ORBITUS ROBOTICS / FIELD GUIDE

ROB VIPER

Security Operations Handbook



Malware Triage, CVE Analysis, and macOS Hardening

Rodolfo Aramayo

First edition / November 2026

Illustrated workflows, harmless exercises and organizational practice

Contents

Use the chapter links to navigate. The EPUB edition reflows and uses chapter titles rather than fixed page numbers.

01 / How to use this handbook 4

02 / Choose the right security question 6

03 / Prepare permissions and prove collection 8

04 / Inspect events without losing the record 10

05 / Rehearse execution rules before enforcement 12

06 / Investigate network evidence and constrain access 15

07 / Prepare Threat Center and its tools 20

08 / Malware concepts that change the investigation 22

09 / Open a case and preserve useful evidence 24

10 / Run ClamAV and interpret a signature result 26

11 / Coverage gaps, archives and false positives 29

12 / Write, review and test a YARA-X rule 32

13 / Inspect macOS artifacts without launching them 34

14 / Correlate behavior signals with events 36

15 / Use controlled execution only in a prepared lab 38

16 / What a CVE scan actually tells you 40

17 / Scan dependency manifests with Trivy 42

18 / A reproducible CVE analysis exercise 44

19 / Prioritize with context, CVSS, EPSS and KEV 46

20 / Remediate and prove the deployed change 48

21 / Use SBOMs and CI without hiding failures 50

22 / Build an organizational analysis workflow 52

23 / Plan an MDM pilot and central administration 54

24 / Schedule work at the right time 56

25 / Respond to a suspicious result 58

26 / Exercise the program with harmless controls 60

27 / Read posture and understand FileVault 62

28 / Export, retain and communicate evidence 64

29 / Case templates, checklists and daily reference 66

30 / Glossary, sources and edition evidence 68

I / OPERATE WITH EVIDENCE

01 / How to use this handbook

This book expands the illustrated ROB Viper user manual into an operating guide for a small security team, a software organization, and an individual Mac owner. It teaches a repeatable habit: ask a specific question, collect the right evidence, record the limits, and make a reviewed decision. A scanner result is an observation about a particular target at a particular time. It becomes useful when somebody owns the next action.

The screenshots show ROB Viper 0.7.0, build 12, a development preview captured in November 2026. The book is freely priced; the software has separate distribution requirements. Check the product page for the current installer and release status. Do not infer that a public, notarized installer or a deployed company service exists merely because a workflow is described here. [V1]

Three reading routes

- New operator: read Chapters 2-7, complete the harmless exercises, then practice reporting a finding before changing a policy.
- Analyst or developer: concentrate on malware triage, ClamAV, YARA-X, CVE analysis, remediation and the CI handoff. Keep the scope and coverage notes beside every result.
- Administrator or security lead: use the organizational model, fleet rollout, scheduling, incident playbook and acceptance criteria to design an approved pilot.

What the examples establish

The illustrated scans use deliberately harmless material: plain text, the standard EICAR test signature, a custom YARA marker, a metadata-only dependency lockfile and a file that exceeds a scan limit. Network records are retained disposable-client tests. The posture screenshot is an evaluation-machine example, not an organizational baseline. No live malware was downloaded or executed to prepare this book.

Instructions marked as a proposed organizational practice describe work your organization must implement and approve. They are not hidden features of Viper. The app does not automatically isolate a host, quarantine a file, deploy a patch, upload a report, create a ticket, or centrally schedule every enrolled Mac. A human decision and a separately controlled delivery mechanism remain necessary.

Edition and rights

First edition, 1.0, November 2026. Copyright 2026 OrbitusRobotics LLC. All rights reserved. Free pricing does not change copyright or third-party software licenses. Apple, ClamAV, Trivy and YARA-X are identified to explain interoperability; no endorsement is implied.

Use the companion exercises

The handbook page provides a small exercise bundle with a fixture generator, a report-review script and case templates. Read the scripts before running them. They create files only in a new directory you choose; they neither install dependencies nor change security policy. The PDF has fixed page numbers. The EPUB reflows, so use chapter titles and its table of contents when teaching from either edition. [B1]

I / OPERATE WITH EVIDENCE

02 / Choose the right security question

A single dashboard can contain several very different kinds of evidence. An execution event says that macOS reported a process action. A network record describes a flow and the decision made for it. A malware scan compares content with signatures. A vulnerability scan compares detected package versions with advisory data. A posture check reads selected settings. Confusing these layers creates false reassurance and unnecessary blocking.

Question	Use	Do not infer
What ran, and under which code identity?	Endpoint Security event inspection	That every argument, script or file content was inspected.
What destination did an app contact?	Network Control history	That a hostname or IP proves the remote content is safe.
Does this file match known signatures?	ClamAV	That a match-free file cannot be malicious.
Does this artifact match our reviewed pattern?	YARA-X	That every rule match is an infection.
Which known vulnerabilities affect detected dependencies?	Trivy	That every installed Mac app or the whole OS was inventoried.
Are three selected system protections enabled?	Mac security posture	That the machine passed a complete security audit.

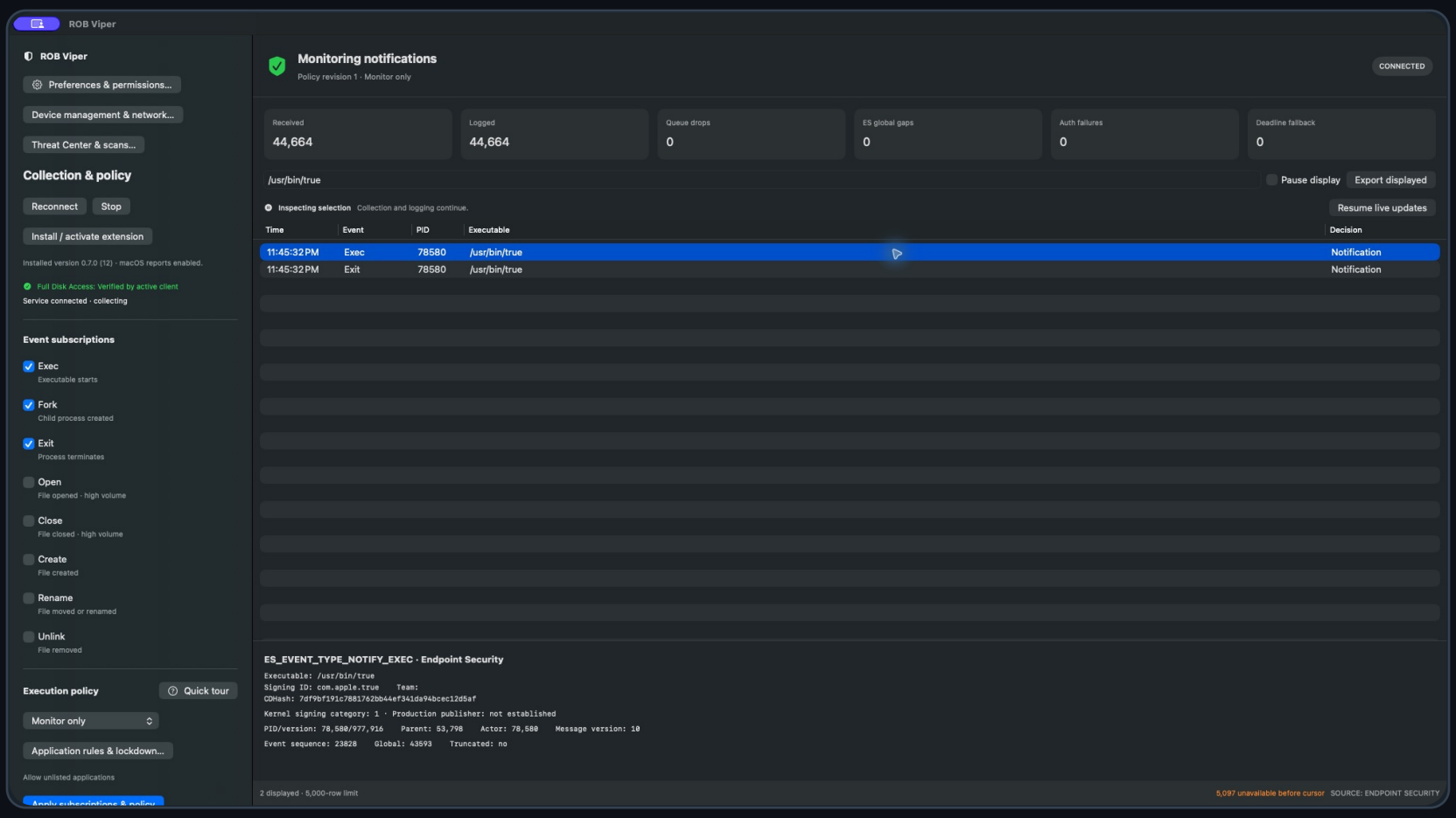


Figure 1. The main monitor controls collection and execution policy. Network Control and Threat Center are separate windows.

Keep four states separate

Installed means software is present. Approved means macOS permits an extension or permission. Active means the service is actually operating. Effective means a representative test demonstrated the intended behavior. Record each state independently. A green extension approval indicator cannot prove that filtering is running; a saved draft cannot prove that a policy reached the service.

Decide what would answer your question before starting a scan. For a downloaded installer, the first question might be whether the exact downloaded bytes match a known malicious signature. For a release candidate, it might be whether a supported lockfile contains known vulnerable packages. For a suspected compromise, it may be necessary to preserve events before any scan changes local activity or consumes the limited report history.

> **An evidence sentence**
>
> A useful conclusion names the target, tool, database or rule revision, time, outcome and limitations: "ClamAV completed a signature scan of this downloaded file at this time; no signature was reported within the configured limits." Avoid replacing that sentence with "the Mac is safe."

I / OPERATE WITH EVIDENCE

03 / Prepare permissions and prove collection

Begin with a signed build supplied through your approved development or distribution process. The host app and its system extensions have different responsibilities. Endpoint Security needs the appropriate entitlement and operating-system approval; the extension also needs Full Disk Access for the intended collection. Installing scanner engines is a separate operation and does not activate the endpoint or network extensions.

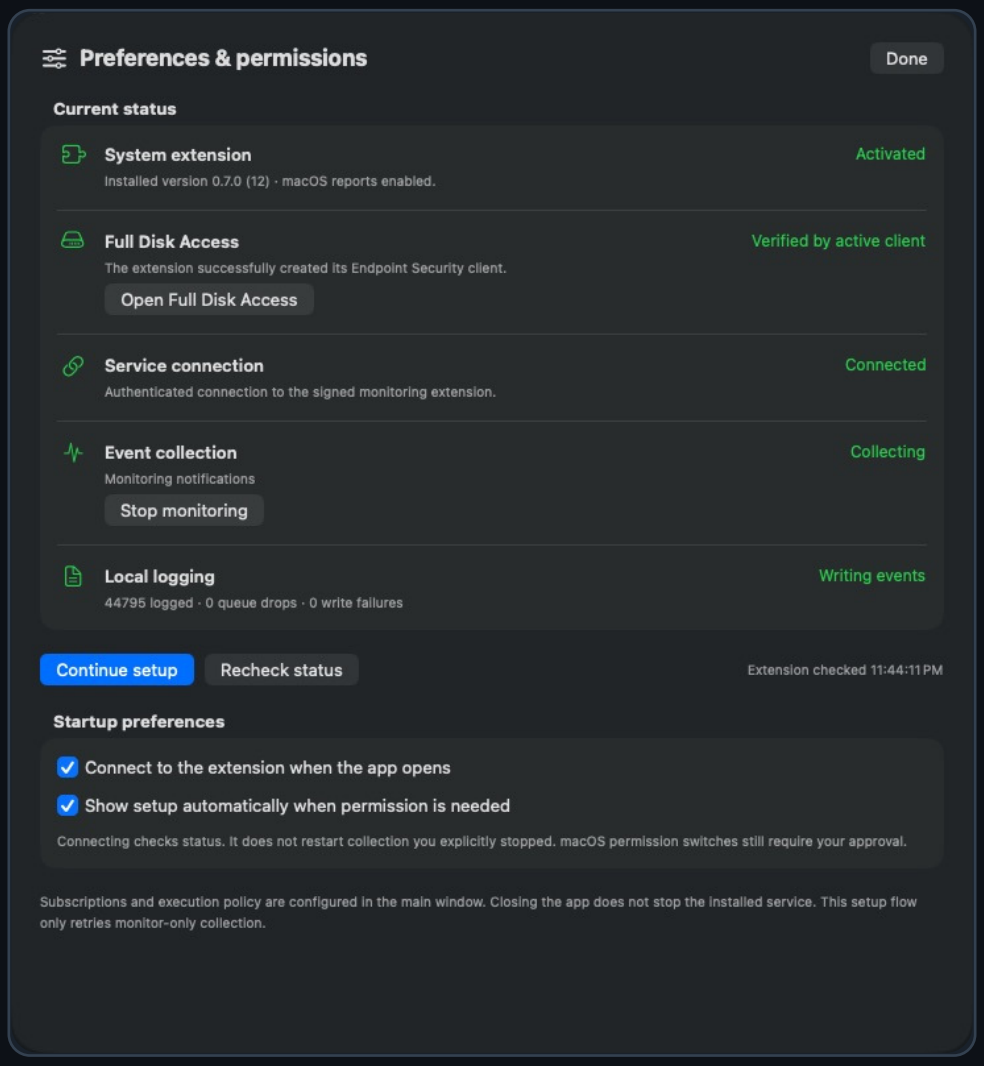


Figure 2. The setup panel shows separate permission, connection, collection and logging states.

1. Open Preferences & permissions from the main window. Read the named extension and the exact status that needs attention.
2. Use Continue setup and the provided macOS links. Complete operating-system approval or authentication through its normal interface.
3. Return to Viper and choose Recheck status. Confirm the active client, service connection, collection and logging indicators separately.
4. Choose Monitor only. Select Exec, Fork and Exit as a modest starting scope, then Apply subscriptions & policy.
5. Run /usr/bin/true from Terminal. Filter for that path and inspect a new event. Confirm the timestamp is fresh and the active subscriptions include Exec.

Harmless collection check

```
/usr/bin/true
```

A collection test should use an event you intentionally generated. A long table full of old rows proves that history exists, but does not establish current collection. If the event is missing, check the filter and held-display state before assuming the extension is broken. Then examine Full Disk Access, active subscriptions, service status and loss counters.

Select subscriptions for a purpose

Exec is useful for launch investigation. Create and Rename support the current behavior signals. More file events can create much more data, and a busy machine may exceed bounded queues. Begin with the smallest useful set and measure the losses under a normal workload. An absent event is weak evidence when the required subscription was not active or the service reported dropped data.

The host supports macOS 13 and later; production-publisher trust and Lockdown enforcement in this preview require macOS 26 signing-category support. Scanner tools have independent platform requirements. In a managed organization, the authoritative MDM policy can lock local controls. Do not treat a disabled button as an invitation to bypass management; ask the policy owner to publish the intended revision.

Record the baseline

Before a pilot, record host and extension versions, macOS build, active mode, subscriptions, policy revision, permissions, and a successful harmless test. Keep the baseline with the device inventory rather than relying on a screenshot alone.

I / OPERATE WITH EVIDENCE

04 / Inspect events without losing the record

Live tables are designed to move. Investigation requires a stable record. Click the event row text once and check that the interface indicates inspection is held. Collection and service logging continue while the selected event and inspector remain available. Use Resume live updates when ready to follow new events again.

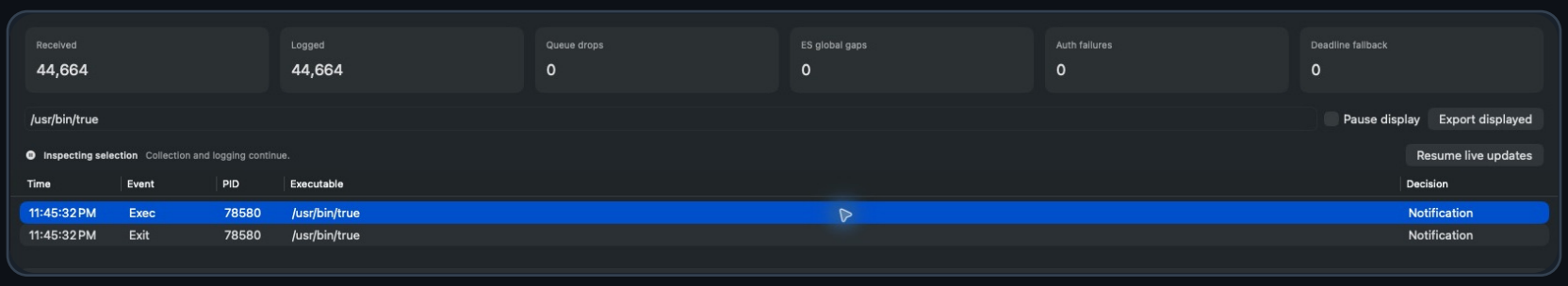


Figure 3. Detail crop: filter and selected Exec row. Holding the display does not stop the service.

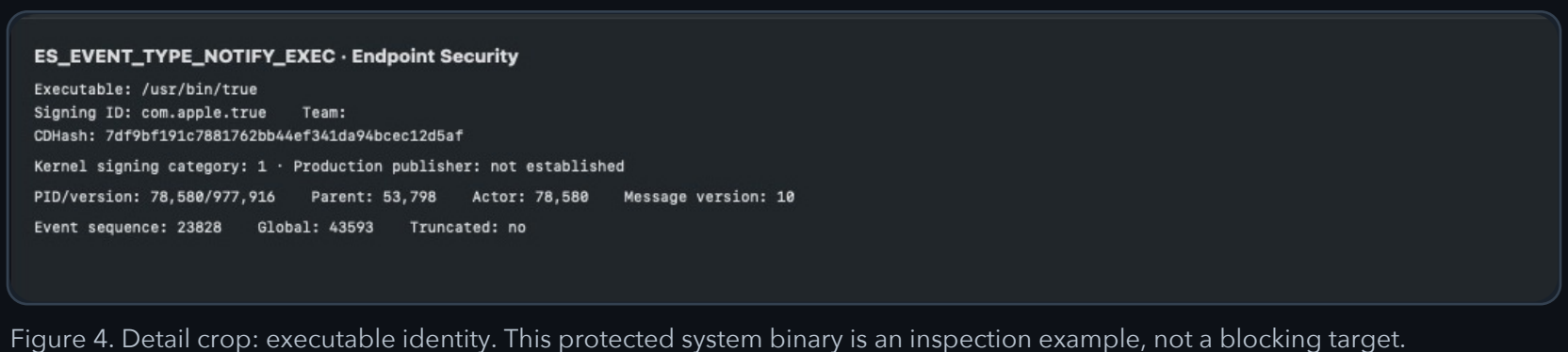


Figure 4. Detail crop: executable identity. This protected system binary is an inspection example, not a blocking target.

