`[REF-10]`).

9. Hardening Priorities for Python Pipelines, Git Clients, and Network Egress

Engineering, DevSecOps, and SOC teams can close these Git and Python loader paths in five steps:

1. **Require binary wheels and hash lockfiles in `pip` (`0-48 Hours`).** Set `PIP_ONLY_BINARY=:all:` (`--only-binary=:all:`) and `--require-hashes` in CI and server builds (`[REF-02]`). Rejecting source archives (`[REF-03]` , `[REF-04]`) and `git+http://` URLs (`[REF-01]`) stops untrusted build hooks from running.
2. **Patch `GitPython` (`>= 3.1.30`) and restrict Git protocols (`0-72 Hours`).** Upgrade `GitPython` to `3.1.30` or newer across all repos and containers to remediate `CVE-2022-24439` (`[REF-05]`). Pass `--` before remote URLs in wrapper scripts and set `git config --system protocol.allow never` with `protocol.https.allow always` (`[REF-07]`).
3. **Disable post-clone `install.py` runs and audit Git filters (`Days 1-7`).** Turn off automatic `install.py` execution in plugin and AI node managers (`[REF-08]` , `[REF-09]`). Check automated checkouts for `.gitattributes` `filter=` rules paired with unauthorized `clean` , `smudge` , or `core.fsmonitor` config commands (`[REF-06]` , `[REF-07]`).
4. **Block raw-IP clones and high-port egress (`Days 7-14`).** Route developer and CI egress through proxies that reject raw IPv4 `Host` headers (`sid:9565002`). Block direct outbound TCP traffic to ports **8084**, **8085**, **8088**, **8888**, and **9999** (`[REF-10]`).
5. **Close exposed `python3 -m http.server` listeners (`Days 14-30`).** Audit cloud firewall rules continuously so internal folders containing `.git/` metadata (`sid:9565007`) are never exposed on `0.0.0.0` .

10. Scope, Limits, and External References

Odin by Brandefense analyzed open Python `SimpleHTTP/0.6` directories on two China cloud servers recorded between `13 and 25 November 2026` . Workbench A's `TCP 8888` directory was recorded on `13-25 November 2026` . Workbench B's `TCP 8088` directory inventory covers `13-20 November 2026` . Separate service observations on `23-24 November 2026` recorded candidate `VShell` ports `TCP 8084` and `TCP 8085` .

Counts cover **328** deduplicated entries (**50** on Workbench A plus **278** on Workbench B, combining **255** live `20 November` files with **23** `13 November` Git objects). Those entries comprise **326** sized files (**35,738,208** bytes), **2** unsized entries, and **76** distinct SHA-256 digests.

The 19 shared SHA-256 digests are standard Git initialization files (`hooks/*.sample` , `description` , `HEAD` , `exclude` , `packs` , and empty blob), and 0 custom operator hashes link both hosts. Directory listings, candidate `VShell` port records (`TCP 9999` , **8084**, and **8085**), and Workbench B's **35.41 MB** `/http.log` document staged infrastructure rather than confirmed victim execution. Both server IPv4 addresses, the external `:8088/cmd` IPv4, and all **76** file digests are withheld.

1. `[REF-01]` PyPA `pip` Documentation (`pip.pypa[.]io`), *VCS Support (git+http, git+https, and Editable/Source Checkout Mechanics)* (verified on 25 November 2026).
2. `[REF-02]` PyPA `pip` Documentation (`pip.pypa[.]io`), *Secure Installs (--require-hashes and --only-binary=:all:)* (verified on 25 November 2026).
3. `[REF-03]` Python Packaging User Guide (`packaging.python[.]org`), *Source Distribution (sdist) Format Specification* (verified on 25 November 2026).
4. `[REF-04]` Python Enhancement Proposals (`peps.python[.]org`), *PEP 517 - A Build-System Independent Format for Source Trees* (verified on 25 November 2026).
5. `[REF-05]` NIST National Vulnerability Database (`nvd.nist[.]gov`), *CVE-2022-24439 - GitPython Improper Remote URL Input Validation RCE (Fixed in 3.1.30)* (verified on 25 November 2026).
6. `[REF-06]` Git Official Documentation (`git-scm[.]com`), *gitattributes - clean and smudge Filter Execution Semantics* (verified on 25 November 2026).
7. `[REF-07]` Git Official Documentation (`git-scm[.]com`), *git-config - protocol.allow, core.hooksPath, and Safe Directory Controls* (verified on 25 November 2026).
8. `[REF-08]` MITRE ATT&CK (`attack.mitre[.]org`), *Compromise Software Dependencies and Development Tools (T1195.001)* (verified on 25 November 2026).

