ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 11 / 30 · AI OFFENSE

White-Labeled AI Offense
Forks, Rebrands, and Swarm Targeting

How five autonomous red-team frameworks are rebranded, co-hosted and pointed at fintech and Web3 targets

Two-week observation window ending 18 November 2026 · Prepared November 2026
Full Report · Public edition · Version 1.1

4,360
Endpoints running one of five AI offense frameworks, on 2,917 IPv4 hosts in 50 countries and 286 networks

1,846
Confirmed endpoints (42.3%); 1,820 of them sit on stable hosting and are ready to block

103
Hosts that carry two or more framework labels

76
Target organizations in one exposed ARTEX campaign archive, across 18 verticals

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

How five autonomous red-team frameworks are rebranded, co-hosted and pointed at fintech and Web3 targets

01	Executive Summary	04
02	How the Swarms Work	07
03	Fleet Footprint: Where the Consoles Run	11
04	White-Label Forks: Same Engine, New Paint	15
05	One Server, Many Tools: Co-Hosting	18
06	Who the Swarm Targeted	22
07	What Defenders Should Do	25
08	Detection Rules	27
09	Indicators	30
10	Scope and Limits	33
11	Sources	35
A	Indicators of compromise (14)	37
B	Detection rules (4 files)	38

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense found 4,360 endpoints running five autonomous AI offense frameworks. Of these, 1,846 are confirmed. Operators rebrand ARTEX consoles for regional markets, stack several frameworks on one server, and run up to 16 parallel workers per target. An exposed swarm archive covers 76 organizations, mostly in payments and Web3. Block the 1,820 confirmed endpoints on stable hosting, hunt the shared ARTEX favicon hash, and alert on 25-URI scan bursts.

01

CHAPTER 01

Executive Summary

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense found 4,360 endpoints running five autonomous AI offense frameworks. Of these, 1,846 are confirmed. Operators rebrand ARTEX consoles for regional markets, stack several frameworks on one server, and run up to 16 parallel workers per target. An exposed swarm archive covers 76 organizations, mostly in payments and Web3. Block the 1,820 confirmed endpoints on stable hosting, hunt the shared ARTEX favicon hash, and alert on 25-URI scan bursts.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 04

01 Executive Summary

Autonomous offense frameworks put a large language model (LLM) in charge of classic scanners such as `subfinder`, `nuclei` and `sqlmap`. The operator gives a target. The model plans the work and hands it to many parallel workers. This report maps where these consoles run, how they are disguised, and what one exposed campaign archive shows about their targets.

1. **Five AI offense frameworks run on 4,360 endpoints across 2,917 hosts.**
   ARTEX (2,121 endpoints) and CyberStrikeAI (1,720) dominate. PentAGI (340), HexStrikeAI (118) and DeepAudit (61) follow. Of the 4,360 endpoints, 1,846 are confirmed, 117 suspected and 2,397 candidates.

2. **Rebranded ARTEX forks change titles and certificates but keep the favicon.**
   Two Korean-localized consoles on AWS Seoul serve the same 15,406-byte favicon as upstream ARTEX controllers in China. Separately, 103 hosts carry two or more framework labels. The clearest case, 38.127.244[.]14, runs PentAGI, an ARTEX port and an upload-bypass dropzone side by side.

3. **One swarm archive shows heavy targeting of payments and Web3 firms.**
   An exposed ARTEX archive lists 76 target organizations and 13,864 enumerated subdomains. Payments and Fintech (21) and Web3, DeFi and Stablecoin (20) make up 53.9% of the scopes. The swarm's own reports label 35 scopes critical or high; these labels are unverified.

Three Decisions for Leaders

1. **Block confirmed controllers, but keep review lanes.** Block the 1,820 confirmed endpoints on stable hosting networks. Put the 30 endpoints on dynamic telecom ISPs, the 3 endpoints under analyst review and all candidates on time-bound watchlists instead.
2. **Hunt fingerprints that operators do not change.** Page titles and logos change with every rebrand. The favicon hash, the TLS certificate subject, the JARM value and the API routes stay the same, so build detection on those.
3. **Harden upload folders and throttle scan bursts.** Disable `.htaccess` overrides on upload directories to stop image-as-PHP polyglots. Alert when one source hits 25 or more distinct URIs in 5 minutes.

The Study in Numbers

Number	What it is
4,360	Endpoints running one of five AI offense frameworks, on 2,917 IPv4 hosts in 50 countries and 286 networks
1,846	Confirmed endpoints (42.3%); 1,820 of them sit on stable hosting and are ready to block
103	Hosts that carry two or more framework labels
76	Target organizations in one exposed ARTEX campaign archive, across 18 verticals
13,864	Subdomains the swarm enumerated for those 76 targets; 1,105 were live
46.1%	Share of the 76 scopes (35) the swarm labeled critical or high, unverified
2,178	Worker runs the swarm recorded against a single target

How to use this report.

Security leaders can read Chapter 1, Chapter 6 and Chapter 7. SOC and detection engineers should start with Chapters 7 to 9. CTI analysts will find the fork, co-hosting and infrastructure detail in Chapters 2 to 5.

02

CHAPTER 02

How the Swarms Work

IN SHORT

These frameworks split one target into many small jobs and run them in parallel worker pools. The pools hand results forward, from subdomain discovery to template scanning, exploitation checks and an LLM-written report.

INSIDE THIS CHAPTER

From script chains to agent workbenches 07

The four worker pools 07

How hard the swarm worked 09

Read the archive with care 09

02 How the Swarms Work

From script chains to agent workbenches

Attackers used to glue reconnaissance tools together with their own scripts. The frameworks in this report replace that with a web workbench. The workbench exposes a REST, GraphQL or Model Context Protocol (MCP) API to an LLM planning loop. MCP is an open JSON-RPC protocol that lets a model call external tools.

Each framework takes a slightly different approach. The table below lists the architecture and the default ports.

Framework	How it is built	Default ports
ARTEX	Python multi-worker swarm with a pool of 1 to 16 subagents and a SQLite or Postgres task queue	8787, 443, 80, 7878
CyberStrikeAI	Go and Python platform with role-based agents wrapping 100+ security tools	5001 to 5003, 8080, 8082, 443, 8899
PentAGI	Go controller with a GraphQL API that starts Docker-in-Docker Kali workers and keeps memory in `pgvector`	8443, 8666, 443
HexStrikeAI	Python FastAPI server that exposes `nmap`, `nuclei`, `sqlmap` and `ffuf` as MCP tools	5000, 80, 8080, 8000, 8888
DeepAudit	LLM workbench for auditing source code and repositories for vulnerabilities	8000, 3000, 80

PentAGI, CyberStrikeAI and HexStrikeAI are public projects on GitHub. AhnLab ASEC has published an infrastructure analysis of ARTEX, and Team Cymru has tracked CyberStrikeAI infrastructure.

The four worker pools

The exposed ARTEX archive, together with the public PentAGI and HexStrikeAI designs, shows a four-stage pipeline. A controller receives a scope through `POST /api/v1/tasks` (ARTEX) or a GraphQL `createFlow` call (PentAGI). It then queues work to four pools.



Figure 1: Autonomous swarms split recon, CVE scanning, exploitation checks and reporting across four worker pools

Pool	Job	Tools	What the archive records
01 Recon	Expand the attack surface and keep only live hosts	`subfinder`, `naabu`, `httpx`, `dnsx`	13,864 subdomains (182.4 per target), narrowed to 1,105 live HTTP or TLS FQDNs (8.0% yield)
02 Vulnerability scan	Run CVE templates and directory fuzzing on live hosts	`nuclei`, `ffuf`, `gobuster`, `katana`	Sweeps for `.env` files, `.git/config`, Swagger or OpenAPI specs, actuator and GraphQL endpoints
03 Exploit checks	Test injectable parameters and upload forms	Kali Docker-in-Docker, `sqlmap`, MCP tool calls, polyglot uploads	Claimed outputs written to `credential_inventory_01.json` and `findings_summary.json`
04 Synthesis	Turn task logs into a report	LLM writer, Markdown, JSON	35 of 76 scopes labeled critical or high

Pool 02 workers run with non-interactive batch flags such as `-jsonl -silent -rate-limit 150`. These flags matter for detection, because they show up in process command lines on the worker host.

In Pool 03, PentAGI starts Kali containers through `containerd-shim-runc-v2`. HexStrikeAI and CyberStrikeAI instead define MCP `tools/call` schemas for `sqlmap --batch` and polyglot upload probes. In both cases, an LLM decides which tool to call next.

How hard the swarm worked

The archive records how many worker runs the swarm spent on single targets. These are the swarm's own counts.

Sector	Recorded worker runs
Crypto exchange	2,178
Online gaming	1,231
Not stated	535
DeFi	477

Read the archive with care

The archive is the attacker's own record, so two limits apply. First, it shows what the tools were configured and logged to do. We did not see these actions on any target system. Second, the severity totals come from an LLM report writer. Such writers often call an unauthenticated banner or an exposed `.env` hit "critical". Treat the 17 critical and 18 high labels as claims.

The archive server also exposed real target domains and harvested secrets. Its address, port and directory paths are withheld in this report. Sizes of the archive files stay in the Indicators chapter; their hashes are withheld.

