ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 36 / 30 · AI OFFENSE

Autonomous Hunt Agents and Cross-Platform Implants
Workbench and C2 Evidence

How a 2,202-file exposed workbench stages autonomous hunt playbooks and Windows or Linux implants alongside 220 unconfirmed agentic C2 endpoints

Prepared November 2026
Full Report · Public edition · Version 1.1

44 + 34
Sampled agent runtime, playbook and session files (44) plus implant staging and Go cache files (34)

19
Secret and callback findings in six sampled session scopes (14 C2 references, 2 passwords, 1 reverse shell, 2 keys)

211
Distinct hosts in the 220-endpoint set, including nine dual-listener hosts and two dual-framework hosts

3 / 217
Feed-labelled endpoints with TLS, JARM and port matches (3) versus other unconfirmed endpoints (217), all withheld

What Is in This Report. 11 chapters and two appendices.

How a 2,202-file exposed workbench stages autonomous hunt playbooks and Windows or Linux implants alongside 220 unconfirmed agentic C2 endpoints

01	Executive Summary	04
02	Two Populations, One Analytical Model	07
03	Inside the Exposed Autonomous Hunting Workbench	10
04	Cross-Platform Implants, Go Build Cache, and Session Findings	14
05	Seven Autonomous Offense Families and TLS Fingerprints	17
06	Feed Labels, TLS Matches, and Other Explanations	20
07	What Defenders Should Do	23
08	Detection Rules	25
09	Indicators	28
10	Scope and Limits	30
11	Sources	32
A	Indicators of compromise (5)	34
B	Detection rules (4 files)	35

IN SHORT

From 13 to 20 November 2026, Odin by Brandefense examined an open HTTP directory in China reporting 2,202 files. We also examined 220 newly observed endpoints (19 to 20 November 2026) across seven autonomous-offense frameworks from 9,928 feed-labelled records. A sample of 178 workbench files pairs a 171.2 MB agent runtime and 16 post-exploitation playbooks with Windows and Linux x64 implants. Three feed-labelled endpoints matched TLS and port checks, but all 220 remain unconfirmed. Close unauthenticated HTTP directories and hunt CyberStrikeAI TLS certificates.

01

CHAPTER 01

Executive Summary

IN SHORT

From 13 to 20 November 2026, Odin by Brandefense examined an open HTTP directory in China reporting 2,202 files. We also examined 220 newly observed endpoints (19 to 20 November 2026) across seven autonomous-offense frameworks from 9,928 feed-labelled records. A sample of 178 workbench files pairs a 171.2 MB agent runtime and 16 post-exploitation playbooks with Windows and Linux x64 implants. Three feed-labelled endpoints matched TLS and port checks, but all 220 remain unconfirmed. Close unauthenticated HTTP directories and hunt CyberStrikeAI TLS certificates.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 05

01 Executive Summary

Autonomous offense tools split work between two layers. On staging servers, command-line agents pair Markdown skill playbooks with local scan scripts and compiled implants. On external virtual private servers (VPS), web and GraphQL consoles accept operator tasks and callback traffic. Defenders can use cyber threat intelligence to examine both layers and separate what to block from what to hunt.

1. **One open HTTP directory exposes 2,202 files across playbooks, implants and scans.**
   On one server in China, 178 sampled entries span five directory layers. A hidden agent runtime folder holds a 171.2 MB agent binary, unified JSONL event logs and 16 Markdown playbooks for Sliver C2, token impersonation, credential dumping and cloud persistence. Sibling folders hold Windows and Linux x64 implant binaries, 93 CMS check scripts and 49.8 MB of packet captures.

2. **Per-target scope folders hold 19 secret and callback findings next to a Go cache.**
   Twenty chat-history logs sit across 18 numbered scope folders under a dated work folder on the server. Those logs hold 14 C2 endpoint references, two passwords, one reverse-shell string, one ElevenLabs API key and one BIP-39 mnemonic. Twenty-six Go build cache files are consistent with Go compilation, though the listing does not establish where it occurred.

3. **All 220 newly observed autonomous-offense endpoints remain unconfirmed without behavioral evidence.**
   Across 9,928 feed records naming ARTEX, CyberStrikeAI, PentAGI, HexStrikeAI, Cairn, AutoHunter and Strix, we examined 220 endpoints on 211 hosts. Three feed-labelled endpoints (one CyberStrikeAI and two PentAGI) matched certificate, JARM, port and hosting checks, but lack protocol or beacon corroboration. The remaining 217 also stay unconfirmed due to cloud churn, single banners or telecom ISP networks.

Three Decisions for Leaders

1. **Keep uncorroborated C2 feed addresses off automatic block lists.** Three feed-labelled CyberStrikeAI and PentAGI endpoints matched certificate, JARM, port and hosting checks, but lack behavioral or protocol confirmation. Treat those three endpoints, single-banner cloud matches and dynamic telecom addresses as hunting leads rather than firewall blocks.
2. **Ban unauthenticated HTTP file servers on red-team, build and AI-agent hosts.** A quick HTTP directory server publishes hidden agent folders, Go build caches, engagement logs and implant archives to the internet.
3. **Audit exposed workspaces and harden CMS and Windows service accounts.** Alert on unauthenticated HTTP directory listings and ad-hoc web servers, restrict Windows token impersonation rights, and patch internet-facing CMS deployments.

The Study in Numbers

Number	What it is
2,202 / 178	Total reported files and sampled file entries on the exposed HTTP workbench in China
44 + 34	Sampled agent runtime, playbook and session files (44) plus implant staging and Go cache files (34)
19	Secret and callback findings in six sampled session scopes (14 C2 references, 2 passwords, 1 reverse shell, 2 keys)
9,928 / 220	Feed-labelled records across seven autonomous-offense frameworks, and the 220 newly observed endpoints examined
211	Distinct hosts in the 220-endpoint set, including nine dual-listener hosts and two dual-framework hosts
3 / 217	Feed-labelled endpoints with TLS, JARM and port matches (3) versus other unconfirmed endpoints (217), all withheld
5	Published indicators in Appendix A: 5 JARM fingerprints (0 IPv4 addresses and 0 file hashes)

How to use this report.

Security leaders can start with the short read on autonomous hunting agents and cross-platform implants or read Chapters 1, 6 and 7. Cloud, platform and Windows teams should read Chapters 3, 4 and 7. SOC analysts, detection engineers and CTI analysts should start with Chapters 2, 5, 6, 8 and 9.

02

CHAPTER 02

Two Populations, One Analytical Model

IN SHORT

No host or TLS certificate links the exposed workbench in China to the 211 hosts in the seven-framework cohort. Keep uncorroborated listener addresses off automatic block lists, and use the TLS certificate and JARM fingerprints for hunting only.

INSIDE THIS CHAPTER

Where the servers sit 08

02 Two Populations, One Analytical Model

Agent-assisted offense relies on local workspaces to prepare tooling and remote listeners to coordinate tasks. Treating an open testing directory as proven hostile infrastructure, or blocking every cloud address that serves a default web banner, breaks legitimate services and clouds attribution.