9. [REF-09] MITRE ATT&CK (attack.mitre[.]org), *Command and Scripting Interpreter: Python (T1059.006)* (verified on 25 November 2026).
10. [REF-10] MITRE ATT&CK (attack.mitre[.]org), *Non-Standard Port (T1571)* (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

Find exposed developer servers and gateways.

Dark Web Monitoring

Catch leaked keys and tokens offered for sale.

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.

Suricata `odin_bare_git_python_install_and_vshell_hunt.rules` 11 lines

```
# Odin by Brandefense - Monograph 65 Suricata 7.x Ruleset
# Covers dumb-HTTP bare Git info/refs fetches, raw-IP git clones, WB-A :9999/slw, WB-B :8088 boot/agent2/cb routes, and SimpleHTTP Git HEAD exposure.

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R65 Outbound Python HTTP Client Fetch of Bare Git info/refs"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/info/refs"; fast_pattern; http.user_agent; content:"python"; nocase; classtype:trojan-activity; sid:9565001; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-R65 Outbound Git Client Clone from Raw IPv4 HTTP Endpoint"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/info/refs"; fast_pattern; http.user_agent; content:"git/"; startswith; http.host; pcre:"/^\d{1,3}(?:\.\d{1,3}){3}(?::\d+)?$/"; classtype:policy-violation; sid:9565002; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 9999 (msg:"ODIN-R65 WB-A Outbound HTTP Fetch of Staged slw Payload on TCP 9999"; flow:established,to_server; http.uri; content:"/slw"; startswith; fast_pattern; classtype:trojan-activity; sid:9565003; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 8088 (msg:"ODIN-R65 WB-B Outbound HTTP Fetch of Staged boot.py or boot2.py on TCP 8088"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/boot"; startswith; fast_pattern; pcre:"/^\/boot2?\.py$/"; classtype:trojan-activity; sid:9565004; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 8088 (msg:"ODIN-R65 WB-B Outbound HTTP Fetch of Staged agent2.py Loader on TCP 8088"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/agent2.py"; bsize:10; fast_pattern; classtype:trojan-activity; sid:9565005; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 8088 (msg:"ODIN-R65 WB-B Outbound HTTP Callback to Staged cb Prefix on TCP 8088"; flow:established,to_server; http.uri; content:"/cb/"; startswith; fast_pattern; classtype:command-and-control; sid:9565006; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-R65 Inbound Python SimpleHTTP Response Serving Bare Git HEAD Reference"; flow:established,to_client; http.header; content:"Server: SimpleHTTP/0.6 Python/"; fast_pattern; http.response_body; content:"ref: refs/heads/"; startswith; classtype:bad-unknown; sid:9565007; rev:1;)
```