CHAPTER 03

Fleet Footprint: Where the Consoles Run

IN SHORT

The 4,360 endpoints sit mostly on low-cost VPS networks in the United States, Hong Kong and China. One provider, PEG TECH INC, hosts 1,891 of them.

INSIDE THIS CHAPTER

Counts by framework 11
Why PentAGI and HexStrikeAI look small 11
Countries and hosting networks 12
What not to block 13

03 Fleet Footprint: Where the Consoles Run

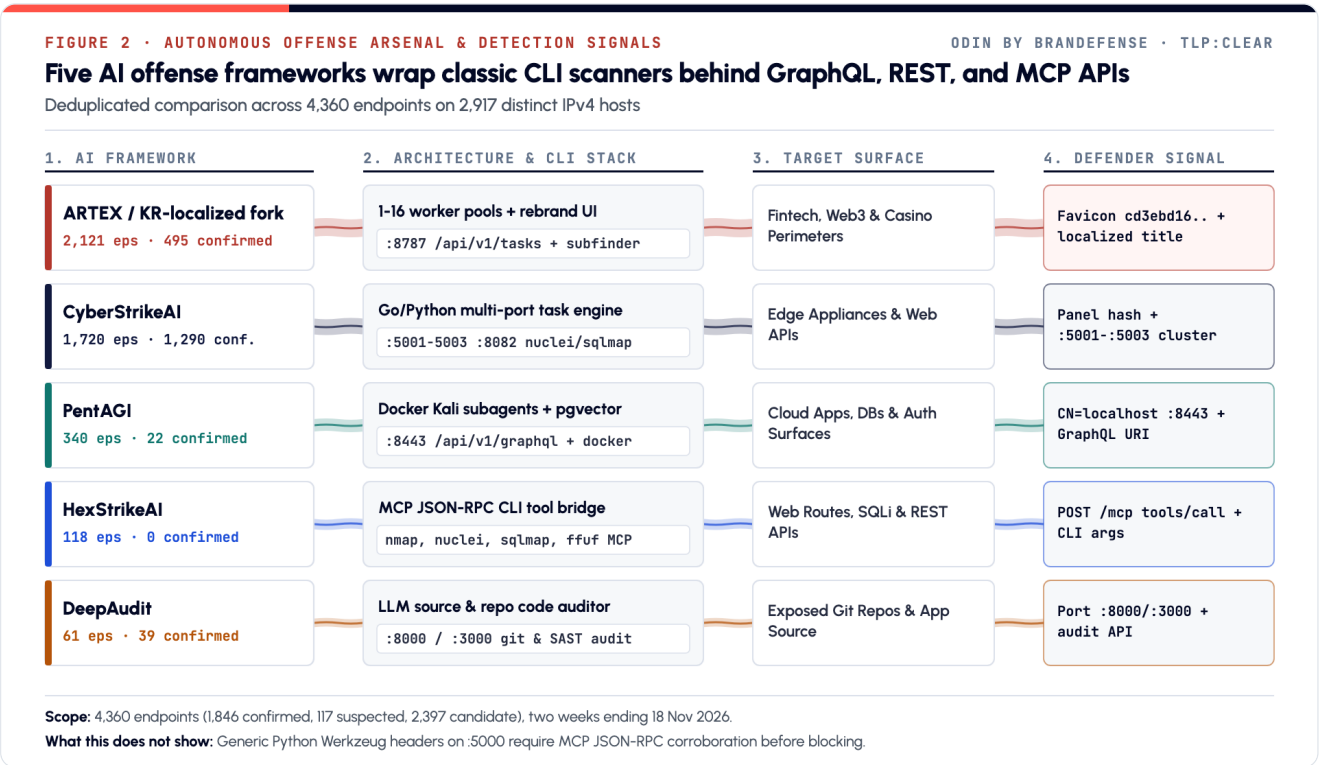


Figure 2: Five AI offense frameworks wrap classic CLI scanners behind GraphQL, REST and MCP APIs

Counts by framework

Each endpoint is an IP, port and framework combination.

Framework	Endpoints	Hosts	Confirmed	Suspected	Candidate
ARTEX (incl. Korean fork)	2,121	1,419	495	14	1,612
CyberStrikeAI	1,720	1,142	1,290	0	430
PentAGI	340	333	22	4	314
HexStrikeAI	118	100	0	99	19
DeepAudit	61	26	39	0	22
Total	**4,360**	**2,917**	**1,846**	**117**	**2,397**

CyberStrikeAI has the highest confirmed share at 75.0%. Its listeners share the JARM value `40d40d...9319`. DeepAudit also appears next to HexStrikeAI and ARTEX on the same servers, as on 38.76.218[.]100.

Why PentAGI and HexStrikeAI look small

A responsive panel is not a confirmed deployment. Of the 340 PentAGI endpoints, only 22 are confirmed. The other 314 are candidates. We make no claim that PentAGI deployments grew over time.

HexStrikeAI is harder still. Many installs show only a generic Python Werkzeug or FastAPI page on 5000, 80 or 8080. Nothing else on these pages ties them to the framework. So none of the 118 HexStrikeAI endpoints is confirmed: 99 are suspected and 19 are candidates.

Countries and hosting networks

The fleet spans 50 countries and 286 networks. Six countries lead.

Country	Endpoints	Share
United States	2,586	59.31%
Hong Kong	697	15.99%
China	444	10.18%
Japan	107	2.45%
Singapore	95	2.18%
Germany	82	1.88%

The top ten hosting networks carry 2,968 of the 4,360 endpoints (68.07%). PEG TECH INC alone hosts 1,891 (43.37%).

Rank	Hosting network	Countries	Endpoints	Confirmed	Suspected	Candidate	Main frameworks
1	PEG TECH INC	US, JP, SG	1,891	1,124	0	767	CyberStrikeAI, ARTEX, PentAGI
2	YANCY LIMITED	HK, US	237	237	0	0	ARTEX
3	Shenzhen Tencent Computer Systems	CN, US, JP	176	25	12	139	ARTEX, CyberStrikeAI, HexStrikeAI
4	Hangzhou Alibaba Advertising	CN	131	17	10	104	ARTEX, CyberStrikeAI, DeepAudit
5	AROSSCLOUD INC.	US, HK	122	92	0	30	CyberStrikeAI, ARTEX, DeepAudit
6	DigitalOcean, LLC	US, DE, NL	116	96	5	15	CyberStrikeAI, ARTEX, HexStrikeAI
7	cognetcloud INC	HK, US, IN	81	17	4	60	ARTEX, CyberStrikeAI, HexStrikeAI
8	CTG Server Limited	HK, JP, SG	79	15	0	64	ARTEX, CyberStrikeAI, PentAGI
9	NetLab Global	HK, US, JP	71	6	1	64	ARTEX, CyberStrikeAI, PentAGI
10	HostPapa	US, CA	64	20	3	41	ARTEX, CyberStrikeAI, DeepAudit

What not to block

Some listeners sit on dynamic home or small-business broadband. These include Korea Telecom (AS4766), Chunghwa Telecom (AS3462) and Chinanet (AS4134). Their addresses rotate between unrelated customers. The 30 endpoints in this group (23 confirmed, 7 suspected) belong on watchlists that expire after 24 to 72 hours, not on permanent block lists. Their addresses are not published.

Port 5000 also causes false positives. One suspected HexStrikeAI host turned out to serve a benign replica of a third-party security training portal. It shared only a Werkzeug header with HexStrikeAI. Its favicon did not match, and it is excluded. Generic Python headers on 5000 need an MCP signal before anyone blocks them.

The blocking picture for confirmed endpoints is shown below.

Group	Endpoints	Action
Confirmed on stable hosting	1,820 (41.74%)	Block
Confirmed on dynamic telecom ISPs	23	Short watchlist
Confirmed framework, hostile use not shown	3 (2 Korean fork nodes, 1 probable authorized tester)	Analyst review
Suspected on stable hosting	110	Watch, look for a second signal
Suspected on dynamic telecom ISPs	7	Short watchlist
Candidates	2,397 (15 of them with an open directory)	Hold until corroborated

By framework, the 1,820 block-ready endpoints are ARTEX 492, CyberStrikeAI 1,272, PentAGI 19 and DeepAudit 37.

04

CHAPTER 04

White-Label Forks: Same Engine, New Paint

IN SHORT

When operators fork an AI offense console, they change the page title, the logo and the TLS certificate. They rarely touch the compiled frontend, the API routes or the favicon file, which is what gives the fork away.