Read the fields as evidence

The executable path describes where a program was found. A familiar filename or application icon is not a trusted identity. Signing ID identifies a signed product; Team ID identifies a signing team when the signature and signing category establish that trust. CDHash identifies a CodeDirectory for a particular build and architecture. It is not the same value as a whole-file SHA-256.

For file events, distinguish the process executable from the object path that was created, renamed or otherwise involved. A process touching a suspicious path is different from a process whose own executable came from that path. Preserve the event type, timestamp, process identity and relevant object fields together. When a PID is reused, a new process generation must not inherit the earlier process history.

Build a small timeline

1. Write the analyst question and a time window in the case record.
2. Identify the original event and record its code identity before following related leads.
3. Compare nearby events and behavior signals. Note required subscriptions that were missing.
4. Correlate network observations using the actual executable identity and time, not just a display name.
5. Export useful records before bounded retention or an app restart removes them.

The current logging excludes command arguments and environment values. This protects privacy but also limits conclusions. If an approved incident process needs command lines, memory or a complete process tree, use an appropriate separately authorized collection tool. Do not fill missing fields with assumptions. A process can be signed and still behave badly; an ad hoc developer build can be harmless.

> **A useful negative result**
>
> Write "no matching event in the retained, subscribed window" rather than "this never happened." Include queue drops, expired events, collection gaps and the time range in the same note.

I / OPERATE WITH EVIDENCE

05 / Rehearse execution rules before enforcement

Monitor only collects selected notifications without asking to authorize execution. Audit execution rules evaluates a policy and records what it would do while allowing the launch. Enforce execution rules can deny a new launch. The distinction matters during both training and an incident: a Would deny event is not a blocked process, and an existing process is not terminated when a new rule is applied.

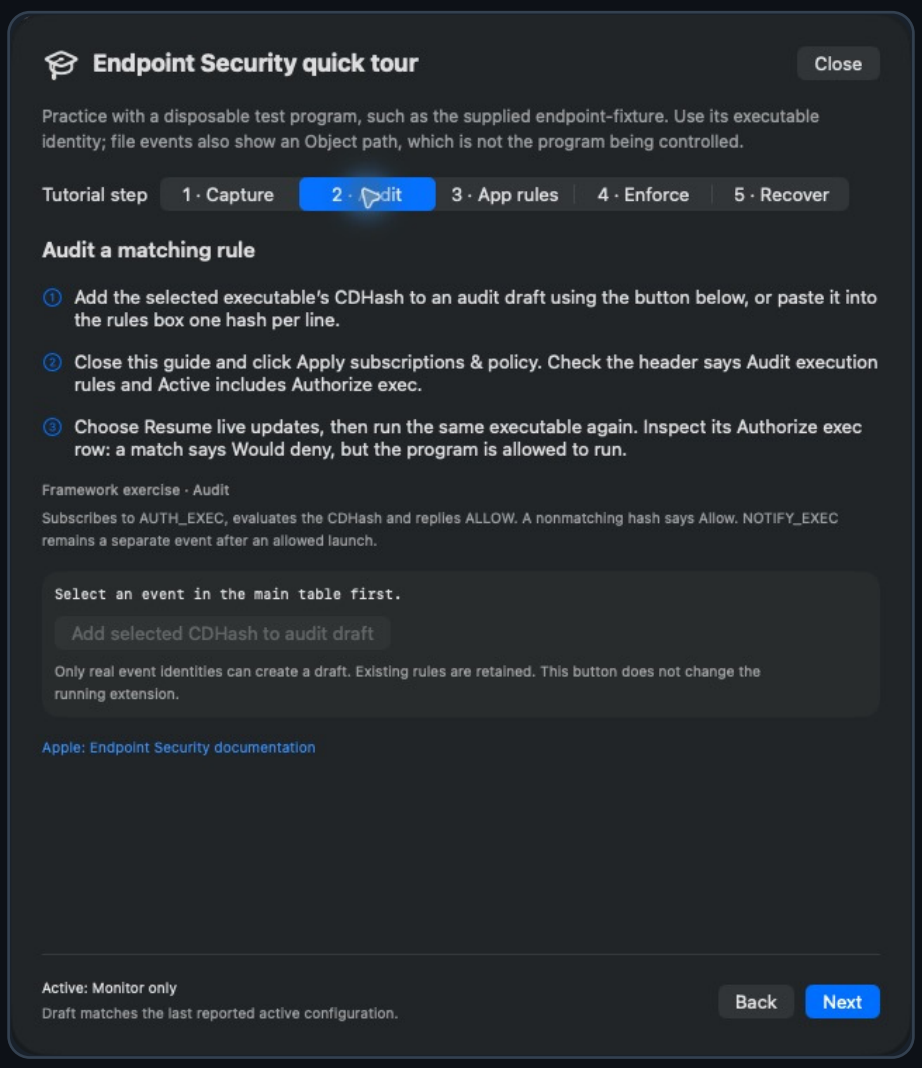


Figure 5. The built-in quick tour explains the Audit rehearsal. The pictured draft button is disabled because no suitable test event is selected.

1. Use a disposable executable that you own. Keep Unlisted applications = Allow for the first test.
2. Run it in Monitor, select its actual Exec event, and open Quick tour > Audit.
3. Add the selected CDHash to the Audit draft. Apply the draft and verify the service reports Audit and Authorize exec.
4. Run the same build again. Verify both the Would deny event and successful completion of the program.
5. Return to Monitor only and Apply when the rehearsal is complete. Remove test rules from the draft and apply the intended final configuration.

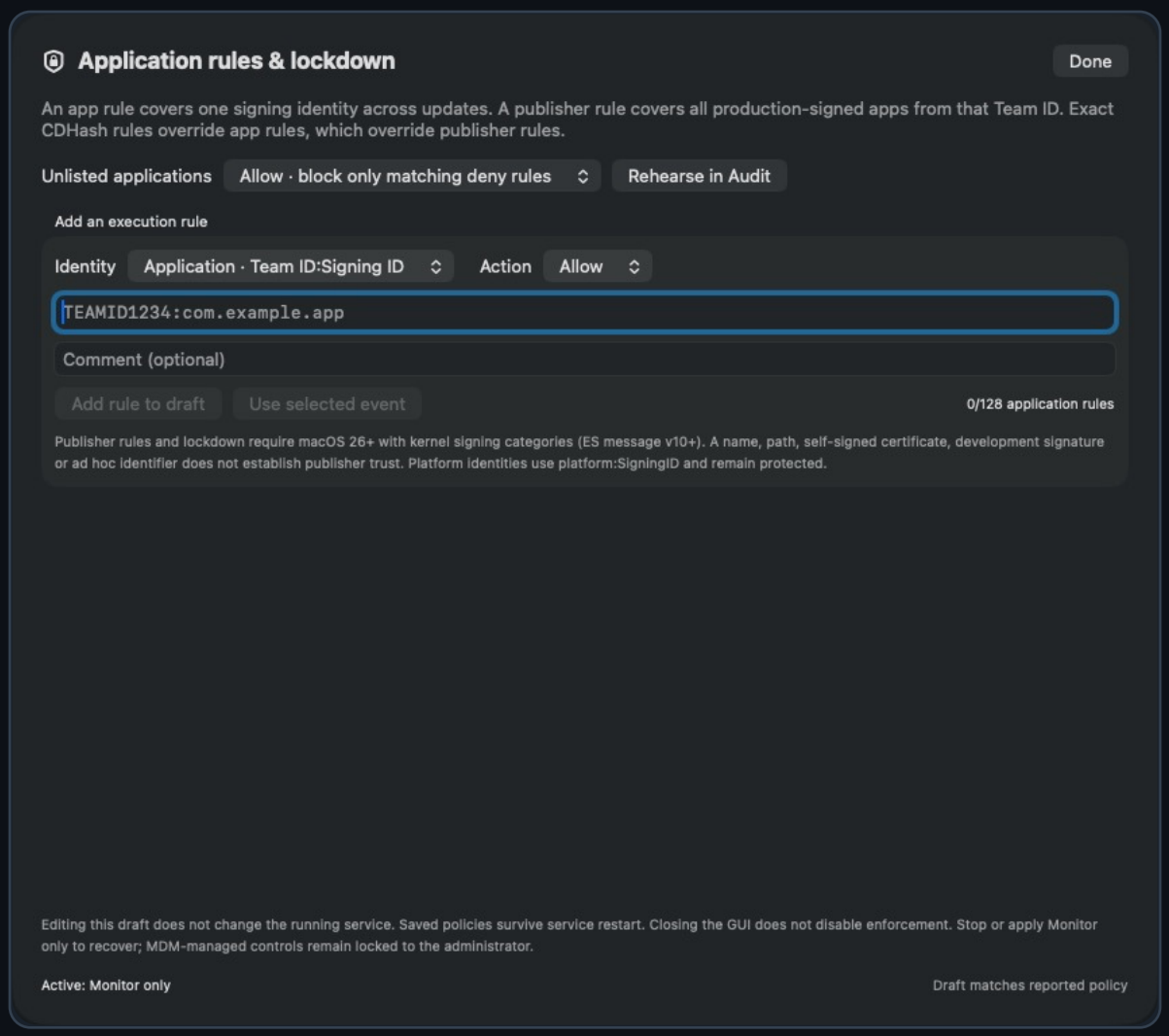


Figure 6. Application rules and Lockdown. Editing prepares a draft; Apply changes the service policy.

Choose the narrowest durable identity

An exact 40-character hexadecimal CDHash targets one code build. A Team ID plus Signing ID can describe one trusted production-signed application across updates. A publisher Team ID can cover many applications from that publisher. Ad hoc and development builds need exact-build handling in this preview. Broader identity rules reduce maintenance but widen the consequences of trusting a compromised or unwanted product.

The current precedence is protected platform and recovery identities, then exact CDHash, application identity, publisher, and the unlisted default. The combined execution-rule budget is 128. A default Allow policy denies selected identities. A default Deny policy is application Lockdown and needs a much broader Audit rehearsal: login, updates, installers, recovery tools and everyday work all matter.

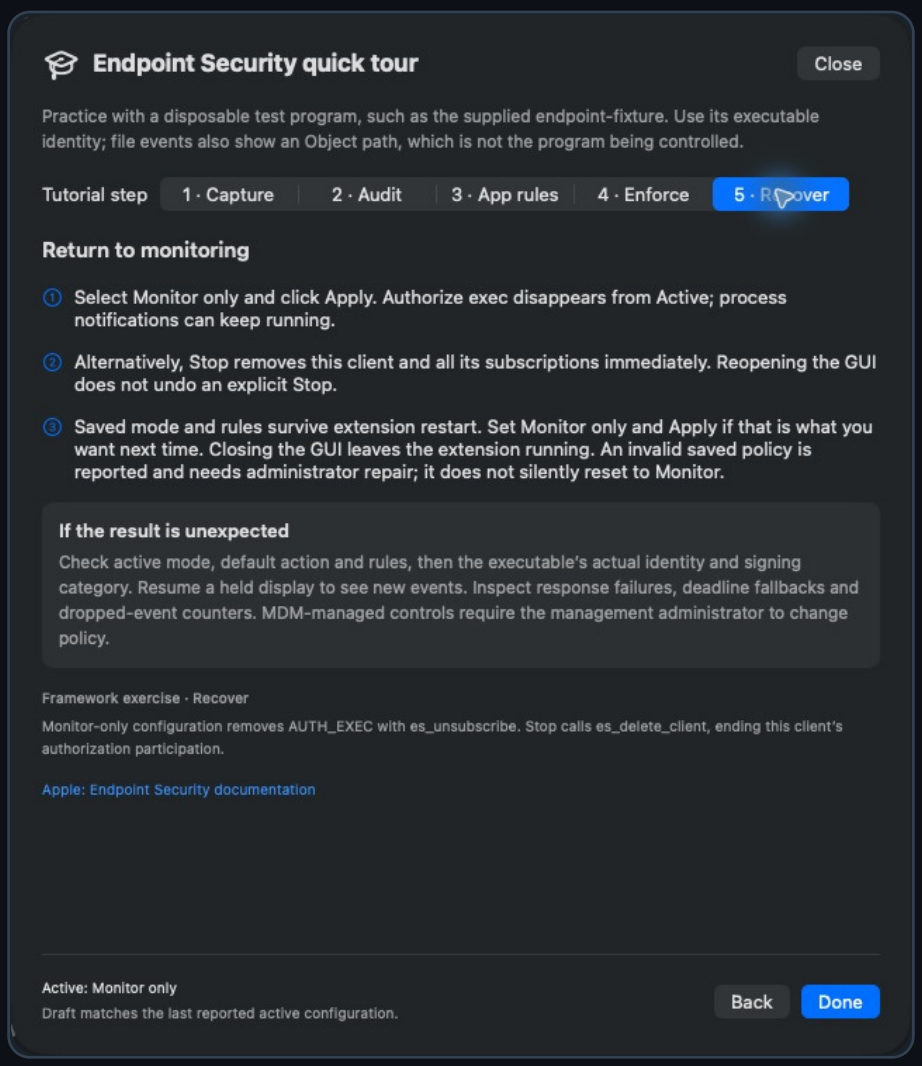


Figure 7. The recovery tour should be read before any blocking test.

Before Enforce, keep a tested recovery route and verify the expected identity under Audit. Apply only a reviewed, scoped change and confirm both the actual denial and the failed launch. Closing the host is not a reliable rollback. Stop removes the ES client, but saved policy can return when a new extension process starts. Publish or apply an allowing policy first when that is the intended recovery state. MDM authority may require the administrator to do this.

> **Scripts and trusted interpreters**
>
> Allowing an interpreter does not mean that Viper inspected or approved every script it can run. Application control, script/content analysis, update management and least privilege address different parts of the problem. Do not equate a publisher allow rule with a malware verdict.

I / OPERATE WITH EVIDENCE

06 / Investigate network evidence and constrain access

Open Network Control with Command-Shift-N. Start in Setup & privacy and read approval and filter state separately. Approved + OFF means macOS permits the extension while traffic filtering is stopped. To rehearse, choose Audit - allow all, review the history/privacy choices, Save & enable, then confirm a running provider and fresh connection evidence. The screenshots in this edition were taken with filtering OFF.

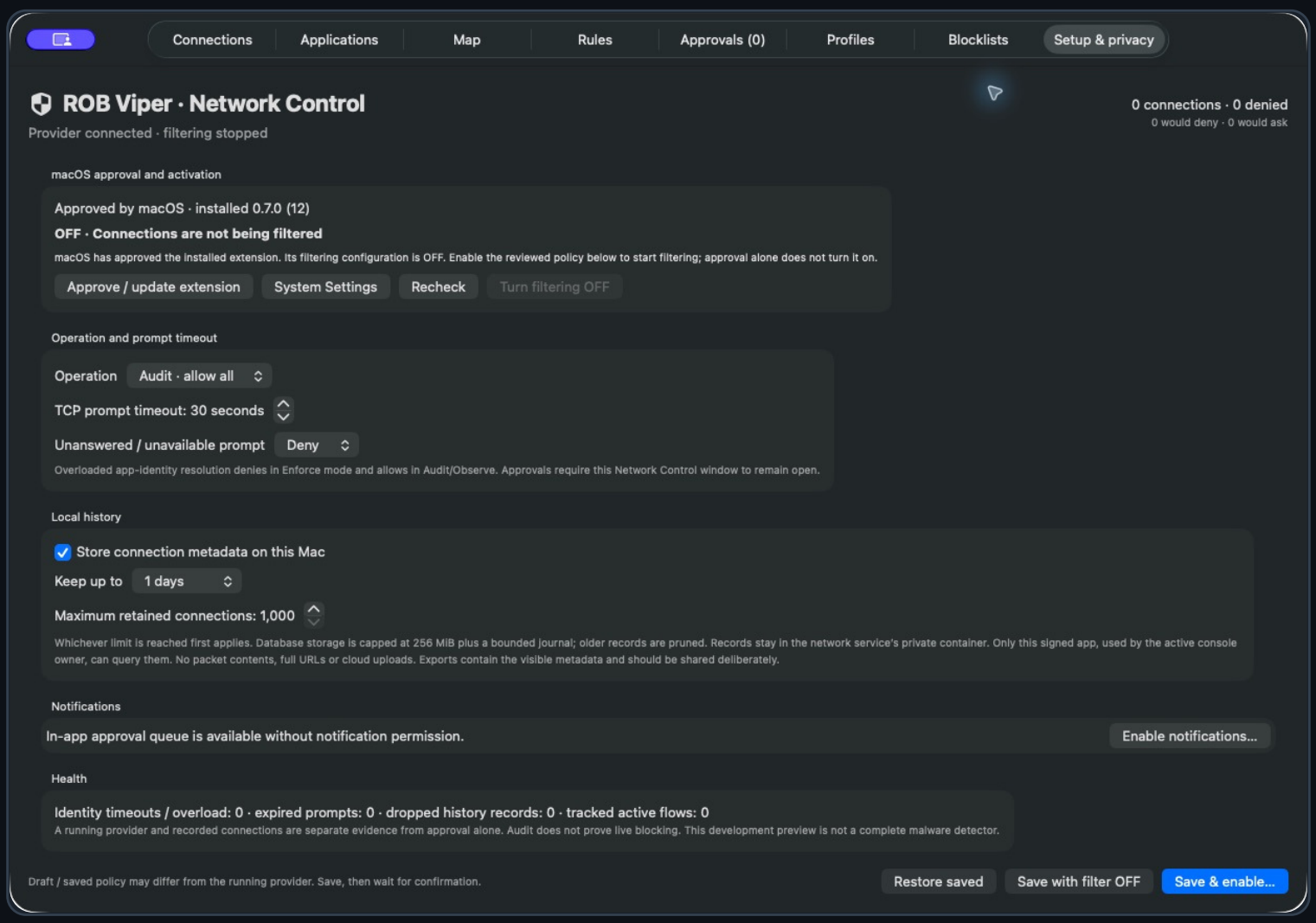


Figure 8. Approval and activation are different states. This historical capture deliberately shows the filter stopped.