We compared hosts and TLS certificates across both populations in this study. Zero IPv4 addresses or certificates link the exposed server in China to the 211 hosts in the seven-framework listener set. We therefore present both populations as an **analytical model** of how local hunting workspaces and remote C2 consoles divide work, not as one actor. Figure 1 shows the four functional groups and what defenders can do with each.

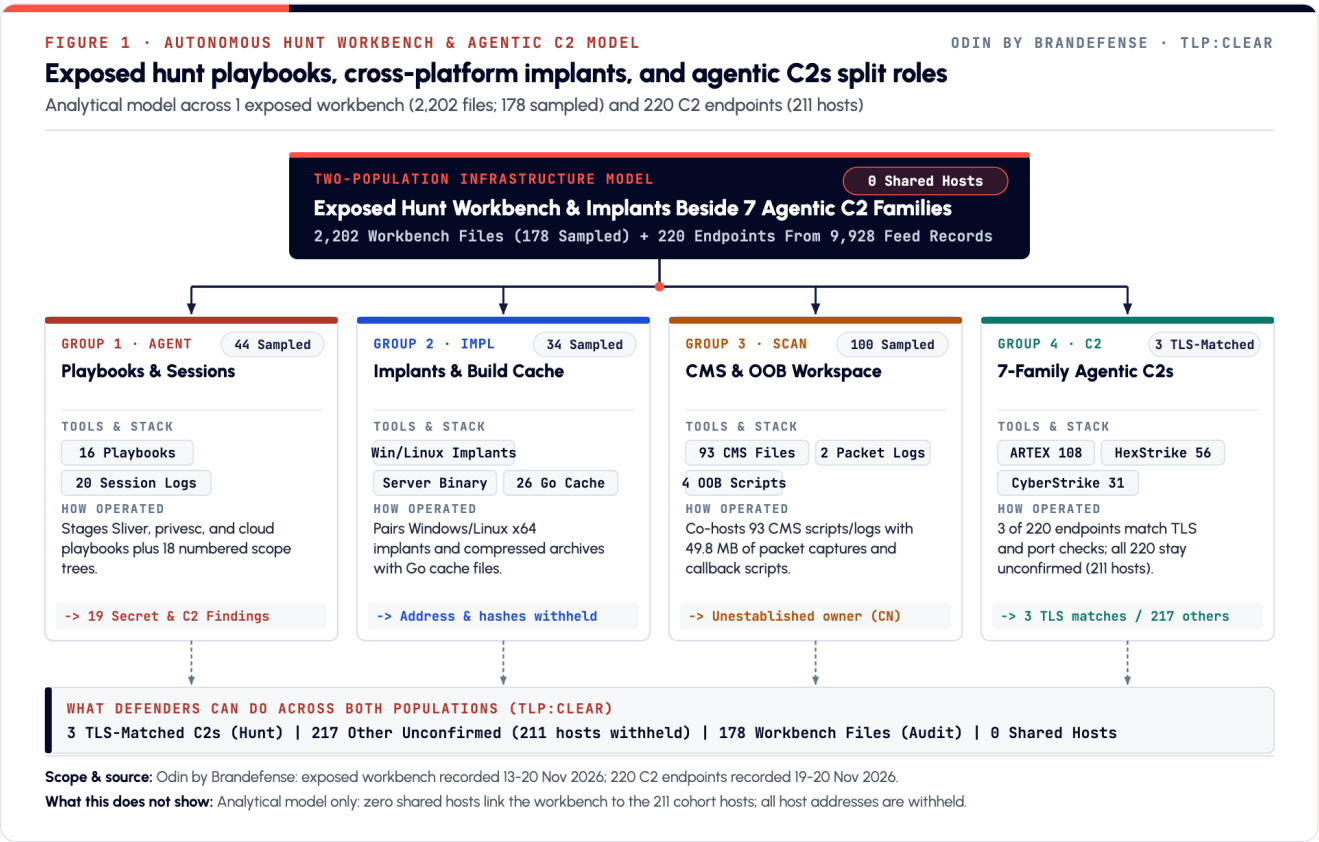


Figure 1: Exposed hunt playbooks, cross-platform implants, and agentic C2s split roles (analytical model)

Population	Upstream count	Examined in this study	Distinct hosts	Published addresses	Defender action
Exposed hunt and implant workbench	2,202 files	178 file entries	1 (withheld)	0	Audit ad-hoc HTTP servers; close open directories
Workbench session findings	19 matches	19 findings (6 scopes)	1 (withheld)	0	Rotate exposed keys; audit agent logs
Seven-framework listener set	9,928 feed records	220 endpoints	211 (all withheld)	0	Hunt TLS certificates and JARM fingerprints

Where the servers sit

The exposed workbench sits in China on an unattributed network, illustrating what defenders can find on their own assets with external attack surface management. The folder tree includes agent playbooks and a report-generation script. External directory listings cannot establish whether the owner is a contracted red team or an intruder reusing red-team skill templates. The server address, service port, directory paths, file names and all file hashes from that host are withheld in this edition.

By contrast, the 220 newly observed autonomous-offense endpoints span 25 countries and 72 hosting networks. Five jurisdictions hold 154 of the 220 endpoints (70.0%), followed by Sweden (8), Germany (8), India (7), Australia (7) and Italy (6).

Country	Endpoints	Share
United States	62	28.2%
Hong Kong	48	21.8%
China	21	9.5%
Singapore	13	5.9%
Japan	10	4.5%

Hyperscale cloud providers and low-cost VPS networks host most of the 220 endpoints. Seven candidate endpoints (six ARTEX and one PentAGI) sit on dynamic Chinese telecom networks (`CHINANET BACKBONE`, `CHINA UNICOM China169 Backbone`, `China Mobile` and `China Telecom`). Because broadband and mobile addresses change hands across subscribers, those seven addresses never belong on a firewall block list.

Hosting provider	Endpoints	Main frameworks on provider
Amazon.com, Inc.	39	HexStrikeAI (36), CyberStrikeAI (1), PentAGI (1), Cairn (1)
PEG TECH INC	19	ARTEX (17), PentAGI (2)
Microsoft Corporation	17	HexStrikeAI (15), PentAGI (1), ARTEX (1)
CTG Server Limited	10	CyberStrikeAI (7), ARTEX (3)
NetLab Global	9	ARTEX (7), CyberStrikeAI (2)
cognetcloud INC	8	ARTEX (6), CyberStrikeAI (2)
Shenzhen Tencent Computer Systems	7	ARTEX (3), AutoHunter (3), PentAGI (1)

CHAPTER 03

Inside the Exposed Autonomous Hunting Workbench

IN SHORT

The exposed server in China reports 2,202 files, with 178 sampled across five directory layers. Inside the agent runtime folder, a 171.2 MB agent binary sits beside 16 post-exploitation Markdown playbooks and 20 chat-history transcripts. Scan your own cloud hosts for open HTTP directories that expose hidden agent folders.