INSIDE THIS CHAPTER

The Korean ARTEX fork 15

The link to upstream ARTEX 16

Who runs the Korean fork 16

A localized portal that does not qualify 16

04 White-Label Forks: Same Engine, New Paint

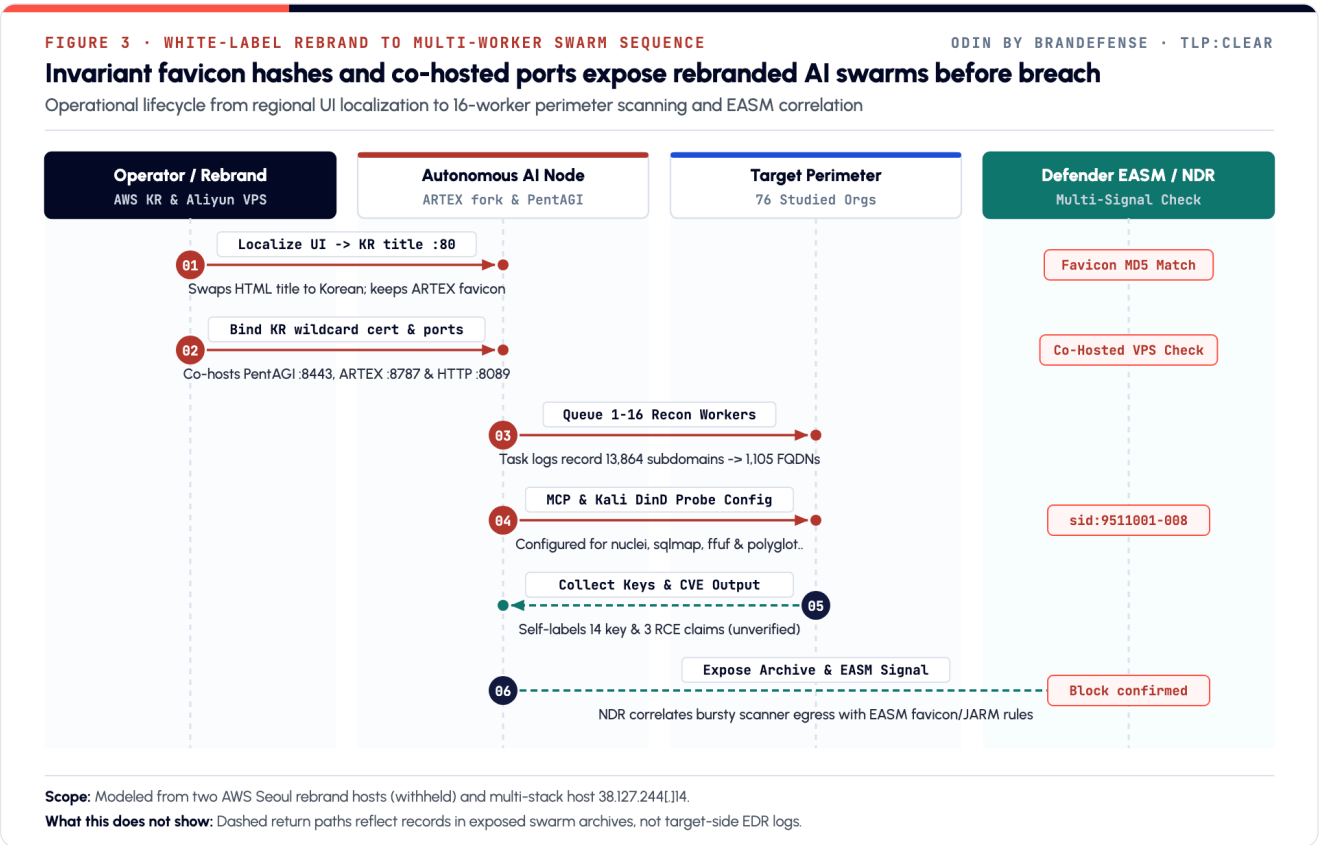


Figure 3: Invariant favicon hashes and co-hosted ports expose rebranded AI swarms

The Korean ARTEX fork

On 17 November 2026, two hosts on Amazon AWS Seoul (`ap-northeast-2`, AS16509) showed a localized ARTEX console. Their addresses and the Korean brand name are withheld, for reasons explained below.

Node A, on port 80, removed every visible "ARTEX" string from its title. It shows a Korean title ending in `공격·방어 보안 운영` ("Attack and Defense Security Operations"). That subtitle is a word-for-word translation of the Chinese upstream subtitle `攻防安全运营`.

Node B, on port 443, kept the English title `ARTEX - Autonomous Red-Team Expert`. It sits behind a wildcard certificate for a South Korean company domain, with JARM `28d28d28d00028d00042d42d000000831b6af40378e2dd35eeac4e9311926e`.

The link to upstream ARTEX

Both Korean nodes serve a 15,406-byte `/favicon.ico` with MD5 `cd3ebd16437d6d3963592f7365dd3980`. The same file appears on upstream ARTEX controllers. Both nodes also mount the single-page app in `<div id="artex-swarm-root">` and expose `/api/v1/tasks`.

Role	Endpoint	Hosting	Page title	TLS subject and JARM
Korean fork, node A	Withheld, port 80	AS16509 AWS Seoul (KR)	Korean title ending `공격·방어 보안 운영`	No TLS
Korean fork, node B	Withheld, port 443	AS16509 AWS Seoul (KR)	`ARTEX - Autonomous Red-Team Expert`	Korean company wildcard, `28d28d...926e`
Upstream ARTEX, Chinese	8.135.3[.]123:80	AS37963 Alibaba Cloud (CN)	`ARTEX - 攻防安全运营`	No TLS
Upstream ARTEX, Chinese	8.135.30[.]162:443	AS37963 Alibaba Cloud (CN)	`ARTEX - 攻防安全运营`	`CN=8.135.30[.]162`, `15d3fd...e331`
Upstream ARTEX, English	45.156.87[.]185:443	AS399629 Rodgers Inv. (US)	`ARTEX - Autonomous Red-Team Expert`	`CN=artex.dwwork[.]live`, `2ad2ad...f06b`

All five share the favicon hash, the `#artex-swarm-root` mount point and the `/api/v1/tasks` route.

Who runs the Korean fork

We weighed three explanations.

Hypothesis	For	Against	Verdict
H1: A threat actor rebranded ARTEX	Internet-exposed console on default ports; shares the ARTEX fingerprint and JARM	The wildcard certificate belongs to a registered Korean company; no hostile traffic seen from these hosts	Plausible, unproven
H2: A Korean security vendor white-labeled ARTEX for clients	Literal translation of the product subtitle; AWS Seoul hosting under a company domain	Leaving the console exposed without mutual TLS is poor practice for a vendor	Plausible
H3: Unrelated web app that happens to match	None	Exact 15,406-byte favicon match and identical DOM and API layout	Rejected

The 76-organization archive in Chapter 2 came from a separate server in Europe. Nothing in it points to the Korean nodes or their domain, so it does not argue against H2.

Because a legitimate vendor (H2) remains plausible, we withhold the brand, the certificate domain and both addresses. Treat this fingerprint as "send to analyst review", not "block", until hostile activity is documented.

A localized portal that does not qualify

Not every localized portal is a fork. A host on Alibaba Cloud (CN), port 8080, shows a Chinese login page titled `用户登录 - 智测工具` ("User Login - Smart Testing Tool"). It is listed as a suspected HexStrikeAI node.

It did not serve the ARTEX favicon, and nothing else ties it to the framework. We treat it as a suspected local testing workbench and withhold its address.

05

CHAPTER 05

One Server, Many Tools: Co-Hosting

IN SHORT

Operators often put several AI offense consoles and a payload dropzone on one rented server. Of 2,917 hosts, 103 carry two or more framework labels, and one hub shows the full pattern.

INSIDE THIS CHAPTER

The 103 multi-label hosts 18

Case study: 38.127.244[.]14 18

What the dropzone held 20

05 One Server, Many Tools: Co-Hosting

The 103 multi-label hosts

Not every multi-label host is a confirmed multi-framework deployment. The breakdown below separates them by strength of evidence.

Category	Hosts	Share	Example
Two confirmed frameworks	1	0.97%	144.172.68[.]197: ARTEX on 80 and CyberStrikeAI on 8443
One confirmed framework plus a weaker companion	70	67.96%	38.76.218[.]100: DeepAudit on 8000, suspected HexStrikeAI on 5000; 45.144.137[.]148: ARTEX on 80, workbench on 8080
Two or more suspected or candidate frameworks, none confirmed	31	30.10%	VPS clusters on PEG TECH INC and Cloudie Limited (8787 with 8443 or 8666)
Flagship hub with confirmed PentAGI, a dropzone and a candidate ARTEX port	1	0.97%	38.127.244[.]14
Total	**103**	**100.0%**	

Together they make up 3.53% of all hosts.

Case study: 38.127.244[.]14

This host sits on Cogent Communications / Tcloudnet in the United States (AS399077). It combines an AI orchestrator with an exploit test dropzone.