Show full rule

YARA `odin_bare_git_python_install_and_vshell_loaders.yar` 79 lines

```
/*
   Odin by Brandefense - Monograph 65 YARA Suite
   4 filesize-bounded structural rules covering staged Python install.py wrappers,
   multi-stage boot/agent2 callback loaders, Git clean/smudge filter abuse, and dumb-HTTP bare Git manifests.
*/

rule ODIN_WB65_Python_Install_Wrapper_Stage_Fetch
{
    meta:
        author = "Odin by Brandefense"
        monograph = "65"
        description = "Detects compact Python install.py package wrappers referencing high-port stage-two routes (:9999/slw, :8088/boot, or :8088/agent2.py)"
        date = "2026-11-25"
    strings:
        $r1 = ":9999/slw" ascii
        $r2 = ":8088/boot" ascii
        $r3 = ":8088/agent2.py" ascii
        $r4 = ":8088/cmd" ascii
        $p1 = "urllib" ascii
        $p2 = "subprocess" ascii
        $p3 = "os.system" ascii
        $p4 = "requests" ascii
```

```
    condition:
        filesize < 16KB and ($r1 or $r2 or $r3 or $r4) and ($p1 or $p2 or $p3 or $p4)
}

rule ODIN_WB65_Python_MultiStage_Callback_Loader
{
    meta:
        author = "Odin by Brandefense"
        monograph = "65"
        description = "Detects multi-stage Python bootstrap and callback scripts combining /cb/, boot2.py, or agent2.py
endpoints with network and process execution"
        date = "2026-11-25"
    strings:
        $c1 = "/cb/" ascii
        $c2 = "boot2.py" ascii
        $c3 = "agent2.py" ascii
        $n1 = "urllib.request" ascii
        $n2 = "socket" ascii
        $e1 = "subprocess" ascii
        $e2 = "exec(" ascii
    condition:
        filesize < 32KB and ($c1 and ($c2 or $c3)) and ($n1 or $n2) and ($e1 or $e2)
}

rule ODIN_WB65_GitAttributes_Filter_Config_Abuse
{
    meta:
        author = "Odin by Brandefense"
        monograph = "65"
        description = "Detects Git configuration files registering clean or smudge filter commands invoking Python, cur
l, wget, or shell interpreters"
        date = "2026-11-25"
    strings:
        $f1 = "[filter " ascii
        $f2 = "filter." ascii
        $m1 = "smudge =" ascii
        $m2 = "clean =" ascii
        $x1 = "python" ascii
        $x2 = "sh -c" ascii
        $x3 = "curl " ascii
        $x4 = "wget " ascii
    condition:
        filesize < 64KB and ($f1 or $f2) and ($m1 or $m2) and ($x1 or $x2 or $x3 or $x4)
}

rule ODIN_WB65_Bare_Git_DumbHTTP_Repo_Manifest
{
    meta:
        author = "Odin by Brandefense"
        monograph = "65"
        description = "Detects small bare Git configuration files for analyst triage; this structural match does not est
ablish malicious use."
        date = "2026-11-25"
    strings:
        $g1 = "[core]" ascii
        $g2 = "bare = true" ascii
    condition:
        filesize < 8KB and $g1 and $g2
}
```

Sentinel KQL **sentinel_git_vcs_python_install_and_vshell_hunt.kql** 36 lines

```
// Odin by Brandefense - Monograph 65
// Hunts pip git+http VCS installs, Git clean/smudge or CVE-2022-24439 option injection, staged install.py wrappers, and high-port connection candidates.
let Lookback = 7d;
let ApprovedBuildHosts = dynamic(["app-ci-01.corp.example", "pkg-mirror-01.corp.example"]);
let VcsInstallAndGitFilterProcesses =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBuildHosts))
    | where (
        (FileName in~ ("pip", "pip3", "python", "python3") and ProcessCommandLine has_any ("git+http://", "evil-node", "exp1.git", "exp2.git", "exp3.git", "exp4.git", "exp5.git"))
        or (FileName in~ ("git", "git-remote-http") and ProcessCommandLine has_any ("--upload-pack=", "protocol.ext.allow", "core.fsmonitor", ".smudge", ".clean"))
        or (FileName in~ ("python", "python3") and ProcessCommandLine has_any ("install.py", "boot.py", "boot2.py", "agent2.py") and ProcessCommandLine has_any ("evil-node", "/exp1", "/exp2", "/exp3", "/exp4", "/exp5", ":8088", ":9999"))
    )
    | project Timestamp, DeviceName, AccountName, SignalType = "GitVcsOrPythonInstallExec", Detail = ProcessCommandLine;
let SuspiciousGitAttributesAndWrapperWrites =
    DeviceFileEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBuildHosts))
    | where ActionType in ("FileCreated", "FileModified", "FileRenamed")
    | where InitiatingProcessFileName in~ ("git", "git-remote-http", "python", "python3", "pip", "pip3")
    | where (
        FileName in~ ("install.py", "boot.py", "boot2.py", "agent2.py", ".gitattributes")
        and (FolderPath startswith "/tmp/" or FolderPath startswith "/var/tmp/" or FolderPath startswith "/dev/shm/" or FolderPath has "evil-node")
    )
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "TempGitCheckoutOrLoaderWrite", Detail = strcat(InitiatingProcessFileName, " -> ", FolderPath, "/", FileName);
let BareGitAndStageTwoEgress =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBuildHosts))
    | where InitiatingProcessFileName in~ ("python", "python3", "pip", "pip3", "git", "git-remote-http", "curl", "wget")
    | where RemotePort in (8084, 8085, 8088, 8888, 9999)
    | project Timestamp, DeviceName, AccountName = InitiatingProcessAccountName, SignalType = "HighPortGitOrLoaderEgress", Detail = strcat(InitiatingProcessFileName, " -> ", RemoteUrl, " (", RemoteIP, ":", tostring(RemotePort), ")");
union VcsInstallAndGitFilterProcesses, SuspiciousGitAttributesAndWrapperWrites, BareGitAndStageTwoEgress
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(), Signals = make_set(SignalType, 6), SampleDetails = make_set(Detail, 5) by DeviceName, AccountName
| order by LastSeen desc
```

