ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 46 / 30 · CREDENTIAL THEFT

Trading Bots, MCP Wallets, and Kubernetes Backups
Exposed Secrets

How five misconfigured open directories expose exchange bot backups, MCP wallet tools, BIP-39 seed phrases, and unencrypted Kubernetes secret dumps

Two-week observation window ending 20 November 2026 · Prepared November 2026
Full Report · Public edition · Version 1.1

55,974
Combined files reported across the five misconfigured open directories (4,334 sampled entries)

5
Independent developer, trader, and cluster hosts reviewed (0 shared links; 0 published IPs)

90
Secret records exposed across 8 secret types on the five servers (0 hashes published)

30
Secret records exposed on Host C in France, the MCP wallet tool-cache host (highest single-host count)

What Is in This Report. 10 chapters and two appendices.

How five misconfigured open directories expose exchange bot backups, MCP wallet tools, BIP-39 seed phrases, and unencrypted Kubernetes secret dumps

01	Executive Summary	04
02	Five Misconfigured Hosts and the Five-Surface Taxonomy	07
03	Quantitative Trading Bot Backups and WeChat Trading Agents	11
04	MCP Wallet Tool Caches, Seed Derivers, and Kubernetes Secret Backups	15
05	Crypto Signing Key Concentration and Owner Boundaries	19
06	What Defenders Should Do	22
07	Detection Rules	24
08	Indicators	27
09	Scope and Limits	30
10	Sources	32
A	Indicators of compromise (0)	34
B	Detection rules (4 files)	35

IN SHORT

In a two-week observation window ending 20 November 2026, Odin by Brandefense observed five misconfigured open web directories reporting 55,974 files and exposing 90 secret records across eight secret types. All five servers belong to misconfigured developers, traders, or cluster administrators with zero infrastructure links, so we group them as an analytical five-surface model. Together they expose trading-bot `.env.bak` chains, WeChat trading agents, Model Context Protocol wallet tools, BIP-39 seed derivers, and plain-text Kubernetes Secret dumps. Close open HTTP directories, encrypt cluster backups, and isolate signing keys.

CHAPTER 01

Executive Summary

IN SHORT

In a two-week observation window ending 20 November 2026, Odin by Brandefense observed five misconfigured open web directories reporting 55,974 files and exposing 90 secret records across eight secret types. All five servers belong to misconfigured developers, traders, or cluster administrators with zero infrastructure links, so we group them as an analytical five-surface model. Together they expose trading-bot `.env.bak` chains, WeChat trading agents, Model Context Protocol wallet tools, BIP-39 seed derivers, and plain-text Kubernetes Secret dumps. Close open HTTP directories, encrypt cluster backups, and isolate signing keys.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04
The Study in Numbers 05

01 Executive Summary

Algorithmic trading bots, agentic crypto-wallet tools, and self-managed Kubernetes clusters store financial signing keys and database credentials on cloud servers. When a developer, trader, or administrator leaves an unauthenticated HTTP directory listing open, every file inside that workspace becomes readable from the internet. Keys exposed this way are exactly what dark web monitoring is meant to catch. We reviewed file metadata and secret counts statically and tested zero credentials. We treated all five hosts strictly as misconfigured users rather than adversary operators, and withheld all IP addresses and file hashes.

1. **Five open directories account for over four-fifths of Ethereum and Binance key records.**
   Nothing connects the five servers to one owner or operator, so we present them as an analytical five-surface defensive model. Together they hold 49 crypto-native secret records (23 BIP-39 seed phrases, 17 Binance API keys, and 9 Ethereum private keys) alongside 41 database, GitHub, AWS, and Telegram records.

2. **Recursive `.env.bak` scripts and MCP wallet caches leave signing-key material in plain text.**
   Host A in Finland holds 15 Binance-key and 1 Ethereum-key records across 45 timestamped backup files. Host B in the United States exposes six WeChat trading-agent profiles. Host C in France, a developer host with a cached MCP wallet package, exposes 30 secret records, including 8 Ethereum-key and 4 BIP-39 seed records.

3. **Seed-derivation trees and plain-text Kubernetes YAML backups widen the blast radius.**
   Host D in the United States exposes a payment platform beside Python wallet-derivation scripts and 19 BIP-39 seed phrases. Host E in France exposes a restore folder with 14 plain-text Kubernetes `*-secrets.yaml` files holding 11 database URLs and 3 AWS keys. That static dump contrasts sharply with a separate host exposing a mounted pod ServiceAccount token.

Three Decisions for Leaders

1. **Shut down ad-hoc HTTP file servers and stop saving `.env.bak` copies.** Block directory listings at cloud firewalls and reverse proxies, ban timestamped `.env` backups on trading hosts, and store keys in managed vaults.
2. **Encrypt Kubernetes Secret backups and scope pod ServiceAccount tokens.** Never dump plain-text `kubectl get secrets -o yaml` manifests to local folders. Encrypt exported backups before storage, separately enable KMS encryption at rest for etcd, and disable unneeded pod token mounts.
3. **Isolate MCP wallet tool caches from source-control tokens and production databases.** Run agentic wallet tools in dedicated sandboxes with withdrawal allowlists, separated from GitHub tokens, SSH keys, and customer databases.

The Study in Numbers

Number	What it is
55,974	Combined files reported across the five misconfigured open directories (4,334 sampled entries)
5	Independent developer, trader, and cluster hosts reviewed (0 shared links; 0 published IPs)
90	Secret records exposed across 8 secret types on the five servers (0 hashes published)
49 (54.44%)	Crypto-native secret records on the five hosts (23 BIP-39 seeds, 17 Binance keys, 9 ETH keys)
81.82% & 80.95%	Share of arrival-set Ethereum-key records (9 of 11) and Binance-key records (17 of 21) on two hosts
30	Secret records exposed on Host C in France, the MCP wallet tool-cache host (highest single-host count)
14 + 1	Plain-text namespace `*-secrets.yaml` files on Host E (FR) vs 1 separate pod token host

How to use this report.

Leaders can start with Chapters 1, 5, and 6. Trading, Web3, AI-agent, and Kubernetes teams should read Chapters 3, 4, and 6. SOC and attack-surface teams should go to Chapters 7 and 8, plus Appendix B. Platform architects will find attribution boundaries and limits in Chapters 5 and 9.

02

CHAPTER 02

Five Misconfigured Hosts and the Five-Surface Taxonomy

IN SHORT

Five open web directories reported 55,974 files (4,334 sampled entries) and exposed 90 secret records across eight types in the two-week window ending 20 November 2026. Because the five hosts belong to separate misconfigured users and admins with zero shared links, we group them into five defensive exposure surfaces.

INSIDE THIS CHAPTER

The broader latest arrival context 07

Why these five hosts are users and admins, not operators 08

02 Five Misconfigured Hosts and the Five-Surface Taxonomy

Cloud breaches and wallet drains often start with accidental self-exposure rather than software exploits. Developers start `python3 -m http.server` to copy logs, traders back up `.env` files inside web folders, and cluster admins save `kubectl` YAML dumps on staging nodes without authentication (T1552.001).

Defenders can use external attack surface management to find public HTTP directory listings that expose workspace trees to anyone scanning the web. We inspected file paths, byte sizes, and secret counts statically, without running scripts, deriving wallet addresses, testing keys, or contacting any server.