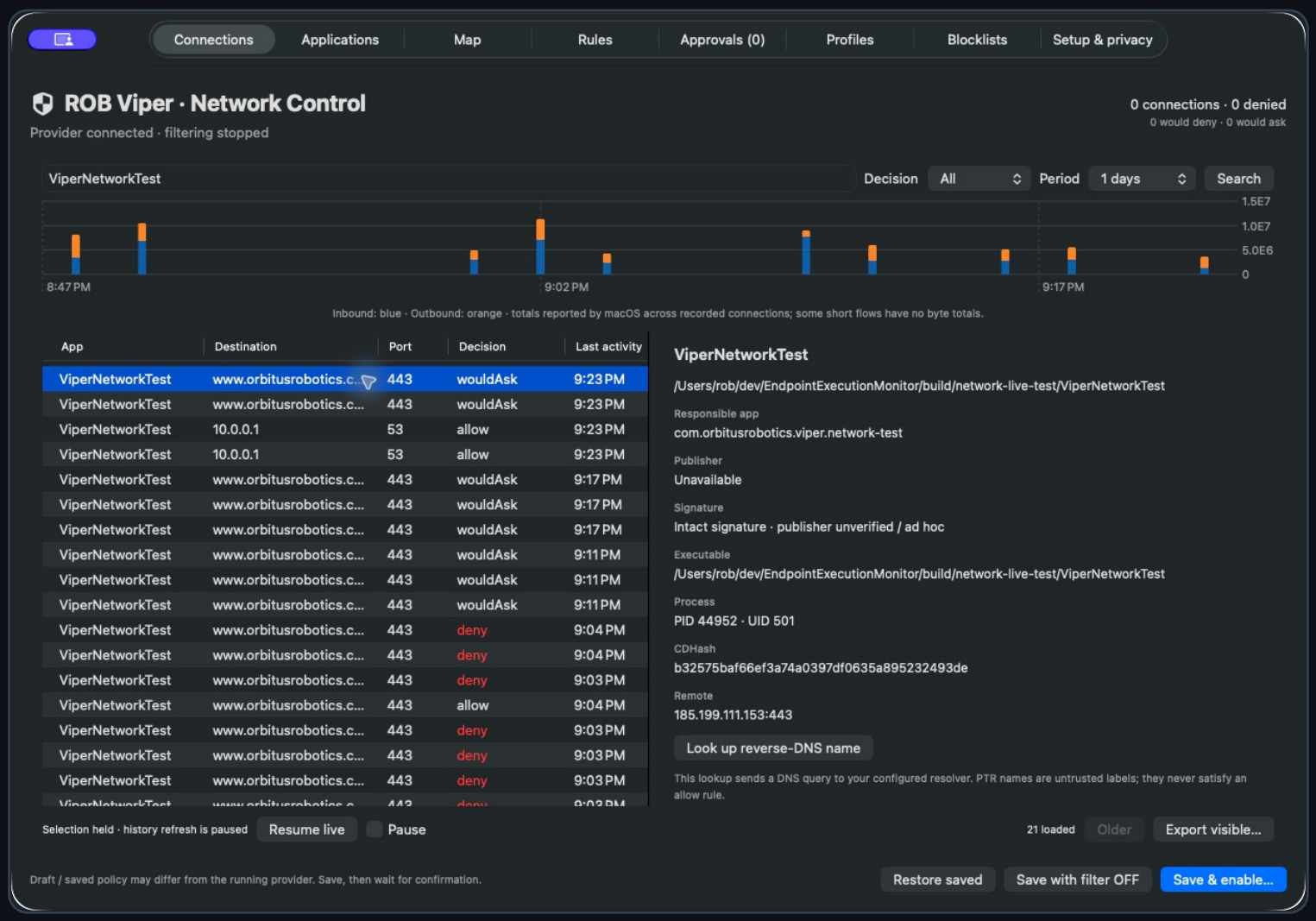


Figure 9. Historical disposable-client records. The graph aggregates the selected period rather than only the filtered application.

Choose the period and filters, select a connection, and read the responsible app and actual executable/helper identity. A helper can originate traffic on an app's behalf. A path is descriptive, not a trust guarantee. Missing byte totals on a short flow mean the total was not available; they do not prove zero traffic. Reverse-DNS lookup asks the configured resolver for a PTR label. That label is untrusted context and never grants permission.

A precise rule and a reviewed website allowlist

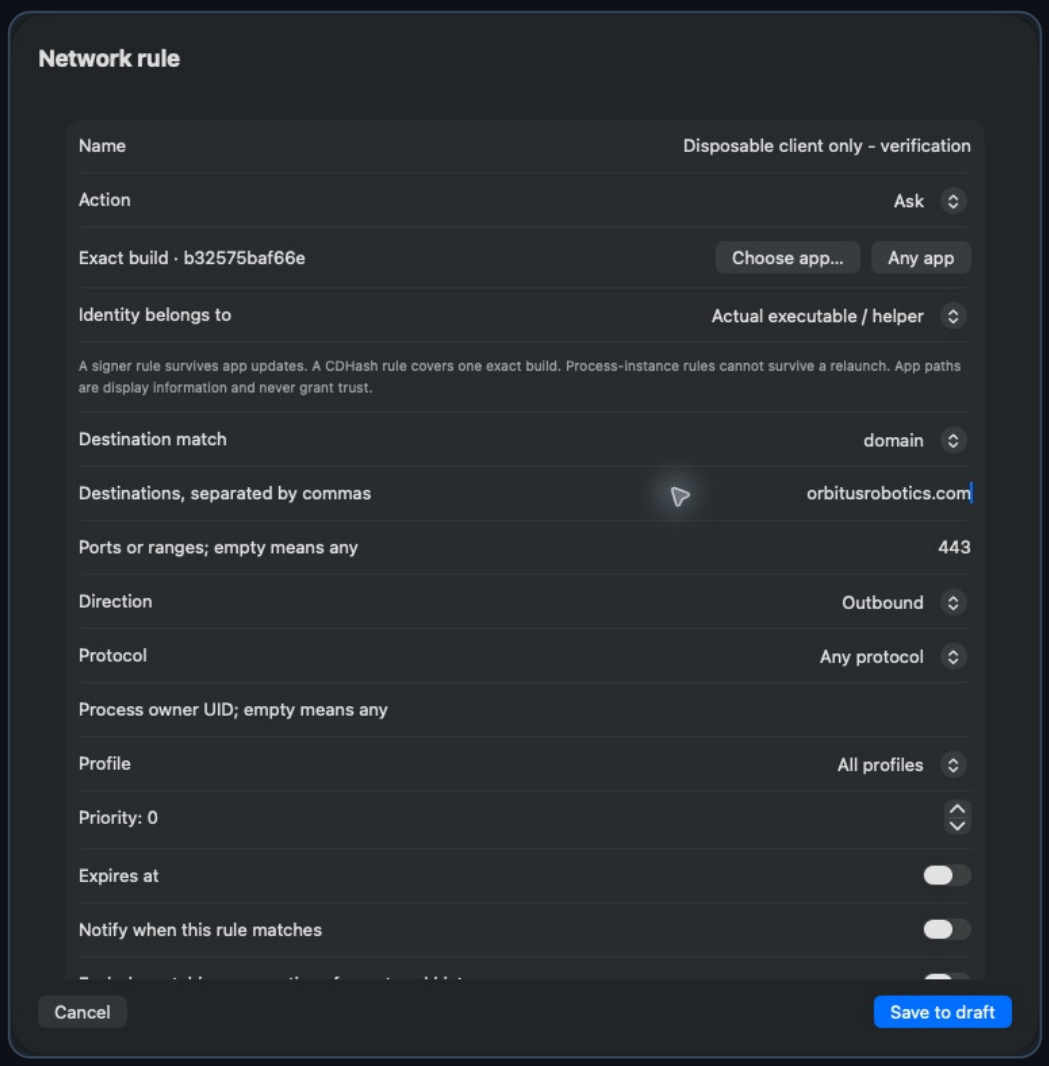


Figure 10. Unsaved domain-rule illustration; it was cancelled after capture.

Choose the actual identity source, a destination match, direction, protocol, ports and profile. Exact matches one IP or hostname; Domain includes the named domain and its subdomains; Network matches a CIDR range. Save to draft closes the editor. The main Save or Apply action changes the saved or active policy. Confirm the provider revision afterward.

1. Create a pilot profile with Default = Deny only after identifying its scope. This affects unmatched inbound and outbound flows.
2. Add explicit Allow rules for reviewed hosts and applications. Prefer an exact hostname when subdomains are not intended.
3. Exercise DNS, login, SSO, updates, MDM, CDNs, accessibility tools and normal work under Audit. No essential-service exceptions are inserted automatically.
4. Review wouldDeny results and a tested rollback. Only then schedule a controlled Enforce pilot.

Hostname matching depends on names supplied by macOS. Raw IP traffic may lack a name, and a shared IP can serve many unrelated sites. An allowed domain can host harmful content. These controls are not TLS certificate pinning, full URL-path inspection or a browser-content reputation service. Use the network decision alongside content analysis, identity and application context.



Figure 11. Empty approval queue. No live prompt or response was fabricated for this book.

Ask can hold a matching new flow in Enforce. Audit records wouldAsk and allows the connection. Keep the host window responsive; UDP requests expire within eight seconds and TCP uses the configured timeout. Select the shortest useful decision lifetime. If the host is unavailable or a request expires, the configured fallback determines the outcome, not an assumption that silence means permission.

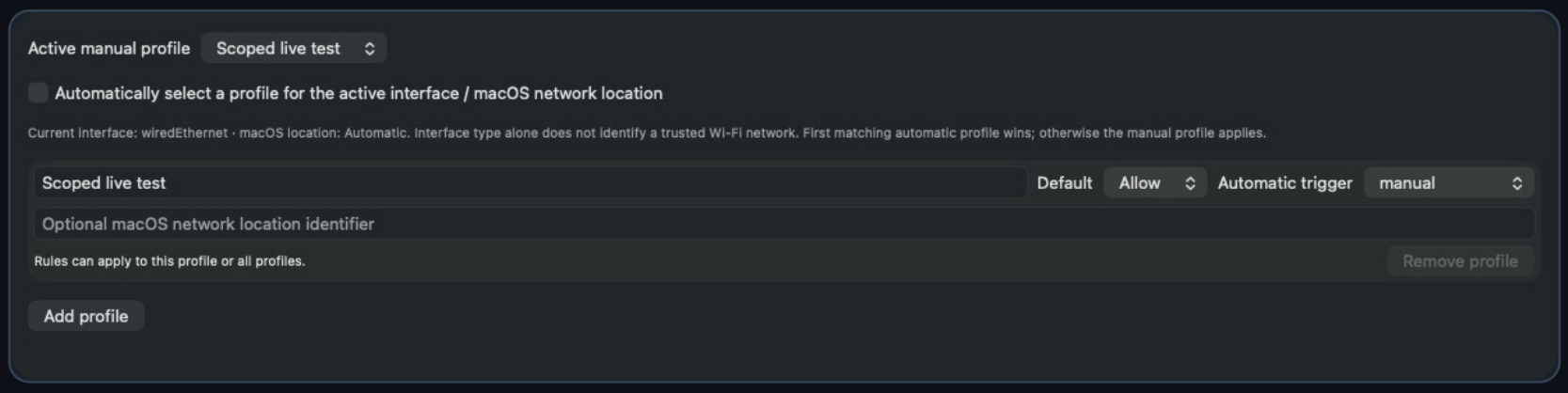


Figure 12. Profile detail: default action and automatic trigger.



Figure 13. Blocklist detail: subscription entry point. No feed is configured in this example.

Profiles can select policy by macOS network location or interface type; Wi-Fi alone does not identify a trusted network. HTTPS blocklists accept reviewed domain/IP data. The preview permits 16 subscriptions, each limited to 8 MiB and 100,000 entries, and retains the last working list when an update is invalid. Higher-priority exceptions can override a blocklist. Review the final policy, not just the feed name.

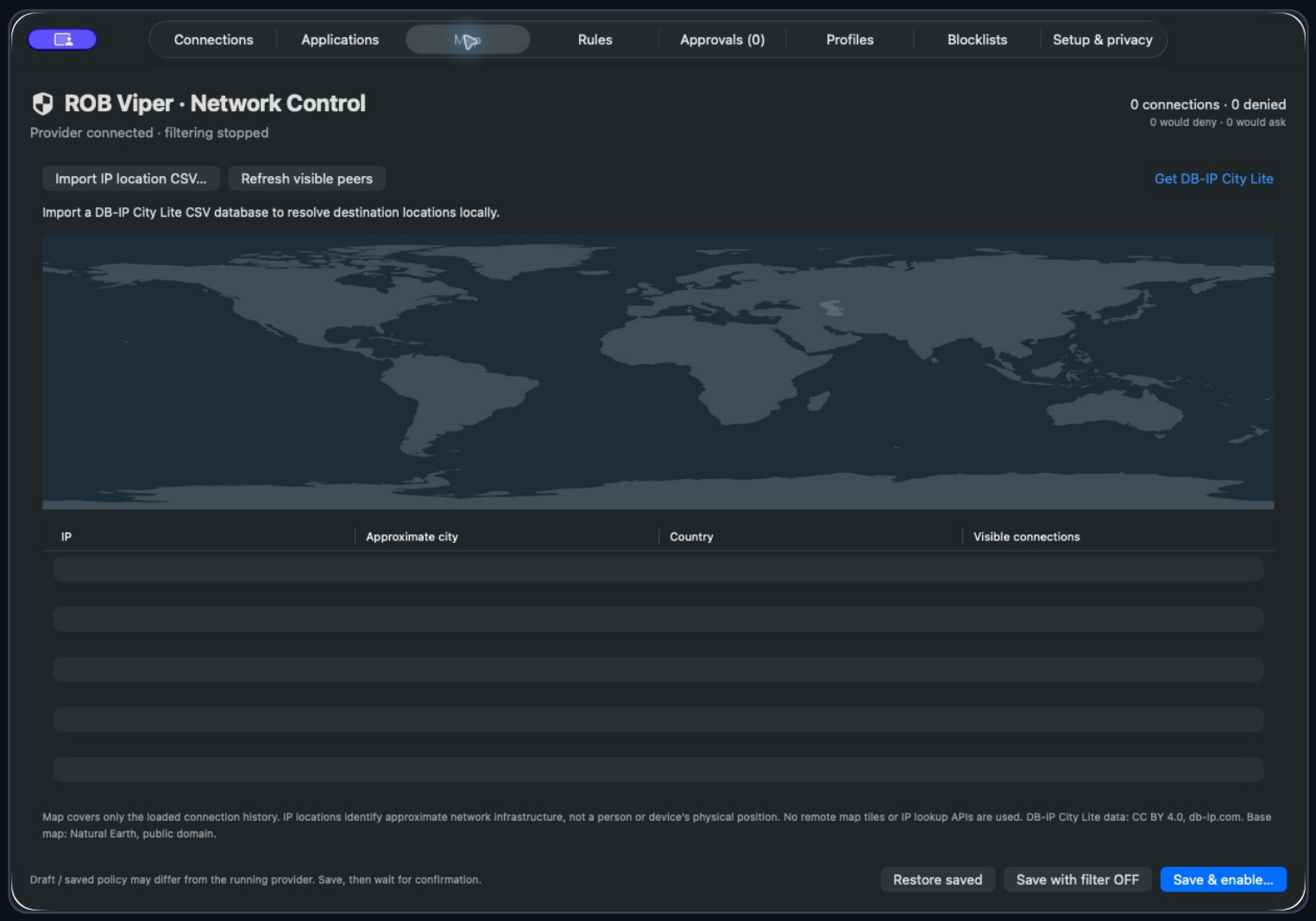


Figure 14. The optional offline map requires an imported dataset and matching loaded history.

The map estimates network infrastructure locations using imported DB-IP City Lite data. It does not locate a person or device. VPNs, CDNs and hosting can make the displayed city very different from an organization's address. The app uses no remote map tiles or per-destination IP lookup for this view. Preserve dataset attribution and dates when you use or export the map.

I / OPERATE WITH EVIDENCE

07 / Prepare Threat Center and its tools

Open Threat Center with Command-Shift-T. Tools & databases shows which engines were discovered, their versions and their executable paths. This edition was tested with ClamAV 1.5.4, Trivy 0.76.2 and YARA-X 1.23.0. These are a recorded baseline, not a promise that they remain the newest releases. Engine upgrades and threat-database refreshes solve different maintenance problems.

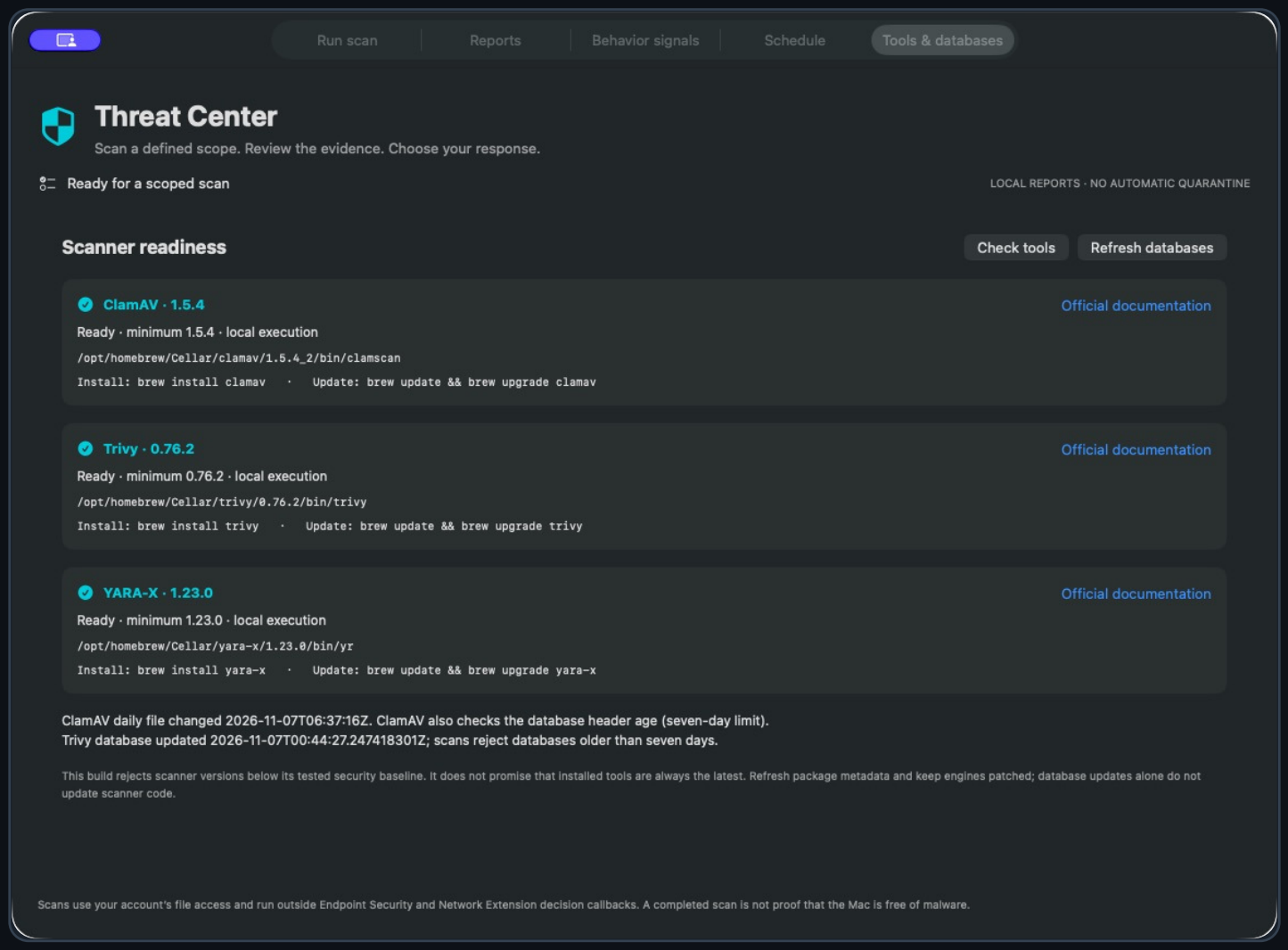


Figure 15. Tool readiness records versions and database freshness before scanning.

