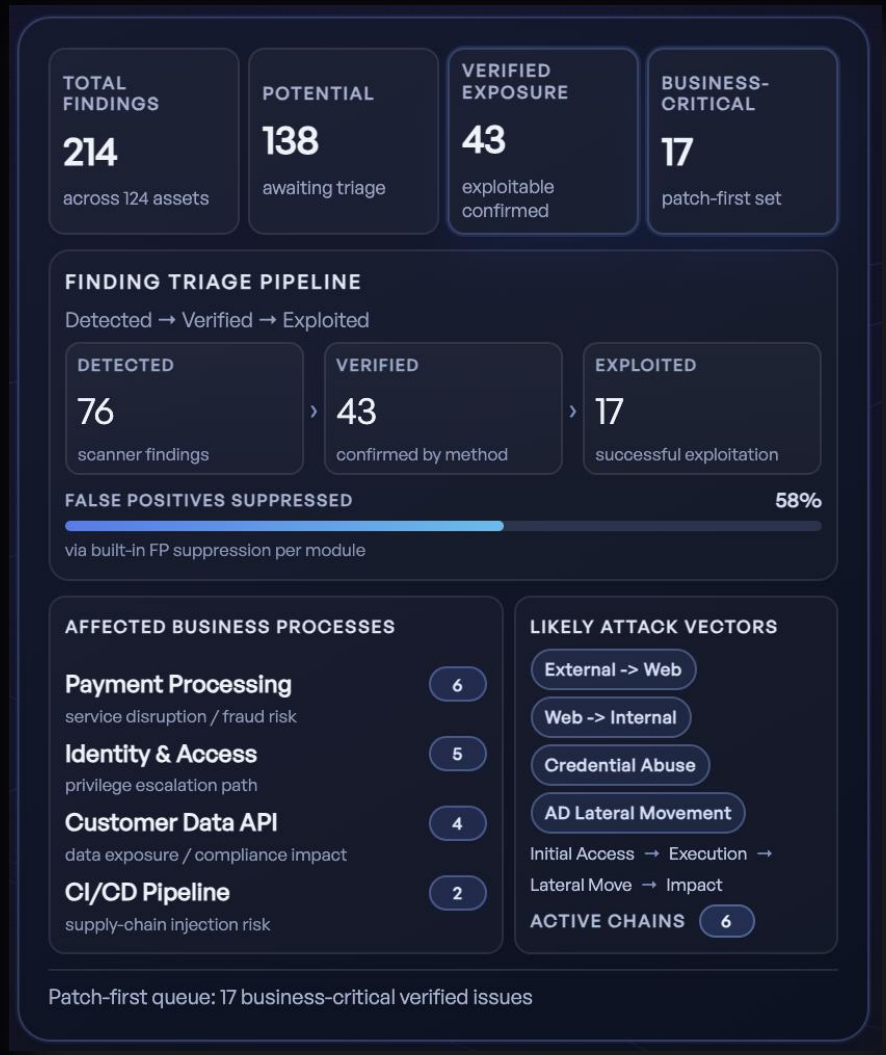
Verified exposure. *Not ~~more~~* noise.

All-in-One Automated Offensive Security Platform

82% of scanner findings are false positives. We fix that.



What is Pentesterra

Safe · Scoped · Controlled · Auditable

Explain in one sentence

Pentesterra is a Continuous Attack Validation Platform that verifies real exploitability across code, web apps, APIs, and infrastructure - then turns the evidence into attack chains, reports, dashboards, and remediation validation.

One standalone platform. No extra tools. No complex integrations. No long deployment – starts in 30 sec.

