ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 39 / 30 · RECON WORKBENCHES

Exposed Recon Queues and Maven Files

Inside an Open HTTP Workbench

How an exposed HTTP workbench combines unattended agent queues, scope polling, staged Java coordinates, and 42 credential findings

Prepared November 2026
Full Report · Public edition · Version 1.1

843 / 599
Reported exposed files across 12 directory slices, with 599 sampled unique entries on the open HTTP server

220
Itemized agent task output files across nine anonymized session folders (113 symlinks and 107 shell files)

7.59 MB
Total volume across 72 sized task output files (ranging from 33,043 to 261,889 bytes)

3
Staged Maven directory files: a Java source file, a JAR, and a POM (coordinate and filenames withheld)

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

How an exposed HTTP workbench combines unattended agent queues, scope polling, staged Java coordinates, and 42 credential findings

01	Executive Summary	04
02	What Was Found on the Open HTTP Server: Four Layers and 843 Reported Files	07
03	Unattended Agent Recon Queues and Nine Session Folders	11
04	Go Scope-Polling Files and the Staged Maven Repository	14
05	Forty-Two Exposed Credentials and Other Ways to Read the Evidence	17
06	What Defenders Should Do	20
07	Detection Rules	22
08	Indicators	25
09	Scope and Limits	28
10	Sources	30
A	Indicators of compromise (0)	32
B	Detection rules (4 files)	33

IN SHORT

Odin by Brandefense examined directory records dated 3 to 16 August 2026 from an open HTTP server in France. It exposed 843 reported files across 12 directory slices (599 sampled unique entries). The server co-located 220 agent task outputs across nine session folders, Go scope-polling files, a staged Maven directory (coordinate and filenames withheld), and 42 credential findings. Pin Maven and Gradle builds to internal HTTPS proxies, enforce artifact checksums, and audit cloud tokens.

01

CHAPTER 01

Executive Summary

IN SHORT

Odin by Brandefense examined directory records dated 3 to 16 August 2026 from an open HTTP server in France. It exposed 843 reported files across 12 directory slices (599 sampled unique entries). The server co-located 220 agent task outputs across nine session folders, Go scope-polling files, a staged Maven directory (coordinate and filenames withheld), and 42 credential findings. Pin Maven and Gradle builds to internal HTTPS proxies, enforce artifact checksums, and audit cloud tokens.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 05

01 Executive Summary

Autonomous coding agents let operators and security researchers run reconnaissance queues overnight without sitting at a keyboard. On this host, reconnaissance queues, scope-polling files, and a staged Java package directory sit on the same open HTTP port. We inspected directory metadata and file structures without running any code, replaying any credential, or contacting any third-party target.

1 **Unattended agent queues pair overnight reconnaissance summaries with nine session trees.**

Two exposed agent workspace folders sit next to three root files: an overnight summary, an API response file, and a setup log (exact host paths withheld). Across 220 itemized task outputs in nine anonymized session folders, 113 symbolic links track delegated subagents and 107 regular files hold background shell logs. Seventy-two sized logs total 7.59 MB, and 12 carry credential or exploit tags.

2 **Go scope-polling files sit beside a staged Maven repository.**

On the same HTTP root, a Go scope-polling directory lists files for polling, sitemap handling, storage, and Cloudflare Compose (project name and paths withheld). Next to it, a staged Maven directory lists a Java source file, a JAR, and a POM (coordinate, filenames, and paths withheld). Unpinned Maven or Gradle builds and cluster library loaders induced to query that HTTP repository could pull the staged JAR.

3 **Open HTTP directory exposure leaks forty-two cloud, analytics, and database credentials.**

Automated secret checks across the host found 42 credential records without publishing any raw secret value. AWS IAM keys account for 71.4% of those records (29 access key IDs and one secret key), alongside seven cloud analytics workspace tokens and five database connection strings. An open directory listener can expose credentials stored within its served directory tree, while the host-level summary does not identify which files held all 42 findings.

Three Decisions for Leaders

1. **Pin every Java build to an internal HTTPS artifact proxy.** Block direct developer and CI runner access to external HTTP Maven repositories, enforce mirror settings, and require SHA-256 and PGP checks on every dependency.
2. **Audit cloud IAM and analytics workspace tokens for unattended agent abuse.** Rotate exposed AWS access keys, bind analytics workspace tokens to IP allowlists, and alert on burst API enumeration from unapproved networks.
3. **Govern internal red-team and bug-bounty servers like production systems.** Ban unauthenticated HTTP directory listeners, isolate offensive agent sandboxes, and purge task transcripts that hold live credentials.

The Study in Numbers

Number	What it is
843 / 599	Reported exposed files across 12 directory slices, with 599 sampled unique entries on the open HTTP server
220	Itemized agent task output files across nine anonymized session folders (113 symlinks and 107 shell files)
7.59 MB	Total volume across 72 sized task output files (ranging from 33,043 to 261,889 bytes)
3	Staged Maven directory files: a Java source file, a JAR, and a POM (coordinate and filenames withheld)
42	Exposed host credential records: 29 AWS keys, 7 analytics tokens, 5 database strings, and 1 AWS secret
0	Published IP addresses or file SHA-256 hashes, withheld because researcher ownership remains plausible
12 / 15	Published generic Maven layout, task-glob, CLI, and build-config patterns in Chapter 8 (coordinates, filenames, project name, port, and 15 host paths withheld)

How to use this report.

Security leaders should start with Chapters 1, 5, and 6. DevSecOps, Java build, and cloud IAM engineers should read Chapters 4, 5, and 6. SOC analysts and detection engineers should focus on Chapters 7 and 8. CTI analysts will find the workspace and session breakdown in Chapters 2 and 3.

02

CHAPTER 02

What Was Found on the Open HTTP Server: Four Layers and 843 Reported Files

IN SHORT

One unauthenticated HTTP server in France exposed 843 reported files across 12 directory slices. Four co-hosted layers join autonomous agent queues, Go scope-polling files, a staged Maven repository, and 42 leaked credentials. Audit open HTTP ports and lock build proxies.

INSIDE THIS CHAPTER

How the 843 reported files break down 07

Four operational layers on one filesystem 08

02 What Was Found on the Open HTTP Server: Four Layers and 843 Reported Files

Manual reconnaissance usually leaves isolated tool output on an analyst workstation. Autonomous CLI coding agents work differently. Operators pair scope-polling utilities with multi-session agent loops that enumerate APIs, test endpoints, and write summary Markdown reports overnight (T1595.002). When that workspace also serves a Maven package tree over cleartext HTTP, reconnaissance and dependency staging share one directory root (T1195.001).

The exposed HTTP server sits in France. Its IP address, port, and scope-poller project name are withheld because open-source scope polling and textbook Java test names can belong to a security researcher. A bug-bounty hunter may have left an automation host open by mistake. Directory index records on the server carry file listing dates of 3 to 16 August 2026. We model the host as a four-layer research workbench rather than attributing it to a named threat actor.