INSIDE THIS CHAPTER

The agent runtime and post-exploitation playbooks 11

Per-scope session trees and dated work folders 11

Co-hosted CMS checks and packet captures 12

03 Inside the Exposed Autonomous Hunting Workbench

Security practitioners and intruders alike now pair command-line LLM agents with plain-text "skill" files and local scan scripts. A quick file server such as `python3 -m http.server` can expose hidden folders, playbooks and logs. This happens when it serves the working directory on an unauthenticated public interface.

Python's standard library documentation explicitly warns that `http.server` lacks production security controls. Figure 2 maps each slice of the workbench and the listener set to its target surface and matching detection or audit control.

FIGURE 2 · WORKBENCH & AGENTIC C2 CAPABILITY MATRIX — ODIN BY BRANDEFENSE · TLP:CLEAR

Detection rules separate exposed workbench paths from unconfirmed C2 feeds

Six operational slices mapping 178 sampled workbench entries and 220 agentic C2 endpoints to controls

1. TOOL / BINARY	2. HOW USED (CLI / CONFIG)	3. TARGET SURFACE	4. DEFENDER SIGNAL
Agent Skill Playbooks 20 files (16 playbooks + 4 helpers)	**Sliver C2, Potato & dump guides** Agent skill playbook folder	Windows, Linux & cloud hosts	sid:9536003 + YARA 1
Agent Runtime & Scopes 24 files (18 scope folders)	**171.2 MB agent + 20 session logs** Agent runtime binary and log	Autonomous hunt task state	KQL + Sigma review
Implants & Go Cache 34 files (8 implant + 26 cache)	**Win/Linux x64 implants + Go** Implant folder and Go build cache	Cross-platform server targets	sid:9536003 + attack-surface audit
CMS & OOB Captures 100 files (93 CMS + 7 root)	**CMS check scripts & 49.8 MB pcap** CMS scan folder and packet logs	Public CMS & OOB callbacks	Manual CMS/OOB artifact review
TLS-Matched Agentic C2s 3 endpoints (hosts/ports withheld)	**CyberStrikeAI (1) & PentAGI (2)** CN=CyberStrikeAI + 2 TLS JARMs	Potential web/GraphQL listeners	sid:9536005 + JARM Hunt
Other Unconfirmed Pool 217 endpoints (208 hosts)	**ARTEX, HexStrike & AutoHunter** 3 shared cohort TLS JARMs	Cloud VPS & ISP churn hosts	JARM & TLS hunt only

Scope & source: Covers 178 sampled entries on the exposed server in China (13-20 Nov 2026) and 220 endpoints across 211 hosts (19-20 Nov 2026).
What this does not show: The exposed workbench server and all 211 listener hosts are withheld; none are placed on automatic block lists.

Figure 2: Detection rules separate exposed workbench paths from unconfirmed C2 feeds

Across the 178 sampled entries on the workbench, scripts lead with 75 files (72 Python, two Unix shell and one batch file). The sample also holds 70 text files, 26 Go cache objects, four executables, two compressed archives and one SQLite database.

Directory layer	Sampled files	Share	Key contents and byte sizes
CMS scanning workspace	93	52.2%	Python CMS check scripts, URL lists and a CMS scan log (3.75 MB)
Agent runtime folder	44	24.7%	Agent binary (171.2 MB), 20 skill playbook files, 21 session files and a unified event log
Go compiler cache	26	14.6%	17 build action files and 9 compiled output objects
Implant staging folder	8	4.5%	Controller binary (17.9 MB), Windows and Linux x64 implant binaries and compressed archives
Root captures and callbacks	7	3.9%	Two packet capture logs (39.0 MB and 10.8 MB), OOB Python scripts and an 85-byte netcat loop script

The agent runtime and post-exploitation playbooks

Forty-four sampled files (24.7%) sit inside the hidden agent runtime folder. A single agent executable measures 171,207,616 bytes (163.3 MB). Beside it sit a unified JSONL event log and a 283,192-byte Bash completion script.

Inside the agent skill subtree, 20 files turn the runtime into a guided post-exploitation assistant. Sixteen are Markdown playbooks grouped across three topic subfolders (12, 3 and 1 files; paths and file names withheld):

Playbook group	Count	Covered workflows (file names withheld)
Sliver C2 operations	5	Sliver operator setup, implant generation, C2 protocols, pivoting proxies and post-exploitation
Credential access and privilege escalation	4	Lateral credential extraction, Windows credential dumping, Windows token Potato privilege escalation and Linux kernel exploit checks
Cloud post-exploitation	3	Cloud post-exploitation workflows, persistence backdoors and cloud data exfiltration
Mobile and exploit research	4	iOS exploitation, iOS mitigation analysis, iOS kernel heap techniques and exploit development

Four supporting files complete the skill folder: a Python wrapper for the public TruffleHog secret-checking tool, a DOCX report-generation script, a 3,881-byte Python memory utility and a 417-byte sample environment file.

Per-scope session trees and dated work folders

Under the agent session folder, an SQLite session index sits next to 20 JSONL chat-history transcripts (118,712 to 593,712 bytes). Those 20 transcripts span 18 numbered scope folders; one holds three sessions and the other 17 hold one each.

The session paths include a dated work folder beneath the session root. Its date is a server folder name, not Odin's observation period. Numbered scope folders and the DOCX report-generation script fit either a consultancy producing client reports or an intruder reusing a red-team harness.

Co-hosted CMS checks and packet captures

Outside the agent runtime folder, 93 sampled entries (52.2%) sit under a Mandarin-named CMS scanning workspace folder. Sixty-six sit in a subfolder with 23 Python check scripts targeting regional CMS and office platforms. Sibling folders hold deduplicated domain and URL lists, backup outputs and scan result text files.

Those file labels reflect staged artifacts on the server, and a directory listing does not show that any remote CMS target was affected. At the web root, two packet capture logs (38,959,964 bytes and 10,807,210 bytes) total 49.77 MB. Three small Python callback scripts (948 to 1,094 bytes) and an 85-byte netcat loop script sit beside them.

04

CHAPTER 04

Cross-Platform Implants, Go Build Cache, and Session Findings

IN SHORT

The implant staging folder holds Windows and Linux x64 implant binaries, pre-compressed archives and a Linux controller binary next to 26 Go compiler cache files. Six sampled session scopes also hold 19 secret and callback findings. Audit ad-hoc HTTP servers and keep build caches and agent logs off web-accessible folders.

INSIDE THIS CHAPTER

Cross-platform implant binaries and Go build cache objects 14

Nineteen secret and callback findings in agent sessions 15

04 Cross-Platform Implants, Go Build Cache, and Session Findings

The implant staging folder has eight sampled entries, and the Go compiler build cache folder has 26. Both sit beside the agent runtime and CMS scanning folders on the exposed server. Figure 4 shows all five layers of the directory tree.