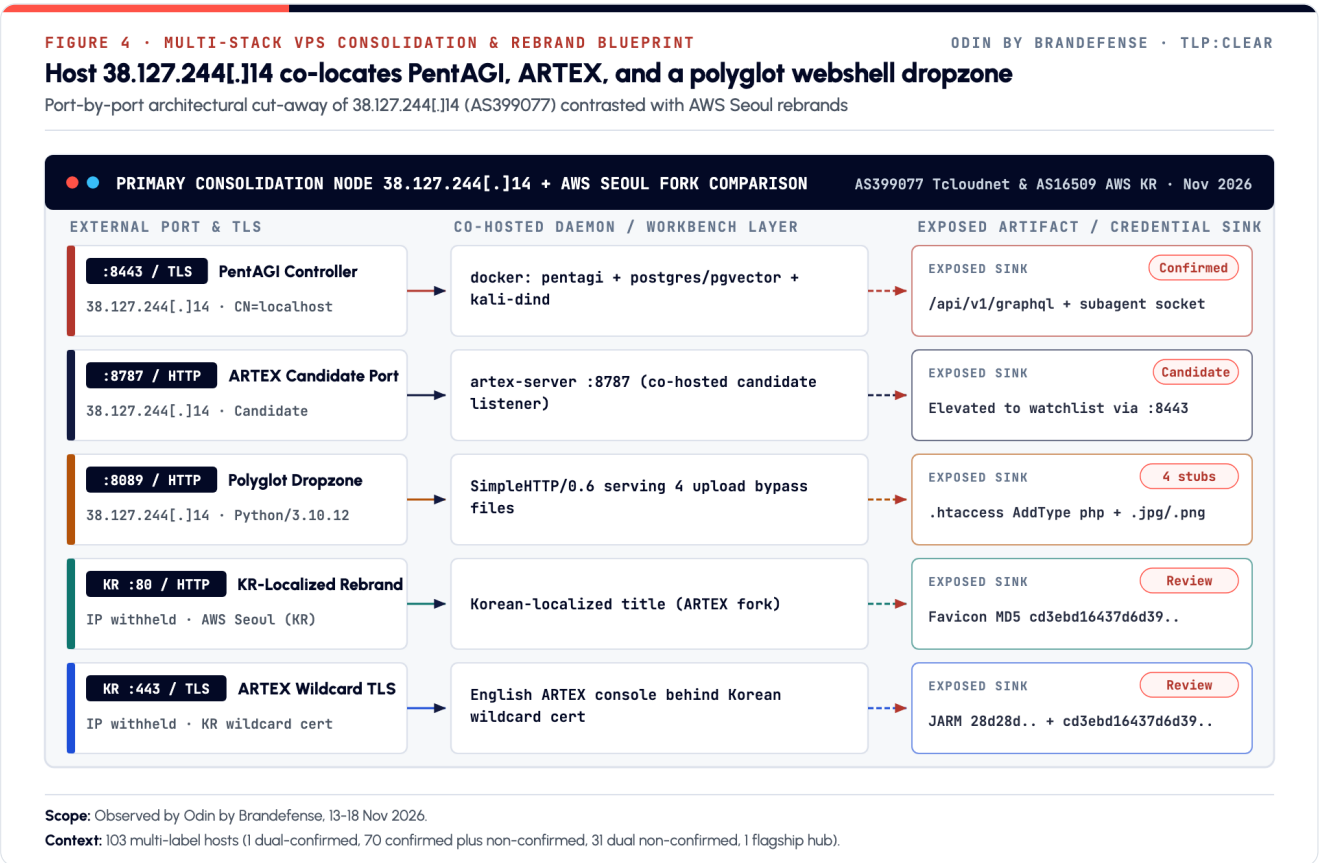


Figure 4: Host 38.127.244[.]14 co-locates PentAGI, ARTEX and a polyglot webshell dropzone

Port	Service	Fingerprint	Status
`8443/tcp`	PentAGI orchestrator	`<title>PentAGI</title>`, `CN=localhost`, JARM `28d28d28d00028d00042d42d000000831b6af40378e2dd35eeac4e9311926e`, `/api/v1/graphql`	Confirmed
`8787/tcp`	ARTEX listener	ARTEX default port	Candidate, watched because it shares a host with confirmed PentAGI
`8089/tcp`	Open directory dropzone	`SimpleHTTP/0.6 Python/3.10.12`	Open directory confirmed

The PentAGI certificate subject `CN=localhost` is common across the fleet. It appears on 61 of the 340 PentAGI endpoints (8 confirmed, 53 candidates). Upstream default deployments present `CN=pentagi` instead.

What the dropzone held

Port 8089 served an unauthenticated directory. It held four small test stubs: `.htaccess`, `test_polyglot.png`, `avatar_loader.jpg` and `health.txt`. It also held larger staged files from the subagents.

File	Size (bytes)	SHA-256	Purpose
`/.htaccess`	161	Withheld	Adds `AddType application/x-httpd-php .jpg .png .gif`, so a permissive Apache server would run uploaded images as PHP (T1505.003)
`/logo_avatar_probe.png`	2,480	Withheld	Valid PNG header followed by a `pentagi_upload_verify` marker, used to check whether an upload landed
Recon output archive (name withheld)	184,320	Withheld	Compressed `nuclei` JSONL logs and `httpx` live-host tables from Docker workers
`/graphql_introspection_dump.json`	42,910	Withheld	GraphQL `__schema` introspection output from an API recon worker
`/mcp_tools_manifest.json`	8,740	Withheld	MCP tool descriptor exposing `nuclei_scan`, `sqlmap_audit` and `ffuf_dir_fuzz` to the LLM planner

These are staged files. Their presence shows intent and tooling. It does not show that any upload worked against a real target.

06

CHAPTER 06

Who the Swarm Targeted

IN SHORT

The exposed ARTEX archive covers 76 organizations in 18 verticals. Payment, Web3 and online gambling firms take most of the effort.

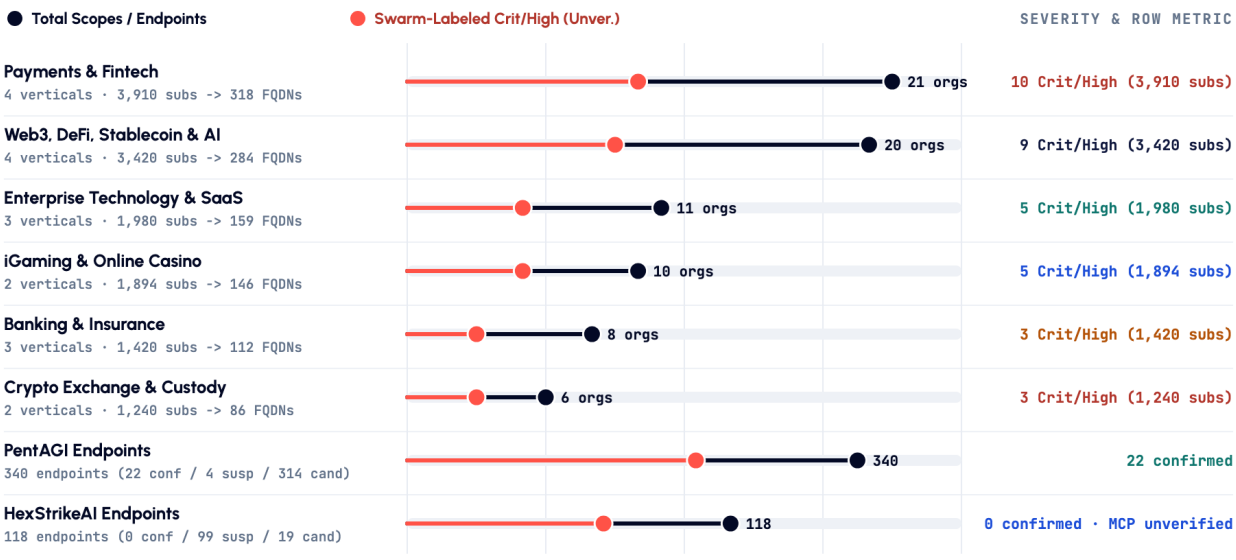
06 Who the Swarm Targeted

FIGURE 5 · ANONYMIZED SECTOR TARGETING & FRAMEWORK ENDPOINT COUNTS

ODIN BY BRANDEFENSE · TLP:CLEAR

Fintech, Web3, and iGaming absorb 67.1% of studied swarm targeting scopes

76 anonymized target scopes across 6 macro-sectors (18 sub-verticals, 14+ countries) + framework endpoint counts



Scope: Macro targeting strictly anonymized (TLP:CLEAR, zero victim identifiers) from 76 studied ARTEX campaign scopes.

What this does not show: Bottom two rows show endpoint counts per framework on a relative index; they are not growth measures.

Figure 5: Fintech, Web3 and iGaming absorb most of the studied swarm targeting scopes

All 76 targets are anonymized. No names, domains or addresses of target organizations appear in this report.

Payments and Fintech (21 organizations, 27.63%) and Web3, DeFi and Stablecoin (20 organizations, 26.32%) together make up 53.95% of the targets. They also account for 52.87% of the enumerated subdomains (7,330 of 13,864).