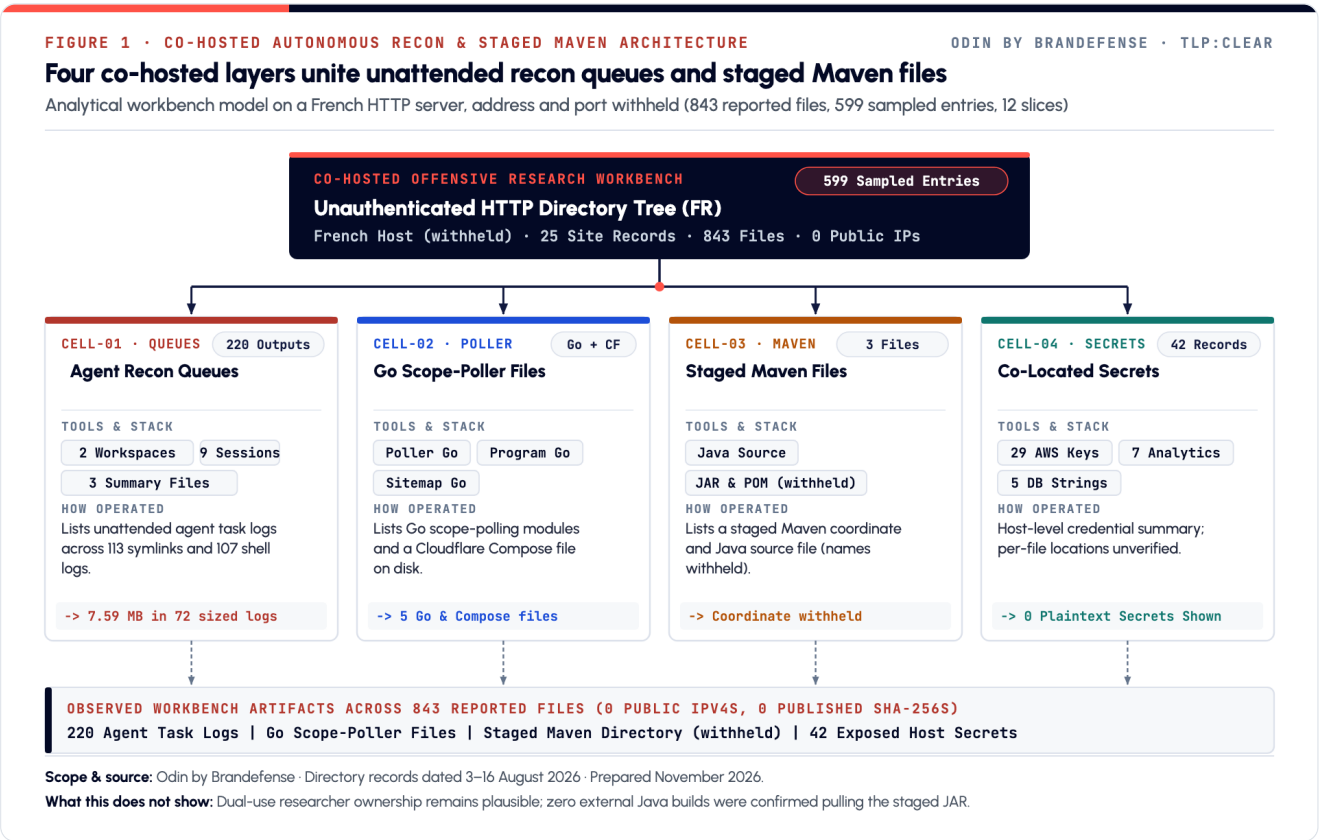


Figure 1: Four co-hosted layers unite unattended recon queues and staged Maven files

How the 843 reported files break down

Twenty-five directory index records across 12 directory slices on the server reported 843 total files. Removing duplicates across those listings left 599 sampled unique entries and 244 un-sampled reported paths that were never opened.

Within the 599 sampled entries, 220 itemized `.output` records sit inside nine anonymized session folders in the cloud-analytics workspace tree. The remaining 379 sampled entries cover a second agent workspace tree, three root coordination files, the Go scope-poller tree, and the staged Maven directory (project name and paths withheld). No third-party company or bug-bounty target is named anywhere in this report.

Dataset slice on exposed host	Count	Share or scope	What it represents
Reported exposed files	843	100.0% across 12 slices	Total files reported across 25 directory index listings
Sampled unique entries	599	71.1% of 843	Unique file paths across agent queues, Go scope poller, root, and Maven tree
Un-sampled directory entries	244	28.9% of 843	Sub-paths reported in directory counts but not opened
Itemized agent `.output` files	220	36.7% of 599	Task output logs across nine anonymized session folders in the analytics tree
Other sampled workbench entries	379	63.3% of 599	Second agent workspace, root status files, Go scope poller, and Maven files
Host credential findings	42	4 credential types	Exposed AWS, analytics workspace, and database credentials (values withheld)

Four operational layers on one filesystem

Sorting the 599 sampled paths by function reveals four distinct layers living side by side on the server. Their co-location supports a workbench model. The listings do not establish a transfer from scope polling to agent tasks or any downstream Java loading.

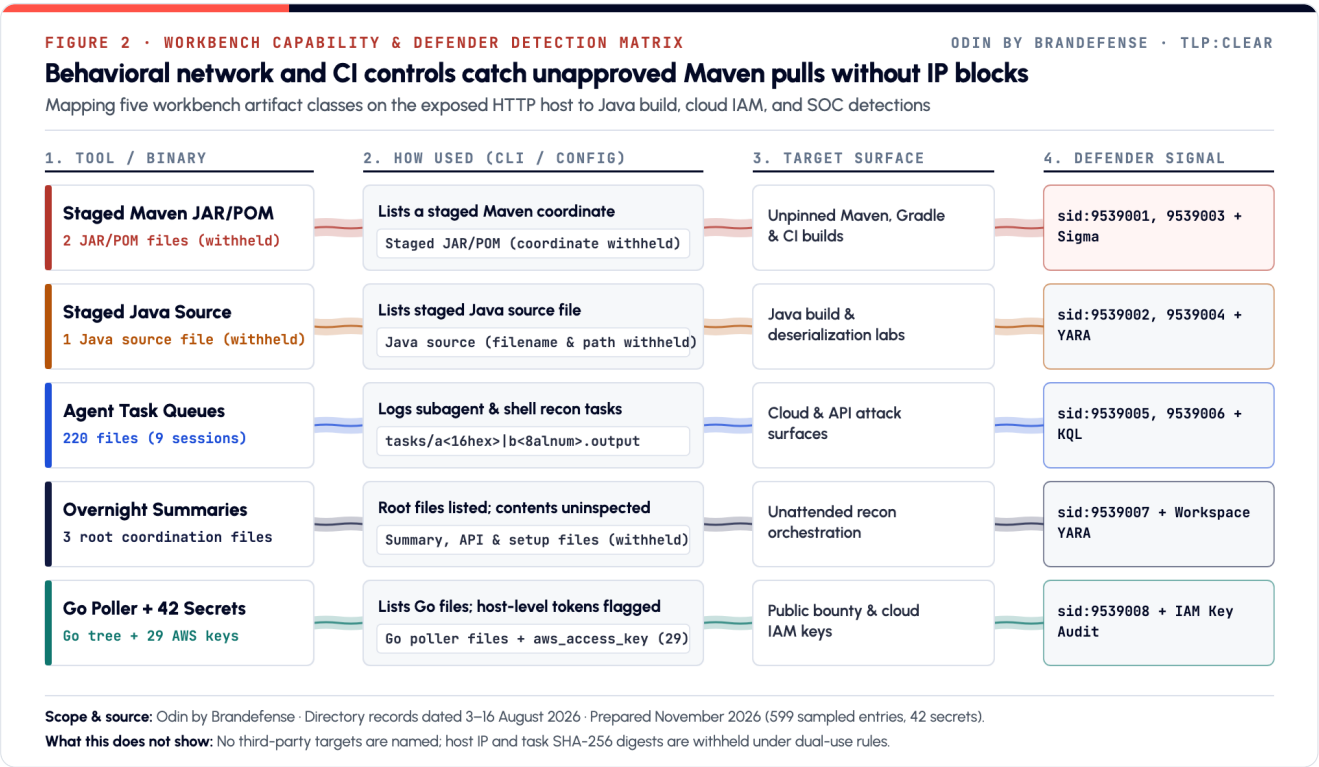


Figure 2: Behavioral network and CI controls catch unapproved Maven pulls without IP blocks

Workbench layer	Artifacts observed on host	Main count	Defensive focus
1. Agent recon queues	Two exposed agent workspace trees and three root summary files (paths withheld)	220 `.output` files in 9 sessions	Hunt unattended API enumeration and exposed agent logs
2. Go scope-polling stack	Go storage files, three web-service modules, and Cloudflare Compose file (project name and paths withheld)	5 core Go and Compose modules	Monitor unapproved scope pollers and open management ports
3. Staged Maven repository	A Java source file, a JAR, and a POM in a versioned Maven layout (coordinate, filenames, and paths withheld)	3 staged Java and Maven files	Pin builds to HTTPS proxies and verify unapproved coordinates
4. Exposed host secrets	Host-level credential summary across AWS IAM, analytics workspace, and database categories (per-file locations unverified)	42 secret findings	Rotate exposed IAM keys and restrict workspace token IPs

Because a single cloud VPS IP address changes quickly and may belong to a dual-use researcher, cyber threat intelligence can support review before blocking. Defenders get durable protection by verifying unapproved Maven coordinates, locking build tools to internal HTTPS repositories, and finding exposed workspaces through external attack surface management.

CHAPTER 03

Unattended Agent Recon Queues and Nine Session Folders

IN SHORT

Nine anonymized session folders hold 220 task `.output` logs beside an overnight summary file (exact host paths withheld). Every task file follows one of two agent naming patterns: 113 subagent symlinks and 107 background shell logs. Purge agent task transcripts and ban open HTTP listeners on research VMs.

INSIDE THIS CHAPTER

Two task-output file types: subagent symlinks and background shell logs 11

Overnight coordination files at the web root 12

Concentration across the nine anonymized session queues 12

03 Unattended Agent Recon Queues and Nine Session Folders

CLI coding agents store session state in predictable folder structures. When an operator starts an agent inside a project directory, the runtime flattens path slashes into hyphens and writes per-session UUID subdirectories. Serving the parent folder over an unauthenticated HTTP server turns every subagent transcript and background shell log into a public web page (T1552.001).