FIGURE 4 · EXPOSED HUNT & IMPLANT WORKBENCH BLUEPRINT — ODIN BY BRANDEFENSE · TLP:CLEAR

One exposed directory tree stacks agent logs, Go cache, and implants

Architectural cut-away of the exposed server in China (2,202 reported files; 178 sampled entries; 19 secret findings)



Scope & source: Covers all 178 sampled entries (3+20+21+34+100=178) on the exposed server, recorded 13-20 November 2026.
What this does not show: Host owner is not established (contracted red team or intruder); address, file hashes, and secrets are withheld.

Figure 4: One exposed directory tree stacks agent logs, Go cache, and implants

Cross-platform implant binaries and Go build cache objects

Five files inside the implant staging folder are compiled executables and gzip archives whose multi-megabyte sizes match statically linked Go binaries (paths and file names withheld):

Artifact role (path withheld)	Kind	Size (bytes)	Role on workbench
Controller binary	Linux binary	17,883,296	Controller binary staged beside the two platform agents
Windows x64 implant binary	Windows x64 PE	14,937,600	Uncompressed Windows x64 implant binary
Linux x64 implant binary	Linux x64 ELF	13,815,968	Uncompressed Linux x64 implant binary
Compressed Windows implant	Gzip archive	5,891,003	Compressed Windows implant (39.4% of raw binary size)
Compressed Linux implant	Gzip archive	5,410,760	Compressed Linux implant (39.2% of raw binary size)

Three text and script files also sit in the implant folder, while a 3,751,583-byte CMS scan log sits in the nearby CMS scanning workspace. One 5,630-byte Python script in the implant folder has byte-identical copies in four CMS workspace subfolders, linking both trees on the host. Two root callback scripts (1,094 bytes) and two CMS scripts (29,873 and 12,141 bytes) also repeat across folders.

Meanwhile, the Go compiler cache folder exposes 26 sampled files: 17 build action records and nine compiled outputs. Under Go's `GOCACHE` layout, those files are consistent with Go compilation, but the listing does not establish where it occurred. Because we did not detonate or reverse-engineer the binaries, co-location with the implant folder is structural context rather than proof of the exact build source.

Nineteen secret and callback findings in agent sessions

Across six sampled scope session trees on the server, 19 secret and callback matches were recorded (T1552.001). Callback and command infrastructure strings make up nearly four fifths of the total, while four entries hold passwords or API and wallet credentials. No secret value or partial prefix appears in this report.

Finding type	Count	Share	What it shows
C2 or callback endpoint reference	14	73.7%	Callback URLs or listener addresses logged inside agent session transcripts
Generic password	2	10.5%	Plaintext password strings stored in session files
Reverse-shell command string	1	5.3%	Shell callback one-liner recorded in a session transcript
ElevenLabs API key	1	5.3%	Commercial voice/AI service key exposed in agent state
BIP-39 mnemonic	1	5.3%	Cryptocurrency seed phrase exposed in agent state

CHAPTER 05

Seven Autonomous Offense Families and TLS Fingerprints

IN SHORT

Out of 220 newly observed endpoints across seven autonomous-offense frameworks, three feed-labelled endpoints matched certificate, JARM, port and hosting checks. All 220 remain unconfirmed without behavioral or protocol evidence. Keep all 211 host addresses off automatic block lists, and use `CN=CyberStrikeAI` and the five JARM fingerprints as hunting selectors.

INSIDE THIS CHAPTER

Three feed-labelled endpoints with certificate, JARM and port matches 17

Multi-listener hosts, dual-framework servers and port patterns 18

05 Seven Autonomous Offense Families and TLS Fingerprints

While the exposed server in China shows a file-based hunting workspace, public and commercial autonomous-offense suites expose web, GraphQL and Model Context Protocol (MCP) listener ports across cloud VPS fleets. Upstream feed labels (`artex`, `hexstrikeai`, `cyberstrikeai`, `pentagi`, `autohunter`, `cairn` and `strix`) describe banner, certificate or API fingerprints; they are not actor attribution. Figure 3 maps the non-directional evidence layers across target surfaces, the exposed workbench, staged cross-platform implant binaries and the unlinked listener fleet.

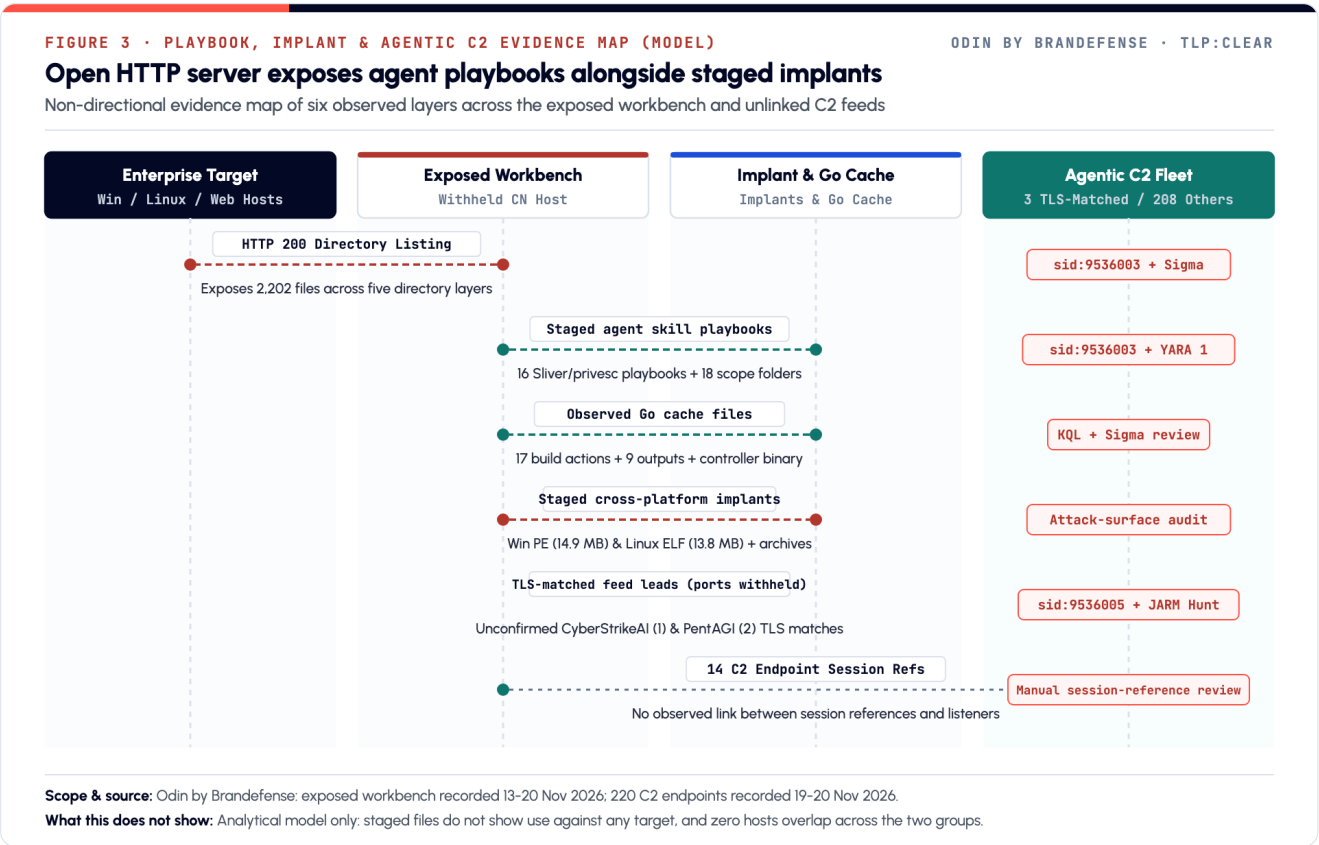


Figure 3: Open HTTP server exposes agent playbooks alongside staged implants (analytical model)