Sector	Verticals	Orgs	Share	Subdomains	Live FQDNs	Labeled critical or high	Workers	Claimed finding types
Payments and Fintech	4	21	27.63%	3,910	318	10	8 to 16	API auth bypass, exposed `.env` and payment keys, SQL injection
Web3, DeFi and Stablecoin	4	20	26.32%	3,420	284	9	8 to 16	RPC and GraphQL introspection, cloud secret leaks, exposed admin consoles
Enterprise Tech and Cloud SaaS	3	11	14.47%	1,980	159	5	4 to 12	CI/CD webhook and `.git` exposure, SSRF, multi-tenant API IDOR
iGaming and Online Casino	2	10	13.16%	1,894	146	5	8 to 16	High-volume `sqlmap` and `nuclei` sweeps (1,231 runs on one task)
Banking and Insurance	3	8	10.53%	1,420	112	3	4 to 12	Dangling subdomain candidates, legacy TLS and VPN portal enumeration
Crypto Exchanges	2	6	7.89%	1,240	86	3	16 (max 2,178 runs)	Sustained 16-worker sweeps (2,178 runs on one task), key-leak claims
Total	**18**	**76**	**100.0%**	**13,864**	**1,105**	**35 (46.05%)**	**1 to 16**	**17 critical, 18 high**

The swarm's 17 critical labels split into 3 claimed remote code execution or full-access scopes and 14 claimed credential or data exposure scopes. Like the high labels, these are the LLM writer's own classification of scanner output. They are not confirmed breaches.

The finding types still help defenders. They show what a swarm looks for first: `.env` files, `.git` folders, GraphQL introspection, open admin consoles and payment API keys.

07

CHAPTER 07

What Defenders Should Do

IN SHORT

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

07 What Defenders Should Do

Within 24 hours

- [ ] Block inbound and outbound traffic to the 1,820 confirmed endpoints on stable hosting, starting with the 14 block entries in the Indicators chapter.
- [ ] Put the 30 dynamic-ISP endpoints on a watchlist that expires after 24 to 72 hours.
- [ ] Send the Korean fork fingerprint and other unattributed forks to analyst review, not to the block list.

Within 72 hours

- [ ] Set `AllowOverride None` on every user-writable upload directory, and turn off script execution there (`php_flag engine off` or `RemoveHandler .php .phtml`).
- [ ] Turn off GraphQL `__schema` introspection in production.
- [ ] Rate-limit `/api/` and `/graphql` routes.

Within 7 to 30 days

- [ ] Add the ARTEX favicon MD5 `cd3ebd16437d6d3963592f7365dd3980` and the JARM values `28d28d...926e` and `40d40d...9319` to external attack surface hunting.
- [ ] Deploy the scan-burst query at the web edge: alert on 25 or more distinct URIs from one source in 5 minutes.
- [ ] Watch container hosts for runtimes that spawn offensive scanners with batch flags.
- [ ] Rotate any secret ever exposed in `.env` files or `.git` folders, since these are the first things a swarm collects.

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 27
How the other rules work 27
How to sort a host 28

08 Detection Rules

The full rules are in Appendix B. Each file targets a signal that survives a rebrand.

Rule file	What it catches
`odin_autonomous_ai_redteam_fleets.rules`	Suricata 7: eight signatures for ARTEX, PentAGI, HexStrikeAI and DeepAudit consoles, task APIs and the `.htaccess` polyglot
`sentinel_autonomous_swarm_scanner_correlation.kql`	Microsoft Sentinel and Defender XDR: contact with confirmed controllers, multi-tool container bursts, and web scan bursts
`sigma_docker_subagent_offensive_tool_spawning.yml`	Sigma (3 rules): container runtimes spawning scanners, `.htaccess` or polyglot file creation, and swarm output files
`odin_autonomous_offense_whitelabel_artifacts.yar`	YARA (4 rules): ARTEX and fork console bundles, PentAGI configs, HexStrikeAI MCP servers, and polyglot dropzone files

How the Suricata signatures work

SID	Looks for	Why
9511001	Korean title bytes `공격·방어 보안 운영` plus `artex-swarm-root`, `/api/v1/tasks` or the favicon hash text in an HTTP response	Catches the Korean fork even without the word ARTEX
9511002	`ARTEX` with the English or Chinese subtitle in an HTTP response	Catches upstream consoles
9511003	Outbound `POST /api/v1/tasks` with `subagent_concurrency` and `target_scope` in the body	Catches an internal host dispatching swarm tasks (T1595.002)
9511004	Outbound `POST /api/v1/graphql` with `subagents` and `createFlow`, `subagentLogs`, `pentagi_worker` or `spawnSubagent`	Catches PentAGI flow creation (T1609)
9511005	TLS certificate subject `CN=pentagi` or `CN=localhost` from ports 8443, 8666 or 8444	Low-confidence hunt only; pair with SID 9511004 or a JARM match before acting
9511006	Outbound `POST /mcp` with `jsonrpc`, `tools/call` and a tool name such as `nmap`, `nuclei` or `sqlmap`	Catches an MCP bridge driving offensive tools (T1059.006)
9511007	`DeepAudit` with `repository_audit`, `vulnerability_agent` or `llm_code_scanner` from ports 8000 or 3000	Catches DeepAudit consoles
9511008	`AddType application/x-httpd-php .jpg .png` in an HTTP body	Catches the polyglot `.htaccess` in transit (T1505.003)

Suricata cannot hash a favicon in transit. SID 9511001 therefore matches the page text, and the favicon MD5 check runs separately in any EASM tool. A simple fetch of `/favicon.ico` piped to `md5sum` is enough.

How the other rules work

The KQL query has three branches and a 7-day lookback. It carries the 13 confirmed controller addresses from the Indicators chapter. The first branch flags devices that contact those addresses, or use ports 8787, 8443, 8666, 8000 or 8899 from `python3`, `curl`, `docker` and similar processes. The second flags a container runtime or orchestrator that starts two or more scanners within 10 minutes.

The third branch reads IIS web logs. It counts distinct URIs per source IP in 5-minute bins before filtering. It alerts at 25 or more distinct URIs, or at 10 or more when the user agent names a tool such as `nuclei`, `sqlmap` or `ffuf`. Counting first means rotated user agents do not hide the burst.

The first Sigma rule needs three things together: a parent such as `containerd-shim-runc-v2`, `dockerd` or `pentagi`; a child such as `nuclei`, `sqlmap` or `naabu`; and a batch flag such as `-jsonl`, `--batch` or `-rate-limit`. Authorized breach-simulation runners are the main false positive.

The second Sigma rule flags a Python, PHP, Apache or PentAGI process that creates `/.htaccess` or `/logo_avatar_probe.png`. One such event alone is weak. Correlate events in the same directory within 5 minutes, or confirm the file content with YARA. The third Sigma rule flags files ending in `_credential_inventory_01.json`, `_findings_summary.json`, `/graphql_introspection_dump.json` or `/mcp_tools_manifest.json`.

Each YARA rule needs at least two independent strings, so a single word does not fire it.

YARA rule	Fires when
`ODIN_ARTEX_Whitelabel_Console`	An ARTEX title (English, Chinese or Korean) appears with `/api/v1/tasks`, `artex-swarm-root` or the favicon hash text, or the hash appears with either route marker
`ODIN_PentAGI_Subagent_Orchestrator_Artifact`	`PentAGI` appears with two of `/api/v1/graphql`, `pgvector`, `/var/run/docker.sock`, `vxcontrol/kali` and `createFlow`
`ODIN_HexStrikeAI_MCP_Offensive_Tool_Server`	`hexstrike` or the Chinese portal title appears with an MCP marker and at least three tool names
`ODIN_Polyglot_Htaccess_PHP_Webshell_Dropzone`	`AddType` with `application/x-httpd-php` and an image extension, or an image header followed by PHP code

How to sort a host

The triage logic turns the signals above into an action. It never blocks on uncorroborated evidence. Its routing follows the logic of this report.

What the host shows	Action
Several consoles plus a dropzone on one server	Block
A confirmed framework controller on stable hosting	Block
A confirmed framework on a dynamic telecom ISP	Watchlist for 24 hours
A rebranded fork with no documented hostile activity	Analyst review
A localized portal with only a title match	Analyst review
A generic Werkzeug header with no favicon or MCP match	Do not block
An internal, authorized breach-simulation runner	Allowlist

09

CHAPTER 09

Indicators

IN SHORT

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

INSIDE THIS CHAPTER

Confirmed endpoints to block 30

MITRE ATT&CK mapping 31

09 Indicators

Appendix A holds the 14 confirmed controller endpoints. The other 14 records in the selected set appear in this report only, or are withheld.

Type	Count
Confirmed controller endpoints to block (IP and port)	14
Watchlist or review endpoints	6 (3 on confirmed hosts listed below, 3 withheld)
File hashes (withheld: values not verified against the acquisition record)	8

Supporting fingerprints: one favicon MD5 (`cd3ebd16437d6d3963592f7365dd3980`), the JARM values named in this report, and the certificate subjects `CN=localhost`, `CN=pentagi`, `CN=supershell`, `CN=artex.dwwork[.]live` and `CN=vps.notelab[.]cc`.