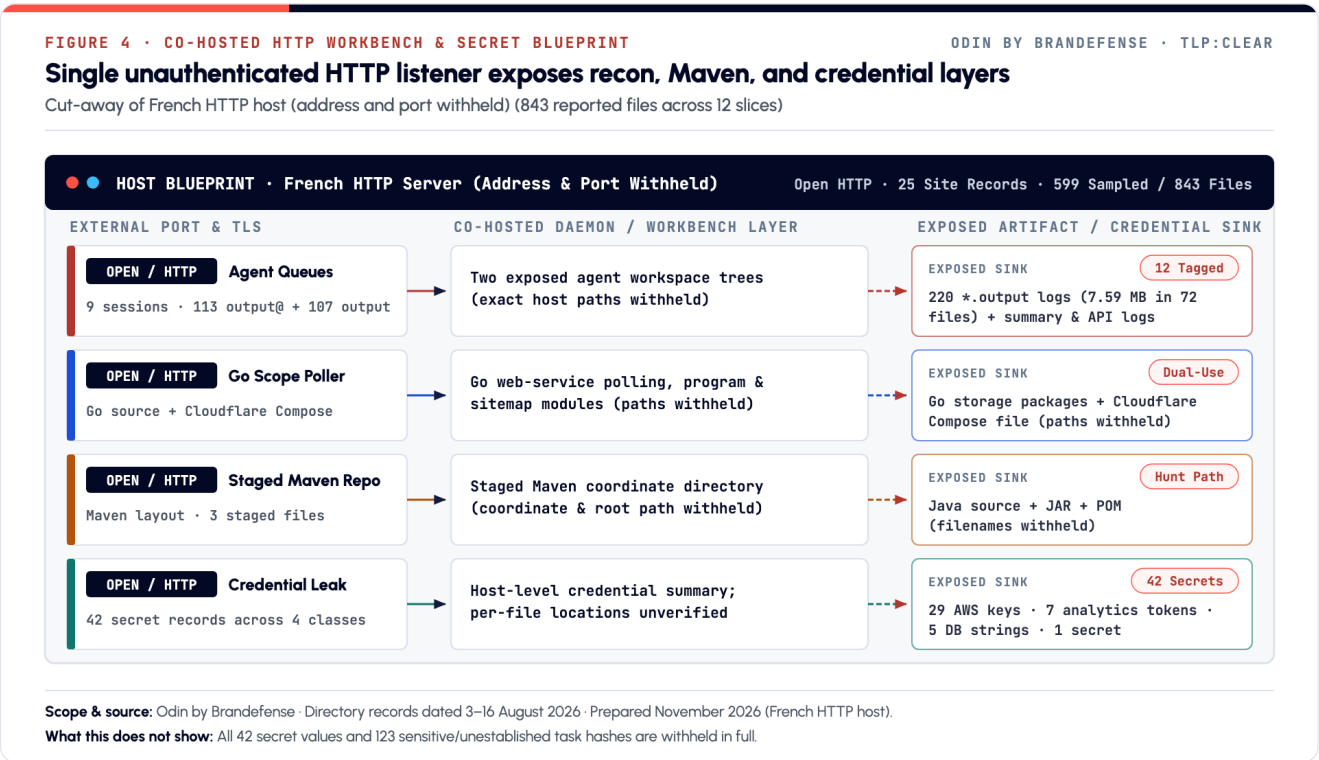


Figure 4: Single unauthenticated HTTP listener exposes recon, Maven, and credential layers

Two task-output file types: subagent symlinks and background shell logs

Every one of the 220 itemized files sits inside a per-session `tasks/` directory. Exact host paths and session UUIDs are withheld, and every file matches one of two naming patterns without exception.

Task file pattern	Count	Share of 220	Listing extension	Sized files and byte range	What creates it
`a<16hex>.output` (17-char ID)	113	51.4%	`output@` (POSIX symlink)	10 sized (91,392 to 254,783 B); 103 unsized	Delegated subagent runs linked to JSONL transcripts
`b<8alnum>.output` (9-char ID)	107	48.6%	`output` (regular file)	62 sized (33,043 to 261,889 B); 45 unsized	Background shell commands capturing stdout and stderr

The split between `output@` and `output` shows how the agent divides work. Reasoning tasks spawn subagents symlinked as `a<16hex>.output` (113 files), while background shell pipelines write to regular `b<8alnum>.output` files (107 files).

Across the 72 sized files (10 `output@` symlinks and 62 regular `output` files), log volume totals 7,585,435 bytes (7.59 MB). File sizes range from 33,043 to 261,889 bytes (median 88,382 bytes), while 148 entries remain unsized stubs.

Overnight coordination files at the web root

Three coordination files in the HTTP root sit beside the agent workspace directories (exact host filenames withheld):

- **Overnight summary file.** Listed as an overnight summary; contents uninspected.
- **Structured API response file.** Listed as an API response file; contents uninspected.
- **Setup log file.** Listed as a setup log; contents uninspected.

All 220 `.output` entries have an initial listing timestamp of 4 August 2026 (`07:45:35Z`). Their last listing timestamps fall on 15 August: 81 at `22:34:01Z` and 139 at `22:36:04Z`. These are directory listing timestamps, not execution times.

Concentration across the nine anonymized session queues

Activity is heavily concentrated. Three anonymized sessions hold 181 of 220 files (82.3%). One holds 4,808,890 bytes (63.4% of 7.59 MB).

The source inventory marks 12 `.output` files across 13 tag instances. Nine carry `cloud-credentials` tags (one in Session 3 also carries `exposed-secrets`), and three carry `exploit-code` tags. Hashes are withheld on 123 files, and 97 carry no hash.

Session rank and label	Files	Share	`output@` / `output`	Sized files (total bytes)	Content tags
1 · Session 1 (UUID withheld)	84	38.2%	84 / 0	0 (0 B)	1 `cloud-credentials`
2 · Session 2 (UUID withheld)	54	24.5%	16 / 38	0 (0 B)	None
3 · Session 3 (UUID withheld)	43	19.5%	9 / 34	40 (4,808,890 B)	4 `cloud-credentials`, 1 `exposed-secrets`
4 · Session 4 (UUID withheld)	11	5.0%	3 / 8	6 (881,632 B)	3 `cloud-credentials`, 2 `exploit-code`
5 · Session 5 (UUID withheld)	10	4.5%	0 / 10	10 (871,033 B)	1 `cloud-credentials`
6 · Session 6 (UUID withheld)	10	4.5%	1 / 9	9 (520,877 B)	1 `exploit-code`
7 · Session 7 (UUID withheld)	6	2.7%	0 / 6	6 (451,172 B)	None
8 · Session 8 (UUID withheld)	1	0.5%	0 / 1	1 (51,831 B)	None
9 · Session 9 (UUID withheld)	1	0.5%	0 / 1	0 (0 B)	None

CHAPTER 04

Go Scope-Polling Files and the Staged Maven Repository

IN SHORT

Beside the agent queues, a Go project directory lists scope-polling and Cloudflare Compose files (project name and paths withheld). A staged Maven directory lists a Java source file, a JAR, and a POM (coordinate, filenames, and host paths withheld). Nothing confirms any external build pulled the JAR, yet unpinned HTTP repositories remain a build-pipeline risk. Lock Java builds to internal HTTPS proxies.

INSIDE THIS CHAPTER

Target-scope polling files in the Go project directory 14

Inside the staged Maven directory layout 15

How Java builds and cloud clusters become exposed to external HTTP repositories 15

04 Go Scope-Polling Files and the Staged Maven Repository

Scope-polling files and a custom Maven directory sit side by side on the same server. The Go scope-polling project is built to track public attack surface. Alongside it, the Maven directory follows the layout that Java build tools and cloud clusters use for HTTP dependency lookups. Figure 3 maps the co-located artifact layers and defender controls.