1. Use your approved software source to install the engines. The app displays Homebrew commands but does not silently install packages.
2. Choose Check tools. Investigate an unexpected path or version rather than trusting a binary with the right filename.
3. Choose Refresh databases. This downloads official ClamAV signatures and Trivy advisory data; it does not update the scanner executables.
4. Select a narrow local scope and run a harmless control. Record engine version, database detail, outcome and findings before working on an actual case.

Operator-run installation examples; review your organization's approved versions first

```
brew update
brew install clamav trivy yara-x
# For an already installed, reviewed toolchain:
brew upgrade clamav trivy yara-x
```

ClamAV uses signature databases, and FreshClam maintains the official data. The update mechanism needs a valid configuration and working access to its distribution service. [C2] Viper creates its own private scanner workspace and stores reports, settings and databases under the current user's Application Support/ROB Viper/Threats directory. A stale cache older than seven days is rejected; ClamAV also checks the signed database-header age.

Understand the execution boundary

Scans run on utility workers outside Endpoint Security authorization and Network Extension flow callbacks. The preview invokes known executable locations with argument arrays, a restricted environment, a private working directory, lower priority, bounded output and an outer 15-minute deadline. Only one scan or update holds the per-user job lease. Cancellation stops the registered child process group; forcibly killing the parent is not a substitute for supervised cancellation.

These controls reduce accidental scope and resource problems. They do not make an unknown sample safe or defend against a malicious local administrator replacing trusted software. Scanners are complex parsers and must receive security updates themselves. Use a prepared analysis environment for material you reasonably suspect is malicious. Do not grant root access merely to make a broad scan look more complete.

> **A missing tool is a failed prerequisite**
>
> If a required tool, database or index is unavailable, preserve the error and fix that prerequisite. A missing scan result is not evidence that the target has no malware or vulnerabilities.

II / MALWARE ANALYSIS

08 / Malware concepts that change the investigation

Malware is software intended to perform unwanted or harmful actions. An analyst usually begins with incomplete evidence of that intent. A suspicious filename, a signature match, an unexpected network flow and a persistence event are different observations. Your task is to test competing explanations without giving an unknown artifact more opportunity to act.

Behavior or label	Useful question	Evidence to seek
Trojanized installer	Did the delivery source or signed product differ from the expected release?	Origin, hash, signature, package contents and authorized installation history.
Credential theft	Could the process access sensitive authentication material?	Execution context, access permissions, relevant telemetry and account activity.
Ransomware-like behavior	Were files renamed, modified or made unavailable?	File-event coverage, affected objects, backups and application context.
Persistence	What would cause code to run again?	A new or changed startup artifact, its referenced executable and deployment history.
Downloader or loader	Did an earlier stage obtain or launch another artifact?	Related process identities, file creation times and connection records.
Potentially unwanted software	Is the behavior unauthorized under company policy?	Consent, business purpose, software inventory and observed actions.

Three methods, three limits

Signature analysis asks whether content matches known patterns. Static analysis studies an artifact without intentionally executing its program logic. Dynamic analysis observes a program in a prepared execution environment. None is complete alone. Static tools may miss encrypted or generated behavior. A dynamic run may never exercise the relevant branch. A known signature may match a test fixture or a legitimate file containing the same bytes.

Start with low-impact observations and a written hypothesis. For example: "This archive was received outside the approved distribution channel and contains an executable whose signer differs from the expected vendor." That sentence points toward provenance and code-signing checks. "A process renamed many files" points toward a timeline and affected-file review. It does not automatically justify running the file in a debugger or deleting the directory.

Separate the artifact from the host

A suspicious file sitting unopened in a download folder is a different situation from a process already executing with access to company data. In the first case, preserve and analyze the file without launching it. In the second, the incident owner must consider containment, volatile evidence, credentials and business impact. A content scan cannot establish that a host has no persistence or that previously exposed credentials are safe.

A valid signature helps establish a code identity. It does not make every behavior authorized, prove the vendor account was never compromised, or guarantee that the exact product is suitable for your organization. Conversely, unsigned or ad hoc code is common in development. Explain why an observation is unexpected in this environment rather than treating one field as a universal verdict.

> **Observation, interpretation, decision**
>
> Keep these in separate fields. Observation: a rule matched specific bytes. Interpretation: the artifact may belong to a suspicious family. Decision: hold the download and obtain an analyst review. The first statement must remain true even if the second is later disproved.

II / MALWARE ANALYSIS

09 / Open a case and preserve useful evidence

Good analysis begins before the first scan. Assign a case identifier, an owner, a reason for collection and an authorized scope. Record the original location, discovery time and timezone, device or project identifier, reporter, and whether the artifact was executed. For an active incident, use the organization's incident process before making changes that might destroy evidence or interrupt a critical service.

1. Preserve the original artifact through an approved evidence process. Work on a controlled copy when appropriate and keep the relationship to the original explicit.
2. Calculate a SHA-256 for the exact bytes you received. Record the byte size and the tool used. A hash identifies bytes; it does not prove when or from whom they were obtained.
3. Create a restricted case directory outside shared source repositories and automatic public sync paths. Keep artifacts, reports and analyst notes separate.
4. Record the engine and database versions before scanning. Afterward, retain the report and its hash with the case timeline.
5. Document every transfer, transformation and decision. If an archive is unpacked, record both the archive and the extracted objects as separate evidence items.

Read-only inspection of a selected artifact; substitute a real, approved path

```
umask 077
case_dir="$(mktemp -d -t viper-case)"
printf "%s\n" "$case_dir"
# Set sample to an approved evidence-copy path.
sample="/absolute/path/to/evidence-copy"
shasum -a 256 "$sample"
file "$sample"
```

A private directory and a hash are useful controls, not a complete forensic chain of custody. The original metadata, acquisition method, time source, permissions and handling history matter. Simple copying can change timestamps or omit extended attributes. If legal, regulatory or disciplinary evidence is involved, use your trained acquisition procedure and approved tooling rather than treating these introductory commands as forensic certification.

What belongs in the first note

- Question: what are we trying to determine, and what decision depends on the answer?
- Scope: exact file, folder, repository commit or device, with exclusions stated.
- State: running or dormant artifact, known execution, network access and relevant permissions.
- Provenance: source channel, expected vendor or owner, hashes and collection details.
- Coverage: tools available, subscriptions active, database dates and known gaps.

Treat filenames, embedded URLs, document text and scanner output as untrusted data. Do not paste a suggested command from an unknown file into a shell. Do not click a suspicious reference simply because a scanner printed it. The app restricts report reference links to HTTPS, but HTTPS alone does not establish that a destination is trustworthy.

> **Sharing is a separate decision**
>
> File uploads to public reputation services can disclose proprietary software, credentials embedded in content or personal information. A hash query can also reveal investigative interest. Use only an approved destination and sharing policy. Viper does not upload the sample or its report automatically.

II / MALWARE ANALYSIS

10 / Run ClamAV and interpret a signature result

ClamAV is a content-scanning engine. Its command-line scanner reads files and reports signature or heuristic matches. A separate daemon-based configuration is possible in the upstream project, but Viper's current integration invokes a bounded local scanner rather than installing an on-access antivirus service. [C1] This distinction determines when scanning occurs: a manual or scheduled job runs when configured, not whenever every file on the Mac is opened.

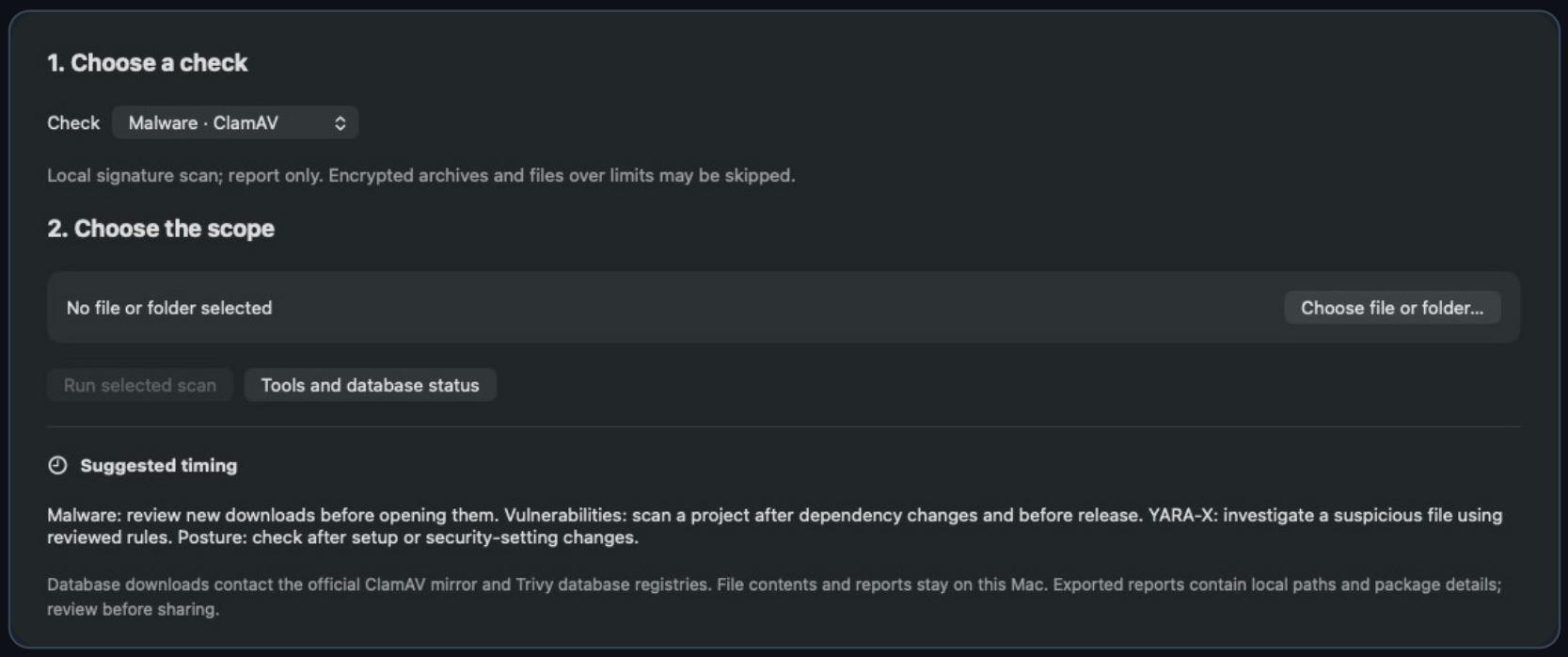


Figure 16. Choose a specific file or folder before starting a malware scan.

1. Confirm ClamAV readiness and fresh signatures in Tools & databases.
2. Choose Malware - ClamAV. Select the exact download, evidence copy or limited directory you intend to inspect.
3. Run selected scan and allow the job to finish or cancel through the app. Record an interrupted outcome as interrupted.
4. Open Reports. Read outcome and coverage before the finding count. Inspect the signature name, subject path and recommendation.
5. Export JSON for a complete local record, or SARIF for a compatible downstream workflow. Review local paths before sharing.

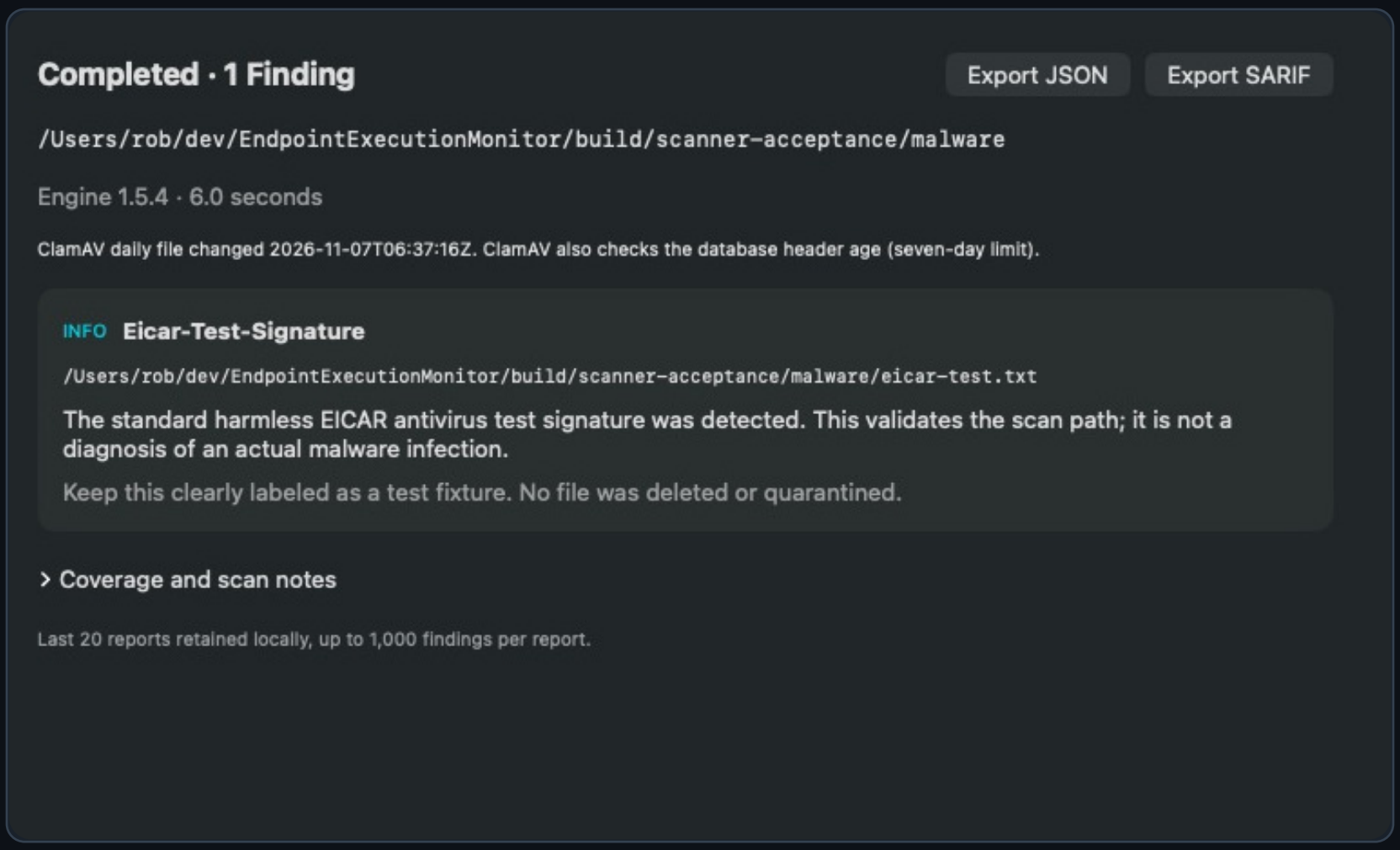


Figure 17. EICAR is deliberately labeled as harmless test data. This validates a scan path, not a real infection.

A match should produce a question, not an automatic deletion

For a non-test match, keep the artifact unopened and preserve the result. Is this the original file or a member inside an archive? Does the source match the expected vendor? Is the signature a specific family name, a broad heuristic, or a scan-limit indicator? Was the file already executed? Is the same exact hash present elsewhere in the authorized inventory? These questions guide the next step without assuming the scanner has answered them.

In Viper 0.7, recognized EICAR signatures are informational; ordinary signature matches are generally high severity and heuristic matches medium. These are display categories implemented by the integration. They do not replace an analyst's confidence assessment or an organization's incident priority. Read the exact title and evidence, especially when a heuristic name describes encryption or a resource limit.

No finding means no matching signature was reported in the inspected content under that configuration. It does not establish that every archive member was inspected, that an unknown threat could not be present, or that the host was never compromised. An organization should write its release or download decision in terms of the actual scan scope and other checks completed.

Practice with EICAR

EICAR provides a standard, harmless antivirus test file so people can exercise a detection path without using a real virus. [E1] Use the companion generator in a clearly labeled training directory and inform anyone responsible for other endpoint alerts. Scan the generated file through the GUI and verify the informational test result. Do not disable another protection because it intercepts the fixture; record that interception as part of the environment.

> **The CLI exit code is not the finding count**
>
> The Viper CLI returns 0 for a completed report without medium-or-higher findings. A recognized informational EICAR result can therefore return 0. Inspect findings and outcome explicitly when testing detection.

II / MALWARE ANALYSIS

11 / Coverage gaps, archives and false positives

An analyst should be able to answer "what did we fail to inspect?" as clearly as "what matched?" Viper deliberately bounds a scan so a malformed or unusually large input cannot monopolize the Mac indefinitely. A limit is a coverage boundary, not a clean result. Moving a difficult artifact into a larger approved analysis environment is a separate, documented decision.