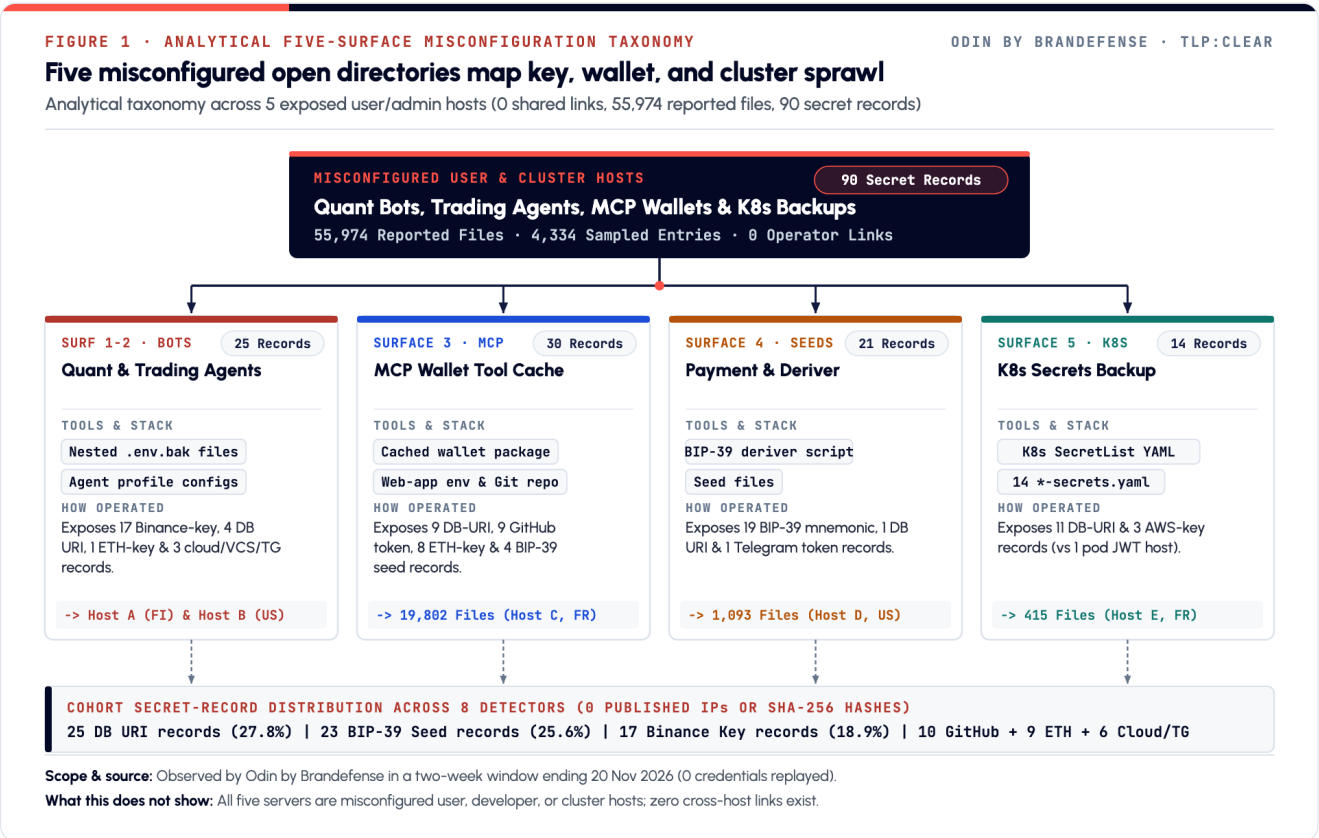


Figure 1: Five misconfigured open directories map key, wallet, and cluster sprawl

The broader latest arrival context

In the two-week observation window ending 20 November 2026, Odin by Brandefense tracked major credential classes in the latest arrival set. In that set, database connection strings lead (255), followed by AWS access keys (106) and Telegram bot tokens (79). Google Cloud service accounts (39) and AWS secret keys (15) follow.

Alongside those cloud and database credentials, cryptocurrency signing and exchange keys form a concentrated high-risk group: BIP-39 wallet seed phrases (92), Binance API keys (21), and Ethereum private keys (11). One separate host in that arrival set also exposed a mounted Kubernetes pod ServiceAccount token at `/run/secrets/kubernetes.io/serviceaccount/token`.

Secret or exposure type	Five-host cohort (90)	Cohort share	Broader arrival set	Share of arrival set	Main exposure location
`db_connection`	25	27.78%	255	9.80%	Host E K8s `*-secrets.yaml` (11), Host C MCP (9), Host B trading agent (4), Host D payment (1)
`bip39_mnemonic`	23	25.56%	92	25.00%	Host D (US) payment/deriver (19) and Host C (FR) MCP tool cache (4)
`binance_key`	17	18.89%	21	80.95%	Host A (FI) `.env.bak` chains (15) and Host B (US) trading-agent proxy (2)
`github_token`	10	11.11%	Unlisted in slice	N/A	Host C (FR) MCP/webapp (9) and Host B (US) trading agent (1)
`eth_private_key`	9	10.00%	11	81.82%	Host C (FR) MCP tool cache (8) and Host A (FI) quant bot (1)
`aws_access_key`	2	2.22%	106	1.89%	Host E (FR) unencrypted Kubernetes namespace backup (2)
`aws_secret_key`	2	2.22%	15	13.33%	Host E (FR) K8s backup (1) and Host B (US) trading agent (1)
`telegram_bot_token`	2	2.22%	79	2.53%	Host B (US) trading agent (1) and Host D (US) notifier (1)
Mounted pod `serviceaccount/token`	0 (separate host)	N/A	1 host	Contrasted	Mounted container runtime token vs Host E static YAML backups

Why these five hosts are users and admins, not operators

Across 104 directory-site records, the five open-directory servers report 55,974 exposed files (4,334 sampled entries and 1,037 detailed rows), holding 90 secret records across 8 secret types.

Nothing links these five servers to each other or to an adversary group (0 shared IPs, certificates, SSH keys, or bot tokens). Every server reflects a developer, trader, or cluster administrator who left a workspace or backup folder exposed over HTTP. Defenders can use cyber threat intelligence to prioritize this five-surface self-audit map across bot backups, trading agents, MCP wallets, seed derivers, and Kubernetes backups.

Exposure surface	Host and country	Reported / sampled files	Detailed rows	Secrets (90)	Exposed artifact classes
Surface 1: Quant futures bot	Host A (FI)	22,654 / 1,370	220	16	Nested `.env.bak` backup chains, Python venv, setup scripts
Surface 2: WeChat trading agent	Host B (US)	12,010 / 1,169	220	9	Trading-agent profile folders, DEX CLI tools, exchange proxy environment files
Surface 3: MCP wallet tool cache	Host C (FR)	19,802 / 1,189	220	30	Cached MCP wallet package, web-app environment file, Git repository entries
Surface 4: Payment & seed deriver	Host D (US)	1,093 / 449	220 (183 unique)	21	Payment platform tree, BIP-39 deriver script, plain-text seed file
Surface 5: K8s secrets backup	Host E (FR)	415 / 157	157	14	Cluster restore folder with 14 namespace `*-secrets.yaml` manifests

CHAPTER 03

Quantitative Trading Bot Backups and WeChat Trading Agents

IN SHORT

Automated trading servers leak exchange API keys when scripts copy `.env` files before strategy edits or expose multi-bot agent folders over HTTP. Host A in Finland holds 15 Binance keys and 1 Ethereum key across 45 backup files, while Host B in the United States exposes six WeChat trading-agent profiles. Stop creating `.env.bak` copies and lock exchange keys to fixed server IPs.

INSIDE THIS CHAPTER

Recursive `.env.bak` chains on Host A (Finland) 11

Autonomous WeChat trading-agent profiles on Host B (United States) 12

03 Quantitative Trading Bot Backups and WeChat Trading Agents

Independent traders and small quant teams often run Python or Node.js trading engines on single cloud instances connected to centralized exchanges such as `api.binance[.]com` and `fapi.binance[.]com`. Instead of pulling short-lived secrets at startup, these setups usually read plain-text `.env` files on disk and duplicate them before changing strategy parameters (T1552.001, T1528).