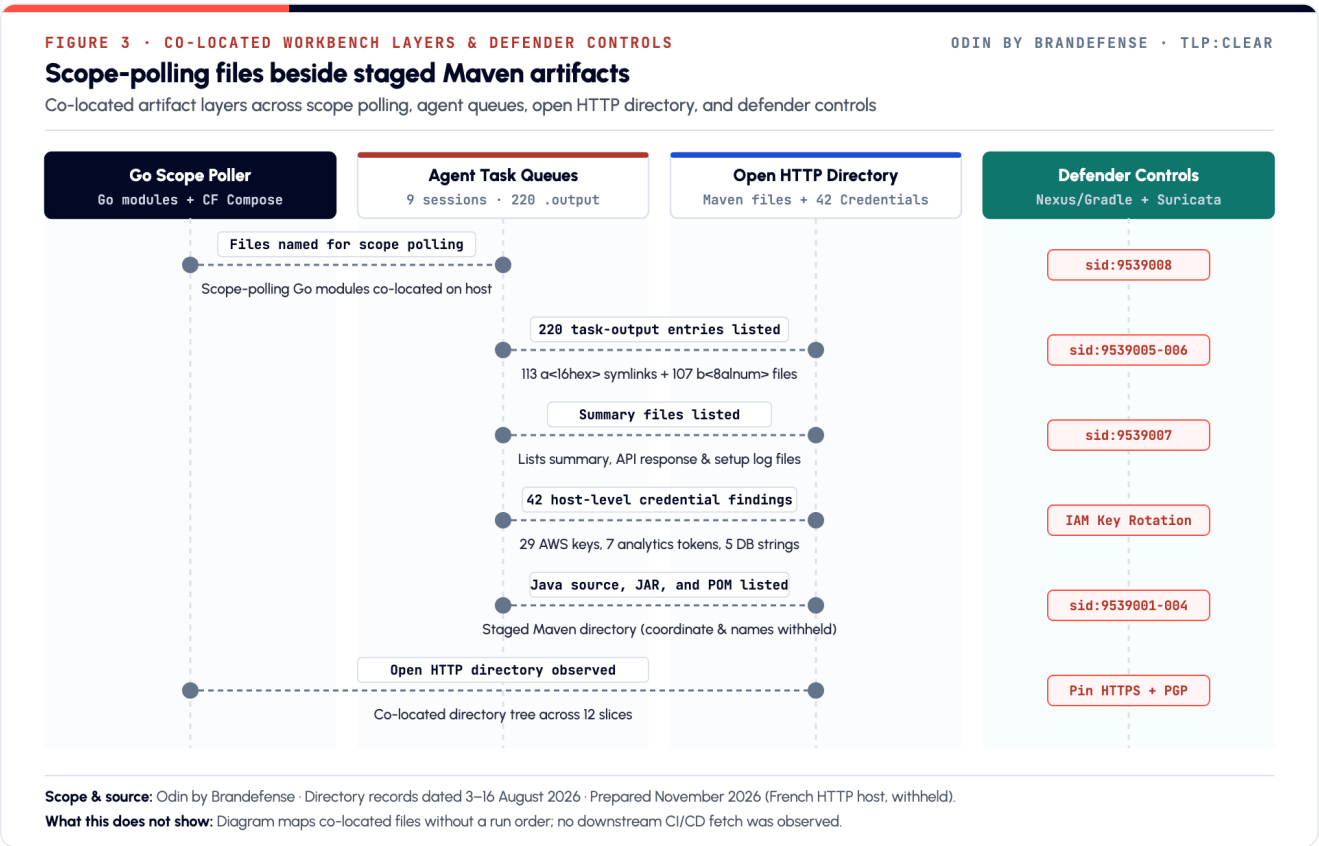


Figure 3: Scope-polling files beside staged Maven artifacts

Target-scope polling files in the Go project directory

The exposed directory lists Go storage files, three web-service modules, and a Compose configuration. Their exact host paths and project name are withheld.

The directory listing shows files named for polling, sitemap handling, storage, and Cloudflare Compose. The listings do not establish that those modules ran or that a tunnel was active.

Inside the staged Maven directory layout

Next to the Go project tree, the staged Maven directory follows the standard Apache Maven 2 repository layout (`/<groupId-path>/<artifactId>/<version>/<artifactId>-<version>.<ext>`). Its coordinate, filenames, and host paths are withheld. Maven and Gradle clients resolve a coordinate by converting dots in the group ID into slashes, then requesting the POM and JAR paths over HTTP or HTTPS (T1195.001).

Staged artifact on exposed host	Artifact role	How Java tools use it	Primary defender control
Staged Java source file (filename and root path withheld)	Staged Java source file	Listed at the repository root alongside the versioned coordinate directory	Use SID 9539002 as a hunt signal and scan with YARA
Staged Maven POM file (coordinate, filename, and path withheld)	Maven POM metadata file	Its listed path suggests a Maven coordinate; its contents and dependencies were not inspected	Force `<mirrorOf>*</mirrorOf>` in `settings.xml` to an internal HTTPS proxy
Staged Maven JAR file (coordinate, filename, and path withheld)	JAR file (contents uninspected)	Could be fetched into build-runner caches or cluster nodes if a client resolves its coordinate	Require SHA-256 and PGP verification in Gradle and Maven (SID 9539001)

We did not download or decompile the staged JAR, POM, or Java source file, and no offensive code is reproduced here. Nothing in the directory records shows whether any external build runner or target server ever requested the staged files.

How Java builds and cloud clusters become exposed to external HTTP repositories

Apache Maven 3.8.1 and newer include a built-in `maven-default-http-blocker` mirror rule that refuses cleartext HTTP repositories declared in `pom.xml` files. Even so, three common enterprise gaps reopen that door:

1. **Developer and CI overrides.** Teams can override `maven-default-http-blocker` in `settings.xml` or set `allowInsecureProtocol = true` in Gradle, while `-Dmaven.wagon.http.ssl.insecure=true` relaxes HTTPS certificate checks. Once HTTP is enabled, a `<repositories>` block can pull JARs from an external HTTP server.
2. **Cloud analytics cluster library installation.** Cloud analytics platforms often let token holders attach Maven coordinates from custom repository URLs onto compute clusters. Storing `analytics_workspace_token` secrets beside the staged Maven files co-locates cluster credentials with a JAR directory.
3. **SSRF and remote class-loading tests.** Researchers also stage custom Java source and JAR files to test server-side request forgery (SSRF), URLClassLoader sinks, or deserialization gadgets over plain HTTP.

Closing these gaps does not require the operator's IP address. When build runners and clusters talk only to `nexus.corp[.]internal` or `artifactory.corp[.]internal` over HTTPS, external HTTP Maven servers become unreachable.

05

CHAPTER 05

Forty-Two Exposed Credentials and Other Ways to Read the Evidence

IN SHORT

Passive secret checks across the exposed HTTP host recorded 42 credentials, led by 29 AWS access keys and seven analytics workspace tokens. Textbook Java test naming fits a security researcher, yet open HTTP exposure puts credentials in the served directory tree at risk. Rotate exposed IAM keys and bind workspace tokens to IP allowlists.

INSIDE THIS CHAPTER

Breakdown of the 42 credential findings on the host 18

Three ways to read this workbench 18

05 Forty-Two Exposed Credentials and Other Ways to Read the Evidence

Autonomous agents read credentials from environment files, pass tokens on command lines, and save verbose API replies into `.output` logs. An open HTTP listener can expose credentials inside its served directory tree (T1552.001). The host-level credential summary records 42 findings without identifying which specific files held each credential.