Three feed-labelled endpoints with certificate, JARM and port matches

Across the 220 endpoints on 211 hosts, three feed-labelled endpoints (1.36%) had certificate, JARM, port and hosting matches (T1071.001). Because the records include no protocol response, beacon, behavioral match or workbench link, all three remain unconfirmed and their IPv4 addresses, ports and hosting details are withheld:

Framework	Unconfirmed feed leads	Hosting	TLS certificate and JARM	Review advice
CyberStrikeAI	One unconfirmed feed lead (host and port withheld)	Withheld	`CN=CyberStrikeAI`, JARM `40d40d40d000…9319`	Hunt TLS and JARM; do not block without behavioral evidence
PentAGI	Two unconfirmed feed leads (hosts and ports withheld)	Withheld	`CN=localhost`, JARM `3fd3fd000000…99fb`	Hunt JARM; do not block without behavioral evidence

Full 62-character JARM values are listed in Chapter 9. Two details matter for operations:

- **CyberStrikeAI TLS match.** The CyberStrikeAI feed lead (host, port and hosting withheld) serves TLS with `CN=CyberStrikeAI`. Its JARM is listed in Chapter 9.
- **Two PentAGI endpoints sharing one TLS stack.** Both PentAGI feed leads (hosts, ports and hosting withheld) present `CN=localhost`. Their shared JARM is listed in Chapter 9.

Multi-listener hosts, dual-framework servers and port patterns

Summing distinct hosts per framework gives 213 rather than 211 because **two servers expose two frameworks at the same time** (hosts, ports and hosting withheld):

Multi-framework pairing	Host count	Frameworks observed
CyberStrikeAI and ARTEX	1 (host, ports and hosting withheld)	CyberStrikeAI and ARTEX on two listeners
ARTEX and AutoHunter	1 (host, ports and hosting withheld)	ARTEX and AutoHunter on two listeners

Seven other withheld hosts expose two listeners of one framework (four CyberStrikeAI, two HexStrikeAI and one ARTEX; hosts and ports withheld). Together, those nine dual-listener hosts explain the gap between 220 endpoints and 211 unique hosts.

Each framework clusters on recognizable ports across the 220 endpoints. ARTEX uses 8787 (42 of 108 endpoints, 38.9%), 8790 (17), 443 (10) and 80 (9). HexStrikeAI uses 80 (34 of 56), 5000 (7), 8080 (5) and 443 (5). CyberStrikeAI uses 8080 (17 of 31) and 443 (4).

PentAGI uses 8443 (12 of 17) and 8666 (2), while AutoHunter puts four of its five Chinese cloud endpoints on 18800. Three additional TLS JARM values each repeat across four other endpoints in the set (listed in Chapter 9 as hunt-only fingerprints).

06

CHAPTER 06

Feed Labels, TLS Matches, and Other Explanations

IN SHORT

Auto-blocking all 9,928 feed-labelled records or all 220 newly observed endpoints would cut off cloud tenants, broadband customers and authorized testers. Three endpoints (1.36% of the 220 examined) matched certificate, JARM, port and hosting checks, but all 220 require behavioral corroboration before blocking.

INSIDE THIS CHAPTER

Four other ways to read the evidence 21

06 Feed Labels, TLS Matches, and Other Explanations

An external block list is only as trustworthy as the false positives it refuses to include. Figure 5 compares the total feed-labelled records (9,928 across seven frameworks) and the 2,202 workbench files against the three endpoints that matched certificate, JARM, port and hosting checks.

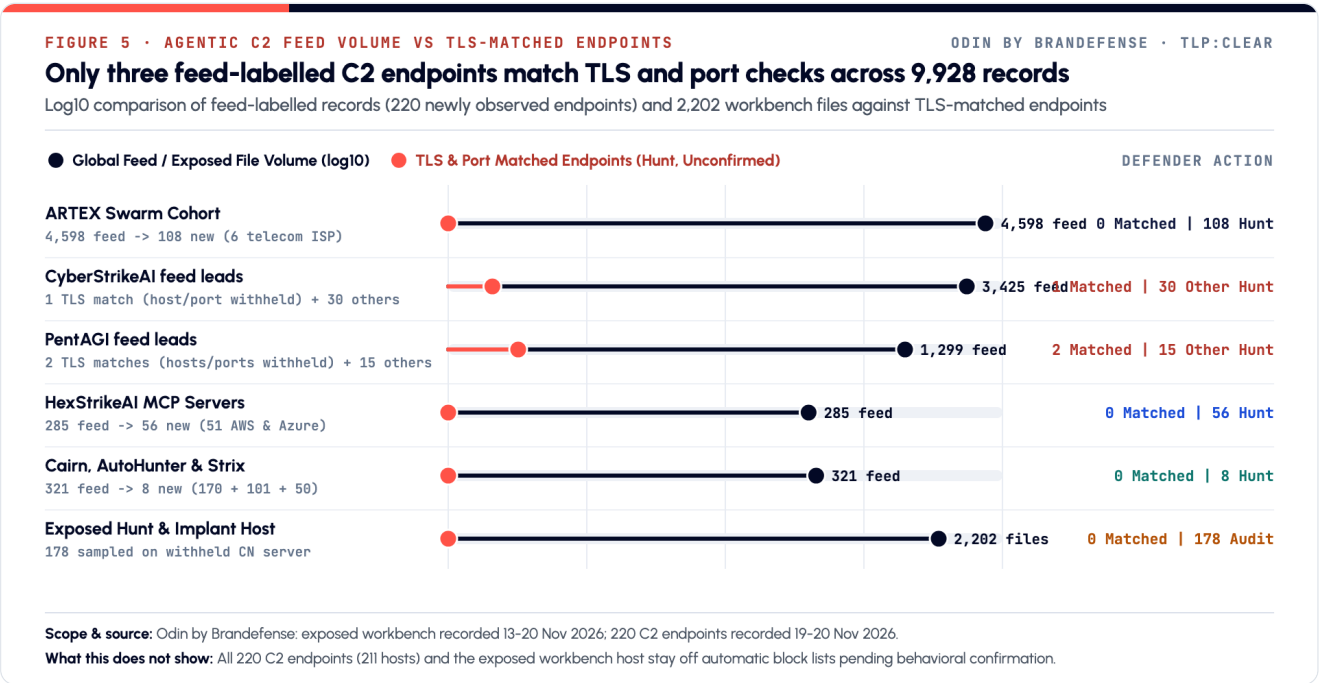


Figure 5: Only three feed-labelled C2 endpoints match TLS and port checks across 9,928 records