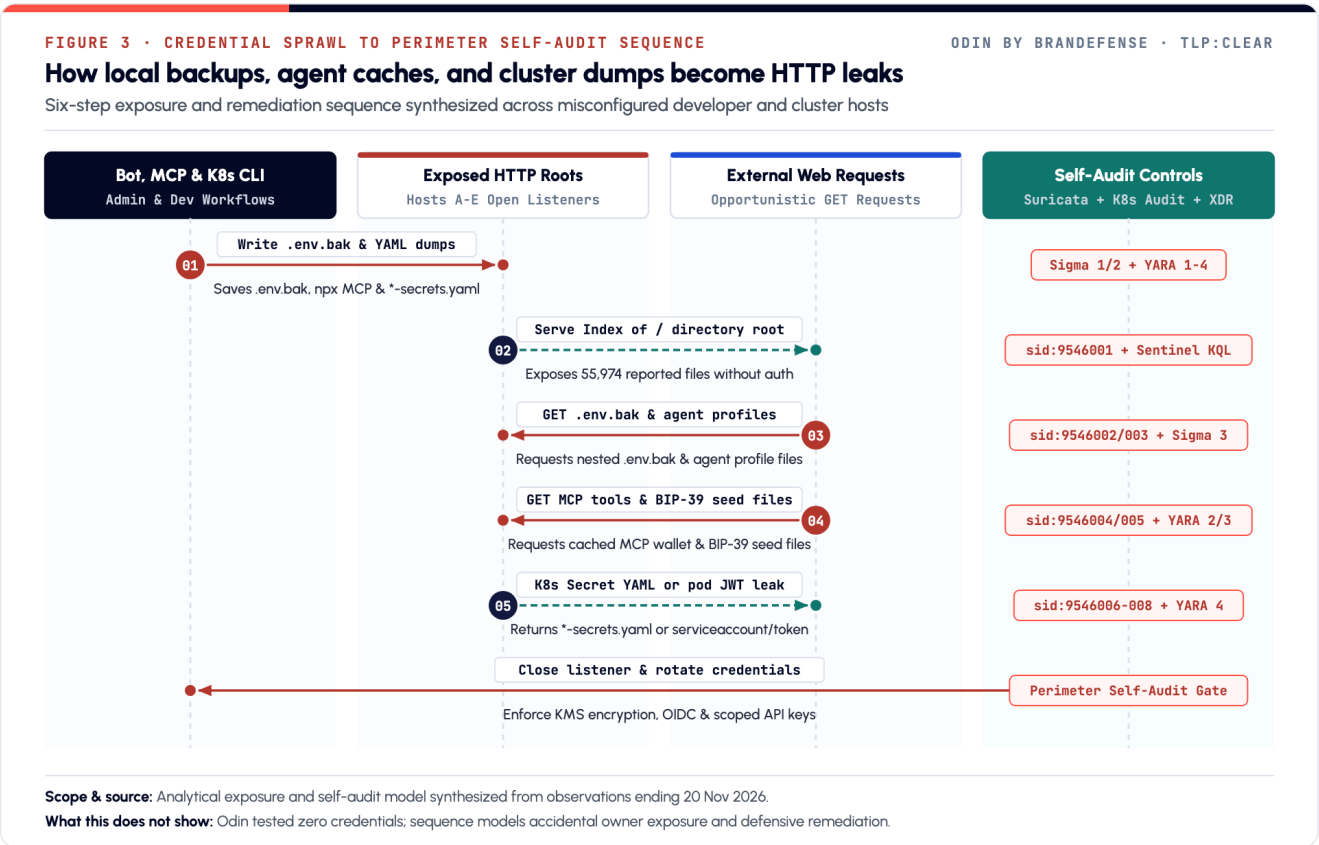


Figure 3: How local backups, agent caches, and cluster dumps become HTTP leaks

Recursive `.env.bak` chains on Host A (Finland)

On Host A in Finland (address withheld), an open directory reports 22,654 exposed files, with 1,370 sampled entries and 220 detailed rows in our review. Across this server, 16 secret records sit exposed: 15 Binance API keys (71.43% of the 21 arrival-set Binance-key records) and 1 Ethereum private key.

Of the 220 detailed rows, 152 belong to a Python 3.12 virtual environment containing `pycryptodome-3.23.0`, `aiohttp`, and `websockets`. The remaining 68 detailed rows sit inside the trading-bot workspace, alongside broader sampled bot folders and a local secret-finder script. That workspace holds 45 sized `.env` and setup backup files ranging from 118 to 2,811 bytes. They sit next to 23 runtime log files measuring 107 to 11,023 bytes, including logs of 4,939 and 11,023 bytes.

Two setup wrapper scripts on the server (1,559 bytes and a 1,550-byte dated backup) are written to copy earlier backups without deleting old versions:

- **Triple- and quadruple-nested backup filenames.** The directory held nested, dated environment backups. Three examples measured 391, 203, and 540 bytes.
- **Separate exchange and strategy snapshots.** Additional plain-text exchange, root `.env.OLD`, and strategy backup copies measured 165, 437, and 2,811 bytes.

When a trader updates the active `.env` file without revoking older keys at the exchange, unrevoked keys in `.bak` and `.OLD` files remain exposed. The October 2025 and February 2026 strings are filename dates on the server, not Odin's observation period.

Autonomous WeChat trading-agent profiles on Host B (United States)

On Host B in the United States (address withheld), an open directory reports 12,010 exposed files, with 1,169 sampled entries and 220 detailed rows. Nine secret records sit exposed across five types: 4 database connection strings, 2 Binance API keys, 1 AWS secret key, 1 GitHub token, and 1 Telegram bot token.

The web root exposes a developer folder with trading-agent configuration, SQLite state databases, and a backup script. Across six WeChat and trading-bot profile folders, each profile keeps its own environment file, SQLite state database, and gateway logs. Eleven scheduled curator log folders carry June to August 2026 folder timestamps on disk.

Inside a bundled Node.js module tree, the host includes a decentralized-exchange CLI across 10 rows alongside MCP skill-packaging tools. In the surrounding workspace, the directory exposes exchange-proxy, token-scanner, trading-bot, and stock-analysis environment files, 20 June 2026 log files, a Telegram notification script, and a secret-linter configuration file. Across the 220 detailed rows sit 25 environment, package-manager, and secret-linter configuration entries plus 13 SQLite database and SQL files.

Host and country	Workspace artifact group	Detailed rows	Size (bytes)	Host secret total	Role on the misconfigured server
Host A (FI)	Root `.env.bak.*` and `.env.OLD_*` backups	6	344 to 522	15 Binance, 1 ETH	Root `.env` backups with triple `.bak` and `.backup` suffixes
Host A (FI)	Exchange `.env` backup directory copies	15	118 to 855	(Host total: 16)	Exchange backup copies (Oct 2025 and Feb 2026 file dates)
Host A (FI)	Strategy `.env.bak.*` backups and setup scripts	22	225 to 2,811	(Host total: 16)	Strategy backups plus two setup wrapper scripts (1,550 and 1,559 B)
Host A (FI)	Runtime strategy logs and secret-finder script	23	107 to 11,023	Runtime context	Strategy execution logs and local secret-finder script
Host B (US)	Trading-agent root and profile environment files (6 bots)	7	Unsized	4 DB, 2 Binance, 1 AWS, 1 GH, 1 TG	Autonomous WeChat and trader bot environment files
Host B (US)	Exchange-proxy, scanner, and saved environment backups	6	Unsized	(Host total: 9)	Exchange proxy, token scanner, and trading-bot environment backups
Host B (US)	Bundled DEX CLI and MCP skill-generator modules	23	Unsized	Agent CLI tools	Bundled DEX trading tool and MCP skill-generator packages
Host B (US)	Agent SQLite state databases and curator log folders	35	Unsized	State & logs	8 agent SQLite databases plus 11 dated curator folders (Jun-Aug 2026)

CHAPTER 04

MCP Wallet Tool Caches, Seed Derivers, and Kubernetes Secret Backups

IN SHORT

Leaving MCP wallet caches, payment source trees, and Kubernetes restore folders on open HTTP ports exposes raw wallet seed phrases, Ethereum keys, and cluster-wide database credentials. Three servers in France and the United States hold 65 secret records across these surfaces, including 23 BIP-39 seeds, 8 Ethereum keys, and 14 Kubernetes `*-secrets.yaml` files. Isolate MCP wallets and encrypt all cluster backups.

INSIDE THIS CHAPTER

MCP Wallet Tool Cache on Host C (France) 15

Crypto payment platform and BIP-39 wallet deriver on Host D (United States) 16

Plain-text Kubernetes `*-secrets.yaml` backups on Host E (France) vs. mounted pod tokens 16

04 MCP Wallet Tool Caches, Seed Derivers, and Kubernetes Secret Backups

Three additional misconfiguration surfaces show what happens when AI-agent wallet plugins, multi-chain payment tools, and Kubernetes backup scripts share unprotected web roots (T1552.001, T1552.007).