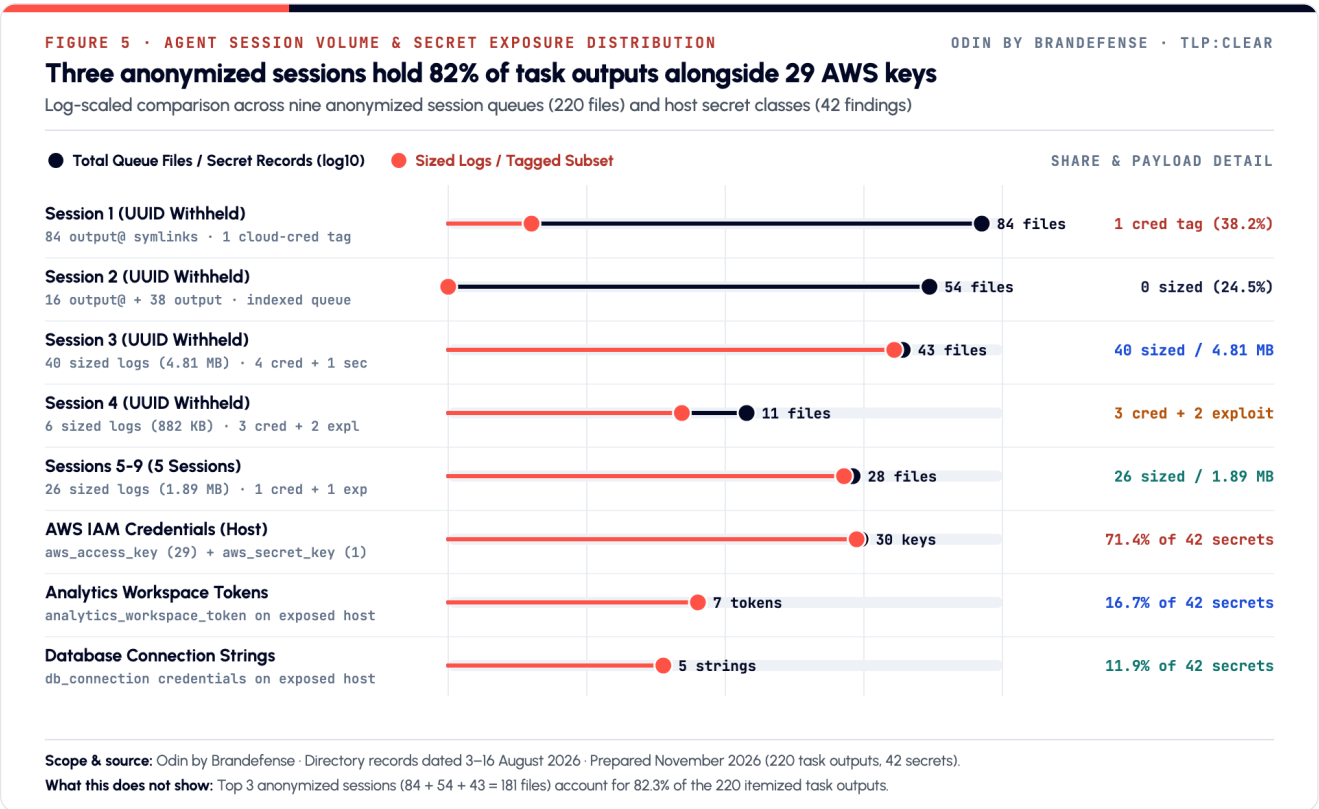


Figure 5: Three anonymized sessions hold 82% of task outputs alongside 29 AWS keys

Breakdown of the 42 credential findings on the host

Forty-two credential records were identified on the French HTTP workbench across four secret types. No secret values or partial key prefixes are published.

Secret or task-tag category	Count	Share	Where it sits	What defenders should do
`aws_access_key`	29	69.0% of 42	Host secret summary	Check key use in AWS CloudTrail, remove unused keys, and move to short-lived roles
`analytics_workspace_token`	7	16.7% of 42	Host secret summary	Replace long-lived personal tokens with short-lived OAuth tokens and IP allowlists
`db_connection`	5	11.9% of 42	Host secret summary	Restrict database ports to private networks and rotate passwords
`aws_secret_key`	1	2.4% of 42	Host secret summary	Revoke the paired AWS key immediately and review IAM Access Analyzer findings
`cloud-credentials` task logs	9	4.1% of 220	Sessions 3 (4), 4 (3), 1 (1), 5 (1)	Scrub agent `.output` logs and never serve agent session folders over HTTP
`exploit-code` task logs	3	1.4% of 220	Sessions 4 (2) and 6 (1)	Separate exploit reproduction labs from live cloud keys

AWS IAM credentials make up 30 of the 42 findings (71.4%: 29 access key IDs and one secret key). A standalone key ID reveals the AWS account via `sts:GetAccessKeyInfo`. The paired secret key allows direct API calls unless source-IP or MFA policies block it.

The seven `analytics_workspace_token` findings match the cloud-analytics workspace queue (220 `.output` files). Two of the nine `cloud-credentials` logs carry measured sizes (33,043 and 45,400 bytes; 78,443 bytes combined). Seventy untagged logs hold the remaining 7,506,992 measured bytes (10 `output@` symlinks and 60 regular `output` files). The other 138 untagged entries are unsized stubs (99 `output@` and 39 `output`).

Three ways to read this workbench

1. **An offensive security researcher or bug-bounty hunter.** Co-hosting an open-source scope poller with agent task queues and textbook Java test names fits a bug-bounty researcher testing custom Maven URLs. We therefore withhold the host IP, port, project name, exact host paths, Maven coordinate and filenames, session UUIDs, and all file hashes.
2. **Secondary compromise of exposed research infrastructure.** Even on a well-intentioned research machine, an open HTTP listener exposes served directory listings, task logs, and staged Maven files to internet-wide directory scrapers.
3. **Unattended agent sprawl inside enterprise teams.** Running `python3 -m http.server` on a cloud VM to download an overnight summary file can expose credentials and `.output` transcripts inside the served directory tree.

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

Java build and CI/CD teams

- [ ] Force every Maven and Gradle build through an internal HTTPS proxy (`nexus.corp[.]internal` or `artifactory.corp[.]internal`) via `<mirrorOf>*</mirrorOf>` in `settings.xml` and Gradle `init.gradle`. Allow upstream proxy fetches only from `repo.maven.apache[.]org` and `plugins.gradle[.]org` over HTTPS.
- [ ] Fail CI builds declaring `<repositories>` blocks in `pom.xml`, `allowInsecureProtocol = true` in Gradle, or `-Dmaven.wagon.http.ssl.insecure=true`.
- [ ] Enable Gradle dependency verification (SHA-256 and PGP). For Maven, use strict checksum checking (`-C`) and an approved HTTPS repository; verify context before blocking.

Cloud analytics and IAM teams

- [ ] Restrict cluster library attachment in cloud analytics workspaces to approved internal repositories, blocking external HTTP Maven URLs.
- [ ] Audit AWS IAM keys with IAM Access Analyzer, revoke stale user-managed keys, and adopt short-lived OIDC roles.
- [ ] Replace long-lived personal analytics tokens with short-lived OAuth tokens bound to corporate IP allowlists, and isolate database ports on private networks.

Security research, red-team, and SOC teams

- [ ] Ban unauthenticated HTTP file servers (`python3 -m http.server`) on developer and red-team VMs, and purge CLI agent `.output` logs after engagements.
- [ ] Deploy the Suricata, Sentinel KQL, Sigma, and YARA rules in Chapter 7 across CI/CD proxies, build runners, and cloud sandboxes.

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 22

How the KQL, Sigma, and YARA rules work 23

MITRE ATT&CK mapping 23

07 Detection Rules

The full rules are in Appendix B. Suricata formatting passed static linting (0 problems); live Suricata, Sentinel, Sigma, and YARA engines were not run here, and rules were not replayed on live traffic. Test and tune them before production blocking.

Rule file	What it catches
`odin_agent_queues_scope_poller_and_staged_maven.rules`	Suricata 7: eight signatures for versioned Maven coordinate JAR/POM requests, top-level `.java` source fetches, HTTP Maven/Gradle client pulls, open Maven, agent task, and Go/Compose directory listings, `.output` downloads, and recon summary Markdown requests
`sentinel_staged_maven_and_autonomous_recon_hunt.kql`	Microsoft Sentinel and Defender XDR: Maven, Gradle, or Java processes using HTTP repository overrides or TLS verification flags joined with artifact, task, and Go source URI network events
`sigma_unapproved_maven_pull_and_agent_recon_loop.yml`	Sigma (3 Linux rules): HTTP repository overrides or TLS verification flags; unattended agent task queues or open HTTP directory servers; and outbound Java, Maven, or Gradle connections outside approved artifact hosts
`odin_maven_and_agent_workspace_structural_hunt.yar`	YARA (4 rules): staged HTTP Maven POM descriptors or Java process-execution sources, insecure HTTP Maven/Gradle repository configs, exposed agent task directory listings, and open HTTP directory listings with Go source and Compose manifests

How the Suricata signatures work

SID	Looks for	Why
9539001	Request URI matching a versioned Maven 2 coordinate directory path for a `.jar` or `.pom`	Hunt signal: surfaces build tools, clusters, or SSRF sinks pulling unapproved Maven coordinates over HTTP
9539002	HTTP `GET` for a top-level `/*.java` source file	Hunt signal: surfaces retrieval of staged Java source files over HTTP
9539003	Outbound request for `.jar` or `.pom` with `Apache-Maven`, `Gradle`, or `Java/` User-Agent	Hunt rule: surfaces build clients pulling packages over cleartext HTTP
9539004	HTTP 200 body containing `Directory listing for /` and `.java`, `.jar`, or `.pom` entries	Flags an open HTTP directory listing staged Maven or Java artifacts
9539005	HTTP 200 body containing `Directory listing for /` and `tasks/` or `.output` filenames	Flags an exposed autonomous agent task directory
9539006	HTTP `GET` under `/tasks/` matching `a<16hex>.output` or `b<8alnum>.output`	Catches downloads of subagent or shell task transcripts
9539007	HTTP `GET` for recon or session summary `.md` files (`*_FINDINGS.md`, `recon_summary.md`, or `session_summary.md`)	Catches access to exposed reconnaissance summary reports
9539008	HTTP 200 body containing `Directory listing for /` and `.go` or `.yml` entries	Hunt rule: flags open HTTP directories listing Go scope-poller or Compose files

Use SIDs 9539001 and 9539002 as hunt signals. Enforce the approved-repository policy at egress, and verify context before blocking a coordinate or filename. Keep SIDs 9539003 and 9539008 as hunt rules to surface legacy HTTP Maven clients and open Go scope-poller directories.