Sigma **sigma_git_vcs_install_hooks_and_python_loaders.yml** 108 lines

```
title: Pip VCS Unencrypted HTTP or Staged Bare Git Package Installation
id: 65a8c310-4d22-4b19-8a01-65c10e48a201
status: experimental
description: Detects pip or Python package-manager invocations fetching source repositories over cleartext git+http URLs or referencing staged evil-node and exp1-exp5 bare Git repositories.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_pip_proc:
    CommandLine|contains:
      - 'pip install'
      - 'pip3 install'
      - '-m pip install'
```

```
  selection_vcs_target:
    CommandLine|contains:
      - 'git+http://'
      - 'evil-node'
      - 'exp1.git'
      - 'exp2.git'
      - 'exp3.git'
      - 'exp4.git'
      - 'exp5.git'
  filter_ci_hosts:
    ComputerName:
      - 'app-ci-01.corp.example'
  condition: (selection_pip_proc and selection_vcs_target) and not filter_ci_hosts
falsepositives:
  - Designated internal build runners pulling legacy internal mirrors before TLS migration
level: high
tags:
  - attack.initial-access
  - attack.execution
  - attack.t1195.001
  - attack.t1059.006
---
title: Git Clone Option Injection or Clean and Smudge Filter Command Registration
id: 65a8c310-4d22-4b19-8a01-65c10e48a202
status: experimental
description: Detects Git CLI invocations containing CVE-2022-24439 remote option-injection flags or registering clean, s
mudge, fsmonitor, or ext-protocol shell commands.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
detection:
  selection_git_bin:
    Image|endswith:
      - '/git'
  selection_dangerous_args:
    CommandLine|contains:
      - '--upload-pack='
      - 'protocol.ext.allow'
      - 'core.fsmonitor'
      - '.smudge'
      - '.clean'
  filter_mirror_hosts:
    ComputerName:
      - 'pkg-mirror-01.corp.example'
  condition: (selection_git_bin and selection_dangerous_args) and not filter_mirror_hosts
falsepositives:
  - Approved Git LFS mirror servers configuring git-lfs smudge and clean filters
level: high
tags:
  - attack.execution
  - attack.t1195.001
  - attack.t1059.006
---
title: Staged Repository install.py Wrapper and Second-Stage Python Loader Execution
id: 65a8c310-4d22-4b19-8a01-65c10e48a203
status: experimental
description: Detects Python interpreter execution of staged repository install.py wrappers or second-stage boot.py, boot
2.py, agent2.py, and :9999/slw loader paths.
author: Odin by Brandefense
date: 2026-11-25
logsource:
  category: process_creation
  product: linux
```

```
detection:
  selection_py_bin:
    Image|endswith:
      - '/python'
      - '/python3'
  selection_staged_paths:
    CommandLine|contains:
      - '/evil-node/install.py'
      - '/exp1/install.py'
      - '/exp2/install.py'
      - '/exp3/install.py'
      - '/exp4/install.py'
      - '/exp5/install.py'
      - '/boot2.py'
      - '/agent2.py'
      - ':9999/slw'
      - ':8088/cb/'
  filter_lab_hosts:
    ComputerName:
      - 'sec-lab-01.corp.example'
  condition: (selection_py_bin and selection_staged_paths) and not filter_lab_hosts
falsepositives:
  - Isolated malware-analysis sandboxes replaying captured loader wrappers
level: high
tags:
  - attack.execution
  - attack.command-and-control
  - attack.t1059.006
  - attack.t1571
```