Confirmed endpoints to block

Endpoint	Framework	Country	Key fingerprint	Seen
112.213.123[.]38:80	ARTEX	HK	`artex.112.213.123.38.sslip[.]io`, ARTEX favicon	13 Nov 2026
45.156.87[.]185:443	ARTEX	US	`CN=artex.dwwork[.]live`, JARM `2ad2ad...f06b`	16 Nov 2026
8.135.3[.]123:80	ARTEX	CN	Title `ARTEX - 攻防安全运营`, ARTEX favicon	17 Nov 2026
8.135.30[.]162:443	ARTEX	CN	Title `ARTEX - 攻防安全运营`, JARM `15d3fd...e331`	17 Nov 2026
144.172.118[.]88:443	CyberStrikeAI	US	Also 8899, open directory	13 Nov 2026
38.76.218[.]100:8000	DeepAudit	HK	Shares host with HexStrikeAI on 5000	13 Nov 2026
205.186.67[.]47:8089	CyberStrikeAI	MY	`CN=supershell`, JARM `40d40d...9319`	13 Nov 2026
45.144.137[.]148:80	ARTEX	NL	Open HTTP workbench on 8080	17 Nov 2026
144.172.68[.]197:80	ARTEX	US	Two confirmed frameworks on one host	17 Nov 2026
144.172.68[.]197:8443	CyberStrikeAI	US	`CN=144.172.68[.]197`	17 Nov 2026
38.127.244[.]14:8443	PentAGI	US	`<title>PentAGI</title>`, `CN=localhost`, dropzone on 8089	17 Nov 2026
47.120.40[.]104:8000	DeepAudit	CN	LLM source-code audit workbench	13 Nov 2026
82.208.23[.]183:8443	PentAGI	GB	`CN=vps.notelab[.]cc`, JARM `40d40d...efa8`	14 Nov 2026
134.122.200[.]96:8443	PentAGI	SG	`CN=localhost`, JARM `3fd3fd...99fb`, GraphQL API	14 Nov 2026

The three published watchlist entries sit on hosts listed above: 38.76.218[.]100:5000 (suspected HexStrikeAI), 38.127.244[.]14:8787 (candidate ARTEX) and 45.144.137[.]148:8080 (open workbench). All eight file hashes from the dropzone and the exposed archive are withheld, because their values could not be checked against the acquisition record.

File	Size (bytes)	SHA-256
Credential inventory, part 1 (`..._credential_inventory_01.json`)	19,340	Withheld
Credential inventory, part 2 (`..._credential_inventory_02.json`)	31,280	Withheld
Findings summary (`..._findings_summary.json`)	54,610	Withheld

Withheld: the two Korean fork addresses and their domain, the archive server and its paths, the 30 dynamic-ISP addresses, the benign collision host, and all suspected or candidate addresses on hosts not listed above.

MITRE ATT&CK mapping

Tactic	Technique	How the fleet uses it	Detection
Resource Development	T1583.006 Acquire Infrastructure: Web Services	4,360 endpoints on rented hosting; regional ARTEX forks	SID 9511001, 9511002; YARA ARTEX console rule
Resource Development	T1587.003 Develop Capabilities: Digital Certificates	Default or self-chosen certificates: `CN=localhost`, `CN=vps.notelab[.]cc`, `CN=artex.dwwork[.]live`, `CN=CyberStrikeAI`	SID 9511005 (hunt), JARM `28d28d...926e` and `3fd3fd...99fb`
Reconnaissance	T1595.002 Active Scanning: Vulnerability Scanning	1 to 16 worker pools; 13,864 subdomains and 1,105 live FQDNs	SID 9511003, 9511007; KQL burst branch
Execution	T1609 Container Administration Command	PentAGI starts Docker-in-Docker Kali workers through GraphQL `createFlow`	SID 9511004; YARA PentAGI rule; Sigma rule 1
Execution	T1059.006 Command and Scripting Interpreter: Python	MCP `tools/call` bridges to `nmap`, `nuclei`, `sqlmap`, `gobuster`	SID 9511006; YARA HexStrikeAI rule
Persistence	T1505.003 Server Software Component: Web Shell	`.htaccess` MIME override and PNG/PHP polyglot probes staged on 38.127.244[.]14:8089	SID 9511008; YARA polyglot rule; Sigma rule 2
Collection	T1560.001 Archive Collected Data: Archive via Utility	Recon logs and LLM-written credential inventories packed into JSON and archive files	Sigma rule 3; file names above

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.** The findings cover a two-week observation window ending 18 November 2026. Some records carry earlier first-seen dates from public sources; we do not use those as a tracking start.
- **Confirmed is not the same as hostile.** A confirmed endpoint runs the framework. For 3 confirmed endpoints, hostile use is not shown, and some operators may be authorized testers.
- **Candidates are leads.** The 2,397 candidates are hunting leads. Do not block them without corroboration.
- **No growth claims.** The data supports counts in the window, not trends over time for any framework.
- **The archive speaks for the attacker.** Worker counts, severity labels and claimed findings come from the swarm's own logs and LLM reports. We did not confirm any of them on a target system.
- **One archive, one campaign.** The 76 targets come from one exposed ARTEX archive. Nothing links it to the Korean fork, and it may not represent other operators.
- **Rule testing.** Every published rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma and Microsoft's Kusto.Language analyzer, checked on 20 November 2026). The rules were not replayed against live traffic, so test and tune them in your own environment before production use.

11

CHAPTER 11

Sources

IN SHORT

This chapter lists the public references used in the report.

11 Sources

1. VXControl. PentAGI repository. https://github.com/vxcontrol/pentagi (verified 18 November 2026)
2. Ed1s0nZ. CyberStrikeAI repository. https://github.com/Ed1s0nZ/CyberStrikeAI (verified 18 November 2026)
3. 0x4m4. HexStrikeAI MCP Agents repository. https://github.com/0x4m4/hexstrike-ai (verified 18 November 2026)
4. Model Context Protocol. Specification. https://modelcontextprotocol.io/specification/ (verified 18 November 2026)
5. MITRE ATT&CK Enterprise Matrix. https://attack.mitre.org/ (verified 18 November 2026)
6. Apache HTTP Server 2.4 documentation, `.htaccess` files. https://httpd.apache.org/docs/2.4/howto/htaccess.html (verified 18 November 2026)
7. Salesforce. JARM TLS server fingerprinting. https://github.com/salesforce/jarm (verified 18 November 2026)
8. AhnLab ASEC. Analysis of the infrastructure of ARTEX, a Chinese AI penetration testing tool. https://asec.ahnlab.com/en/95875 (verified 18 November 2026)
9. Team Cymru. Tracking CyberStrikeAI usage. https://www.team-cymru.com/post/tracking-cyberstrikeai-usage (verified 18 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 (14) 37

B. Detection rules (4 files) 38

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
R11-IOC-01	IPv4	112.213.123[.]38	80	ARTEX	BLOCK
R11-IOC-02	IPv4	45.156.87[.]185	443	ARTEX	BLOCK
R11-IOC-03	IPv4	8.135.3[.]123	80	ARTEX	BLOCK
R11-IOC-04	IPv4	8.135.30[.]162	443	ARTEX	BLOCK
R11-IOC-05	IPv4	144.172.118[.]88	443	CyberStrikeAI + Open Workbench	BLOCK
R11-IOC-06	IPv4	38.76.218[.]100	8000	DeepAudit (Co-hosted with HexStrikeAI)	BLOCK
R11-IOC-07	IPv4	205.186.67[.]47	8089	CyberStrikeAI + SuperShell	BLOCK
R11-IOC-08	IPv4	45.144.137[.]148	80	ARTEX + Open Workbench	BLOCK
R11-IOC-09	IPv4	144.172.68[.]197	80	ARTEX (Co-hosted with CyberStrikeAI)	BLOCK
R11-IOC-10	IPv4	144.172.68[.]197	8443	CyberStrikeAI (Co-hosted with ARTEX)	BLOCK
R11-IOC-11	IPv4	38.127.244[.]14	8443	PentAGI (Co-hosted with ARTEX & Dropzone)	BLOCK
R11-IOC-12	IPv4	47.120.40[.]104	8000	DeepAudit	BLOCK
R11-IOC-13	IPv4	82.208.23[.]183	8443	PentAGI	BLOCK
R11-IOC-14	IPv4	134.122.200[.]96	8443	PentAGI	BLOCK

B Detection Rules

Each rule file loads in its own engine (Suricata 7, YARA-X, 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_autonomous_ai_redteam_fleets.rules