How the KQL, Sigma, and YARA rules work

The Sentinel KQL query looks back 7 days across Linux build runners, analytics nodes, and workstations. It checks `DeviceProcessEvents` for `mvn`, `gradle`, or `java` using HTTP repository overrides, `allowInsecureProtocol`, or `maven.wagon.http.ssl.insecure=true`. It also checks command lines for `/tasks/a`, `/tasks/b`, `.output`, or `http.server`.

Those process events are inner-joined by process ID within five minutes to `DeviceNetworkEvents` matching artifact, task, or Go source URI patterns. Exact runner and repository hostnames sit in explicit allowlists (`ApprovedCicdRunners`, `ApprovedAsmRunners`, and `ApprovedArtifactHosts`) without wildcards.

The three Linux Sigma rules cover `auditd` and eBPF logs. Rule 1 flags Maven, Gradle, or Java command lines using HTTP repository overrides, `maven-default-http-blocker`, `allowInsecureProtocol`, or `maven.wagon.http.ssl.insecure=true`. It excludes exact runner hostnames and `/opt/nexus-sync/bin/mirror-verify`.

Rule 2 flags `/tasks/a`, `/tasks/b`, `.output`, or `http.server` outside `/opt/asm-poller/bin/approved-scope-sync`. Rule 3 flags outbound connections from `/usr/bin/java`, `/usr/bin/mvn`, `/usr/local/bin/mvn`, `/usr/bin/gradle`, or `/usr/local/bin/gradle` outside exact `DestinationHostname` allowlists.

YARA rule	Fires when
`ODIN_HTTP_Maven_POM_or_Java_Process_Execution_Source`	File under 256 KB has a `<project>` POM declaring `<groupId>`, `<artifactId>`, and `<url>http://`, or pairs a Java class header with `Runtime.getRuntime().exec` or `ProcessBuilder`
`ODIN_Insecure_Maven_or_Gradle_HTTP_Repo_Config`	File under 512 KB contains `http://` plus a `<repositories>` or `maven {` block alongside `allowInsecureProtocol`, `maven-default-http-blocker`, or `wagon.http.ssl.insecure`
`ODIN_Exposed_Agent_Overnight_Workspace`	File under 512 KB contains `Directory listing for /` plus `tasks/` and `.output@` or `.output`
`ODIN_Exposed_Go_Scope_Poller_or_Compose_Listing`	File under 512 KB contains `Directory listing for /` plus `.go` source and `.yml` or `.yaml` Compose entries

MITRE ATT&CK mapping

Tactic	Technique	How it appears on the workbench	Detection coverage
Initial Access	T1195.001 Compromise Software Dependencies and Development Tools	Staged Maven directory listing a Java source file, JAR, and POM over HTTP (coordinate and filenames withheld)	SIDs 9539001 to 9539004; KQL; Sigma rules 1 and 3; YARA rules 1 and 2
Reconnaissance	T1595.002 Active Scanning: Vulnerability Scanning	Unattended agent recon task queues and Go scope-polling files	SIDs 9539005, 9539007, 9539008; KQL; Sigma rule 2; YARA rules 3 and 4
Credential Access	T1552.001 Unsecured Credentials: Credentials In Files	42 cloud, analytics, and DB secrets alongside exposed `.output` transcripts	SID 9539006; KQL; Sigma rule 2

CHAPTER 08

Indicators

IN SHORT

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

INSIDE THIS CHAPTER

Staged Maven layout and artifact patterns (6 generic patterns, 1 withheld root path; coordinate and filenames withheld) 25

Agent queue globs, scope-poller CLI, build flags, and withheld host exposures (6 published patterns, 14 withheld paths) 26

08 Indicators

Appendix A holds 0 published indicators: 0 network addresses and 0 file hashes. Both are withheld because the single French HTTP host may belong to a security researcher and its task logs hold sensitive tokens. Twelve generic Maven layout, task-glob, CLI, and build-configuration patterns appear in the tables below. Exact Maven coordinates, filenames, project name, port, and 15 host paths from the 27 v1.0 items are withheld.

Indicator category	Published in tab	Published patterns below	Withheld host paths	Where to use it
Network addresses (IPv4)	0	0	1 (host IP)	Not published under dual-use rules
File hashes (SHA-256)	0	0	123 (97 unhashed)	Withheld (123 sensitive/ unestablished, 97 unhashed)
Staged Maven layout and artifact patterns	0	6	1 (repo root path; coordinate and filenames withheld)	Nexus/Artifactory firewalls, egress proxies, and build caches
Agent queue globs and root exposure types	0	3	9 (workspace/root paths)	Perimeter HTTP hunting and internal sandbox audits
Scope-poller CLI pattern and Go exposure types	0	1	5 (Go/Compose paths)	Endpoint process monitoring and perimeter audits
Insecure Maven/Gradle config and CLI flags	0	2	0	CI/CD SAST linters and build-runner logs

Staged Maven layout and artifact patterns (6 generic patterns, 1 withheld root path; coordinate and filenames withheld)

Pattern	Type	Recommended action
`<groupId>:<artifactId>:<version>` (coordinate withheld)	Maven coordinate pattern	Enforce approved-repository policy; verify context before blocking
`<groupId>.<artifactId>` (namespace withheld)	Java package namespace pattern	Review unapproved external namespaces in repository firewalls
`/<groupId-path>/<artifactId>/<version>/ <artifactId>-<version>.jar` (URI withheld)	Maven JAR URI path pattern	Use as a hunt signal at egress proxies (SID 9539001)
`/<groupId-path>/<artifactId>/<version>/ <artifactId>-<version>.pom` (URI withheld)	Maven POM URI path pattern	Use as a hunt signal at egress proxies (SID 9539001)
`/<ClassName>.java` (filename withheld)	Staged Java source URI pattern	Use as a hunt signal at egress proxies (SID 9539002) and scan with YARA
`<artifactId>-<version>.jar` (filename withheld)	Cached JAR filename pattern	Hunt unverified HTTP JARs inside `~/.m2/ repository/` and `~/.gradle/caches/`
Staged Maven repository root (path withheld)	Exposure type	Block external HTTP Maven repositories at egress proxies

Agent queue globs, scope-poller CLI, build flags, and withheld host exposures (6 published patterns, 14 withheld paths)

Pattern	Type	Recommended action
`tasks/a[0-9a-f]{16}.output`	Subagent symlink glob	Audit CLI agent session logs for cleartext secrets
`tasks/b[a-z0-9]{8}.output`	Shell task log glob	Audit CLI agent session logs for cleartext secrets
`Directory listing for /`	HTTP response string	Alert on open `http.server` directory listings
`<scope-poller> poll` (project name withheld)	Scope-poller CLI pattern	Watchlist outside approved attack-surface hosts
`allowInsecureProtocol = true`	Gradle DSL flag	Fail in CI `build.gradle` checks
`-Dmaven.wagon.http.ssl.insecure=true`	Maven CLI flag	Alert on relaxed HTTPS certificate checks in build-runner logs
Agent recon workspace trees (2 paths withheld)	Exposure type	Audit internal cloud VMs and open HTTP servers
Agent session task path globs (2 paths withheld)	Exposure type	Audit internal agent sandboxes
Overnight summary file (path withheld)	Exposure type	Audit unauthenticated agent directories
API response file (path withheld)	Exposure type	Audit unauthenticated agent directories
Setup log file (path withheld)	Exposure type	Audit unauthenticated agent directories
Open directory listing banners (2 paths withheld)	Exposure type	Alert on open agent and Maven directories
Go scope-poller and Compose files (5 paths withheld)	Exposure type	Audit unapproved scope-polling hosts

Withheld from this public edition are the French server's IPv4 address, port, project name, 15 exact host paths, staged Maven coordinate and filenames, and nine session UUIDs. All 42 raw credential values, all third-party target names, and all file hashes on the host 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

- **Dataset timestamps.** Analysis covers directory listing records dated 3 to 16 August 2026 and prepared in November 2026. Those dates are directory listing timestamps recorded on the server, not task execution times.
- **Staged files, not downstream execution.** We inspected directory listings and metadata only. We ran no binaries or Java classes and replayed no credentials. Nothing in the listings confirms that any external CI pipeline or analytics cluster fetched the staged JAR.
- **Dual-use researcher ownership is plausible.** Open-source scope-polling files and textbook Java test filenames fit an offensive security researcher or bug-bounty hunter whose HTTP server was left open. The host IP, port, project name, exact host paths, Maven coordinate and filenames, session UUIDs, and task hashes are withheld.
- **Workbench model.** Grouping the host into four layers is an analytical model of co-hosted files on one server, not attribution to a threat group.
- **Rule validation.** Published Suricata signatures passed static syntax linting (0 problems), and KQL, Sigma, and YARA rules were checked structurally against test fixtures. Live engines were not run, and the rules were not tested on production traffic.