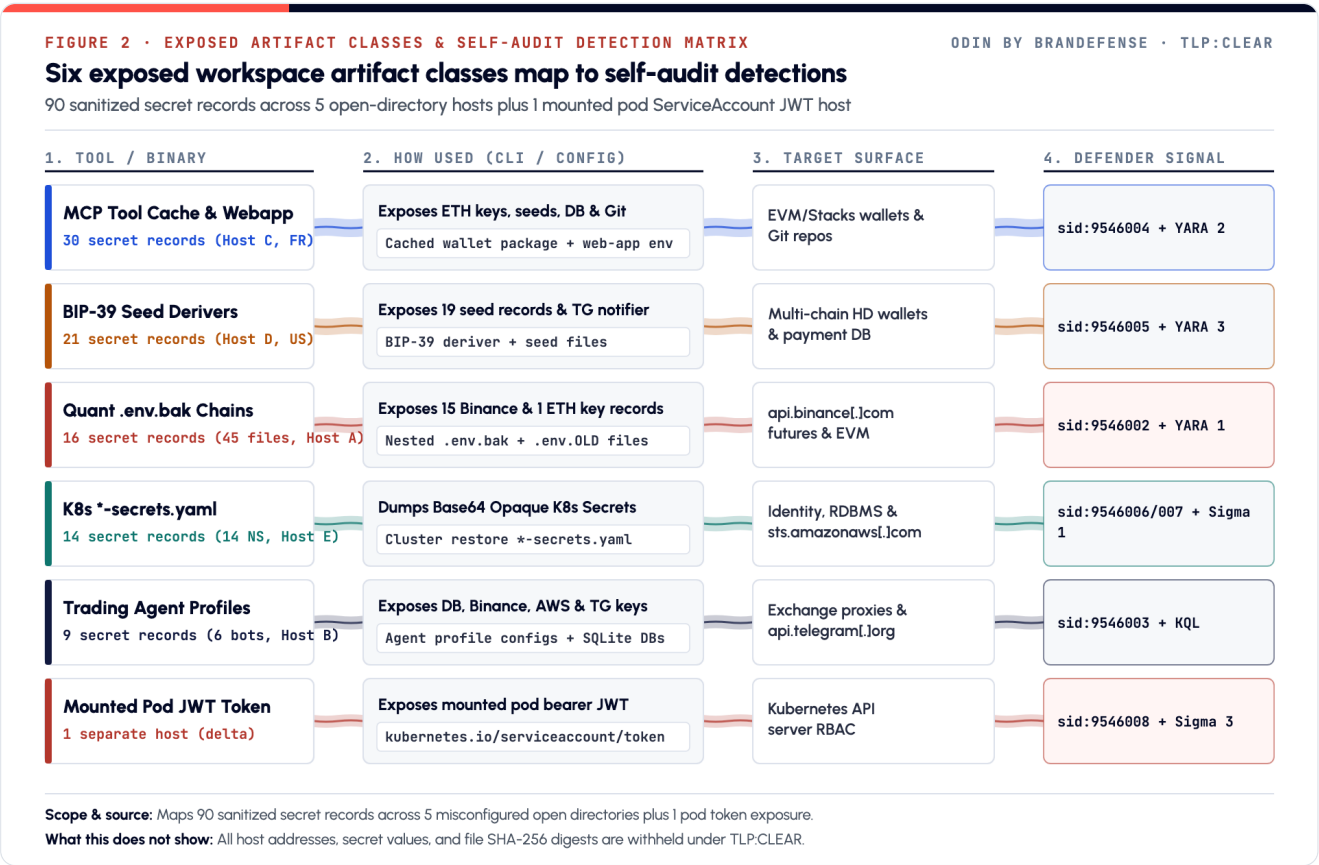


Figure 2: Six exposed workspace artifact classes map to self-audit detections

MCP Wallet Tool Cache on Host C (France)

On Host C in France (address withheld), an open directory reports 19,802 exposed files, with 1,189 sampled entries and 220 detailed rows. It holds 30 secret records (33.33% of the cohort): 9 database URLs, 9 GitHub tokens, 8 Ethereum keys, and 4 BIP-39 seeds. The Ethereum key records represent 72.73% of the 11 arrival-set records.

Inside a package-runner cache (70 detailed rows), the directory exposes cached MCP wallet credential modules across 8 files alongside wallet and transaction SDK packages. MCP wallet packages let AI agents invoke external wallet tools.

On this developer host, those modules share the directory with a web-app environment file, 45 Git repository entries, and 8 web-application build chunks. Broader sampled entries include wallet folders, exchange audit logs, agent configuration folders, and an SSH private key, linking wallet keys to Git repositories and databases.

Crypto payment platform and BIP-39 wallet deriver on Host D (United States)

On Host D in the United States (address withheld), HTTP and HTTPS directory listings (37 and 183 rows) report 1,093 files (449 sampled entries, 183 unique detailed paths). Twenty-one secret records sit exposed: 19 BIP-39 seed phrases (20.65% of the 92 arrival-set seed records), 1 Telegram bot token, and 1 database URL.

Seventy-four unique paths belong to an unpacked crypto payment platform. They include SQL schema and migration files (3,174 and 5,017 bytes), token-backup and Telegram bot code (7,475 and 9,625 bytes), and a 6,348-byte exchange-token script. The folder also holds two agent instruction files measuring 16,384 and 81,920 bytes, with an October 2025 file-date suffix on disk.

Beside that tree, a 25-file wallet-checking directory exposes a 20,480-byte BIP-39 derivation script, a 72-byte plain-text seed file, a 184-byte example file, and a 7,680-byte Telegram notifier. The same directory includes seven multi-chain balance scanners (up to 20,480 bytes across Bitcoin, EVM, Solana, and UTXO chains). It also sits beside 16 OAuth files, 13 email-service files, and 24 `.zip` archives.



Figure 4: Open HTTP directories expose bot backups, MCP wallets, and K8s secrets

Plain-text Kubernetes `*-secrets.yaml` backups on Host E (France) vs. mounted pod tokens

On Host E in France (address withheld), an open directory reports 415 exposed files and 157 sampled entries (149 sized from 1 to 65,734 bytes plus 8 unsized entries). Fourteen secret records sit exposed: 11 database URLs, 2 AWS access keys, and 1 AWS secret key.

One exposed restore directory held 119 Kubernetes YAML files, including 14 namespace Secret manifests ranging from 68 to 1,503 bytes. Their namespace names and file paths are withheld. Alongside them sat a 340-byte single-Secret manifest, a 2,081-byte deployment manifest, and cluster namespace and persistent-volume manifests (21,612 and 27,212 bytes). The folder also held `k3s-1.34.3`, `restic` cache files, and an 8-byte HTTP server PID file.

Kubernetes `Secret` YAML files store values in plain Base64 under `data:`, not ciphertext. Saving `kubectl get secrets -o yaml` output into an open restore folder leaks database URLs, identity-provider credentials, and AWS keys at rest. That static backup mistake differs from the 1 separate arrival-set host that exposed `/run/secrets/kubernetes.io/serviceaccount/token` (T1552.007). Static `*-secrets.yaml` files leak stored application credentials, whereas a mounted `serviceaccount/token` grants bearer access to the Kubernetes API server until expiration or pod shutdown.

Host and country	Workspace artifact group	Detailed rows	Size (bytes)	Host secret total	Exposure risk
Host C (FR)	Package-runner cache with MCP wallet and SDK modules	70	Unsized	9 DB, 9 GH, 8 ETH, 4 BIP-39	Agentic MCP wallet credential modules in an open directory
Host C (FR)	Web-app environment file, Git repository entries (45), and build chunks (8)	54	Unsized	(Host total: 30)	Web-app wallet environment file, exposed Git objects, and SSH private key
Host D (US)	Unpacked crypto payment platform source tree	74 (unique)	93 to 827,392	19 BIP-39, 1 DB, 1 TG	Unpacked payment stack, SQL schemas, token-backup, and Telegram bot code
Host D (US)	BIP-39 wallet deriver, plain-text seed file, and 7 scanners	25 (unique)	0 to 25,600	(Host total: 21)	BIP-39 seed derivation (20,480 B), seed file (72 B), Telegram notifier (7,680 B)
Host E (FR)	Cluster restore folder with 14 namespace `*-secrets.yaml` manifests	14	68 to 1,503	11 DB, 2 AWS access, 1 secret	Base64 Kubernetes `SecretList` and `Secret` namespace dumps
Host E (FR)	Single-Secret and cluster YAML manifests (105), `k3s-1.34.3`, `restic`	124	1 to 65,734	(Host total: 14)	Cluster manifests, persistent volumes, and `restic` cache
Host E (FR)	Redis `/tmp` test or intrusion entries	19	Not separately stated	(Host total: 14)	RDB snapshots, probe markers, and modules

05

CHAPTER 05

Crypto Signing Key Concentration and Owner Boundaries

IN SHORT

Crypto-native records make up 49 of 90 secrets (54.44%) across the five directories. They comprise 23 BIP-39 seed phrases, 17 Binance keys, and 9 Ethereum private keys. Because all five hosts are misconfigured user or cluster servers rather than adversary infrastructure, all addresses and file hashes are withheld.

INSIDE THIS CHAPTER

Owner-versus-operator boundaries and co-hosted test files 20

05 Crypto Signing Key Concentration and Owner Boundaries

Comparing the 90 secret records across these five servers against the broader arrival set shows how heavily crypto signing-key records cluster on developer and trading hosts. Two servers (Host C in France and Host A in Finland) account for 9 of 11 Ethereum private-key records (81.82%) in the arrival set.

Two servers (Host A in Finland and Host B in the United States) account for 17 of 21 Binance-key records (80.95%). Two servers (Host D in the United States and Host C in France) account for 23 of 92 BIP-39 seed-phrase records (25.00%).