The problem

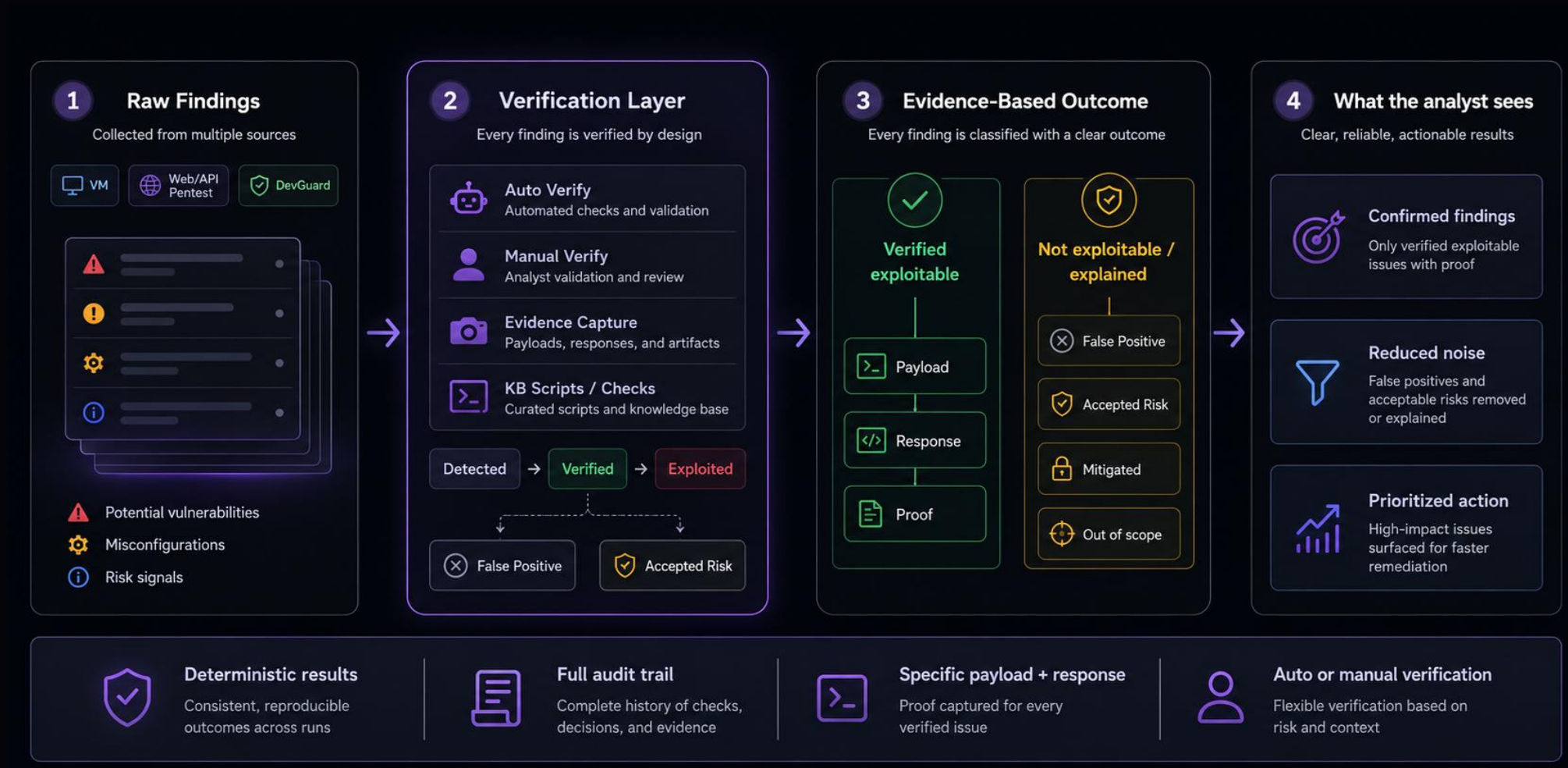
Security tools generate findings. Nobody has time to verify them.

$4.88M - average cost of a data breach (IBM, 2024). Reported breaches up **+30% YoY**