SURICATA

```
# ============================================================================
# ODIN Threat Research - Report 11: Autonomous Red-Team Forks & Multi-Agent Fleets
# Deployable Suricata 7.x IDS Ruleset (8 Production Signatures)
# Validation: Loads in Suricata 7.0.12; not replayed against live traffic
# ============================================================================

# 1. Korean-Localized ARTEX White-Label Console HTTP Response-Body Banner
alert http any any -> $HOME_NET any (msg:"ODIN-AI Korean-Localized ARTEX Fork Console HTTP Banner";
flow:established,to_client; http.stat_code; content:"200"; http.response_body; content:"|ea b3 b5 ea b2 a9 c2 b7
eb b3 b4 ec 95 88 20 ec 9a b4 ec 98 81|"; fast_pattern; pcre:"/(artex-swarm-root|\/api\/v1\/tasks|
cd3ebd16437d6d3963592f7365dd3980)/i"; classtype:trojan-activity; sid:9511001; rev:2; metadata:framework
artex_whitelabel_fork, confidence high, mitre_attack T1583.006;)

# 2. Upstream ARTEX Autonomous Red-Team Console Banner (English & Chinese Title Variants)
alert http any any -> $HOME_NET any (msg:"ODIN-AI Upstream ARTEX Autonomous Red-Team Console HTTP Response";
flow:established,to_client; http.stat_code; content:"200"; http.response_body; content:"ARTEX"; fast_pattern;
pcre:"/ARTEX\s*[-:]\s*(Autonomous Red-Team Expert|
\xe6\x94\xbb\xe9\x98\xb2\xe5\xae\x89\xe5\x85\xa8\xe8\xbf\x90\xe8\x90\xa5)/"; classtype:trojan-activity; sid:
9511002; rev:1; metadata:framework artex, confidence high, mitre_attack T1583.006;)

# 3. ARTEX / White-Label Fork Multi-Worker Swarm Task Dispatch API
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-AI ARTEX or White-Label Fork Multi-Worker Swarm Task
Dispatch"; flow:established,to_server; http.method; content:"POST"; http.uri; content:"/api/v1/tasks";
fast_pattern; http.request_body; content:"subagent_concurrency"; content:"target_scope"; distance:0;
classtype:attempted-recon; sid:9511003; rev:2; metadata:framework artex_swarm, confidence high, mitre_attack
T1595.002;)

# 4. PentAGI Autonomous Pentest Orchestrator GraphQL Flow Creation
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-AI PentAGI Autonomous Subagent Orchestrator GraphQL
Query"; flow:established,to_server; http.method; content:"POST"; http.uri; content:"/api/v1/graphql";
fast_pattern; http.request_body; content:"subagents"; pcre:"/(createFlow|subagentLogs|pentagi_worker|
spawnSubagent)/i"; classtype:trojan-activity; sid:9511004; rev:1; metadata:framework pentagi, confidence high,
mitre_attack T1609;)

# 5. PentAGI Default TLS Certificate Subject Hunt (LOW CONFIDENCE: CN=localhost or CN=pentagi is generic; hunting
signal only, correlate manually with JARM or sid:9511004 before any action)
alert tls $EXTERNAL_NET [8443,8666,8444] -> $HOME_NET any (msg:"ODIN-AI PentAGI Default TLS Subject Hunt (Low
Confidence)"; flow:established,to_client; tls.cert_subject; content:"CN="; fast_pattern; pcre:"/CN=(pentagi|
localhost)\b/i"; threshold:type limit, track by_src, count 1, seconds 300; classtype:misc-activity; sid:9511005;
rev:4; metadata:framework pentagi, confidence low, mitre_attack T1587.003;)

# 6. HexStrikeAI Model Context Protocol (MCP) Offensive Tool Call Bridge
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-AI HexStrikeAI MCP Offensive Tool Execution Bridge";
flow:established,to_server; http.method; content:"POST"; http.uri; content:"/mcp"; fast_pattern;
http.request_body; content:"\"jsonrpc\""; content:"\"tools/call\""; pcre:"/\"name\"\s*:\s*\"(nmap|nuclei|sqlmap|
gobuster|ffuf|masscan|hydra)/i"; classtype:trojan-activity; sid:9511006; rev:1; metadata:framework hexstrikeai,
confidence high, mitre_attack T1059.006;)

# 7. DeepAudit Autonomous Code & Repo Vulnerability Workbench
alert http any [8000,3000] -> $HOME_NET any (msg:"ODIN-AI DeepAudit Autonomous Code Audit Workbench HTTP
Response"; flow:established,to_client; http.stat_code; content:"200"; http.response_body; content:"DeepAudit";
fast_pattern; pcre:"/(repository_audit|vulnerability_agent|llm_code_scanner)/i"; classtype:trojan-activity; sid:
9511007; rev:1; metadata:framework deepaudit, confidence high, mitre_attack T1595.002;)
```

```
# 8. Co-Hosted Polyglot .htaccess PHP Image Handler Injection (Companion to PentAGI/ARTEX)
alert http any any -> any any (msg:"ODIN-AI Co-Hosted Polyglot .htaccess PHP Image Handler Override";
flow:established; http.response_body; content:"AddType application/x-httpd-php"; fast_pattern; pcre:"/
AddType\s+application\/x-httpd-php\s+\.jpg\s+\.png/i"; classtype:web-application-attack; sid:9511008; rev:1;
metadata:framework artex_pentagi_cohosted, confidence high, mitre_attack T1505.003;)
```

odin_autonomous_offense_whitelabel_artifacts.yar

YARA

```
/*
 * ======================================================================
 * ODIN Threat Research - Report 11: Autonomous Red-Team Forks & Multi-Agent Fleets
 * YARA Ruleset: White-Label Console Bundles, Subagent Orchestrators & Polyglot Dropzones
 * Validation: Loads in YARA-X; not replayed against live samples
 * ======================================================================
 */

rule ODIN_ARTEX_Whitelabel_Console {
    meta:
        author = "ODIN Threat Research (Brandefense)"
        description = "Detects ARTEX and Korean-localized white-label web console HTML/JS bundles sharing
invariant static asset hashes and swarm API routes"
        date = "2026-11-18"
        report = "Odin by Brandefense - Autonomous Red-Team Forks"
        confidence = "high"
        mitre_attack = "T1583.006"
    strings:
        $ko_ops = { ea b3 b5 ea b2 a9 c2 b7 eb b3 b4 ec 95 88 20 ec 9a b4 ec 98 81 }
        $artex_en = "ARTEX - Autonomous security assessment console" ascii wide
        $artex_en2 = "ARTEX - Autonomous Red-Team Expert" ascii wide
        $artex_zh = { e8 87 aa e4 b8 bb e6 b8 97 e9 80 8f e6 b5 8b e8 af 95 e6 8e a7 e5 88 b6 e5 8f b0 }
        $fav_md5 = "cd3ebd16437d6d3963592f7365dd3980" ascii nocase
        $api_tasks = "/api/v1/tasks" ascii
        $dom_root = "artex-swarm-root" ascii
    condition:
        filesize < 10MB and (
            (($ko_ops or $artex_en or $artex_en2 or $artex_zh) and ($api_tasks or $dom_root or $fav_md5)) or
            ($fav_md5 and ($api_tasks or $dom_root))
        )
}

rule ODIN_PentAGI_Subagent_Orchestrator_Artifact {
    meta:
        author = "ODIN Threat Research (Brandefense)"
        description = "Detects PentAGI autonomous penetration testing orchestrator configurations, GraphQL flow
schemas, and Docker subagent runtime artifacts"
        date = "2026-11-18"
        report = "Odin by Brandefense - Autonomous Red-Team Forks"
        confidence = "high"
        mitre_attack = "T1609"
    strings:
        $p1 = "PentAGI" ascii wide
        $s_gql = "/api/v1/graphql" ascii
        $s_pgv = "pgvector" ascii nocase
        $s_sock = "/var/run/docker.sock" ascii
        $s_kali = "vxcontrol/kali" ascii nocase
        $s_flow = "createFlow" ascii
    condition:
        filesize < 15MB and $p1 and 2 of ($s_gql, $s_pgv, $s_sock, $s_kali, $s_flow)
}

rule ODIN_HexStrikeAI_MCP_Offensive_Tool_Server {
    meta:
        author = "ODIN Threat Research (Brandefense)"
        description = "Detects HexStrikeAI Model Context Protocol (MCP) JSON-RPC tool-execution servers wrapping
offensive CLI scanners"
        date = "2026-11-18"
        report = "Odin by Brandefense - Autonomous Red-Team Forks"
        confidence = "high"
```

```
        mitre_attack = "T1059.006"
    strings:
        $hex_id = "hexstrike" ascii nocase
        $zh_portal = { e7 94 a8 e6 88 b7 e7 99 bb e5 bd 95 20 2d 20 e6 99 ba e6 b5 8b e5 b7 a5 e5 85 b7 }
        $mcp_rpc = "\"jsonrpc\"" ascii
        $mcp_call = "tools/call" ascii
        $t_nmap = "nmap" ascii nocase
        $t_nuclei = "nuclei" ascii nocase
        $t_sqlmap = "sqlmap" ascii nocase
        $t_gobuster = "gobuster" ascii nocase
        $t_ffuf = "ffuf" ascii nocase
    condition:
        filesize < 10MB and ($hex_id or $zh_portal) and ($mcp_rpc or $mcp_call) and 3 of ($t_*)
}

rule ODIN_Polyglot_Htaccess_PHP_Webshell_Dropzone {
    meta:
        author = "ODIN Threat Research (Brandefense)"
        description = "Detects Apache .htaccess MIME-override companions and PNG/JPEG polyglot PHP execution
probes staged on co-hosted offensive dropzones"
        date = "2026-11-18"
        report = "Odin by Brandefense - Autonomous Red-Team Forks"
        confidence = "high"
        mitre_attack = "T1505.003"
    strings:
        $ht_addtype = "AddType" ascii nocase
        $ht_mime = "application/x-httpd-php" ascii nocase
        $ht_ext = /\.(jpg|jpeg|png|gif)/i
        $png_hdr = { 89 50 4E 47 0D 0A }
        $jpg_hdr = { FF D8 FF E0 }
        $php_open = "<?php" ascii nocase
        $php_post = "$_POST" ascii
        $php_isset = "isset(" ascii
    condition:
        filesize < 500KB and (
            ($ht_addtype and $ht_mime and $ht_ext) or
            (($png_hdr or $jpg_hdr) and $php_open and ($ht_mime or $php_post or $php_isset))
        )
}
```