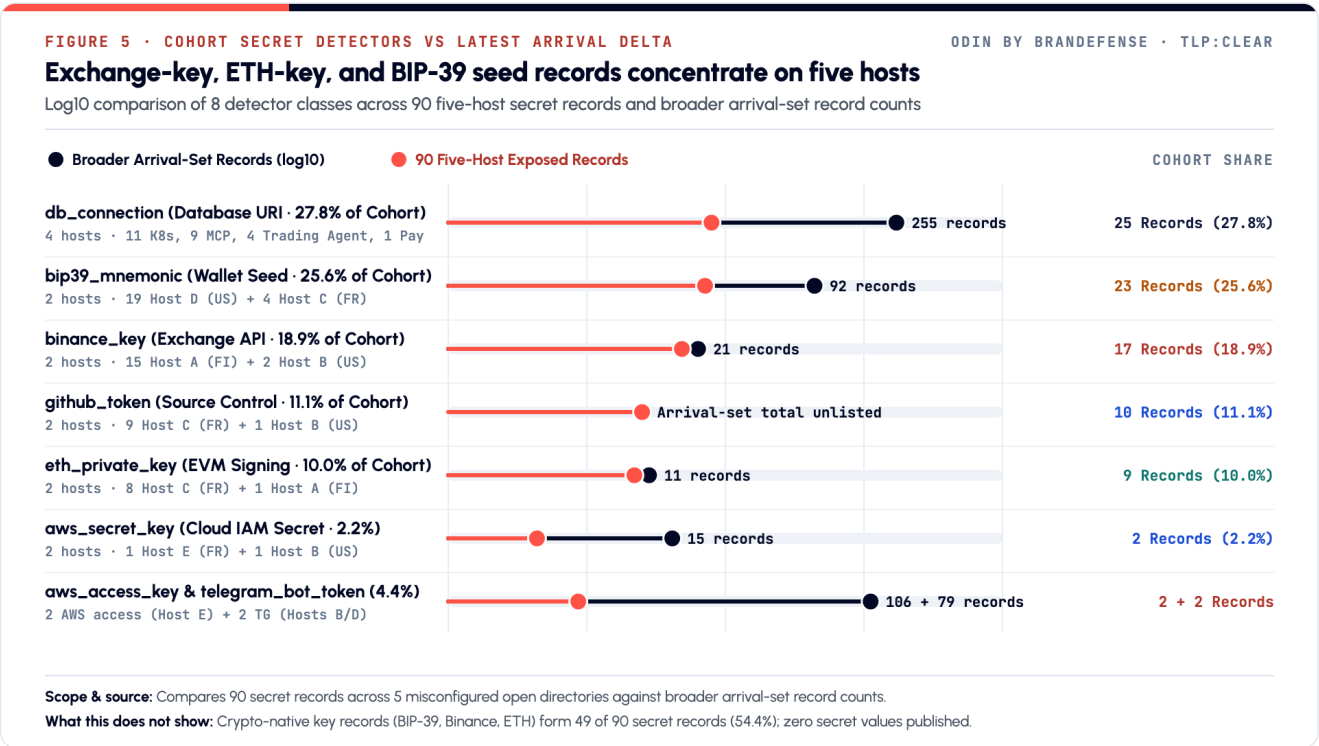


Figure 5: Exchange-key, ETH-key, and BIP-39 seed records concentrate on five hosts

Unlike an API token that can be revoked in place, a leaked BIP-39 seed phrase or raw Ethereum key cannot be rotated. Move assets to a new wallet and revoke token allowances on the exposed address; do not reuse the old key.

Supporting credentials form the other 41 records (45.56%): 25 database URLs, 10 GitHub tokens, 2 AWS access keys, 2 AWS secret keys, and 2 Telegram bot tokens. Our dataset records secret counts per host and secret type (16, 9, 30, 21, and 14) rather than per individual file.

Secret type	Category	Five-host records (90)	Cohort share	Hosts	Broader arrival set	Share of arrival set
`db_connection`	Database	25	27.78%	4	255	9.80%
`bip39_mnemonic`	Crypto wallet	23	25.56%	2	92	25.00%
`binance_key`	Crypto exchange	17	18.89%	2	21	80.95%
`github_token`	Source control	10	11.11%	2	Unlisted	N/A
`eth_private_key`	Crypto wallet	9	10.00%	2	11	81.82%
`aws_access_key`	Cloud IAM	2	2.22%	1	106	1.89%
`aws_secret_key`	Cloud IAM	2	2.22%	2	15	13.33%
`telegram_bot_token`	Messaging / alert	2	2.22%	2	79	2.53%

Owner-versus-operator boundaries and co-hosted test files

Treating exposed developer or cluster hosts as threat actors would misdirect defenders and harm careless owners:

1. **Misconfigured users and administrators.** Host A (Finland), Host B (United States), and Host C (France) are trading and developer trees left behind open HTTP listeners. Kubernetes namespace names on Host E (France) and brand names in `.zip` files on Host D (United States) are withheld.

1. **Co-hosted Redis `/tmp` test files on Host E and multi-use scripts on Host D.** On Host E in France, 19 entries inside a `systemd-private` Redis `/tmp` folder hold RDB snapshots, `authorized_keys`, pentest probe markers, and `.so` modules. They fit either a testbed or a probed K3s and Redis server. On Host D in the United States, the payment-platform tree sits beside a BIP-39 derivation script, a plain-text seed file, and 24 `.zip` archives; all addresses and hashes remain withheld.

1. **Zero credentials or wallet balances tested.** Odin replayed 0 credentials and derived 0 wallet addresses from the 23 seed phrases or 9 Ethereum keys, so funded balances and active-key ratios are unmeasured.

06

CHAPTER 06

What Defenders Should Do

IN SHORT

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

06 What Defenders Should Do

Cloud perimeter and web engineering

- [ ] Scan all cloud VPS, lab, and staging subnets for unauthenticated `Index of /` HTTP listeners across standard and high TCP ports, and bind local development servers strictly to `127.0.0.1`.
- [ ] Block inbound HTTP requests for `.env*`, `.bak*`, and Git repository paths at reverse proxies, CDNs, and WAFs.

Quantitative trading and Web3 wallet security

- [ ] Delete `.env.bak.*` and `.env.OLD` backups from trading hosts, and revoke every historical exchange key stored inside them. Enforce source IP allowlists on `api.binance[.]com` and `api.bitget[.]com`, and disable withdrawal permissions on trading API keys.
- [ ] Move BIP-39 seed phrases and raw EVM or Stacks keys out of flat seed or web-app environment files into hardware wallets, MPC signers, or KMS enclaves. Sweep funds to a fresh seed if any wallet file ever sat in an open directory.

AI agent and Model Context Protocol (MCP) environments

- [ ] Run MCP wallet packages, trading agents, and similar agentic wallet tools inside isolated containers separated from SSH private keys, GitHub tokens, and production database URLs, and require human approval before signing high-value transactions.

Kubernetes platform administration and SOC

- [ ] Stop exporting plain-text `kubectl get secrets -o yaml` files (`*-secrets.yaml`) to disk. Encrypt each exported backup before storage and use encrypted backup operators. Separately enable KMS encryption at rest for etcd. Set `automountServiceAccountToken: false` on pods that do not call the Kubernetes API.
- [ ] Deploy the self-audit Suricata, Sentinel KQL, Sigma, and YARA rules in Chapter 7 to catch accidental open directories and plain-text cluster dumps early.

07

CHAPTER 07

Detection Rules

IN SHORT

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

INSIDE THIS CHAPTER

How the Suricata signatures work 24

How the KQL, Sigma, and YARA rules work 25

MITRE ATT&CK mapping 25

07 Detection Rules

The full rules are in Appendix B. Static syntax checks (`lint_suricata.py` with 0 problems, plus modifier and fixture checks) passed cleanly. Suricata, YARA, Sigma, and Kusto engines were not installed in this environment, and the rules were not replayed against live traffic. Test and tune them in staging before production enforcement.

Rule file	What it catches
`odin_exposed_trading_bots_mcp_wallets_and_k8s_secrets.rules`	Suricata 7: eight rules for open directories, `.env.bak` chains, trading-agent profiles, cached MCP wallet modules, wallet seed files, `*-secrets.yaml`, HTTP `kind: Secret` leaks, and pod `serviceaccount/token` URIs
`sentinel_exposed_trading_bot_mcp_and_k8s_secret_hunt.kql`	Sentinel and Defender XDR: `http.server`, `*-secrets.yaml`, MCP wallet, or `.env.bak` processes matched by initiating process ID and creation time to listening ports or exchange/cloud API sessions
`sigma_k8s_secret_backup_and_mcp_wallet_exposure.yml`	Sigma (3 rules): K8s audit `secrets` list/get calls, Linux `http.server` or `-secrets.yaml` commands, and web hits on exposed bot, MCP wallet, or K8s paths
`odin_exposed_bot_mcp_wallet_and_k8s_secret_artifacts.yar`	YARA (4 rules): quant bot `.env` backups, cached MCP wallet tools, BIP-39 derivers with scanners/Telegram, and plain-text K8s `Secret` YAMLs

How the Suricata signatures work

SID	Looks for	Why
9546001	Outbound 200 with `Index of /` and `.env.bak`, trading-agent profile `.env`, `mcp-server`, or `-secrets.yaml`	Self-audit alert when an internal host exposes a sensitive directory
9546002	Inbound `GET` for `.env.bak`, `.env.backup`, or `.env.old`	Catches requests targeting exposed `.env` backup chains (block candidate)
9546003	Inbound `GET` under `/profiles/` for `.env`, SQLite `.db` / `.sqlite` / `.sql` files, or `logs/`	Catches requests for exposed trading-agent profiles and state databases
9546004	Inbound `GET` under `/node_modules/` or `/_npx/` for cached `mcp-server` module files	Flags HTTP access to cached MCP wallet package modules
9546005	Inbound `GET` for plain-text `seed`, `wallet_seed`, or `bip39_seed` files	Flags HTTP access to exposed BIP-39 wallet seed files
9546006	Inbound `GET` ending in `-secrets.yaml`	Catches retrieval of plain-text K8s namespace Secret files (block candidate)
9546007	Outbound 200 containing `apiVersion: v1`, `kind: Secret`, and `data:`	Self-audit alert when an internal HTTP port serves plain-text K8s Secret YAML
9546008	Inbound URI containing `/kubernetes.io/serviceaccount/token`	Catches requests for mounted pod ServiceAccount tokens (block candidate)

Test SIDs 9546002, 9546006, and 9546008 in staging. Deploy equivalent path blocks at reverse proxies, or convert verified Suricata alerts to `drop` rules in an inline IPS. Treat any SID 9546001 or 9546007 outbound 200 alert on your own network as an immediate posture incident: close the listener and rotate every exposed credential.