Current ClamAV boundary	Operational meaning
64 MB per file	A larger object may not be fully inspected; preserve the incomplete result.
256 MB expanded archive and 10,000 embedded files	Compressed size alone does not describe the work required.
Archive depth 16; directory depth 32	Deeply nested content is outside the configured coverage.
30 seconds per file; 15-minute outer job deadline	A timeout is an unfinished inspection, even with zero detections.
No symlink traversal and normal user permissions	Linked targets and inaccessible content need separate, approved scopes.

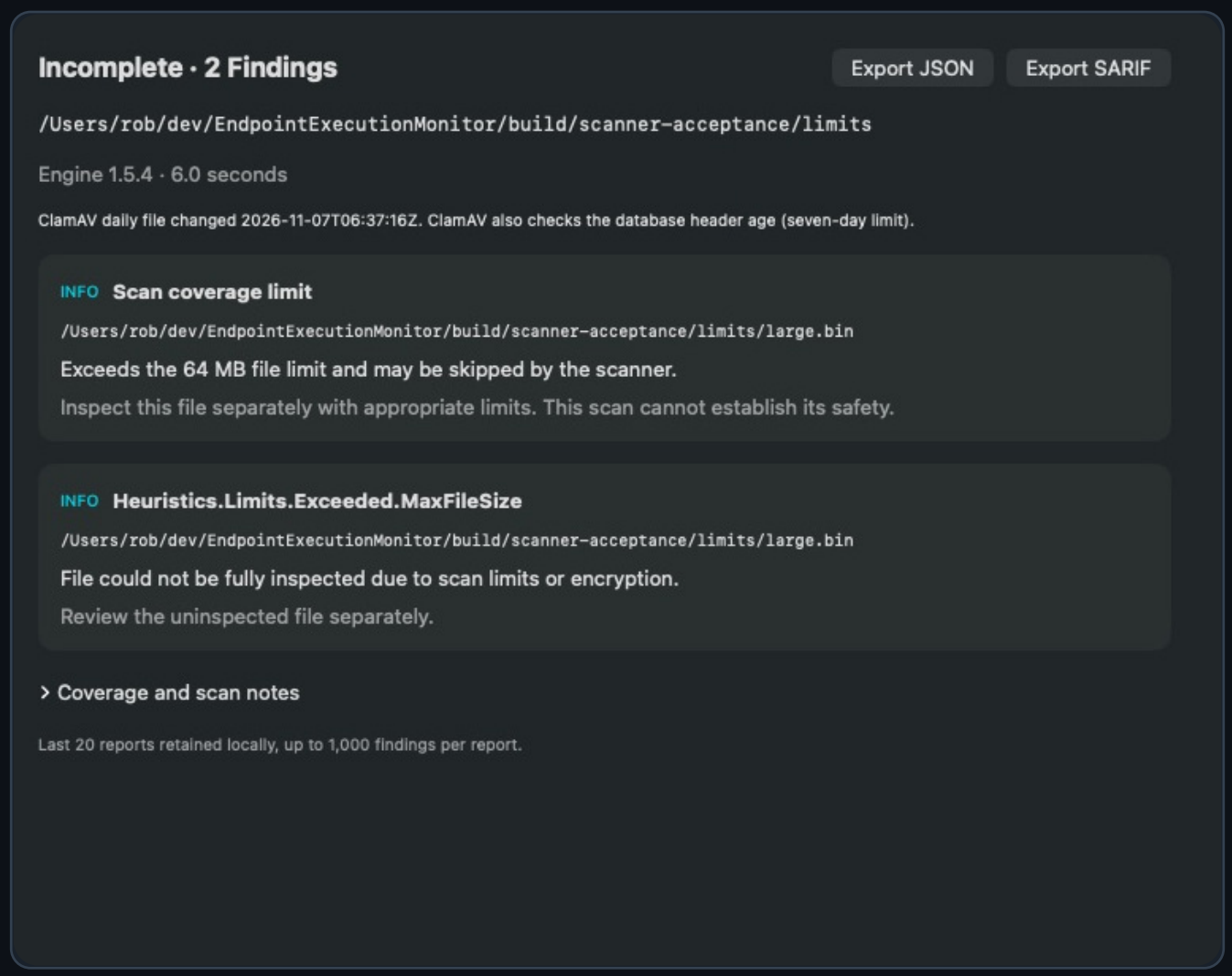


Figure 18. A real oversized-file control produced Incomplete and named the inspection boundary.

Encrypted archives deserve special handling. A scanner cannot infer the contents merely from the archive name. If a legitimate workflow requires inspection, obtain the password through the approved channel and unpack only in a prepared environment using maintained tools. Preserve the original archive, record the extraction method and hash the extracted objects. Do not invent a password, strip controls, or silently exclude the archive from the final report.

Review a suspected false positive

1. Confirm the exact file hash, signature name, engine version and database detail. Check whether the report concerns a nested object rather than the visible outer file.
2. Obtain the expected artifact from an authenticated vendor or internal build channel and compare provenance and code identity.
3. Use independent evidence: build records, package contents, approved software inventory and behavior. Agreement between two tools is helpful but not proof of independence or safety.
4. Assign an analyst disposition with reasons and an expiry. If a vendor submission is necessary, review what will be disclosed before uploading any sample.
5. Prefer a narrow, reviewed exception for the exact case over a broad directory or publisher exemption. Retest after signature or software changes.

Do not suppress a coverage error as a false positive. A large-file limit is an accurate statement that work was not completed. Likewise, a real test marker should remain identifiable as a test rather than being hidden from reports to make a dashboard green. Your tuning process should reduce noise while preserving truthful evidence.

Completed, incomplete, failed and cancelled

Completed means the configured operation finished. Incomplete means some content was not inspected or a tool reported an inspection problem. Failed means no usable result could be produced, for example after a missing prerequisite or malformed output. Cancelled and timedOut are also unfinished evidence. Keep these states in dashboards and ticket filters instead of flattening all of them into a zero-detection count.

> **Two acceptance criteria**
>
> A useful scan needs both an acceptable outcome and adequate coverage for the decision. A successful process exit cannot compensate for an unsupported target, missing permissions or a report that says no dependency targets were recognized.

II / MALWARE ANALYSIS

12 / Write, review and test a YARA-X rule

YARA-X matches file content against a rule you supply. A rule can contain metadata, patterns and a Boolean condition. [Y1] A match means that the condition was satisfied for the inspected bytes. It does not establish intent, execution or compromise. This makes YARA useful for focused triage and repeatable searches when the rule's scope is understood.

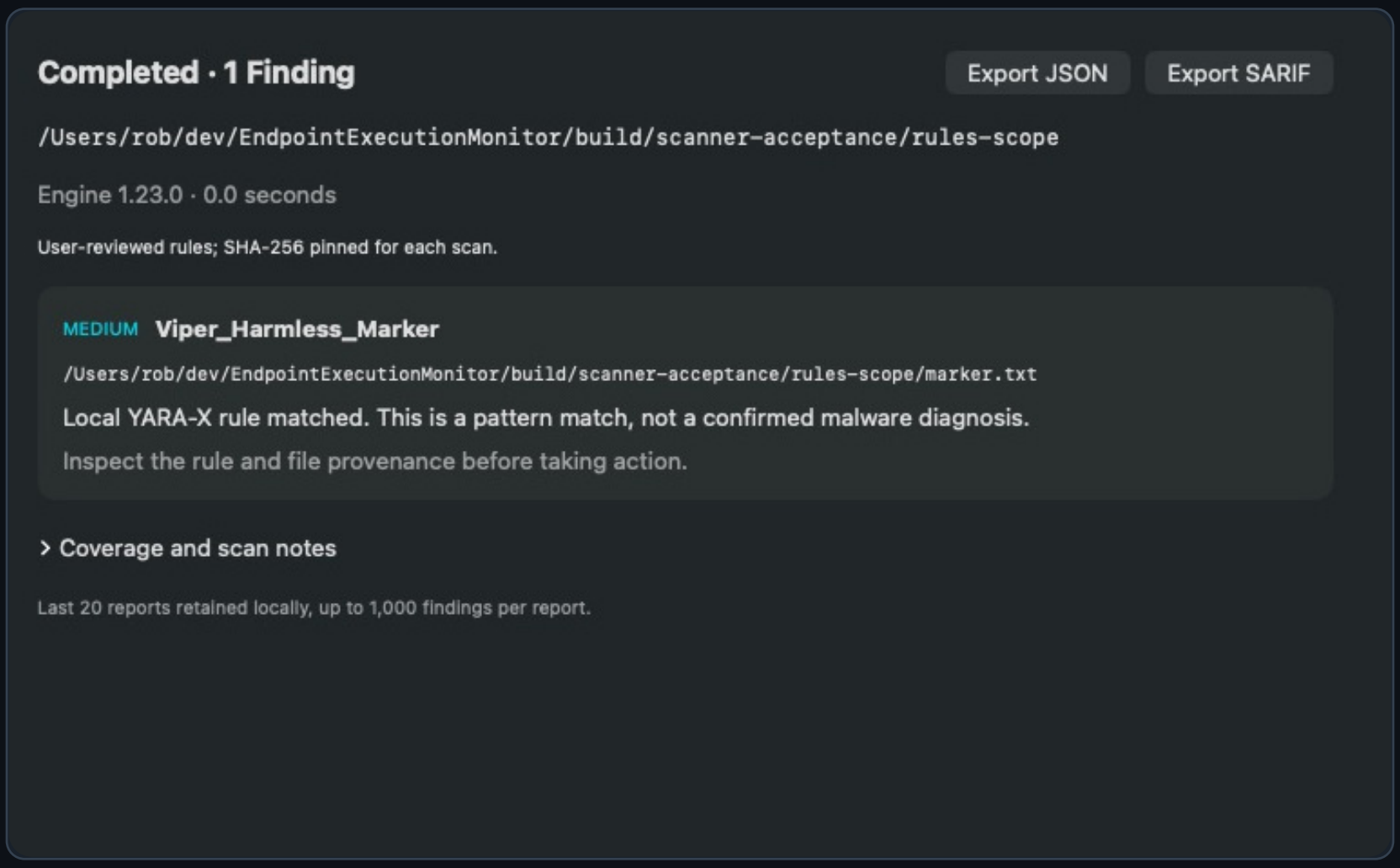


Figure 19. The custom marker matched an ordinary text file. The report explicitly avoids diagnosing an infection.

A deliberately harmless teaching rule

YARA source created for this handbook

```
rule Viper_Training_Marker
{
    meta:
        purpose = "Harmless handbook exercise"
        owner = "Orbitus Robotics training"
        revision = "1.0"
    strings:
        $marker = "VIPER-HANDBOOK-TRAINING-ONLY"
    condition:
        filesize < 1024 and $marker
}
```

The file-size condition limits this particular exercise to small files, and the literal marker makes the expected match easy to explain. YARA conditions can combine pattern presence with other checks; their meaning depends on the exact operators and data. [Y2] The exercise is not a generic malware detector. A legitimate document quoting the marker would also match.

1. Create a positive text fixture containing the marker and a negative fixture without it. Keep the rule file outside the selected scan target.
2. Review the source as text. Record its owner, purpose, expected matches and known false-positive cases. Calculate its SHA-256.
3. Choose Threat rules - YARA-X, select the target, then Choose reviewed YARA rules. Viper records the selected rule hash.
4. Scan both controls. The positive should match and the negative should not. Retain the outcomes and tool version.
5. Change the rule only through review. Select it again in the GUI or supply the new approved hash to the CLI before another run.

This integration accepts one local UTF-8 source file up to 1 MB and rejects includes. It checks the selected hash again and copies reviewed bytes into a private work directory. It does not silently download a universal rule feed. Those restrictions make the effective rule easier to review, but the analyst still owns the rule's logic and the data used to validate it.

Move beyond a single positive sample

Maintain a small test corpus: positive examples, near misses, legitimate software from the relevant environment and files at boundary sizes. A rule that matches only the sample used to write it may be too narrow; a rule based on a common word may be too broad. Prefer discriminating combinations supported by analysis. Measure scan time and review warnings before introducing expensive expressions.

The preview uses two threads, a per-scan rule timeout, directory-depth limits and a 64 MB file limit; it does not follow directory symlinks. Treat skipped or inaccessible objects as outside the evidence. Every current YARA finding is displayed as medium severity by the integration, so do not derive organizational priority from that display level alone. [Y3]

> **Rule governance**
>
> Store reviewed rules in version control with a purpose, owner, revision, source attribution, positive and negative tests, performance expectations and an expiry or review date. Keep the approved SHA-256 with each report. A changed rule is a changed analysis method.

II / MALWARE ANALYSIS

13 / Inspect macOS artifacts without launching them

Static inspection helps explain what you received before any deliberate execution. Use a patched, prepared analysis environment for suspicious material; metadata and archive parsers are software too. Start with the exact file type and hash. An extension such as .pdf or .jpg is a filename convention, not proof of the underlying format.

Read-only metadata checks; do not remove quarantine attributes

```
sample="/absolute/path/to/evidence-copy"
shasum -a 256 "$sample"
file "$sample"
xattr -l "$sample"
```

For an application bundle, distinguish the bundle from its main executable and helpers. Preserve the directory structure. Extended attributes can contain provenance hints but are mutable and may be lost during transfer. A quarantine attribute is a useful observation, not an immutable history. Do not strip it merely to make an unknown application run.

macOS signature and assessment examples; these do not launch the app

```
bundle="/absolute/path/to/ReviewedCopy.app"
codesign --display --verbose=4 "$bundle" 2>&1
codesign --verify --strict --verbose=2 "$bundle"
spctl --assess --type execute --verbose=4 "$bundle"
```

The first command displays signing information. Verification checks the signature against the object presented. Assessment asks macOS for its policy decision and may use system trust services; it is not a guarantee of an offline operation or of benign behavior. Record command output and time rather than replacing a nuanced result with "Apple approved this software." Apple's launch protections and trust decisions are separate from Viper's content scans. [A4]

Inspect the structure and ask targeted questions

- Does the bundle identifier and signer match the expected vendor and distribution channel?
- Which executables or helpers are present, and are they expected for this product?
- Does a startup configuration refer to this artifact, and was that configuration deployed by an authorized installer?
- Do readable strings or configuration entries suggest a destination or path worth correlating with actual events?
- Are timestamps plausible relative to acquisition and installation, while recognizing that timestamps can be changed?

On a machine with the appropriate developer tools, otool -L can list a Mach-O executable's linked libraries. A string extractor can reveal readable fragments. Neither proves that a referenced library was loaded or a URL

contacted during execution. Treat these as leads for correlation. Avoid publishing secrets or customer data accidentally found in strings or embedded configuration.

Package and archive inspection should follow the same discipline: list before extracting, use a controlled destination, prevent overwriting other evidence and record the extraction tool. A preview renderer, installer or script interpreter can activate behavior or parse a complex format. Do not double-click an unknown sample as an analysis shortcut.

> **When to stop local inspection**
>
> If the artifact appears active, the host handles sensitive credentials, or the investigation requires memory, forensic acquisition or controlled execution, involve the incident owner and the appropriate specialist environment. More commands on the original host are not automatically better evidence.

II / MALWARE ANALYSIS

14 / Correlate behavior signals with events

Viper's behavior panel provides explainable leads derived from the authenticated event stream. The analyzer has its own cursor, so holding the main table does not pause its work. Closing the host does stop this host-side analysis. The extension's separate execution policy can continue independently.

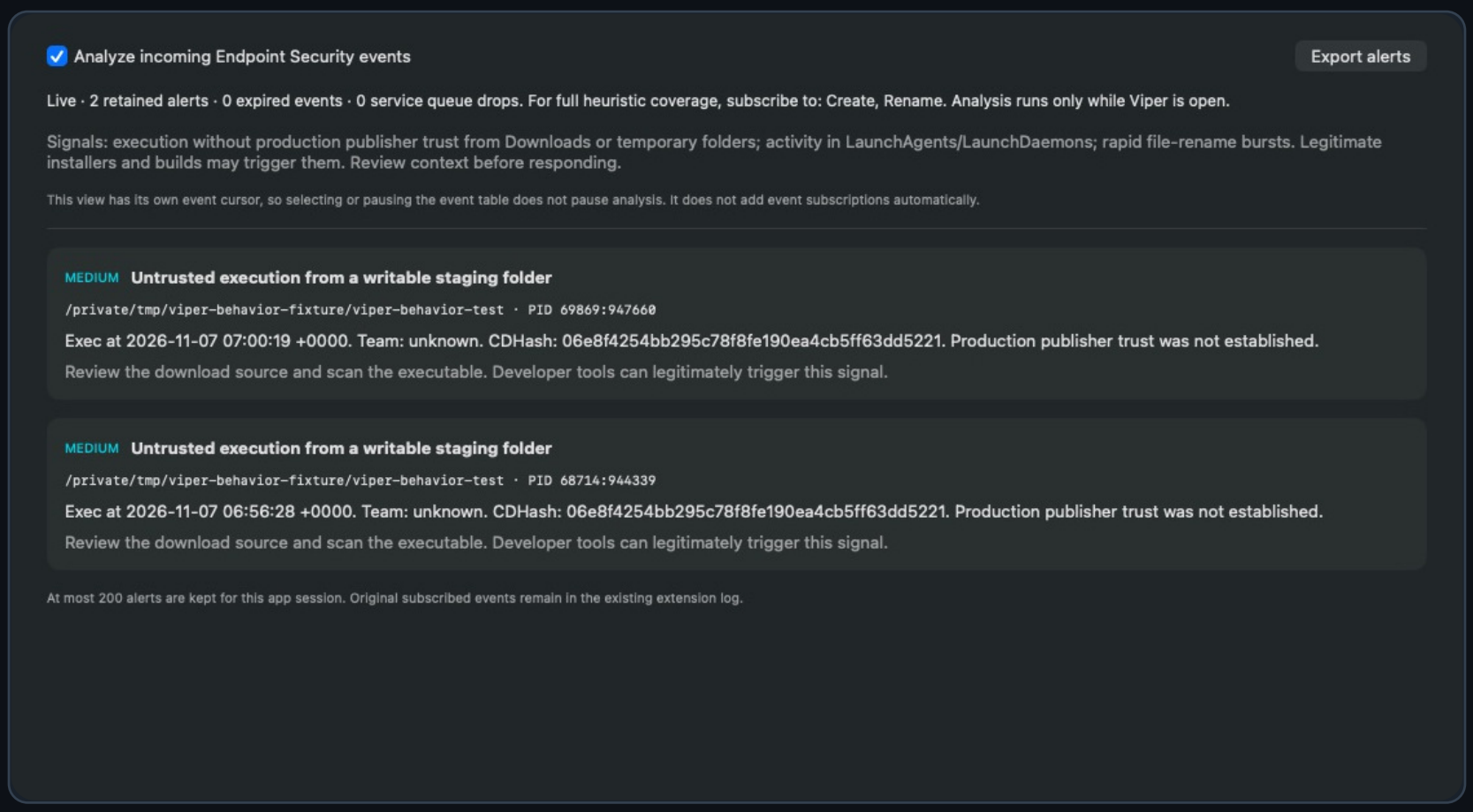


Figure 20. Two harmless staging-folder executions generated these examples. Missing Create and Rename subscriptions are shown as a coverage gap.