10

CHAPTER 10

Sources

IN SHORT

This chapter lists the public references used in the report.

10 Sources

1. Apache Maven Project. Introduction to Repositories. https://maven.apache.org/guides/introduction/introduction-to-repositories.html (verified 21 November 2026)
2. Apache Maven Project. Settings Reference (Mirrors and HTTP Blockers). https://maven.apache.org/settings.html (verified 21 November 2026)
3. Gradle Inc. Verifying Dependencies (SHA-256 and PGP Signatures). https://docs.gradle.org/current/userguide/dependency_verification.html (verified 21 November 2026)
4. Amazon Web Services. IAM Best Practices: Manage Access Keys. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html (verified 21 November 2026)
5. OWASP Cheat Sheet Series. CI/CD Security Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/CI_CD_Security_Cheat_Sheet.html (verified 21 November 2026)
6. MITRE ATT&CK. T1195.001 Supply Chain Compromise: Compromise Software Dependencies and Development Tools. https://attack.mitre.org/techniques/T1195/001/ (verified 21 November 2026)
7. MITRE ATT&CK. T1595.002 Active Scanning: Vulnerability Scanning. https://attack.mitre.org/techniques/T1595/002/ (verified 21 November 2026)
8. MITRE ATT&CK. T1552.001 Unsecured Credentials: Credentials In Files. https://attack.mitre.org/techniques/T1552/001/ (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) 32

B. Detection rules (4 files) 33

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 (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_agent_queues_scope_poller_and_staged_maven.rules

SURICATA

```
# Odin by Brandefense - Monograph 39 / 46 Suricata 7.x Detection Suite
# Coverage: Versioned Maven coordinate JAR/POM retrieval, top-level Java source retrieval,
#           cleartext HTTP Maven/Gradle fetches, exposed agent task output listings,
#           recon summary Markdown requests, and Go scope-poller or Compose directory listings
# Classification: TLP:CLEAR

alert http any any -> any any (msg:"ODIN [R39] HUNT Staged Maven Coordinate Layout JAR or POM Retrieval";
flow:established,to_server; http.method; content:"GET"; http.uri; bsize:>12; pcre:"/^\/[a-z0-9_]+\/[a-z0-9_.-]+\/
[a-z0-9_.-]+\/[0-9][0-9a-z_.-]*\/[a-z0-9_.-]+-[0-9][0-9a-z_.-]*\.(jar|pom)$/i"; classtype:policy-violation; sid:
9539001; rev:2; metadata:created_at 2026_11_21, deployment perimeter_and_proxy, policy behavioral_hunt;)

alert http any any -> any any (msg:"ODIN [R39] HUNT Staged Top-Level Java Source File Retrieval (*.java)";
flow:established,to_server; http.method; content:"GET"; http.uri; bsize:>5; content:".java"; fast_pattern;
pcre:"/^\/[A-Za-z_][A-Za-z0-9_]{1,63}\.java$/"; classtype:policy-violation; sid:9539002; rev:2;
metadata:created_at 2026_11_21, deployment perimeter_and_proxy, policy behavioral_hunt;)

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN [R39] HUNT Cleartext HTTP Maven or Gradle Client
Artifact Fetch"; flow:established,to_server; http.uri; bsize:>4; pcre:"/\.(jar|pom)$/i"; http.user_agent;
bsize:>5; content:"/"; fast_pattern; pcre:"/(Apache-Maven|Gradle|Java\/)/i"; classtype:policy-violation; sid:
9539003; rev:1; metadata:created_at 2026_11_21, deployment egress_proxy, policy behavioral_hunt;)

alert http any any -> any any (msg:"ODIN [R39] Open-Directory Staged Maven or Java Artifact Listing";
flow:established,from_server; http.stat_code; content:"200"; http.response_body; bsize:>22; content:"Directory
listing for /"; fast_pattern; pcre:"/[A-Za-z0-9_.-]+\.(java|jar|pom)/i"; classtype:attempted-recon; sid:9539004;
rev:2; metadata:created_at 2026_11_21, deployment perimeter, tactic resource_development;)

alert http any any -> any any (msg:"ODIN [R39] Open-Directory Autonomous Agent Task Queue Listing";
flow:established,from_server; http.stat_code; content:"200"; http.response_body; bsize:>22; content:"Directory
listing for /"; fast_pattern; pcre:"/(tasks\/|(a[0-9a-f]{16}|b[a-z0-9]{8})\.output)/"; classtype:attempted-recon;
sid:9539005; rev:1; metadata:created_at 2026_11_21, deployment perimeter, tactic reconnaissance;)

alert http any any -> any any (msg:"ODIN [R39] Autonomous Agent Task Output Transcript Retrieval (/tasks/
*.output)"; flow:established,to_server; http.method; content:"GET"; http.uri; bsize:>23; content:"/tasks/";
fast_pattern; pcre:"/\/tasks\/(a[0-9a-f]{16}|b[a-z0-9]{8})\.output$/"; classtype:policy-violation; sid:9539006;
rev:1; metadata:created_at 2026_11_21, deployment perimeter, tactic collection;)

alert http any any -> any any (msg:"ODIN [R39] Open-Directory Agent Session or Recon Summary Markdown Retrieval";
flow:established,to_server; http.method; content:"GET"; http.uri; bsize:>4; content:".md"; fast_pattern; pcre:"/
\/([A-Z0-9_]{6,40}_FINDINGS|recon_summary|session_summary)\.md$/i"; classtype:attempted-recon; sid:9539007; rev:
1; metadata:created_at 2026_11_21, deployment perimeter, tactic reconnaissance;)

alert http any any -> any any (msg:"ODIN [R39] HUNT Open-Directory Go Scope-Poller or Compose Manifest Listing";
flow:established,from_server; http.stat_code; content:"200"; http.response_body; bsize:>22; content:"Directory
listing for /"; fast_pattern; pcre:"/[A-Za-z0-9_.-]+\.(go|ya?ml)/i"; classtype:policy-violation; sid:9539008;
rev:2; metadata:created_at 2026_11_21, deployment perimeter, policy behavioral_hunt;)
```

odin_maven_and_agent_workspace_structural_hunt.yar

YARA

```
/*
   Odin by Brandefense - Monograph 39 / 46 YARA Structural Detection Suite
   Targets staged HTTP Maven POM descriptors or Java process-execution sources, insecure HTTP Maven/Gradle
   repository configs, exposed agent task directory listings, and Go scope-poller or Compose directory listings.
   Classification: TLP:CLEAR
*/

rule ODIN_HTTP_Maven_POM_or_Java_Process_Execution_Source {
```

```
    meta:
        author = "Odin by Brandefense"
        monograph = "39 / 46"
        description = "Detects staged Maven POM descriptors with cleartext HTTP repositories or standalone Java
source files invoking process execution sinks"
        date = "2026-11-21"
        tlp = "CLEAR"
    strings:
        $pom_proj = "<project" ascii
        $pom_grp = "<groupId>" ascii
        $pom_art = "<artifactId>" ascii
        $pom_http = "<url>http://" ascii
        $java_pkg = "package " ascii
        $java_cls = "public class " ascii
        $exec1 = "Runtime.getRuntime().exec" ascii
        $exec2 = "ProcessBuilder" ascii
    condition:
        filesize < 256KB and (
            ($pom_proj and $pom_grp and $pom_art and $pom_http) or
            (($java_pkg or $java_cls) and 1 of ($exec1, $exec2))
        )
}

rule ODIN_Insecure_Maven_or_Gradle_HTTP_Repo_Config {
    meta:
        author = "Odin by Brandefense"
        monograph = "39 / 46"
        description = "Detects Maven POM/settings or Gradle build files pointing to cleartext HTTP repository
endpoints, overriding HTTP repository blockers, or relaxing HTTPS certificate checks"
        date = "2026-11-21"
        tlp = "CLEAR"
    strings:
        $repo_tag = "<repositories>" ascii
        $grad_repo = "maven {" ascii
        $http_url = "http://" ascii
        $bypass1 = "allowInsecureProtocol" ascii
        $bypass2 = "maven-default-http-blocker" ascii
        $bypass3 = "wagon.http.ssl.insecure" ascii
    condition:
        filesize < 512KB and $http_url and 1 of ($repo_tag, $grad_repo) and 1 of ($bypass1, $bypass2, $bypass3)
}

rule ODIN_Exposed_Agent_Overnight_Workspace {
    meta:
        author = "Odin by Brandefense"
        monograph = "39 / 46"
        description = "Detects exposed autonomous agent task directory listings containing symlinked or shell
task output entries"
        date = "2026-11-21"
        tlp = "CLEAR"
    strings:
        $listing = "Directory listing for /" ascii
        $tasks = "tasks/" ascii
        $out_sym = ".output@" ascii
        $out_reg = ".output" ascii
    condition:
        filesize < 512KB and $listing and $tasks and 1 of ($out_sym, $out_reg)
}

rule ODIN_Exposed_Go_Scope_Poller_or_Compose_Listing {
    meta:
        author = "Odin by Brandefense"
        monograph = "39 / 46"
        description = "Detects open HTTP directory listings exposing Go scope-poller source files or Compose
manifests"
        date = "2026-11-21"
        tlp = "CLEAR"
    strings:
        $listing = "Directory listing for /" ascii
        $go_ext = ".go" ascii
```

```
        $yml_ext1 = ".yml" ascii
        $yml_ext2 = ".yaml" ascii
    condition:
        filesize < 512KB and $listing and $go_ext and 1 of ($yml_ext1, $yml_ext2)
}
```