How the KQL, Sigma, and YARA rules work

The Sentinel KQL query looks back 7 days across `DeviceProcessEvents` and `DeviceNetworkEvents`. It finds `python`, `node`, `npx`, `kubectl`, `bash`, `sh`, or `restic` commands naming `http.server`, `SimpleHTTPServer`, `mcp-server`, `wallet-sdk`, `get secrets`, `-secrets.yaml`, `serviceaccount/token`, `.env.bak`, `.env.backup`, or `.env.old`. It matches `ProcessId` and `ProcessCreationTime` to `InitiatingProcessId` and `InitiatingProcessCreationTime` across six listed listening ports and five exchange, messaging, and cloud API services. Two exact example hosts are excluded.

Sigma rule 1 flags Kubernetes audit `secrets` `list` or `get` calls from `kubectl/`, `curl/`, `python-requests/`, or `Go-http-client/` outside two exact backup ServiceAccounts. Sigma rule 2 flags Linux `-m http.server`, `SimpleHTTPServer`, `mcp-server`, `wallet-sdk`, `get secrets`, `-secrets.yaml`, or `.env.bak` commands outside the two exact audit hosts. Sigma rule 3 flags web requests for `.env.bak`, `.env.backup`, `.env.old`, `/profiles/`, `mcp-server/`, `/.git/`, `/.ssh/`, `-secrets.yaml`, or `/kubernetes.io/serviceaccount/token`.

YARA rule	Fires when
`ODIN_Exposed_Quant_Bot_Env_Backup_Chain`	Under 256 KB, at least two Binance or Ethereum private-key variable names appear in a scanned backup folder
`ODIN_Exposed_MCP_Crypto_Wallet_Credential_Tool`	Under 512 KB, an MCP server or SDK marker appears with two crypto-wallet SDK or mnemonic markers
`ODIN_Exposed_BIP39_Wallet_Deriver_And_Notifier`	Under 512 KB, a BIP-39 HD path or seed generator appears with two wallet-scanner or Telegram markers
`ODIN_Unencrypted_Kubernetes_Secret_Manifest_Dump`	Under 10 MB, `apiVersion: v1` and `kind: Secret`, `SecretList`, or `List` appear with `type:` and a root or indented `data:` map

MITRE ATT&CK mapping

Tactic	Technique	Observed exposure pattern	Detection
Credential Access	T1552.001 Unsecured Credentials: Credentials In Files	45 `.env.bak` files, cached MCP wallet modules, plain-text seed files, and 14 `*-secrets.yaml` files	SIDs 9546001 to 9546007; Sigma rules 1 to 3; YARA rules 1 to 4
Credential Access	T1552.007 Unsecured Credentials: Container API	Mounted pod `/run/secrets/kubernetes.io/serviceaccount/token` vs static K8s YAML backups	SID 9546008; Sigma rules 1 and 3; YARA rule 4; KQL
Credential Access	T1528 Steal Application Access Token	17 Binance API keys, 10 GitHub tokens, and 2 Telegram bot tokens in exposed workspaces	SIDs 9546002 and 9546003; KQL; Sigma rules 2 and 3; YARA rule 1
Defense Evasion, Persistence	T1078.004 Valid Accounts: Cloud Accounts	2 AWS access keys and 2 AWS secret keys exposed in `*-secrets.yaml` and trading-agent configs	SIDs 9546006 and 9546007; KQL; Sigma rule 1; YARA rule 4

CHAPTER 08

Indicators

IN SHORT

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

INSIDE THIS CHAPTER

Self-audit exposure patterns in this report 27

08 Indicators

Appendix A holds 0 published indicators: 0 network addresses and 0 file hashes. All five open-directory server addresses and all file SHA-256 digests are withheld because every host in this study is a misconfigured developer, trader, or cluster server rather than adversary infrastructure. Sixteen defensive self-audit file, directory, and container-token path patterns appear in this report only.

Type	Count	Where to use it
Network addresses (Appendix A)	0	All five misconfigured host addresses are withheld
File hashes (Appendix A)	0	All secret-bearing and unestablished file hashes are withheld
Self-audit workspace paths (in text)	15	Block at WAFs and scan your own cloud VPS, build, and backup folders
Container runtime token path (in text)	1	Block inbound HTTP traversal and audit pod `automountServiceAccountToken` settings

Self-audit exposure patterns in this report

Pattern or artifact class	Type	Exposure surface	Defensive self-audit use
`*.env.bak*`	Workspace file pattern	Surface 1 (Host A)	Block `.env.bak*` at WAFs and purge nested backup chains
`*.env.OLD*`	Workspace file pattern	Surface 1 (Host A)	Audit trading servers for dated `.env.OLD` copies and revoke keys
`*binance*.env*`	Workspace file pattern	Surface 1 (Host A)	Block at WAFs and revoke historical exchange keys
`*.env.backup*`	Workspace file pattern	Surface 1 (Host A)	Audit trading hosts for dated `.env.backup` files
Trading-agent profile directories	Workspace directory class	Surface 2 (Host B)	Block HTTP access to multi-bot profile environment and state files
Agent SQLite state databases	Workspace database class	Surface 2 (Host B)	Block HTTP access to trading-agent SQLite state databases
Editor-saved environment backups	Workspace file class	Surface 2 (Host B)	Remove editor-saved environment backup artifacts from trading workspaces
Cached MCP wallet modules	Package cache class	Surface 3 (Host C)	Audit developer hosts exposing cached MCP wallet packages
Web-app local environment files	Workspace file class	Surface 3 (Host C)	Block HTTP access to web-application wallet environment files
Exposed Git repository metadata	Repository metadata class	Surface 3 (Host C)	Block HTTP access to Git repository directories in web roots
Exposed SSH private keys	Credential file class	Surface 3 (Host C)	Never serve user home directories or SSH private keys over HTTP
`*seed*.txt`	Workspace file pattern	Surface 4 (Host D)	Audit servers for plain-text BIP-39 seed phrase files

Pattern or artifact class	Type	Exposure surface	Defensive self-audit use
`*deriver*.py`	Workspace file pattern	Surface 4 (Host D)	Audit servers for exposed BIP-39 HD wallet derivation scripts
`*-secrets.yaml`	Workspace file pattern	Surface 5 (Host E)	Scan backup hosts for plain-text Kubernetes Secret dumps
Single-Secret Kubernetes YAML manifests	Workspace manifest class	Surface 5 (Host E)	Block HTTP access and encrypt exported Kubernetes Secret manifests
`/run/secrets/kubernetes.io/serviceaccount/token`	Container token path	Surface 5 (K8s runtime)	Block web traversal and restrict pod token automounting

`api.binance[.]com`, `fapi.binance[.]com`, `api.bitget[.]com`, `sts.amazonaws[.]com`, and `api.telegram[.]org` are legitimate public services. Never block them globally at the firewall; audit internal hosts that expose configuration files while connecting to those endpoints.

Withheld: all five server IP addresses, Kubernetes namespace names on Host E (France), and third-party brand names in `.zip` archives on Host D (United States). All victim-specific paths, secret values, and file SHA-256 digests are also withheld.

09

CHAPTER 09

Scope and Limits

IN SHORT

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

09 Scope and Limits

- **Window.** All observations come from a two-week observation window ending 20 November 2026.
- **Static review, zero credentials tested.** File paths, sizes, and secret counts were reviewed statically without running scripts, testing any of the 90 secret records, deriving wallet addresses, or contacting any server. Active key ratios and wallet balances are unmeasured.
- **Five surfaces are an analytical taxonomy.** The five servers share 0 IP addresses, certificates, SSH keys, or bot tokens. Grouping them into five surfaces maps posture controls across bot, wallet, and cluster environments rather than naming an actor.
- **Misconfigured owners, not adversary operators.** All five servers are treated as developer, trader, or cluster administrator hosts, including Host E in France where the cluster restore folder sat alongside 19 Redis `/tmp` test or intrusion files. All host addresses and file hashes are withheld.
- **Server file dates vs observation window.** October 2025 and February 2026 suffixes on Host A in Finland are filename dates on disk. June to August 2026 curator folders on Host B in the United States and the October 2025 file-date suffix on Host D are server file dates, not Odin's monitoring window.
- **Rule testing.** Static checks (`lint_suricata.py` with 0 problems, plus modifier and fixture checks) passed on 21 November 2026. Suricata, YARA, Sigma, and Kusto engines were not installed locally, so test and tune rules before production use.