Framework	All feed records	Historical verified in feed	Newly observed endpoints	Distinct hosts	TLS & port matches (hunt, withheld)	Other unconfirmed (hunt, withheld)
ARTEX	4,598	83	108 (49.1%)	107	0	108 (6 telecom ISP, 1 open directory)
HexStrikeAI	285	0	56 (25.5%)	54	0	56 (54 suspected, 2 candidate; 51 AWS/Azure)
CyberStrikeAI	3,425	15	31 (14.1%)	27	1 (host, port and hosting withheld; `CN=CyberStrikeAI`)	30 candidates
PentAGI	1,299	23	17 (7.7%)	17	2 (hosts, ports and hosting withheld; `CN=localhost`)	15 (1 suspected, 13 candidate, 1 telecom ISP)
AutoHunter	101	1	5 (2.3%)	5	0	5 candidates (hosts and ports withheld)
Cairn	170	1	2 (0.9%)	2	0	2 (1 suspected, 1 candidate; hosts and ports withheld)
Strix	50	3	1 (0.5%)	1	0	1 candidate (host and port withheld)

Four other ways to read the evidence

1. **Contracted red team versus template reuse.** Including post-exploitation skill playbooks, keeping numbered scope folders and bundling a DOCX report-generation script fit a consultancy writing client reports. Co-hosting 93 CMS check scripts, cross-platform implant binaries and 49.8 MB of packet captures on an unattributed Chinese host also fits template reuse by an intruder. Because a directory listing cannot settle ownership, we withhold the address, service port, directory paths, file names and all 48 unique file hashes from that host.
2. **AWS and Azure container churn.** Fifty-one of 56 HexStrikeAI endpoints (91.1%) sit on Amazon AWS (36) and Microsoft Azure (15). Most return only generic FastAPI pages on ports 80, 5000 or 8080 (0 of 285 historical records verified), and cloud addresses move rapidly.
3. **Dynamic telecom ISP addresses.** Six ARTEX candidates and one PentAGI candidate sit on Chinese broadband and mobile networks. Blocking carrier addresses hurts unrelated subscribers after lease reassignment.
4. **Default certificates and co-hosted directories.** One ARTEX candidate uses a default Kubernetes `CN=k3s` certificate, and another shares its host with an open directory (hosts, ports and hosting withheld). Never block on `CN=localhost` or a JARM fingerprint alone.

07

CHAPTER 07

What Defenders Should Do

IN SHORT

This chapter lists the actions defenders should take, in priority order.

INSIDE THIS CHAPTER

Network and firewall 23

Cloud, build and red-team platforms 23

Endpoint, Windows and application security 23

Threat intelligence 23

07 What Defenders Should Do

Network and firewall

- [ ] Require independent protocol or beacon telemetry before blocking feed-labelled CyberStrikeAI or PentAGI addresses at the egress firewall, especially on high-churn cloud ranges.
- [ ] Inspect outbound HTTP requests for unauthenticated directory transfers of compiled Go binaries or gzip archives, and alert on TLS handshakes carrying `CN=CyberStrikeAI`.

Cloud, build and red-team platforms

- [ ] Scan your own cloud subnets, CI build runners and red-team hosts for `Directory listing for /` responses on high ports. Close any directory that exposes hidden agent dot-folders, staged binaries or compiler build caches.
- [ ] Block `python3 -m http.server` and similar ad-hoc HTTP listeners bound to `0.0.0.0` on hosts holding agent session logs, `.env` files or compiled offensive binaries.

Endpoint, Windows and application security

- [ ] Deploy the process and directory-listing review rules in Chapter 8 to catch ad-hoc HTTP servers on hosts holding agent workspaces or build caches.
- [ ] Audit Windows service accounts that hold `SeImpersonatePrivilege` (targeted by Potato-style playbooks) and patch internet-facing CMS installations. Move API keys and wallet secrets out of agent folders into a managed vault.

Threat intelligence

- [ ] Keep single-banner AWS/Azure candidates, Kubernetes `CN=k3s` collisions and dynamic telecom ISP matches out of automatic firewall block lists. Use the five JARM fingerprints in Chapter 9 for TLS hunting only.

CHAPTER 08

Detection Rules

IN SHORT

This chapter describes each published rule and where to deploy it.

INSIDE THIS CHAPTER

How the Suricata signatures work 25
How the other rules work 25
MITRE ATT&CK mapping 26

08 Detection Rules

The full rules are in Appendix B. The Suricata file passes a syntax lint. Suricata, YARA, Sigma and Kusto engines were not available to load the files for this edition, and the rules were not replayed against live traffic. Test and tune them in your own environment before production use.

Rule file	What it catches
`odin_hunt_workbench_and_agentic_c2.rules`	Suricata 7: two signatures for unauthenticated HTTP `Directory listing for /` responses (review) and `CN=CyberStrikeAI` server TLS certificates (hunt)
`sentinel_hunt_workbench_and_agentic_c2_hunt.kql`	Microsoft Sentinel and Defender XDR: ad-hoc HTTP directory server processes (`http.server`, `SimpleHTTPServer`, `-S 0.0.0.0`, `--bind 0.0.0.0`) on non-allowlisted hosts for review
`sigma_hunt_workbench_http_server_exposure.yml`	Sigma (1 rule): ad-hoc Linux HTTP directory server launches via Python, PHP or Node on non-allowlisted hosts for review
`odin_exposed_http_directory_listing.yar`	YARA (1 rule): captured HTML directory listing snapshots from unauthenticated HTTP servers exposing hidden dot-folders

How the Suricata signatures work

SID	Looks for	Why
9536003	An HTTP 200 response body containing `<title>Directory listing for /` and `Directory listing for /`	Review alert for an unauthenticated ad-hoc HTTP directory server
9536005	A server TLS certificate subject containing `CN=CyberStrikeAI`	Hunt only: CyberStrikeAI's default self-signed certificate

TLS 1.3 encrypts server certificates on the wire, so SID 9536005 fires on TLS 1.2 handshakes or behind a TLS-decrypting proxy. Keep SID 9536005 in alert-only mode where authorized red teams test CyberStrikeAI.

How the other rules work

The KQL query looks back seven days across `DeviceProcessEvents` and flags ad-hoc HTTP directory server processes (`http.server`, `SimpleHTTPServer`, `-S 0.0.0.0` or `--bind 0.0.0.0`) launched via Python, PHP, Node, Caddy or Nginx. Approved red-team and Go CI build hosts are excluded by exact device name only. Defenders should also audit their own agent dot-folders, session logs and Go compiler caches to ensure they are never served over HTTP.