sentinel_autonomous_swarm_scanner_correlation.kql

KQL (SENTINEL / DEFENDER XDR)

```
// ============================================================================
// ODIN Threat Research - Report 11: Autonomous Red-Team Forks & Multi-Agent Fleets
// Microsoft Sentinel / Defender XDR KQL Hunting Query
// Scope: Correlates outbound/inbound connections to verified Autonomous C2
//        controllers, Docker subagent offensive tool bursts, and perimeter web
//        scanning bursts exhibiting multi-worker swarm concurrency.
// ============================================================================

let Lookback = 7d;
let VerifiedAutonomousC2IPs = dynamic([
    "45.156.87.185",  // Upstream ARTEX TLS Console (CN=artex.dwwork.live)
    "112.213.123.38", // Confirmed ARTEX console (HK)
    "82.208.23.183",  // Confirmed PentAGI console (GB)
    "8.135.3.123",    // Upstream ARTEX Console (Alibaba Cloud CN)
    "8.135.30.162",   // Upstream ARTEX TLS Console (Alibaba Cloud CN)
    "38.127.244.14",  // Flagship Multi-Framework Hub (PentAGI + ARTEX + Dropzone)
    "144.172.68.197", // Co-hosted ARTEX + CyberStrikeAI Stack
    "45.144.137.148", // Confirmed ARTEX + Open Workbench Directory
    "38.76.218.100",  // Co-hosted DeepAudit + HexStrikeAI Stack
    "205.186.67.47",  // Confirmed CyberStrikeAI + SuperShell Stack
    "134.122.200.96", // Confirmed PentAGI Orchestrator (CTG Server Limited)
    "47.120.40.104",  // Confirmed DeepAudit Workbench
    "144.172.118.88"  // Confirmed CyberStrikeAI + Open Secret Sink
]);
let SwarmOffensiveTools = dynamic([
    "nuclei", "sqlmap", "nmap", "masscan", "gobuster", "ffuf",
```

```
    "dirsearch", "subfinder", "httpx", "naabu", "katana", "hydra"
]);
union isfuzzy=true
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where RemoteIP in (VerifiedAutonomousC2IPs)
       or (RemotePort in (8787, 8443, 8666, 8000, 8899) and InitiatingProcessFileName in~ ("python3", "python",
"curl", "docker", "pentagi", "cyberstrike"))
    | extend HuntSignal = iff(RemoteIP in (VerifiedAutonomousC2IPs), "Verified_Autonomous_C2_Contact",
"Suspicious_Autonomous_Framework_Port_Activity")
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName = RemoteIP, ArtifactDetail =
strcat(RemoteIP, ":", tostring(RemotePort), " (", RemoteUrl, ")")
),
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where InitiatingProcessFileName in~ ("dockerd", "containerd-shim", "containerd-shim-runc-v2", "python3",
"pentagi", "node")
    | where FileName in~ (SwarmOffensiveTools)
    | summarize DistinctTools = dcount(FileName), ToolList = make_set(FileName, 10), SampleCmd =
any(ProcessCommandLine), FirstSeen = min(Timestamp), LastSeen = max(Timestamp)
      by DeviceName, InitiatingProcessFileName, bin(Timestamp, 10m)
    | where DistinctTools >= 2
    | extend HuntSignal = "Autonomous_Subagent_MultiTool_Container_Burst",
             FileName = tostring(ToolList),
             ArtifactDetail = strcat("distinct_tools=", tostring(DistinctTools), " sample_cmd=", SampleCmd)
    | project Timestamp = FirstSeen, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
),
(
    W3CIISLog
    | where TimeGenerated >= ago(Lookback)
    | summarize DistinctURIs = dcount(csUriStem), TotalRequests = count(), ObservedUserAgents =
make_set(csUserAgent, 5), SampleURIs = make_set(csUriStem, 5)
      by bin(TimeGenerated, 5m), Computer, cIP
    | extend HasSwarmUA = tostring(ObservedUserAgents) has_any ("nuclei", "sqlmap", "gobuster", "ffuf",
"subfinder", "PentAGI", "ARTEX")
    | where cIP in (VerifiedAutonomousC2IPs) or DistinctURIs >= 25 or (HasSwarmUA and DistinctURIs >= 10)
    | extend Timestamp = TimeGenerated,
             DeviceName = Computer,
             HuntSignal = "Multi_Worker_Swarm_Perimeter_Burst_Scan",
             InitiatingProcessFileName = cIP,
             FileName = tostring(ObservedUserAgents),
             ArtifactDetail = strcat("uris=", tostring(DistinctURIs), " reqs=", tostring(TotalRequests))
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
)
| sort by Timestamp desc
```

sigma_docker_subagent_offensive_tool_spawning.yml

SIGMA

```
title: Autonomous Red-Team Subagent Container Spawning Offensive Reconnaissance Utilities
id: 7e8490a2-9c31-4f88-b614-1101a6f72c01
status: experimental
description: >
  Detects container runtimes or Python/Go orchestrator processes (characteristic of
  PentAGI, ARTEX, regional white-label forks, and HexStrikeAI) spawning automated offensive
  reconnaissance and exploitation binaries inside ephemeral worker subagents.
references:
  - https://github.com/vxcontrol/pentagi
  - https://brandefense.io/odin-research/
author: ODIN Threat Research (Brandefense)
date: 2026-11-18
tags:
  - attack.execution
  - attack.t1609
  - attack.reconnaissance
  - attack.t1595.002
logsource:
```

```
    category: process_creation
    product: linux
  detection:
    selection_parent:
      ParentImage|endswith:
        - '/containerd-shim-runc-v2'
        - '/containerd-shim'
        - '/dockerd'
        - '/pentagi'
        - '/cyberstrike-ai'
    selection_tool:
      Image|endswith:
        - '/nuclei'
        - '/sqlmap'
        - '/nmap'
        - '/masscan'
        - '/ffuf'
        - '/gobuster'
        - '/subfinder'
        - '/naabu'
        - '/httpx'
        - '/hydra'
    selection_flags:
      CommandLine|contains:
        - '-jsonl'
        - '--batch'
        - '-oJ'
        - '-oX'
        - '-silent'
        - '-rate-limit'
    condition: selection_parent and selection_tool and selection_flags
  falsepositives:
    - Authorized enterprise continuous security validation (BAS/PTaaS) runner nodes
    - Scheduled internal vulnerability scanning containers (verify asset inventory tags)
  level: high
  ---
  title: Suspicious .htaccess or Polyglot Probe File Creation by Web or Subagent Process
  id: 7e8490a2-9c31-4f88-b614-1101a6f72c02
  status: experimental
  description: >
    Detects a single file-creation event for either an Apache /.htaccess configuration file or
    the observed /logo_avatar_probe.png polyglot probe filename by a Python, PHP, web server,
    or PentAGI process. Because a single file_event log does not prove a paired companion
    dropzone, correlate matching events in the same directory within a 5-minute window or
    pair with YARA content inspection (ODIN_Polyglot_Htaccess_PHP_Webshell_Dropzone).
  references:
    - https://httpd.apache.org/docs/2.4/howto/htaccess.html
    - https://brandefense.io/odin-research/
  author: ODIN Threat Research (Brandefense)
  date: 2026-11-18
  tags:
    - attack.persistence
    - attack.t1505.003
  logsource:
    category: file_event
    product: linux
  detection:
    selection_htaccess:
      TargetFilename|endswith:
        - '/.htaccess'
        - '/logo_avatar_probe.png'
    selection_process:
      Image|endswith:
        - '/python3'
        - '/python'
        - '/php'
        - '/apache2'
        - '/httpd'
        - '/pentagi'
    condition: selection_htaccess and selection_process
  falsepositives:
    - Legitimate web application deployments updating Apache directory rules via CI/CD runners
```

```
  level: medium
  ---
  title: Autonomous Multi-Worker Swarm Output and Credential Inventory Archive Creation
  id: 7e8490a2-9c31-4f88-b614-1101a6f72c03
  status: experimental
  description: >
    Detects autonomous multi-worker swarm controllers and ephemeral Docker subagents writing
    structured JSON credential inventories, automated GraphQL schema introspection dumps, or
    compressed reconnaissance output bundles to staging directories.
  references:
    - https://github.com/vxcontrol/pentagi
    - https://brandefense.io/odin-research/
  author: ODIN Threat Research (Brandefense)
  date: 2026-11-18
  tags:
    - attack.collection
    - attack.t1560.001
  logsource:
    category: file_event
    product: linux
  detection:
    selection_artifacts:
      TargetFilename|endswith:
        - '_credential_inventory_01.json'
        - '_credential_inventory_02.json'
        - '_findings_summary.json'
        - 'recon_bundle.tar.gz'
        - '/graphql_introspection_dump.json'
        - '/mcp_tools_manifest.json'
    condition: selection_artifacts
  falsepositives:
    - Authorized security assessment archival pipelines writing standardized JSON output
  level: medium
```

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/white-labeled-ai-offense/
Short Read · brandefense.io/odin-research/blog/rebranded-ai-red-team-tools-are-running-at-scale/
Defense Kit · brandefense.io/odin-research/defense-kit/