10

CHAPTER 10

Sources

IN SHORT

This chapter lists the public references used in the report.

10 Sources

1. MITRE ATT&CK. T1552.001 Unsecured Credentials: Credentials In Files. https://attack.mitre.org/techniques/T1552/001/ (verified 21 November 2026)
2. MITRE ATT&CK. T1552.007 Unsecured Credentials: Container API. https://attack.mitre.org/techniques/T1552/007/ (verified 21 November 2026)
3. MITRE ATT&CK. T1528 Steal Application Access Token. https://attack.mitre.org/techniques/T1528/ (verified 21 November 2026)
4. MITRE ATT&CK. T1078.004 Valid Accounts: Cloud Accounts. https://attack.mitre.org/techniques/T1078/004/ (verified 21 November 2026)
5. Kubernetes Documentation. Good practices for Kubernetes Secrets. https://kubernetes.io/docs/concepts/security/secrets-good-practices/ (verified 21 November 2026)
6. Kubernetes Documentation. Managing Service Accounts. https://kubernetes.io/docs/reference/access-authn-authz/service-accounts-admin/ (verified 21 November 2026)
7. Model Context Protocol Documentation. Tools Specification and Security Considerations. https://modelcontextprotocol.io/docs/concepts/tools (verified 21 November 2026)
8. OWASP Cheat Sheet Series. Secrets Management Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html (verified 21 November 2026)

APPENDICES

Indicators and Detection Rules

FOR DEFENDERS

No individual indicators are published for this study; Appendix A explains why. Appendix B carries every published detection rule in full.

INSIDE

A. Indicators of compromise (0) 34

B. Detection rules (4 files) 35

A Indicators of Compromise

No individual addresses or file hashes are published for this study. The hosts in this report are exposed or unattributed systems rather than confirmed adversary infrastructure, so they stay anonymous. Use the detection rules in Appendix B.

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_bot_mcp_wallet_and_k8s_secret_artifacts.yar

YARA

```
/*
   Odin by Brandefense - Monograph 46 / 46 Defensive Self-Audit YARA Suite
   Purpose: Audit web roots, developer staging folders, and backup directories for unencrypted
            quant trading bot .env backups, MCP crypto-wallet tool caches, BIP-39 derivers,
            and plain-text Kubernetes Secret / SecretList YAML dumps.
   Validation: Structural and fixture checks passed; test in staging before production scanning.
*/

rule ODIN_Exposed_Quant_Bot_Env_Backup_Chain {
    meta:
        author = "Odin by Brandefense"
        monograph = "46 / 46"
        date = "2026-11-20"
        mitre_attack = "T1552.001, T1528"
        description = "Detects plain-text quantitative trading bot environment backup files staging exchange API
keys and Ethereum private key variables"
    strings:
        $k1 = "BINANCE_API_KEY" ascii nocase
        $k2 = "BINANCE_API_SECRET" ascii nocase
        $k3 = "BINANCE_SECRET" ascii nocase
        $k4 = "ETH_PRIVATE_KEY" ascii nocase
    condition:
        filesize < 256KB and (2 of ($k*))
}

rule ODIN_Exposed_MCP_Crypto_Wallet_Credential_Tool {
    meta:
        author = "Odin by Brandefense"
        monograph = "46 / 46"
        date = "2026-11-20"
        mitre_attack = "T1552.001"
        description = "Detects Model Context Protocol crypto-wallet credential tool modules and wallet SDK
handlers exposed in web-accessible directories"
    strings:
        $mcp1 = "@modelcontextprotocol/sdk" ascii
        $mcp2 = "McpServer" ascii
        $mcp3 = "mcp-server" ascii
        $wal1 = "wallet-sdk" ascii
        $wal2 = "mnemonic" ascii nocase
        $wal3 = "generateWallet" ascii
    condition:
        filesize < 512KB and (1 of ($mcp*)) and (2 of ($wal*))
}

rule ODIN_Exposed_BIP39_Wallet_Deriver_And_Notifier {
    meta:
        author = "Odin by Brandefense"
        monograph = "46 / 46"
        date = "2026-11-20"
        mitre_attack = "T1552.001"
        description = "Detects Python or TypeScript BIP-39 hierarchical deterministic wallet derivation scripts
paired with multi-chain scanners or Telegram bot dispatchers"
    strings:
        $hd1 = "m/44'/60'/0'/0" ascii
        $hd2 = "m/44'/0'/0'/0" ascii
        $hd3 = "Bip39SeedGenerator" ascii
        $hd4 = "mnemonic_to_seed" ascii
        $sc1 = "eth_getBalance" ascii
        $sc2 = "getBalance" ascii
        $sc3 = "api.telegram.org/bot" ascii
```

```
        $sc4 = "sendMessage" ascii
    condition:
        filesize < 512KB and (1 of ($hd*)) and (2 of ($sc*))
}

rule ODIN_Unencrypted_Kubernetes_Secret_Manifest_Dump {
    meta:
        author = "Odin by Brandefense"
        monograph = "46 / 46"
        date = "2026-11-20"
        mitre_attack = "T1552.001, T1552.007"
        description = "Detects unencrypted Kubernetes Secret or SecretList YAML exports saved on disk with Base64
data maps"
    strings:
        $api = "apiVersion: v1" ascii
        $kind1 = "kind: Secret" ascii
        $kind2 = "kind: SecretList" ascii
        $kind3 = "kind: List" ascii
        $type1 = "type: Opaque" ascii
        $type2 = "kubernetes.io/service-account-token" ascii
        $type3 = "kubernetes.io/dockerconfigjson" ascii
        $data = /\n[ \t]*data:/
    condition:
        filesize < 10MB and $api and (1 of ($kind*)) and (1 of ($type*)) and $data
}
```

odin_exposed_trading_bots_mcp_wallets_and_k8s_secrets.rules

SURICATA

```
# Odin by Brandefense - Monograph 46 / 46 Suricata 7.x Self-Audit & Perimeter Exposure Suite
# Scope: Accidental HTTP exposure of quant trading bot .env.bak chains, trading-agent profiles, cached MCP wallet
packages, BIP-39 wallet seed files, unencrypted Kubernetes *-secrets.yaml backups, and mounted pod
serviceaccount/token paths
# Validation: Static syntax and lint_suricata.py checked (0 problems); test in staging before inline enforcement.

alert http $HOME_NET any -> any any (msg:"ODIN R46 Self-Audit Open Directory Exposing Trading Bot MCP or K8s
Backup Tree"; flow:established,to_client; http.stat_code; content:"200"; bsize:3; file.data; content:"Index
of /"; fast_pattern; pcre:"/(\.env\.(bak|backup|old)|profiles\/[a-z0-9_-]+\/\.env|mcp-server|[a-z0-9_-]+-
secrets\.yaml)/i"; classtype:policy-violation; sid:9546001; rev:1; metadata:policy self_audit, mitre_technique_id
T1552.001;)
alert http any any -> $HOME_NET any (msg:"ODIN R46 Inbound HTTP Request for Exposed Trading Bot Env Backup
Chain"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:".env."; bsize:>=8;
fast_pattern; pcre:"/\.env\.(bak|backup|old)/i"; classtype:credential-theft; sid:9546002; rev:1; metadata:policy
security-ips, mitre_technique_id T1552.001;)
alert http any any -> $HOME_NET any (msg:"ODIN R46 Inbound HTTP Request for Exposed Trading Agent Profile or
State DB"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"/profiles/";
bsize:>=14; fast_pattern; pcre:"/\/profiles\/[a-z0-9_-]+\/(\.env|[a-z0-9_-]+\.(db|sqlite|sql)|logs\/)/i";
classtype:credential-theft; sid:9546003; rev:1; metadata:policy security-ips, mitre_technique_id T1528;)
alert http any any -> $HOME_NET any (msg:"ODIN R46 Inbound HTTP Request for Exposed MCP Crypto Wallet Package
Cache"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"mcp-server"; nocase;
bsize:>=15; fast_pattern; pcre:"/\/(node_modules|_npx)\/.*mcp-server.*\.(js|ts|json)/i"; classtype:credential-
theft; sid:9546004; rev:1; metadata:policy behavioral_hunt, mitre_technique_id T1552.001;)
alert http any any -> $HOME_NET any (msg:"ODIN R46 Inbound HTTP Request for Exposed BIP39 Wallet Seed File";
flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"seed"; nocase; bsize:>=9;
fast_pattern; pcre:"/\/(seeds?|wallet[-_]?seeds?|bip39[-_]?seeds?)\.(txt|json|csv|env|bak)$/i";
classtype:credential-theft; sid:9546005; rev:1; metadata:policy behavioral_hunt, mitre_technique_id T1552.001;)
alert http any any -> $HOME_NET any (msg:"ODIN R46 Inbound HTTP Request for Unencrypted Kubernetes Namespace
Secrets YAML"; flow:established,to_server; http.method; content:"GET"; bsize:3; http.uri; content:"-
secrets.yaml"; endswith; bsize:>=14; fast_pattern; pcre:"/\/[a-z0-9_-]+-secrets\.yaml$/i"; classtype:credential-
theft; sid:9546006; rev:1; metadata:policy security-ips, mitre_technique_id T1552.001;)
alert http $HOME_NET any -> any any (msg:"ODIN R46 Self-Audit Unencrypted Kubernetes Secret Manifest Served Over
HTTP"; flow:established,to_client; http.stat_code; content:"200"; bsize:3; file.data; content:"apiVersion: v1";
content:"kind: Secret"; fast_pattern; content:"data:"; classtype:policy-violation; sid:9546007; rev:1;
metadata:policy self_audit, mitre_technique_id T1552.001;)
alert http any any -> $HOME_NET any (msg:"ODIN R46 Inbound HTTP Request for Mounted Kubernetes Pod ServiceAccount
Token"; flow:established,to_server; http.uri; content:"/kubernetes.io/serviceaccount/token"; bsize:>=35;
fast_pattern; classtype:web-application-attack; sid:9546008; rev:1; metadata:policy security-ips,
mitre_technique_id T1552.007;)
```