sentinel_staged_maven_and_autonomous_recon_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Monograph 39 / 46 Microsoft Sentinel & Defender XDR Hunt
// Correlates unapproved Maven/Gradle cleartext repository overrides or TLS verification flags,
// unapproved HTTP artifact fetches, unattended agent recon loops, and open HTTP directory servers
// Classification: TLP:CLEAR

let Lookback = 7d;
let ApprovedCicdRunners = dynamic(["JAVA-CICD-RUNNER-01", "MAVEN-RELEASE-LNX-01"]);
let ApprovedAsmRunners = dynamic(["ASM-RECON-LNX-01", "SEC-VALIDATION-LNX-01"]);
let ApprovedArtifactHosts = dynamic(["repo.maven.apache.org", "plugins.gradle.org", "nexus.corp.internal",
"artifactory.corp.internal"]);
let BuildAndJavaBinaries = dynamic(["mvn", "mvnw", "gradle", "gradlew", "java"]);
let SuspiciousBuildAndReconProcs = DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedCicdRunners)) and not(DeviceName in~ (ApprovedAsmRunners))
    | where (FileName in~ (BuildAndJavaBinaries) and ProcessCommandLine has_any ("http://", "maven-default-http-
blocker", "maven.wagon.http.ssl.insecure=true", "allowInsecureProtocol"))
        or ProcessCommandLine has_any ("/tasks/a", "/tasks/b", ".output", "http.server")
    | project ProcTime = Timestamp, DeviceId, DeviceName, ProcessId, InitiatingProcessFileName, FileName,
ProcessCommandLine;
DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedCicdRunners)) and not(DeviceName in~ (ApprovedAsmRunners))
    | where not(RemoteUrl in~ (ApprovedArtifactHosts))
    | where (InitiatingProcessFileName in~ (BuildAndJavaBinaries) and RemoteUrl has_any (".jar", ".pom",
".java"))
        or RemoteUrl has_any ("/tasks/", ".output", ".go")
    | summarize
        NetFirstSeen = min(Timestamp),
        NetLastSeen = max(Timestamp),
        DistinctRemoteUrls = dcount(RemoteUrl),
        SampleRemoteUrls = make_set(RemoteUrl, 12),
        ConnectionCount = count()
        by DeviceId, DeviceName, InitiatingProcessId, InitiatingProcessFileName, RemotePort
    | join kind=inner (SuspiciousBuildAndReconProcs) on DeviceId, DeviceName, $left.InitiatingProcessId ==
$right.ProcessId
    | where NetFirstSeen >= ProcTime and NetFirstSeen <= ProcTime + 5m
    | extend AlertDisposition = case(
        SampleRemoteUrls has_any (".jar", ".pom", ".java"), "HUNT_UNAPPROVED_HTTP_MAVEN_ARTIFACT_PULL",
        ProcessCommandLine has_any ("/tasks/a", "/tasks/b", ".output", "http.server") or SampleRemoteUrls has_any
("/tasks/", ".output"), "HUNT_EXPOSED_AUTONOMOUS_RECON_WORKBENCH",
        "HUNT_UNAPPROVED_HTTP_MAVEN_OR_SCOPE_POLLER_ACTIVITY"
    )
    | project NetFirstSeen, NetLastSeen, ProcTime, DeviceName, AlertDisposition, InitiatingProcessId,
InitiatingProcessFileName, FileName, RemotePort, ProcessCommandLine, SampleRemoteUrls, ConnectionCount
    | sort by NetLastSeen desc
```

sigma_unapproved_maven_pull_and_agent_recon_loop.yml

SIGMA

```
title: Unapproved HTTP Maven Repository or Insecure TLS Flag Execution
id: 6d9b1e39-4a12-4c88-9f31-953900000001
status: experimental
description: Detects Maven, Gradle, or Java executions referencing cleartext HTTP repository overrides
(allowInsecureProtocol or maven-default-http-blocker) or Maven HTTPS certificate verification flags
(maven.wagon.http.ssl.insecure=true) outside approved build runners.
author: Odin by Brandefense
date: 2026-11-21
```

```
  tags:
    - attack.initial-access
    - attack.t1195.001
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_binary:
      Image:
        - '/usr/bin/mvn'
        - '/usr/local/bin/mvn'
        - '/usr/bin/gradle'
        - '/usr/local/bin/gradle'
        - '/usr/bin/java'
    selection_flags:
      CommandLine|contains:
        - 'http://'
        - 'maven.repo.remote=http://'
        - 'maven-default-http-blocker'
        - 'maven.wagon.http.ssl.insecure=true'
        - 'allowInsecureProtocol'
    filter_cicd_hosts:
      ComputerName:
        - 'JAVA-CICD-RUNNER-01'
        - 'MAVEN-RELEASE-LNX-01'
    filter_sync_tool:
      Image:
        - '/opt/nexus-sync/bin/mirror-verify'
    condition: selection_binary and selection_flags and not 1 of filter_*
  falsepositives:
    - Offline sandbox builds mirroring legacy internal HTTP repositories on isolated lab runners
  level: high
  ---
  title: Unattended Agent Recon Queue and Open Directory Server Execution
  id: 6d9b1e39-4a12-4c88-9f31-953900000002
  status: experimental
  description: Detects unattended autonomous recon loops referencing agent task output queues or unauthenticated
  HTTP directory servers.
  author: Odin by Brandefense
  date: 2026-11-21
  tags:
    - attack.reconnaissance
    - attack.t1595.002
    - attack.credential-access
    - attack.t1552.001
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_recon_loop:
      CommandLine|contains:
        - '/tasks/a'
        - '/tasks/b'
        - '.output'
        - 'http.server'
    filter_asm_runners:
      Image:
        - '/opt/asm-poller/bin/approved-scope-sync'
    condition: selection_recon_loop and not 1 of filter_*
  falsepositives:
    - Approved external attack-surface management hosts running documented scope polling binaries
  level: medium
  ---
  title: Java or Build Runner Outbound Connection Outside Approved Artifact Hosts
  id: 6d9b1e39-4a12-4c88-9f31-953900000003
  status: experimental
  description: Detects Java, Maven, or Gradle processes initiating outbound network connections outside approved
  artifact repository hosts.
  author: Odin by Brandefense
  date: 2026-11-21
  tags:
    - attack.initial-access
```

```
    - attack.t1195.001
  logsource:
    category: network_connection
    product: linux
  detection:
    selection_proc:
      Initiated: 'true'
      Image:
        - '/usr/bin/java'
        - '/usr/bin/mvn'
        - '/usr/local/bin/mvn'
        - '/usr/bin/gradle'
        - '/usr/local/bin/gradle'
    filter_repo_hosts:
      DestinationHostname:
        - 'nexus.corp.internal'
        - 'artifactory.corp.internal'
        - 'repo.maven.apache.org'
        - 'plugins.gradle.org'
    condition: selection_proc and not 1 of filter_*
  falsepositives:
    - Internal Java services connecting to documented application backends outside build runners
  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/exposed-recon-queues-and-maven-files/
Short Read · brandefense.io/odin-research/blog/when-unattended-recon-agents-share-a-server-with-maven-files/
Defense Kit · brandefense.io/odin-research/defense-kit/