The current rules look for execution without established production-publisher trust from staging folders, activity involving LaunchAgents or LaunchDaemons locations, and at least 40 distinct rename destinations by one process generation within ten seconds. These are intentionally limited heuristics. Installers, builds, synchronization tools and administrators can produce similar events.

1. Read the coverage banner first. For the current set, enable Exec, Create and Rename in the main monitor and Apply the intended Monitor policy.
2. Inspect the original event, not just the alert title. Confirm the executable identity, affected object and timestamp.
3. Check whether an approved install, update, development build or bulk file operation explains the activity.
4. If the explanation remains suspicious, preserve relevant events and scan the approved artifact scope. Correlate network history without assuming an IP identifies a malicious actor.
5. Assign a disposition and preserve the reasoning. Export useful signals before the app session ends.

Do not overread a heuristic

A rename burst does not prove that file contents were encrypted. A close event alone does not establish a write. A new startup item does not prove an unauthorized persistence mechanism until ownership and deployment context are checked. An ad hoc program in a temporary directory can be a normal developer test. The important question is whether the behavior is expected and authorized for this machine and user.

The panel retains up to 200 alerts for the app session, with bounded state and deduplication. A low alert count may reflect retention, deduplication or missing telemetry rather than a quiet host. Keep the event losses and subscription state with any conclusion. Do not use this preview as a substitute for complete enterprise detection coverage.

> **Signals do not enact response**
>
> The current heuristics do not kill processes, quarantine files, change policy or add subscriptions. If a signal justifies a response, the authorized operator must review and apply that response through the appropriate control.

II / MALWARE ANALYSIS

15 / Use controlled execution only in a prepared lab

Dynamic analysis can help answer questions that static inspection cannot, but it changes the risk and evidence model. This handbook's executable exercises are harmless controls. Real suspicious code belongs in a prepared, authorized lab operated by people who understand its containment and observation limits. Viper's GUI is not a malware sandbox.

Define the lab contract

- Specify the question, sample hash, expected observations and a stopping condition before execution.
- Use a dedicated environment with no production credentials, personal browser profile, shared clipboard, mounted company shares or automatic cloud synchronization.
- Decide network behavior explicitly. An isolated or simulated network and a carefully controlled observation path are different from unrestricted Internet access.
- Record baseline state, time synchronization, tool versions and relevant telemetry. Verify the environment can actually observe the behavior being studied.
- Preserve reports through an approved transfer method, then rebuild or restore the lab using a documented process. A snapshot is a recovery aid, not a universal containment guarantee.

A specimen can behave differently when observed, require a missing dependency, wait for user interaction or remain dormant during the available time. A single run with no visible activity is therefore weak evidence. Record how long it ran, what was exercised, which destinations were reachable, and what telemetry was missing. Keep conclusions tied to those conditions.

Use a harmless tabletop before a real incident

Practice the same handoffs using a text marker and an ordinary disposable executable. The operator collects evidence; the analyst interprets it; the incident owner approves any response; the platform administrator validates recovery. The exercise succeeds when the team can explain the evidence and restore the intended state, not when the dashboard contains the greatest number of alerts.

For a suspected active compromise, containment and evidence preservation can conflict. A network isolation action may interrupt remote collection or a business service. Powering off can remove volatile evidence. Leaving a system connected can expose more data. Your incident owner should make that tradeoff using the actual case and approved procedures. Do not infer a universal order from a generic manual.

Escalation boundary

Viper can contribute process, network and scan observations. Memory analysis, forensic disk acquisition, sample detonation, enterprise isolation and account recovery require separately selected tools and an authorized response process. Document these integrations as organizational capabilities, not as features already present in this app.

III / VULNERABILITY MANAGEMENT

16 / What a CVE scan actually tells you

A CVE identifier gives people a common reference for a disclosed vulnerability. It is not a malware signature, a proof that an attacker used the issue, or a complete description of the risk to your organization. A package scanner usually begins by identifying software and versions, then comparing that inventory with advisory data. The quality of the inventory is therefore part of the quality of the conclusion.

A vulnerable library can be present in source metadata but absent from the shipped application. It can also be copied into a release without an obvious package manifest. A development-only dependency can still matter if it runs in a build job with release credentials. An apparently patched version may include a vendor backport that changes how a simple version comparison should be interpreted. Investigate the actual artifact and its use.

Term	Meaning in an investigation
CVE	A vulnerability identifier used to join records and advisories.
Package identity	Ecosystem, name and version; a name alone is often ambiguous.
Direct dependency	A component deliberately declared by the project.
Transitive dependency	A component brought in through another component.
Affected range	Versions or conditions the advisory identifies as vulnerable.
Fixed version	A vendor or maintainer remediation candidate, still requiring compatibility tests.
Reachability	Whether the affected behavior can be invoked in the relevant deployment.
Exploitability	Whether the conditions for exploiting the issue can be met in context.

Presence is the beginning of triage

For every important finding, identify the repository commit or deployed artifact, the manifest or package evidence, the dependency path, the application owner and the environment. Then read the advisory. Check the affected version range, prerequisites, supported fix and whether the upstream record has been revised, disputed or rejected. Preserve the advisory URL and retrieval time with the ticket.

Do not silently discard a finding because a developer believes the vulnerable function is unused. Ask for evidence tied to the exact release: build configuration, dependency resolution, call paths or a tested mitigation.

If the analysis supports a not-affected disposition, record its assumptions and a review trigger. A future feature or build change can invalidate that reasoning.

Keep operating-system and dependency work separate

The current Viper integration scans supported dependency manifests and lockfiles with Trivy. It is not a complete macOS CVE inventory, a remote service scanner or an inventory of every installed application. Maintain a separate managed inventory of macOS builds and application versions, compare those with the relevant vendor security advisories, and verify deployment through your management system. A dependency report cannot stand in for that work.

> **Three questions before priority**
>
> Is the affected component actually present? Is the advisory applicable to this build and configuration? What could happen in this deployment? Answer these before turning a scanner severity into an organizational deadline.

III / VULNERABILITY MANAGEMENT

17 / Scan dependency manifests with Trivy

Trivy detects known vulnerabilities in components it can identify in a target. Its upstream capabilities include several target types; the current Viper integration deliberately uses local filesystem vulnerability analysis with offline dependency identification. [T1] Do not assume that an upstream feature automatically appears in the Viper GUI.

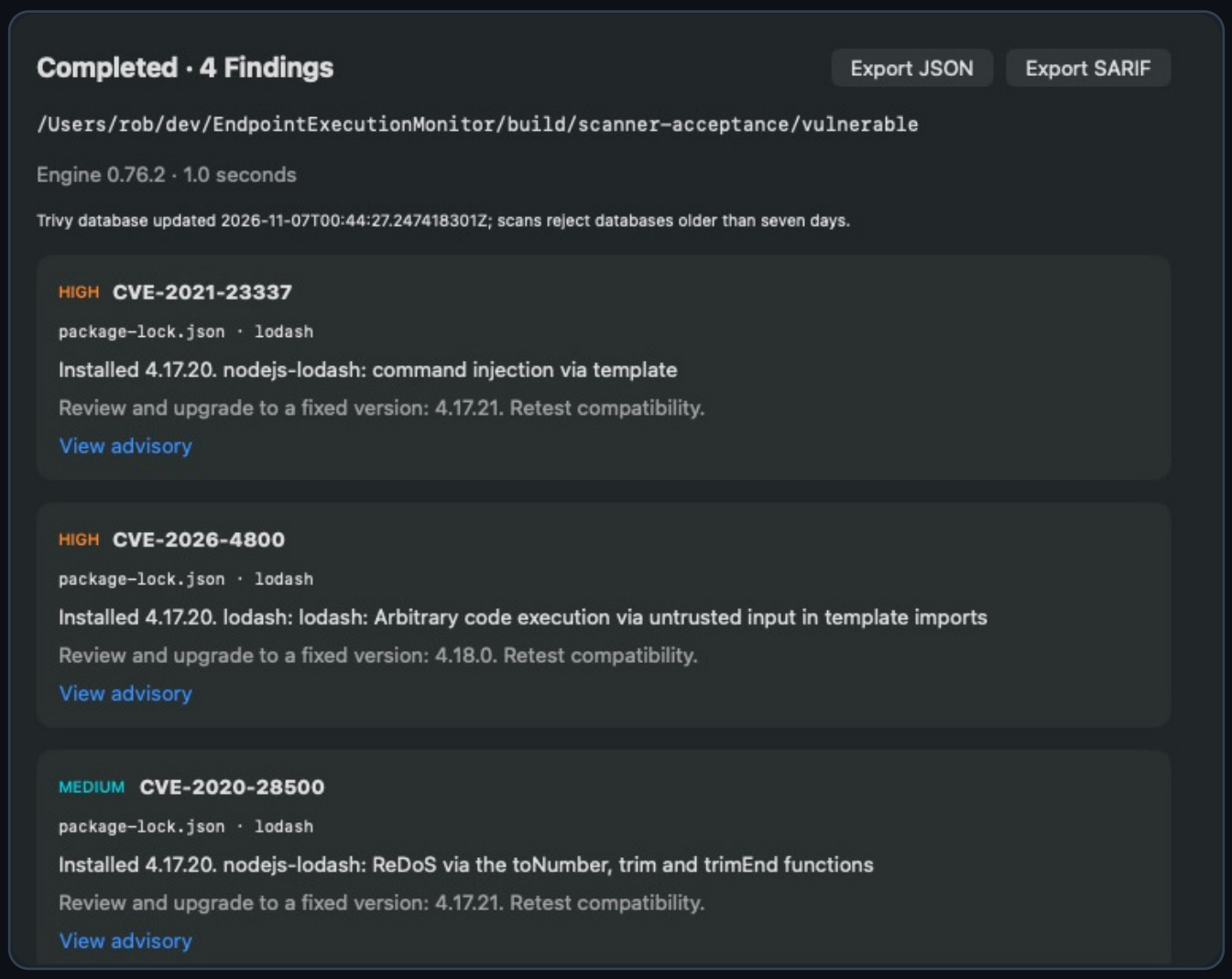


Figure 21. Historical metadata-only training project. No vulnerable package was installed or executed to produce this report.

1. Choose an exact project checkout or release-analysis directory and record its commit or artifact hash.
2. Confirm the expected lockfile or supported package metadata exists. A broad folder name does not prove coverage.
3. Check tool readiness and refresh advisory data. Record the database detail and scan start time.
4. Select Vulnerabilities - Trivy and run the scoped scan.
5. Read coverage notes for recognized dependency targets and package records. Investigate zero recognized targets before interpreting zero findings.
6. For each relevant finding, record package, installed version, advisory, recommended fixed version, owner and the evidence of presence.

Read the report without flattening it

Viper preserves the advisory identifier as the finding title, the target and package as the subject, the installed version in the evidence, and the listed fixed versions in the recommendation. An empty fixed-version field means that no fix was listed in the supplied data; it does not mean the vulnerability is harmless. A patch may require upgrading the parent dependency, replacing a component or applying a documented mitigation.

The wrapper uses two workers, vulnerability scanning only, no telemetry, no version lookup, and no target-provided configuration or ignore file. Project-controlled scanner configuration is excluded from the analysis boundary. The first integration does not download the separate Java index; a scope requiring an unavailable index can fail. Read the outcome and notes rather than assuming that a familiar language or lockfile was fully supported.

Filesystem analysis depends on the available package evidence. The upstream tool also offers other scanners and SBOM operations. Those are separate capabilities and are not implied by choosing Vulnerabilities in this GUI. [T2] In particular, this command does not automatically perform a source-code security review, a secret scan, an application penetration test or a macOS patch audit.

When to run it

- After a dependency or lockfile change, with the candidate commit recorded.
- Before a release, against the release inputs and, where supported, the built artifact.
- After a significant advisory update, even when your own code did not change.
- After a remediation, using the same scope and a documented current database.
- Periodically for supported retained releases that remain deployed.

> **A useful comparison**
>
> Compare package inventories and coverage alongside finding counts. Fewer findings can mean a successful fix, but it can also mean that a lockfile disappeared, an analyzer failed, or the scan used a different directory.

III / VULNERABILITY MANAGEMENT

18 / A reproducible CVE analysis exercise

The companion exercise uses a package-lock.json describing lodash 4.17.20. It is metadata only: there is no package download, install script or executable dependency. The recorded baseline detected CVE-2021-23337 and additional advisories available in the November 2026 database. Your current database can produce a different count. The exercise tests inventory-to-advisory matching, not exploitation.

Companion fixture and Viper CLI example

```
python3 prepare_lab.py --output /absolute/new/viper-training
# Inspect the generated files and manifest before scanning.
/absolute/path/to/viper-scan vulnerabilities \
  /absolute/new/viper-training/dependencies
```

The public app distribution may not yet include an installed CLI. In an authorized source checkout, swift build produces .build/debug/viper-scan, a development artifact. A company deployment needs a reviewed, signed and appropriately distributed build. Do not copy a private developer binary into fleet policy simply because the example command ran once.

1. Inspect the generated lockfile. Verify its package name, version and lockfile structure. Keep the generator manifest with the training record.
2. Run the vulnerability scan and save the JSON. Check outcome, database detail and recognized targets.
3. Find the selected advisory in the report. Follow its approved primary reference and compare the installed version with the affected range.
4. Write a training ticket with package evidence, advisory, owner, proposed fix, test plan and coverage limits.
5. Make a separate metadata copy representing a candidate patched version. Scan it again and explain which findings changed and which did not.

What the second scan does not prove

Editing a training lockfile does not patch an application. It demonstrates how a changed inventory affects matching. A real remediation must update the dependency through the project's package manager, resolve the lockfile, rebuild, test and deploy the resulting artifact. Verify the deployed version rather than closing the ticket after a text edit.

An older advisory may list a version that fixes that one issue while newer advisories still affect it. Do not teach a fixed version as permanently safe. Read current guidance for all relevant findings and select a supported release that fits the application. Keep the original and post-change reports so reviewers can see what changed, not merely a green check mark.

Exercise acceptance criteria

- The scan recognizes the intended dependency target and produces a usable completed report.

- At least one expected advisory is identified and traced back to package metadata.
- The trainee explains why this is not evidence of malware execution or host compromise.
- The proposed change includes compatibility tests and deployment verification, not only a version string.
- Any remaining findings, unsupported targets or coverage gaps remain visible.

Use the actual data

Screenshots are historical examples. Do not force a modern result to match their exact count or severity. If the expected advisory is absent, investigate the database, target support and advisory history before declaring the test successful.

III / VULNERABILITY MANAGEMENT

19 / Prioritize with context, CVSS, EPSS and KEV

Prioritization allocates limited engineering time. Keep technical severity, exploitation evidence and local business exposure separate so a reviewer can see why one item moves ahead of another. A useful queue records presence, affected configuration, external exposure, data sensitivity, privileges, compensating controls, owner and remediation options.

CVSS describes vulnerability severity through a scored vector. A Base score is not a complete risk assessment for your organization. Preserve the score version and vector when available; a bare number hides the assumptions. Threat and environmental context can change the operational decision. [F1]

EPSS estimates the probability of observed exploitation activity for a published CVE over the next 30 days across its observed population. It is not the probability that your particular Mac will be compromised. Record both the probability and date; a percentile is a relative ranking, not the same probability. Low EPSS is not a safety certificate. [F2]

CISA's Known Exploited Vulnerabilities catalog identifies vulnerabilities with known exploitation evidence. CISA publishes official machine-readable data and a schema. Use a dated copy and an exact CVE match; absence from the catalog is not proof that exploitation is impossible. [K1] Viper 0.7 does not automatically enrich its reports with CVSS vectors, EPSS or KEV. Add those through a reviewed analyst or organizational integration.

Example situation	Reasoned response
Relevant known exploitation on an exposed critical service	Urgent incident and remediation review; verify whether compromise evidence exists.
High technical impact in an affected but tightly restricted component	Plan prompt remediation with the owner; validate the claimed restriction.
Build-only dependency with release credentials available	Assess the build trust boundary; do not dismiss it because it is not shipped.
Component present but affected feature demonstrably excluded	Record a narrowly scoped not-affected analysis with evidence and a review trigger.
No recognized package inventory	Treat as a coverage problem, not a low-risk vulnerability result.

A proposed service-level model

Your organization should approve its own deadlines. One workable starting model is same-day triage for suspected active exploitation, a short remediation window for exposed critical assets, and a planned window for lower-exposure issues. These are proposed operating practices, not statutory deadlines or product defaults. Preserve an emergency path when new evidence makes an existing deadline inappropriate.

Exceptions need an accountable owner, a reason, a compensating control, an expiry and a re-review trigger. An exception does not remove the vulnerability from the inventory. Report accepted risk separately from fixed issues. Do not let a generic scanner ignore entry silently become a permanent business decision.

> **Avoid false precision**
>
> Do not multiply a CVSS number by an EPSS probability and call the result a universal risk score. Document the decision factors and the local consequences. Scores support judgment; they do not supply missing asset context.

III / VULNERABILITY MANAGEMENT

20 / Remediate and prove the deployed change

A useful vulnerability ticket ends with evidence about the delivered system. The loop is: identify, assess, assign, change, test, deploy, verify and retain. The original scanner result should remain attached even when later advisory data changes. This lets the team explain why the decision was reasonable at the time.

1. Confirm applicability and ownership. Identify the direct or transitive dependency path and the release that contains it.
2. Choose a supported fix or documented mitigation. Review compatibility, licensing and platform requirements.
3. Update through the project's normal dependency workflow. Review lockfile changes and unexpected new packages.
4. Run functional tests and the relevant security regression. Rebuild the release artifact from controlled inputs.
5. Rescan the intended scope with recorded tool and database versions. Compare coverage and the full set of remaining findings.
6. Deploy through an approved release process. Verify the actual installed or running artifact and close only the scope that is proven fixed.