The Pentesterra Way

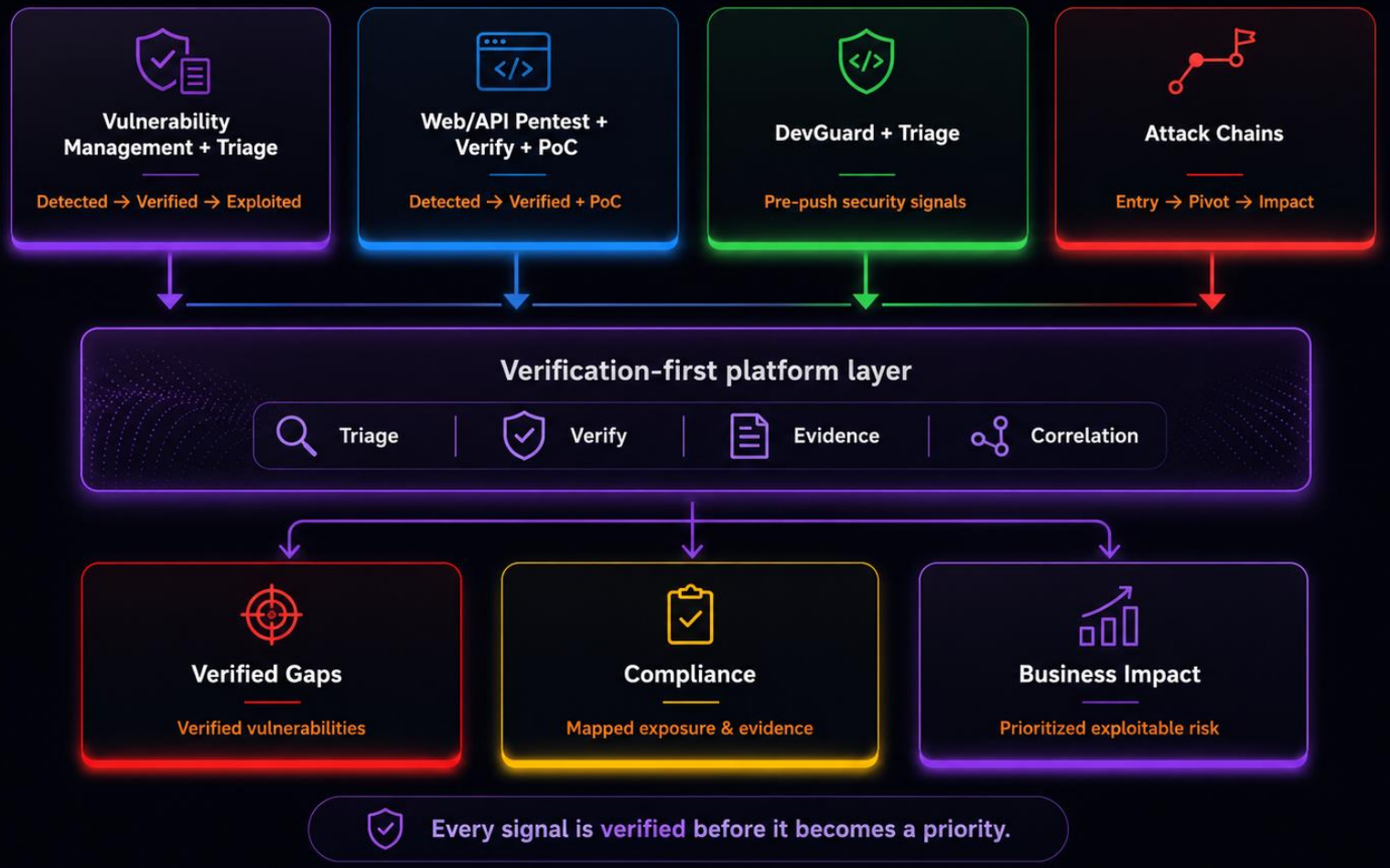
Every finding is either verified exploitable or explicitly marked why it isn't.

Pentesterra is a verification-first platform. Before a vulnerability reaches the analyst, the platform automatically attempts to exploit it. The output is not a list of potential problems - it's a list of confirmed ones.



The first platform that verifies before it reports.

Pentesterra is a verification-first offensive security platform:
it automatically attempts to exploit (safe) every finding before surfacing it to the analyst - so every alert is worth acting on.



Platform

Multiple entry points. One subscription platform.

DevGuard

Developers · Dev teams · AppSec · DevSecOps

Pre-push code, dependency, supply-chain, secret, IDE, and AI-agent risk analysis.

Fast B2C and B2B entry point before CI/CD and production.

Verify · FP Suppress

Vulnerability Management

CISO · IT Security · Cybersecurity

Internal and external vulnerability scanning for exposed assets, misconfigurations, vulnerable services, and infrastructure risk.

Expands into triage, verification, compliance, and attack-chain analysis.

Verify · FP Suppress · Triage

Web/API Pentest

AppSec · Product Security · SaaS teams

Automated DAST and API pentesting release, during compliance preparation, continuous pentest coverage.

Expands into authenticated testing, API security, business logic checks, and compliance evidence.

Verify · FP Suppress · PoC

Attack chain

Part of ANPTT module

Customers add modules, hosts, projects, workspaces, scans, users, internal nodes, compliance views, and attack-chain coverage.

All signals converge into one verified exposure platform.

Verify · Triage · PoC

DevGuard · VM · Web/API Pentest → Verified Findings → Attack-Chain Platform

DevGuard – The developer layer

See more: https://pentesterra.com/devguard

Security before code leaves the developer's machine

What DevGuard does

- DevGuard is a local pre-push security scanner for developers, AI-native teams, and AppSec workflows.
- Runs as a **CLI**, **VS Code extension**, or **CI/CD step**
- Scans locally before code reaches the repository, pipeline, or production
- Sends only **redacted metadata and findings** to the Pentesterra API
- Does **not transfer source code** or modify the project environment
- Works without exposing the user's **OpenAI, Anthropic, or other AI provider keys**
- Produces reports in **app.pentesterra.com** for the owning organization

Availability

- ✓ **CLI: pip install pentesterra-devguard**
- ✓ **VS Code extension: available from the marketplace**
- ✓ **38+ analysis modules**

Key risk categories

- **Secrets & credentials** - AWS keys, GitHub tokens, JWTs, API keys, liveness checks
- **Supply chain** - malicious npm / PyPI packages, typosquatting, dependency confusion
- **AI-agent risks** - MCP hijacking, unsafe model loading, prompt injection, risky agent configs
- **Business logic** - hardcoded roles, broken auth, insecure flags, risky configs
- **Classic AppSec** - SQLi, XSS, SSRF, vulnerable handlers before code enters production

Developer-side findings become Pentesterra signals for triage, verification, compliance, and attack-chain analysis.