sentinel_exposed_trading_bot_mcp_and_k8s_secret_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Monograph 46 / 46 Microsoft Sentinel & Defender XDR Self-Audit Hunt Query
// Purpose: Detect ad-hoc HTTP directory servers, unencrypted kubectl Secret YAML exports, or exposed MCP
wallet / trading-bot workspaces paired with external network activity
let Lookback = 7d;
let ApprovedBackupHosts = dynamic([
    "k8s-velero-controller-01.corp.example",
    "sec-posture-audit-01.corp.example"
]);
let ExposureProcesses =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBackupHosts))
    | where FileName in~ ("python", "python3", "node", "npx", "kubectl", "bash", "sh", "restic")
    | where ProcessCommandLine has_any (
        "http.server",
        "SimpleHTTPServer",
        "mcp-server",
        "wallet-sdk",
        "get secrets",
        "-secrets.yaml",
        "serviceaccount/token",
        ".env.bak",
        ".env.backup",
        ".env.old"
    )
    | extend ExposureClass = case(
        ProcessCommandLine has_any ("get secrets", "-secrets.yaml", "serviceaccount/token"),
"KubernetesSecretDumpOrToken",
        ProcessCommandLine has_any ("mcp-server", "wallet-sdk"), "MCPOrWalletKeyWorkspace",
        ProcessCommandLine has_any ("http.server", "SimpleHTTPServer"), "AdHocHTTPDirectoryServer",
        "TradingBotBackupChain"
    )
    | project ProcTime = Timestamp, DeviceId, DeviceName, ProcessId, ProcessCreationTime, AccountName, FileName,
ProcessCommandLine, ExposureClass;
let ExternalAndAPIActivity =
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedBackupHosts))
    | where ActionType in ("ListeningConnectionCreated", "InboundConnectionAccepted", "ConnectionSuccess")
    | where LocalPort in (80, 443, 7070, 8080, 8899, 9999)
        or RemoteUrl has_any (
            "api.binance.com",
            "fapi.binance.com",
            "api.bitget.com",
            "api.telegram.org",
            "sts.amazonaws.com"
        )
    | summarize
        NetworkEvents = count(),
        ListeningPorts = make_set_if(LocalPort, ActionType in ("ListeningConnectionCreated",
"InboundConnectionAccepted"), 10),
        RemoteEndpoints = make_set(RemoteUrl, 10),
        LastNetTime = max(Timestamp)
      by DeviceId, DeviceName, ProcessId = InitiatingProcessId, ProcessCreationTime =
InitiatingProcessCreationTime;
ExposureProcesses
| join kind=leftouter ExternalAndAPIActivity on DeviceId, DeviceName, ProcessId, ProcessCreationTime
| extend RemediationPriority = case(
    ExposureClass in ("KubernetesSecretDumpOrToken", "AdHocHTTPDirectoryServer") and array_length(ListeningPorts)
> 0, "URGENT_CLOSE_OPEN_DIRECTORY_AND_ROTATE",
    ExposureClass == "MCPOrWalletKeyWorkspace" and NetworkEvents > 0, "AUDIT_MCP_WALLET_AND_REVOKE_KEYS",
    "REVIEW_LOCAL_SECRET_HYGIENE"
)
| project
    ProcTime,
    LastNetTime,
    DeviceName,
    AccountName,
```

```
    FileName,
    ProcessId,
    ProcessCreationTime,
    ExposureClass,
    NetworkEvents = coalesce(NetworkEvents, 0),
    ListeningPorts,
    RemoteEndpoints,
    RemediationPriority,
    ProcessCommandLine
| sort by NetworkEvents desc, ProcTime desc
```

sigma_k8s_secret_backup_and_mcp_wallet_exposure.yml

SIGMA

```
title: Kubernetes Bulk Secret Resource Export By Non-Controller Principal
id: 7a31d842-6e19-4c85-9b34-46a01b954601
status: experimental
description: Detects Kubernetes API audit events where secrets are listed or retrieved at namespace or cluster
scope by CLI or script user agents outside approved encrypted backup controllers, matching unencrypted *-
secrets.yaml backup dumps.
author: Odin by Brandefense
date: 2026-11-20
logsource:
    product: kubernetes
    service: audit
detection:
    selection_secret_read:
        objectRef.resource: 'secrets'
        verb:
            - 'list'
            - 'get'
    selection_cli_agent:
        userAgent|contains:
            - 'kubectl/'
            - 'curl/'
            - 'python-requests/'
            - 'Go-http-client/'
    filter_approved_controllers:
        user.username:
            - 'system:serviceaccount:velero:velero-server'
            - 'system:serviceaccount:kube-system:encrypted-backup-operator'
    condition: selection_secret_read and selection_cli_agent and not 1 of filter_*
falsepositives:
    - Interactive cluster administrator troubleshooting sessions outside GitOps or SealedSecrets workflows
level: medium
tags:
    - attack.credential-access
    - attack.t1552.001
    - attack.t1552.007
---
title: Linux Ad-Hoc HTTP Directory Server Or Plaintext K8s Secret YAML Dump
id: 4c92f118-3b74-49d2-8e62-46a02b954602
status: experimental
description: Detects Linux command lines launching unauthenticated HTTP file servers, invoking cached MCP wallet
packages on shared hosts, or redirecting kubectl get secrets output into plain-text YAML files.
author: Odin by Brandefense
date: 2026-11-20
logsource:
    product: linux
    category: process_creation
detection:
    selection_bin:
        Image|endswith:
            - '/python'
            - '/python3'
            - '/npx'
            - '/node'
            - '/kubectl'
            - '/bash'
```

```
            - '/sh'
    selection_args:
        CommandLine|contains:
            - '-m http.server'
            - 'SimpleHTTPServer'
            - 'mcp-server'
            - 'wallet-sdk'
            - 'get secrets'
            - '-secrets.yaml'
            - '.env.bak'
    filter_approved_hosts:
        ComputerName:
            - 'k8s-velero-controller-01.corp.example'
            - 'sec-posture-audit-01.corp.example'
    condition: selection_bin and selection_args and not 1 of filter_*
falsepositives:
    - Local developer loopback test servers bound strictly to 127.0.0.1 on isolated workstations
level: high
tags:
    - attack.credential-access
    - attack.t1552.001
    - attack.t1528
---
title: Web Server Access To Exposed Bot Env Backups MCP Wallet Tools Or K8s Secrets
id: 8e50b379-2d41-4a90-9c17-46a03b954603
status: experimental
description: Detects HTTP requests hitting exposed .env.bak chains, trading-agent profile directories, cached MCP
wallet packages, Git metadata, SSH private keys, unencrypted *-secrets.yaml manifests, or mounted pod
serviceaccount tokens.
author: Odin by Brandefense
date: 2026-11-20
logsource:
    category: webserver
detection:
    selection_uri:
        cs-uri-stem|contains:
            - '.env.bak'
            - '.env.backup'
            - '.env.old'
            - '/profiles/'
            - 'mcp-server/'
            - '/.git/'
            - '/.ssh/'
            - '-secrets.yaml'
            - '/kubernetes.io/serviceaccount/token'
    condition: selection_uri
falsepositives:
    - Authorized external attack-surface scanners verifying perimeter hardening
level: high
tags:
    - attack.credential-access
    - attack.t1552.001
    - attack.t1552.007
    - attack.t1528
```

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/trading-bots-mcp-wallets-and-kubernetes-backups/
Short Read · brandefense.io/odin-research/blog/how-exposed-trading-bots-mcp-wallets-and-kubernetes-backups-leak/
Defense Kit · brandefense.io/odin-research/defense-kit/