When no fix is available

Read the advisory for supported mitigations and identify what exposure they reduce. Options may include disabling the affected feature, removing the component, restricting access or replacing a dependency. A firewall rule can reduce a reachable path but does not repair vulnerable code. Record the limitation, test that the control is effective, and keep a dated review for the eventual patch.

A suspected exploit on an affected system requires an incident assessment as well as patching. Replacing a package does not revoke stolen credentials, remove unrelated persistence or establish that data was not accessed. Keep remediation of the weakness separate from investigation and recovery of the system.

Close with evidence, not a comment

- Link the approved change and release identifier.
- Record the post-change package evidence and scan report.
- Record tests relevant to the affected behavior and ordinary application function.
- Identify the devices or deployments verified and those still pending.
- Explain remaining exceptions, compensating controls and their next review dates.

For a transitive issue, upgrading the parent package is usually more durable than editing an installed nested directory by hand. For a fleet application, distinguish the package prepared by the administrator from the version

observed on devices. For an offline laptop, a successful MDM command submission is not proof that the update has executed.

Reopen conditions

Reopen or re-evaluate the issue when the affected configuration changes, a mitigation fails, the advisory changes materially, a new release reintroduces the dependency, or deployment evidence contradicts the intended state.

III / VULNERABILITY MANAGEMENT

21 / Use SBOMs and CI without hiding failures

A software bill of materials records components associated with an artifact. It can make later advisory analysis more repeatable, but its value depends on how it was generated and which build it describes. Trivy supports SBOM generation and analysis upstream. [T3] The current Viper GUI does not generate SBOMs or perform a complete VEX workflow; treat those as separately designed pipeline capabilities.

A proposed release evidence bundle

- Source revision and build configuration.
- Artifact digest and signing identity.
- Package inventory or SBOM, with generation tool and scope.
- Vulnerability report, engine version, database detail and timestamps.
- Disposition records for applicable findings and dated exceptions.
- Functional tests, deployment evidence and rollback plan.

An SBOM that describes a source checkout may omit components introduced by packaging, a container base, a plugin or a downloaded build step. Conversely, a development dependency in the source inventory may not ship. Keep inventories tied to the relevant stage and artifact. When a new advisory appears, search retained inventories, identify deployed affected versions, and rescan or review with current data.

Separate scan execution from policy decisions

The Viper CLI emits JSON for scan operations. Exit 0 means a completed report without medium-or-higher findings; exit 1 means completed with such findings; exit 2 means incomplete, failed, cancelled or timed out. Infrastructure errors can also prevent a report from being produced. Always require valid JSON and inspect outcome, kind, target and coverage before applying a release threshold.

Capture evidence and review it; the companion script does not automatically approve a release

```
umask 077
report_dir="$(mktemp -d -t viper-report)"
/absolute/path/to/viper-scan vulnerabilities \
  /absolute/path/to/approved-checkout \
  > "$report_dir/report.json"
scan_exit=$?
printf "Scanner exit: %s\n" "$scan_exit"
python3 review_report.py "$report_dir/report.json"
```

Do not append an unconditional success operator to silence a failing security job. Equally, do not interpret every exit 1 as a broken scanner: it can be the intended findings signal. Define separate pipeline states for execution

failure, insufficient coverage, findings requiring review, and an accepted completed scan. Retain the raw report with restricted access even when the dashboard uses a summary.

Pin and maintain the scanner toolchain. Give the job only the repository or artifact scope it needs; avoid unnecessary deployment credentials. Review database-download access separately from scanning. The Viper wrapper removes inherited credential and configuration variables, but an organization must still secure its build runner, artifact store and report-upload integration.

Proposed organization policy

Begin with an advisory job that produces reviewed evidence. Establish ownership and tolerable failure rates before introducing a release gate. If you add exceptions, keep them in reviewed policy with expiry and evidence rather than allowing an untrusted project to suppress its own findings.

IV / RUN THE PROGRAM

22 / Build an organizational analysis workflow

An effective security program makes responsibility visible. A scanner can produce a finding without creating an owner, a business decision or a maintenance window. Start with a small operating agreement that defines what is scanned, who can access the evidence, who makes response decisions, and how a result reaches the team that can fix it.

Role	Responsibility	Evidence of completion
Operator	Run an approved scope and preserve the outcome.	Report, scope, timestamps and coverage notes.
Analyst	Assess applicability, confidence and priority.	Reasoned disposition and supporting references.
Application owner	Plan and test a code or dependency change.	Reviewed change, test results and release identifier.
Platform administrator	Deploy approved agent and policy changes.	Installed versions, policy revisions and device verification.
Incident owner	Coordinate containment, recovery and communications.	Decision log, authorized actions and recovery criteria.
Risk owner	Accept a time-bounded exception when necessary.	Rationale, controls, expiry and review trigger.

A proposed intake-to-closure path

1. Receive a report with a clear question and scope. Reject neither the report nor the concern merely because one required field is missing; assign the missing evidence task.
2. Validate that the scan completed and recognized the intended targets. Separate operational failures from security findings.
3. Deduplicate by artifact or package identity, advisory, environment and relevant release. Keep distinct affected deployments visible.
4. Assign the owner and a context-based priority. Document why the decision follows from the evidence.
5. Track remediation, mitigation or an approved exception. Verify the actual deployed state before closing.
6. Review recurring coverage failures and false positives as improvements to the process.

Choose a modest initial scope: perhaps one development project, one download-review workflow and a small managed-Mac pilot. Establish a baseline, measure job duration and collect operator feedback. Expanding scope before resolving routine failures can create a larger queue without improving decisions.

Separate routine work from incidents

Most dependency findings belong in a tracked remediation workflow. Evidence that a suspicious process executed, accessed sensitive material or caused damage may require incident handling. Provide a clear escalation route so operators do not wait for a weekly vulnerability meeting during an active problem. NIST SP 800-61 Revision 3 integrates incident response with broader risk management and preparation; use it as a reference for your organization's approved process. [N1]

Agree on evidence retention, access review and deletion with the people responsible for privacy and operations. Reports can expose usernames, internal paths, software versions and architecture. A ticket summary usually needs less detail than the original artifact. Keep the original in an approved restricted store and link to it rather than copying sensitive content into every dashboard.

> **Start with a service your team can maintain**
>
> A named owner, a reliable scoped scan and a reviewed remediation loop are more useful than an unowned collection of tools. The procedures in this part are a proposed organizational model; they are not automatically deployed by Viper.

IV / RUN THE PROGRAM

23 / Plan an MDM pilot and central administration

MDM and Viper serve different purposes. The management system handles enrollment, administrator identity, device groups and configuration delivery. Viper applies supported local policies and produces observations. A central dashboard must be backed by an actual service and a verified device channel; a static website cannot itself enroll or command a fleet.

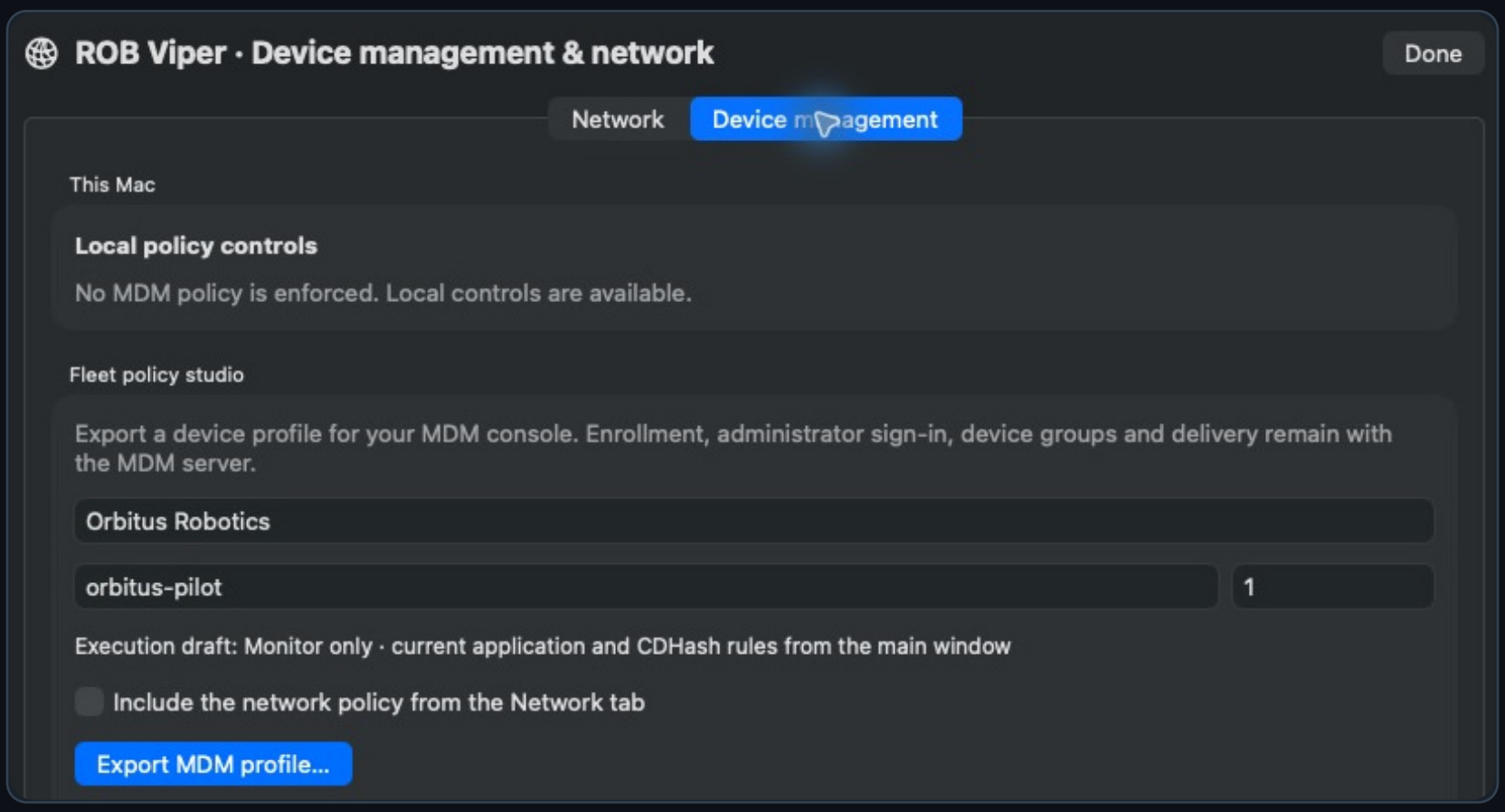


Figure 22. Profile export prepares a file. It does not enroll a device or prove delivery.

A proposed staged rollout

1. Inventory candidate Macs, business owners, macOS versions and recovery contacts. Select a small pilot group with a representative workload.
2. Prepare the production distribution: reviewed signing, extension identifiers, approvals, Full Disk Access and recovery procedure. Keep development artifacts out of production deployment.
3. Choose the management platform and verify its actual enrollment, profile delivery, application deployment, reporting and role capabilities. Licensing and self-hosting obligations need separate review.
4. Export a reviewed policy and deliver it to the pilot. Application-control profiles require outer schema 4 and matching agents; use the matching 0.7 components for this edition.
5. Start in Audit. Verify active device policy, representative events, queue health and recovery. Expand only after the acceptance criteria are met.

The legacy pane contains an older compatibility hint that does not cover the newer application-control policy. The profile schema, installed agent and active service response are the meaningful checks. Keep a deployment manifest with those versions and identifiers. A successful file export or MDM command acceptance does not establish that an offline device applied the policy.

Design administrator authority deliberately

Separate read access to fleet posture from permission to publish blocking rules. Use the identity provider, MFA, role controls and audit history supplied by your management platform. Keep a small, reviewed group able to change security policy, with a documented emergency recovery route. Viper's local forced-policy handling is not an implementation of enterprise administrator accounts or a complete central audit system.

An MDM-forced policy can lock local Apply and Stop. Administrators should publish higher revisions for reviewed changes, including a rollback to an allowing policy. Removing a management profile is not equivalent to proving the intended policy state on every device; verify the actual local service afterward. Maintain a recovery path that does not depend on a network destination your new rule may block.

Scanning across the fleet

The reusable CLI can be packaged for a separately managed, standard-account job. That is an organizational integration task: distribute approved tools, control scope and database updates, schedule resource use, collect only authorized report fields, and verify result freshness. The current GUI schedule runs while the host app is open and does not become a fleet scheduler merely because the Mac is enrolled.

Production acceptance

Require a real pilot to demonstrate install, approval, active policy, representative work, upgrade, offline behavior, rollback and evidence collection. Unit tests and a successful profile upload do not substitute for these device-level observations.

IV / RUN THE PROGRAM

24 / Schedule work at the right time

Scanning should fit the event that creates risk. New downloads deserve review before opening. Dependency changes deserve review before release. A new advisory can justify re-evaluating unchanged software. Posture belongs after configuration changes and at a cadence appropriate to the organization. A single nightly full-disk job is not the only or necessarily best pattern.

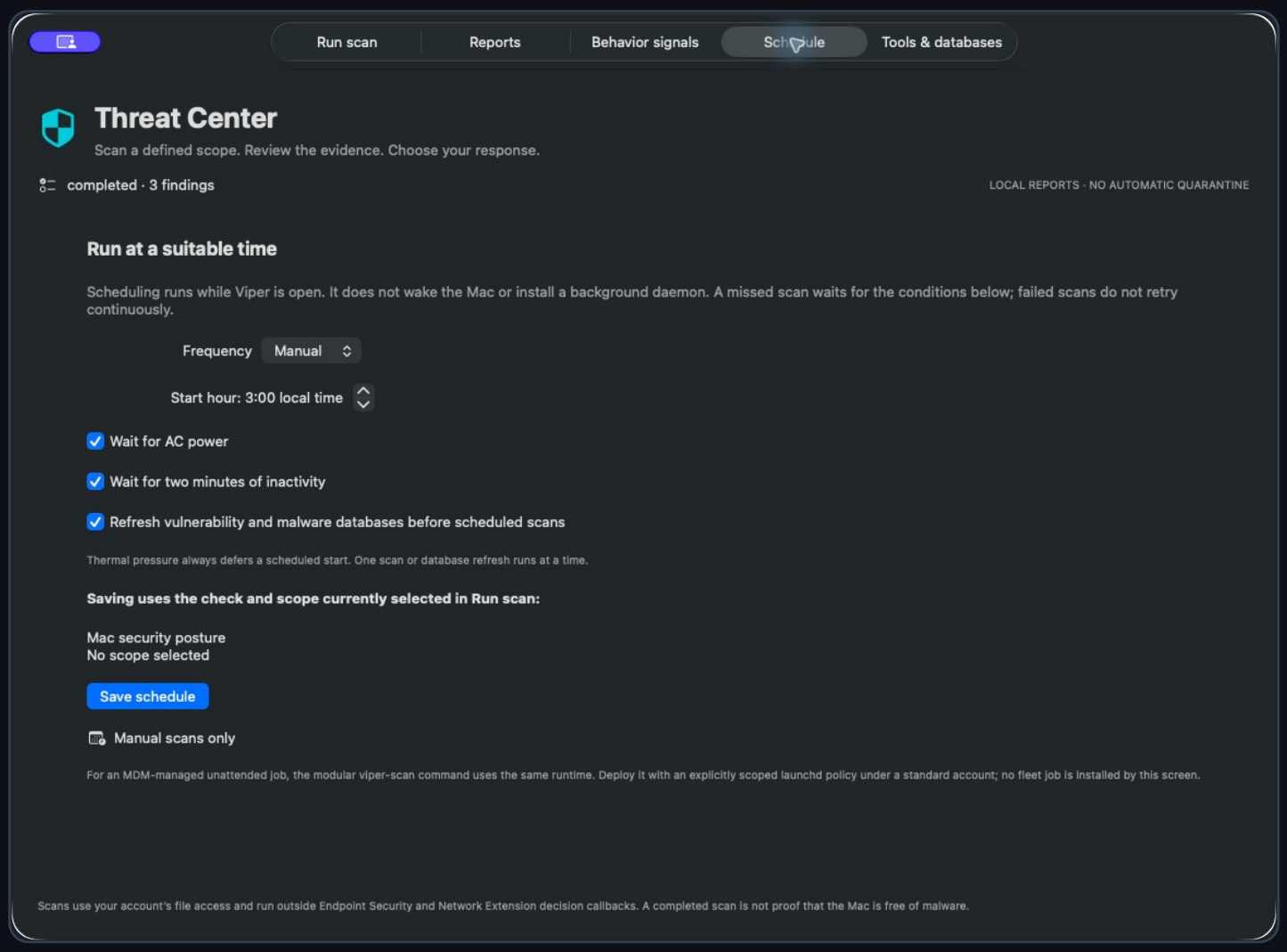


Figure 23. Manual is selected in this example. No automatic scan was enabled while preparing the book.

1. In Run scan, choose the kind and exact target. For YARA-X, select the reviewed rule source as well.
2. In Schedule, choose Manual, Daily or Weekly and the local start hour. Review AC power and two-minute inactivity conditions.
3. Choose whether the scheduled job should refresh databases first, according to the organization's update policy.
4. Save schedule. Confirm the saved job and scope in the displayed status; an unsaved edit is not a scheduled job.
5. Observe at least one intended execution and one deferred case in a pilot before relying on the schedule operationally.

The host app must remain open. This scheduler does not wake the Mac or install a launchd daemon. Thermal pressure defers a start, and a scan or database update already in progress holds the single-job slot. A missed

scheduled time waits for required conditions. A laptop that is asleep or routinely disconnected may therefore need a different managed workflow.

Separate freshness from frequency

A weekly scan with old advisory data answers an old question. A fresh database without a new scan says nothing about files that changed since the last result. Record both database age and report age. For a fleet, distinguish devices that were recently seen, recently scanned, successfully scanned and scanned with adequate coverage.

Measure real duration, thermal impact and user disruption on representative hardware. Avoid scheduling heavy analysis during production builds, presentations or robot-control workloads. If a job must be cancelled, preserve that outcome and arrange a follow-up; do not count it as a completed check. Use a separately approved supervised job for unattended fleet requirements rather than assuming the GUI will always be running.

> **An example cadence, not a default policy**
>
> Daily: tool readiness and new findings. On dependency change and release: scoped CVE analysis. Weekly: unresolved findings and exceptions. After configuration changes: posture. Monthly: exercise a recovery path and review coverage. Adapt the cadence to the actual assets and staffing.