Sigma rule	Level	Fires when
Ad-hoc Linux HTTP directory server exposure for review	High	A Python, PHP or Node process starts with `http.server`, `SimpleHTTPServer`, `-S 0.0.0.0` or `--bind 0.0.0.0` (review alert, exact lab hostname filter)

YARA rule	Fires when
`ODIN_Exposed_HTTP_Directory_Listing_Snapshot`	A captured file under 512 KB contains `<title>Directory listing for /` and `<h1>Directory listing for /` alongside a hidden dot-folder link (`href="."`)

MITRE ATT&CK mapping

Tactic	Technique	How it shows up	Detection
Credential Access	T1552.001 Unsecured Credentials: Credentials In Files	Keys, passwords and C2 endpoints in exposed agent session files on an unauthenticated HTTP directory	SID 9536003; KQL; Sigma; YARA
Command and Control	T1071.001 Application Layer Protocol: Web Protocols	Cross-platform HTTP implant staging and CyberStrikeAI or PentAGI HTTPS/GraphQL listeners	SID 9536005; JARM hunting

09

CHAPTER 09

Indicators

IN SHORT

This chapter lists the published indicators and explains what stays withheld.

INSIDE THIS CHAPTER

JARM fingerprints to hunt 28

Text-only hunting selectors 28

09 Indicators

Appendix A holds 5 published indicators: 5 JARM fingerprints. All 211 listener host addresses, the exposed workbench address and port, all workbench paths and file names, and all 48 workbench file hashes are withheld. One additional text-only hunting selector appears in the table below: CyberStrikeAI's default TLS certificate subject.

Type	Count	Where to use it
JARM fingerprints	5	TLS hunting only, never block on a JARM alone
Default TLS certificate subject	1	TLS and proxy hunting (`CN=CyberStrikeAI`)

JARM fingerprints to hunt

JARM	Framework	Context
`40d40d40d00000000043d43d00043d6aff5ab0f4fc31897186312c9e639319`	CyberStrikeAI	One unconfirmed feed lead (host, port and hosting withheld)
`3fd3fd00000000000043d43d00043dc3b2afa8a5ec09b510a8559aff7899fb`	PentAGI	Shared by two unconfirmed feed leads and two other TLS listeners (hosts and ports withheld)
`29d29d00029d29d00041d41d00041dcb3f8752a2d70d1dbf446fc46ab5df96`	HexStrikeAI	Shared by four suspected TLS endpoints (hosts and ports withheld)
`29d29d00029d29d21c42d42d000000c1fc3d47ef34532079f4922fcb05ef65`	CyberStrikeAI	Shared by four candidate TLS endpoints (hosts and ports withheld)
`3fd3fd0003fd3fd00042d42d0000002059a3b916699461c5923779b77cf06b`	ARTEX and CyberStrikeAI	Shared across four candidate TLS endpoints (hosts and ports withheld)

Text-only hunting selectors

Selector	Type	Use
`CN=CyberStrikeAI`	Default server TLS certificate subject	TLS hunting (SID 9536005, alert only)

Withheld: the exposed workbench IPv4 address, its service port, its directory paths and file names, and all 48 unique SHA-256 file hashes recorded on that unestablished host. We also withhold all 211 listener host addresses and individual host ports, including the three feed-labelled endpoints that lack independent behavioral confirmation. Unverified server certificate names and every secret value are withheld as well.

10

CHAPTER 10

Scope and Limits

IN SHORT

This chapter sets out what the evidence can show and what it cannot.

10 Scope and Limits

- **Window.** Workbench records were recorded from 13 to 20 November 2026, and the 220 autonomous-offense listener endpoints were recorded on 19 and 20 November 2026. The dated work folder beneath the agent session directory is a server folder name, not our observation period.
- **Feed labels are not attribution.** Names such as ARTEX, CyberStrikeAI or PentAGI come from upstream feeds. Even the three endpoints with certificate, JARM and port matches remain unconfirmed without behavioral or protocol evidence.
- **The model is a model.** Zero shared IPv4 addresses or certificates link the exposed workbench to the 211 listener hosts. Figures 1 and 3 show an analytical model, not one operator.
- **Workbench owner not established.** The post-exploitation playbooks and DOCX report-generation script fit either a contracted red team or template reuse. The server's IPv4 address, service port, directory paths, file names and all 48 unique hashes on that host are therefore withheld.
- **Staged files are not target outcomes.** Staged CMS scan results, cross-platform implant binaries and packet capture logs show staged files only, and no binary or key was tested.
- **Cloud addresses move.** Many unconfirmed feed leads, including 51 of 56 HexStrikeAI endpoints, sit on hyperscale cloud ranges where IPv4 leases move rapidly.
- **Rule testing.** Rules were not loaded on live engines or replayed against live traffic for this edition; test them locally first.

11

CHAPTER 11

Sources

IN SHORT

This chapter lists the public references used in the report.

11 Sources

1. MITRE ATT&CK. S0633 Sliver. https://attack.mitre.org/software/S0633/ (verified on 21 November 2026)
2. MITRE ATT&CK. T1552.001 Unsecured Credentials: Credentials In Files. https://attack.mitre.org/techniques/T1552/001/ (verified on 21 November 2026)
3. MITRE ATT&CK. T1071.001 Application Layer Protocol: Web Protocols. https://attack.mitre.org/techniques/T1071/001/ (verified on 21 November 2026)
4. OWASP Cheat Sheet Series. Secrets Management Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html (verified on 21 November 2026)
5. Go Documentation. Command go: Build and Test Caching. https://pkg.go.dev/cmd/go (verified on 21 November 2026)
6. Python Documentation. http.server: HTTP servers. https://docs.python.org/3/library/http.server.html (verified on 21 November 2026)
7. Salesforce Engineering. JARM: An Active Transport Layer Security (TLS) Server Fingerprinting Specification. https://github.com/salesforce/jarm (verified on 21 November 2026)

APPENDICES

Indicators and Detection Rules

FOR DEFENDERS

Appendix A lists the indicators published for this study. Appendix B carries every published detection rule in full.

INSIDE

A. Indicators of compromise (5) 34

B. Detection rules (4 files) 35

A Indicators of Compromise

Values are defanged for display. The CSV, JSON and STIX files on the online edition carry the raw values. Values the report marks as withheld are not listed.