IV / RUN THE PROGRAM

25 / Respond to a suspicious result

A response playbook turns uncertainty into accountable action. It should identify who can declare an incident, who can authorize isolation or account changes, and who communicates with affected people. The steps below organize evidence and decisions; they do not imply that Viper has a one-click containment or eradication service.

Stage 1: validate and preserve

Confirm the target, finding type and outcome. Verify that a test fixture, incomplete scan or expected deployment did not explain the alert. Preserve the report, hashes, relevant events and coverage state. Determine whether the artifact was merely present or actually executed. If there is credible ongoing harm, escalate immediately through the approved incident route while evidence work continues.

Stage 2: assess scope and authorize containment

Identify affected assets, users, data and business functions. Use evidence to distinguish a single held download from execution on multiple devices. The incident owner should choose containment with knowledge of management access, critical services and evidence requirements. Network isolation, disabling accounts or revoking tokens may be appropriate in a particular case, but they are consequential actions that require the organization's authority and tooling.

Stage 3: remediate the cause and recover

Fix the vulnerable component or configuration, remove or replace affected artifacts through a reviewed process, and address credentials or persistence if the investigation supports that need. A signature disappearing after a file is deleted does not establish complete recovery. Verify the restored system, expected software, active policies, account protections and business function before returning it to normal use.

Stage 4: retain the lesson

Record the initial signal, final findings, decisions, times, unresolved uncertainty and the evidence supporting recovery. Improve the relevant control: inventory, collection coverage, software update process, rule quality, operator training or escalation routing. Keep a false positive as a learning record when it reveals why the system was noisy; avoid hiding it by weakening an unrelated protection.

Decision	Minimum record
Hold an artifact	Exact hash, location, owner, reason and next review.
Block a code identity	Identity scope, evidence, reviewer, Audit result and rollback.
Restrict network access	Affected services, explicit scope, authority and recovery path.
Accept temporary risk	Business owner, controls, expiry and re-review trigger.

Decision	Minimum record
Declare recovery	Deployed state, validation results, residual risks and responsible approver.

A report is not a response receipt

A scan report can show that content was inspected. It cannot prove that a device was isolated, a credential revoked, a patch deployed or a user notified. Preserve the actual evidence for each action separately.

IV / RUN THE PROGRAM

26 / Exercise the program with harmless controls

An exercise should prove the workflow you intend to rely on. Use the companion fixtures to test signature detection, negative controls, YARA matching, dependency advisories and coverage errors. Keep production security policy unchanged unless a separately reviewed exercise explicitly includes a scoped policy test.

Control	Expected observation	What it does not prove
Clean training text	Completed scan with no matching signature.	Detection of unknown malware.
EICAR text	Recognized informational antivirus test finding.	Detection of every malware family.
YARA marker and negative text	Positive matches; negative does not.	That a real-world rule has acceptable false positives.
Metadata-only vulnerable dependency	Recognized target and applicable advisory finding.	An installed vulnerable package or exploitation.
Oversized sparse file	Incomplete coverage result.	Inspection of the skipped bytes.
Changed rule hash	A rejected or failed scan request.	Protection against a malicious local administrator.
Harmless process under Audit	Would deny and successful launch.	A real Enforce denial.

A team exercise

Have one participant act as a reporter, one as operator, one as analyst and one as incident or application owner. The reporter submits a training file with a case question. The operator records scope and runs the tool. The analyst explains the result and limits. The owner decides whether a simulated response or remediation is justified. A final reviewer checks that the evidence supports every claim.

1. Prepare a fresh training directory and record the fixture manifest.
2. Run positive and negative controls through the GUI. Export the results and compare them with the expected behavior.
3. Introduce a coverage problem, such as the optional oversized file, and verify that it remains visible in the report and ticket.
4. Run the companion report reviewer against a valid report, an incomplete report and malformed input. Confirm none of the latter cases becomes a silent approval.

5. Conduct a tabletop policy or incident decision and explain the rollback without changing a live production service.
6. Capture lessons and assign improvements with owners and dates.

Measure time to assign an owner, time to identify a coverage gap, time to reach a justified disposition and time to verify a simulated fix. Track false-positive burden and missing telemetry. These measures are more useful than celebrating a high alert count. If an exercise exposes a control that was never active, record that as a finding rather than editing the evidence to match expectations.

> **Release evidence in this edition**
>
> The prior 0.7 implementation receipt records 181 passing Swift tests, 37 scanner/behavior tests under Address Sanitizer, and nine real scanner checks with harmless fixtures. Those are bounded engineering observations, not certification, fleet validation or proof that real malware was exercised. This book also ships its own reproducible training checks.

IV / RUN THE PROGRAM

27 / Read posture and understand FileVault

Mac security posture is a read-only hardening check. It reads System Integrity Protection, Gatekeeper assessment status and FileVault status. It does not scan every file, evaluate every macOS setting or automatically repair a protection. INFO can record an expected enabled state; a protection needing review can be medium severity. Unknown or unavailable status remains a reason to verify manually.

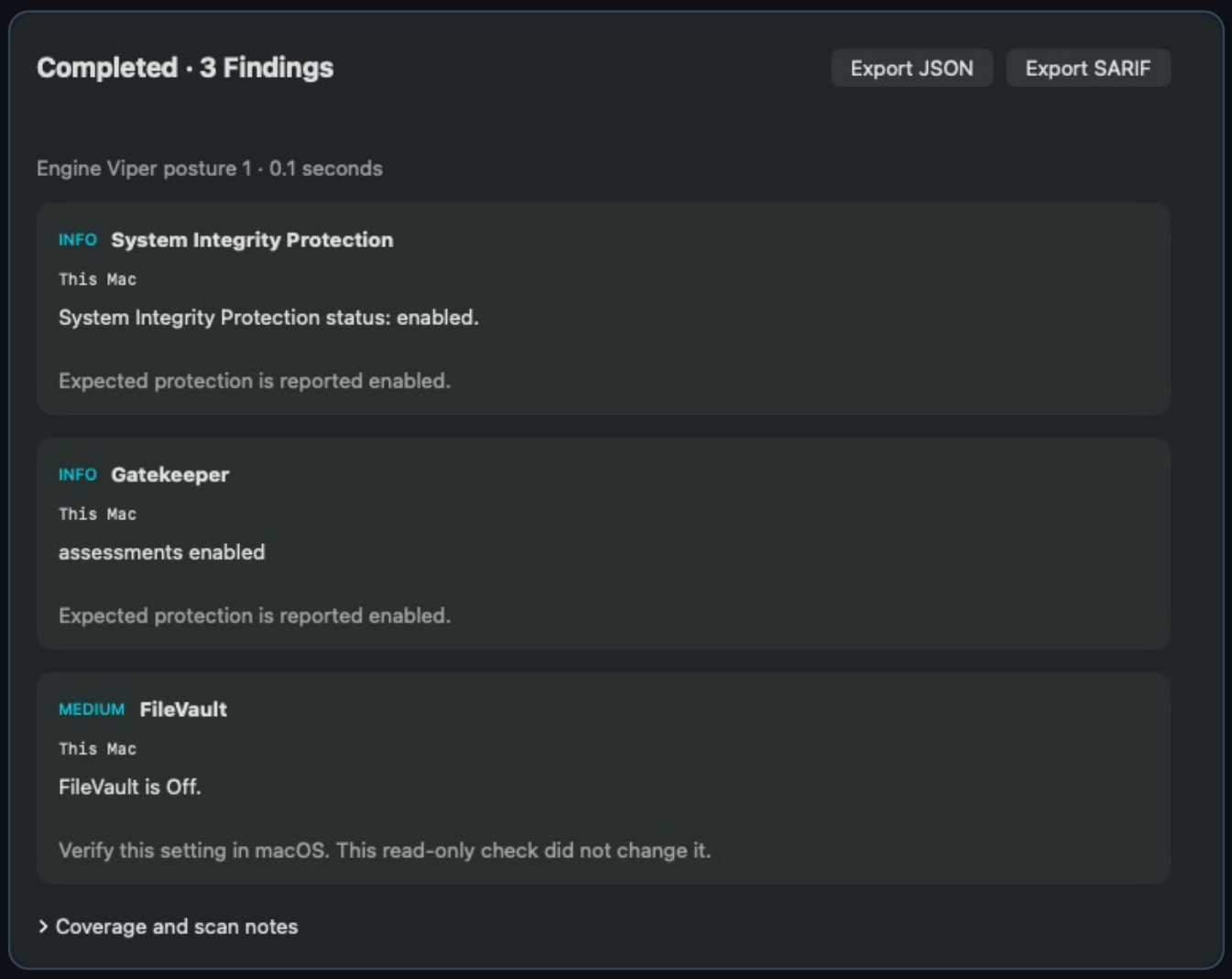


Figure 24. Evaluation-machine example: SIP and Gatekeeper enabled; FileVault off. This is a training observation, not a recommended baseline.

SIP protects selected system locations and constrains privileged modifications. [A3] Gatekeeper evaluates software trust as part of macOS launch protection. [A4] Apple's built-in malware protections operate separately from Viper; the app does not disable them or call undocumented force-scan mechanisms. [A5] A passed posture check cannot prove that a host is uncompromised.

What FileVault off means

On Apple silicon and T2 Macs, internal storage already uses hardware encryption. With FileVault off, the volume-key protection is based on the hardware UID; enabling FileVault adds protection tied to user credentials. This strengthens protection for stored data when someone has physical access to the Mac. [A1] The finding is a configuration issue, not a malware detection and not a statement that every internal byte is unencrypted.

1. Review System Settings > Privacy & Security > FileVault and confirm the startup-volume state.
2. Before enabling, keep a current backup and choose the recovery method offered by macOS. Store recovery information safely away from the startup disk; managed devices should use the organization's recovery-key process.
3. Plan startup unlocking and unattended-access requirements, especially on always-on machines.
4. After an authorized change, rerun the posture check and retain the new report. External drives and backups need their own encryption review.

Apple's setup guide explains the available recovery choices. [A2] FileVault protects data at rest; it does not stop malware already operating inside an unlocked user session. Treat encryption, account security, software updates, backups and incident response as related controls with different purposes.

> **Scope of the posture result**
>
> The current implementation runs csrutil status, spctl --status and fdesetup status. It does not assess every update, firewall, account, backup, MDM setting or recovery process. The check itself enables or disables nothing.

IV / RUN THE PROGRAM

28 / Export, retain and communicate evidence

Use JSON when you need the native report detail. A scan report includes schema version, identifier, kind, target, start and finish times, outcome, engine version and hash, optional rule hash, database detail, findings, notes and optional engine exit code. A finding includes severity, title, subject, evidence, recommendation and an optional reference. Treat every string as data rather than executable instructions.

SARIF export provides a standard structure for compatible analysis systems. Viper marks incomplete, failed, cancelled and timed-out runs as unsuccessful. The current export carries findings and scope but does not invent source-line locations for results that have none. Check your downstream importer's behavior before assuming that every report will appear as a source-code annotation.

Keep retention limits visible

Evidence store	Current preview boundary
Execution event logs	Five rotating files of 4 MiB; bounded queues can drop telemetry.
Network history	Opt-in local metadata with time, row and storage bounds; exports contain loaded rows.
Scan reports	Last 20 reports, up to 1,000 findings each, in private per-user storage.
Behavior signals	Up to 200 alerts for the host-app session.
Packet payloads and full browsing URLs	Not collected by the network history described here.

Export material needed for a case before local retention expires. Hash the exported report, keep the unmodified original and label any redacted derivative. Record who received the derivative and why. Do not edit the only copy to remove an inconvenient error or to make it look like a different machine was scanned.

A proposed organizational data flow

A local operator or approved scheduled job writes a report. A reviewed collector validates its schema, size, provenance and timestamps. It stores the original in a restricted repository and sends a minimal summary to a ticket or dashboard. The assigned owner investigates and attaches disposition evidence. This is a proposed integration design, not an automatic cloud service in Viper 0.7.

Define access by purpose. Operators may need local scope and troubleshooting details; application owners need package and remediation evidence; executives usually need aggregate progress and overdue risk. Avoid

putting full paths, usernames, raw files or unrelated network history into broad notifications. Use appropriate authentication, transport protection, access logs and retention controls in the system that receives reports.

> **Communicate uncertainty clearly**
>
> Use labels such as confirmed observation, analyst hypothesis, coverage gap, accepted exception and verified remediation. A short accurate summary is more useful than a dramatic claim that a scanner cannot support.

V / FIELD REFERENCE

29 / Case templates, checklists and daily reference

Malware-analysis case template

- Case ID, owner, reporter, discovery time and timezone.
- Question, authorized scope, original source and whether execution is known.
- Artifact hash, size, provenance and handling history.
- Tool, engine hash/version, database or rule hash, scan time and outcome.
- Findings, coverage gaps, related events and alternative explanations.
- Disposition, authorized action, actual response evidence and next review.

Vulnerability ticket template

- Asset or application, environment, owner and release/commit identifier.
- Package ecosystem, name, installed version, dependency path and inventory evidence.
- CVE or advisory identifier, affected range, primary reference and retrieval date.
- Severity source/vector, dated exploitation context and local exposure.
- Fix or mitigation, compatibility tests, deployment plan and rollback.
- Exception owner/expiry if needed; post-change scan and deployed verification.

Before trusting a scan

1. Confirm the tool and database/rule identity.
2. Confirm the intended target and permissions.
3. Read outcome and coverage before counting findings.
4. Verify expected targets were recognized.
5. Preserve evidence and assign the next decision.

Troubleshoot the visible state

Symptom	First checks
Events appear frozen	Held selection, Pause display, text filter, then collection and subscriptions.
Extension approved but no filtering	Setup & privacy OFF state, running provider and fresh test traffic.
Rule edit has no effect	Draft versus Apply, active revision, matching identity, existing process/flow.
Zero findings with an error	Outcome, coverage notes, database age, permissions and target support.
Scheduled job did not run	Host open, saved scope, local hour, AC/idle/thermal conditions and job lease.
Controls locked	Forced MDM policy and the administrator responsible for its revision.
No map point or hostname	Imported dataset, loaded history and macOS metadata; absence is not proof of no traffic.

Keyboard reference: Command-Shift-T opens Threat Center; Command-Shift-N opens Network Control. Quick tour sits beside execution policy. Resume live updates clears a held process view. Network Resume live clears a held connection. Restore saved discards unsaved network draft edits. These actions do not all change the same security layer.

Daily operating note

Review readiness and losses, assign new findings, examine important coverage gaps, verify planned changes and preserve useful evidence. Keep a short list of unresolved decisions with owners rather than relying on an alert count alone.

V / FIELD REFERENCE

30 / Glossary, sources and edition evidence

Term	Working definition
Artifact	The particular file, package, build or report under examination.
CDHash	CodeDirectory identity used for an exact code build; not whole-file SHA-256.
Coverage gap	Content, time or behavior the selected method did not inspect.
Disposition	An analyst decision with supporting evidence and any next action.
False positive	A reported security interpretation that review finds inapplicable or benign.
Hash	A digest identifying exact bytes, without proving provenance or intent.
IOC	An indicator that can guide investigation; context determines its meaning.
MDM	A device-management channel and service, separate from the local security agent.
SBOM	An inventory of software components tied to a stated artifact and generation method.
VEX	A structured statement about vulnerability applicability, requiring evidence and maintenance.
YARA rule	A content-matching condition whose match must be interpreted in context.

Evidence and limits of this edition

The book was checked against the 0.7 source implementation, the installed development-signed interface, scanner validation receipts and the original screenshot manifest. Public examples use harmless fixtures or explicitly labeled historical test records. No real malware, production blocking, broad personal-folder scan, fleet deployment or automatic remediation was performed to create this edition.

The companion scripts are teaching tools. Their generated files and checks are deliberately narrow, and their validation is recorded with the publication artifacts. A training pass does not establish complete detection coverage, legal compliance or production readiness. Check the current product page before relying on a feature or installer. [V1]

Primary reading

The following references support the tool definitions and selected platform facts. Operational checklists, exercises and organizational models are original guidance for this handbook. References were reviewed in November 2026; vendor features, advisories and release requirements can change. Follow current primary documentation when a version differs.

[C1] ClamAV: scanning. https://docs.clamav.net/manual/Usage/Scanning.html

[C2] ClamAV: signature management. https://docs.clamav.net/manual/Usage/SignatureManagement.html

[E1] EICAR: the antivirus test file. https://www.eicar.org/download-anti-malware-testfile/

[Y1] YARA-X: anatomy of a rule. https://virustotal.github.io/yara-x/docs/writing_rules/anatomy-of-a-rule/

[Y2] YARA-X: rule conditions. https://virustotal.github.io/yara-x/docs/writing_rules/rule-conditions/

[Y3] YARA-X: command-line reference. https://virustotal.github.io/yara-x/docs/cli/commands/

[T1] Trivy: vulnerability scanning. https://trivy.dev/docs/latest/scanner/vulnerability/

[T2] Trivy: filesystem targets. https://trivy.dev/docs/latest/target/filesystem/

[T3] Trivy: software bills of materials. https://trivy.dev/docs/latest/supply-chain/sbom/

[F1] FIRST: CVSS 4.0 user guide. https://www.first.org/cvss/v4.0/user-guide

[F2] FIRST: EPSS frequently asked questions. https://www.first.org/epss/faq

[K1] CISA: official KEV data repository. https://github.com/cisagov/kev-data

[N1] NIST: SP 800-61 Revision 3. https://csrc.nist.gov/pubs/sp/800/61/r3/final

[A1] Apple: introduction to FileVault. https://support.apple.com/en-gb/guide/deployment/dep82064ec40/web

[A2] Apple: protect data with FileVault. https://support.apple.com/guide/mac-help/mh11785/mac

[A3] Apple: System Integrity Protection. https://support.apple.com/en-us/102149

[A4] Apple: Gatekeeper and runtime protection. https://support.apple.com/guide/security/sec5599b66df/web

[A5] Apple: protecting against malware. https://support.apple.com/guide/security/sec469d47bd8/web

[V1] ROB Viper: release status and support. https://www.orbitusrobotics.com/products/rob-viper/

[B1] Handbook, exercises and publication status. https://www.orbitusrobotics.com/books/rob-viper-security-operations/

Support and corrections

Visit the handbook page for downloads, exercise assets and the actual Apple Books publication status. Report a correction with the edition, chapter, observed app/tool version and a minimal example. Do not send passwords, FileVault recovery keys or unrelated private logs. [B1]

Public contact: orbitus@orbitusrobotics.com. Author: Rodolfo Aramayo. Publisher: OrbitusRobotics LLC. This freely priced edition retains its copyright; third-party tools and documentation retain their own licenses and trademarks.