ID	Type	Indicator	Port	Context	Action
R36-IOC-01	JARM	40d40d40d00000000043d43d00043d6aff5ab0f4fc31897186312c9e639319	withheld	CyberStrikeAI · TLS fingerprint of one unconfirmed feed lead (host, port and hosting withheld)	HUNT
R36-IOC-02	JARM	3fd3fd00000000000043d43d00043dc3b2afa8a5ec09b510a8559aff7899fb	withheld	PentAGI · TLS fingerprint shared by two unconfirmed feed leads and two other TLS listeners (hosts and ports withheld)	HUNT
R36-IOC-03	JARM	29d29d00029d29d00041d41d00041dcb3f8752a2d70d1dbf446fc46ab5df96	withheld	HexStrikeAI · TLS fingerprint shared by four suspected TLS endpoints (hosts and ports withheld)	HUNT
R36-IOC-04	JARM	29d29d00029d29d21c42d42d000000c1fc3d47ef34532079f4922fcb05ef65	withheld	CyberStrikeAI · TLS fingerprint shared by four candidate TLS endpoints (hosts and ports withheld)	HUNT
R36-IOC-05	JARM	3fd3fd0003fd3fd00042d42d0000002059a3b916699461c5923779b77cf06b	withheld	ARTEX and CyberStrikeAI · TLS fingerprint shared across four candidate TLS endpoints (hosts and ports withheld)	HUNT

B Detection Rules

Each rule file loads in its own engine (YARA-X, Suricata 7, Kusto, pySigma). Tune thresholds and allow-lists to your environment before alerting or blocking. The same files can be downloaded from the online edition.

odin_exposed_http_directory_listing.yar

YARA

```
/*
   Odin by Brandefense - Monograph 36 / 46 YARA Artifact Suite (public edition v1.1)
   Scope: Generic review rule for captured unauthenticated HTTP directory listing snapshots
          exposing hidden dot-folders or compiler cache directories.
   Classification: TLP:CLEAR
*/

rule ODIN_Exposed_HTTP_Directory_Listing_Snapshot
{
    meta:
        author = "Odin by Brandefense"
        monograph = "36 / 46"
        classification = "TLP:CLEAR"
        action = "REVIEW_EXPOSED_HTTP_DIRECTORY_LISTING"
        description = "Matches captured HTML directory listings from unauthenticated HTTP servers exposing hidden
dot-folders"
    strings:
        $hdr1 = "<title>Directory listing for /" ascii nocase
        $hdr2 = "<h1>Directory listing for /" ascii nocase
        $dot = "href=\"." ascii nocase
    condition:
        filesize < 512KB and $hdr1 and $hdr2 and $dot
}
```

odin_hunt_workbench_and_agentic_c2.rules

SURICATA

```
# Odin by Brandefense - Monograph 36 / 46 Suricata 7.x Detection Suite (public edition v1.1)
# Coverage: Unauthenticated HTTP directory listing responses for review and
#           CyberStrikeAI default server TLS certificates for hunting
# Classification: TLP:CLEAR

alert http any any -> any any (msg:"ODIN [R36] REVIEW Unauthenticated HTTP Directory Listing Response";
flow:established,to_client; http.stat_code; bsize:3; content:"200"; http.response_body; bsize:>48;
content:"<title>Directory listing for /"; fast_pattern; content:"Directory listing for /"; distance:0;
classtype:policy-violation; sid:9536003; rev:2; metadata:created_at 2026_11_20, deployment
perimeter_and_east_west, policy behavioral_hunt, tactic credential_access;)

alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN [R36] HUNT CyberStrikeAI Default TLS Server Certificate
Subject (CN=CyberStrikeAI)"; flow:established,to_client; tls.cert_subject; bsize:>15; content:"CN=CyberStrikeAI";
fast_pattern; classtype:trojan-activity; sid:9536005; rev:2; metadata:created_at 2026_11_20, deployment
perimeter, policy behavioral_hunt;)
```

sentinel_hunt_workbench_and_agentic_c2_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Monograph 36 / 46 Microsoft Sentinel & Defender XDR Hunt (public edition v1.1)
// Reviews ad-hoc HTTP directory servers (such as python3 -m http.server or php -S 0.0.0.0)
// launched on non-allowlisted hosts that could expose local agent workspaces or compiler build caches.
// Host allowlists hold exact device names only.
// Classification: TLP:CLEAR

let Lookback = 7d;
let ApprovedRedTeamHosts = dynamic(["REDTEAM-HUNT-BOX-01", "PURPLE-SIM-NODE-01"]);
```

```
let ApprovedBuildHosts = dynamic(["GO-CI-RUNNER-01"]);
let AdHocHttpProcesses = dynamic(["python", "python.exe", "python3", "python3.exe", "php", "php.exe", "node",
"node.exe", "caddy", "nginx"]);
let AdHocHttpMarkers = dynamic(["http.server", "SimpleHTTPServer", "-S 0.0.0.0", "--bind 0.0.0.0"]);
DeviceProcessEvents
| where Timestamp >= ago(Lookback)
| where not(DeviceName in~ (ApprovedRedTeamHosts)) and not(DeviceName in~ (ApprovedBuildHosts))
| where FileName in~ (AdHocHttpProcesses) and ProcessCommandLine has_any (AdHocHttpMarkers)
| summarize
    FirstSeen = min(Timestamp),
    LastSeen = max(Timestamp),
    LaunchCount = count(),
    SampleCommands = make_set(ProcessCommandLine, 10),
    SampleFolders = make_set(FolderPath, 10)
    by DeviceId, DeviceName, InitiatingProcessFileName, FileName
| extend AlertAction = "REVIEW_ADHOC_HTTP_SERVER_EXPOSURE"
| project LastSeen, FirstSeen, DeviceName, AlertAction, InitiatingProcessFileName, FileName, LaunchCount,
SampleFolders, SampleCommands
| sort by LastSeen desc
```

sigma_hunt_workbench_http_server_exposure.yml

SIGMA

```
title: Ad-Hoc Linux HTTP Directory Server Exposure for Review
id: 9c38b5a1-6e24-4d09-8b17-202611203602
status: experimental
description: Detects ad-hoc Linux HTTP directory servers launched via Python, PHP or Node on non-allowlisted
hosts for review of exposed agent workspaces or compiler build caches.
author: Odin by Brandefense
date: 2026/11/20
modified: 2026/11/22
tags:
  - attack.credential-access
  - attack.t1552.001
logsource:
  category: process_creation
  product: linux
detection:
  selection_http_workbench:
    Image|endswith:
      - '/python'
      - '/python3'
      - '/php'
      - '/node'
    CommandLine|contains:
      - 'http.server'
      - 'SimpleHTTPServer'
      - '-S 0.0.0.0'
      - '--bind 0.0.0.0'
  filter_approved_lab:
    Computer:
      - 'REDTEAM-HUNT-BOX-01'
      - 'PURPLE-SIM-NODE-01'
  condition: selection_http_workbench and not 1 of filter_*
falsepositives:
  - Approved red-team Linux workbench servers listed by exact hostname in the filter and legitimate developer
local HTTP servers reviewed for exposed folders
level: high
```

ODIN

About Odin Research

Odin Research is the public research series of Odin by Brandefense. It documents adversary infrastructure seen on the internet and turns each finding into indicators and detection rules that defenders can use. Victims and third parties stay anonymous, addresses appear only for confirmed adversary infrastructure, and no working attack code is published.

Odin is the agentic analyst inside the Brandefense threat intelligence platform, which unifies Cyber Threat Intelligence, Digital Risk Protection and External Attack Surface Management.

Online edition, indicators and rules · brandefense.io/odin-research/reports/autonomous-hunt-agents-and-cross-platform-implants/
Short Read · brandefense.io/odin-research/blog/autonomous-hunting-agents-implants-and-c2s/
Defense Kit · brandefense.io/odin-research/defense-kit/