Pentesterra Vulnerability Management

Continuous scanning, triage, and verification across external and internal infrastructure

What it does

- Discovers and scans external and internal assets
- Runs from Scan Wizard, scheduler, or automated rules
- Supports black-box, grey-box, and authenticated scanning
- Finds subdomains, live hosts, open ports, protocols, and exposed services
- Detects banners, technologies, protection systems, misconfigurations, and CVEs
- Supports distributed scanning with selectable scan nodes
- Applies triage statuses per host, service, port, and CVE
- Optionally verifies vulnerabilities with safe, controlled checks
- Feeds results into attack chains, compliance context, and DRSE rules

Output

- **Assets** - live hosts, subdomains, ports, protocols
- **Services** - banners, versions, technologies, exposed interfaces
- **Protection systems** - WAF, CDN, firewalls, filtering behavior
- **Risk signals** - CVEs, misconfigurations, vulnerable services
- **Triage** - Detected, Potential, Verified, Exploited, FP, Accepted Risk
- **Attack chains** - verified infrastructure paths and exposure context

Key properties

- **Internal & external scanning**
  Internet-facing assets, internal networks, dedicated nodes, and perimeter checks
- **Distributed architecture**
  Scans can run from selected nodes, customer environments, or controlled perimeters
- **Triage-first workflow**
  Findings are grouped, deduplicated, prioritized, and tracked by status
- **Verification-ready findings**
  Vulnerabilities can be safely verified before escalation or reporting
- **Attack-chain aware**
  Infrastructure findings become entry points for lateral movement, impact, and compliance analysis

VM findings become verified infrastructure signals for attack-chain analysis, compliance evidence, and remediation prioritization.

VM – how it works

From scoped infrastructure scanning to verified exposure context

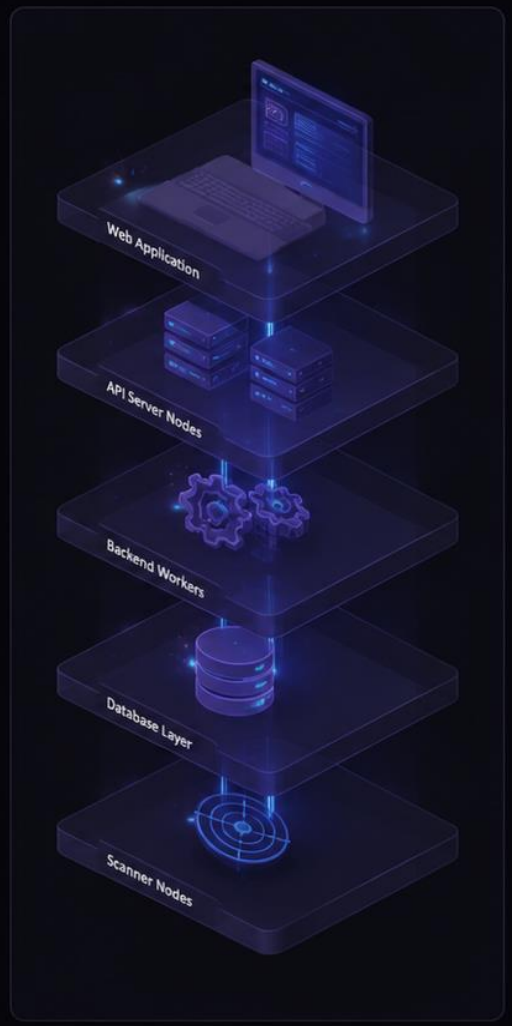
1. Define scope: targets, ports, protocols, credentials, and scan limits
2. Choose scan mode: black-box, grey-box, or authenticated
3. Select scan node: SaaS, external node, internal node, or dedicated customer scanner
4. Enable distributed scanning when multiple perimeters or environments are involved
5. Run manually, schedule recurring scans, or trigger scans through automation rules
6. Review results in the UI, reports, and organization workspace
7. Optionally generate a report and send it by email after the scan completes

After the scan

- **Asset discovery**
  Live hosts, subdomains, ports, protocols, services, and exposed technologies
- **Exposure analysis**
  Banners, versions, protection systems, misconfigurations, and vulnerable services
- **CVE mapping**
  Findings linked to CVEs, affected services, severity, and evidence
- **Triage workflow**
  Detected, Potential, Verified, Exploited, False Positive, and Accepted Risk statuses
- **Attack-chain input**
  Infrastructure findings become entry points for attack-chain analysis, compliance context, and remediation priority

Every scan result becomes structured input for triage, verification, reporting, compliance, and attack-chain analysis.

Pentesterra Platform Components



Pentesterra Platform Components

Modular architecture for continuous verification, distributed scanning, and enterprise deployment

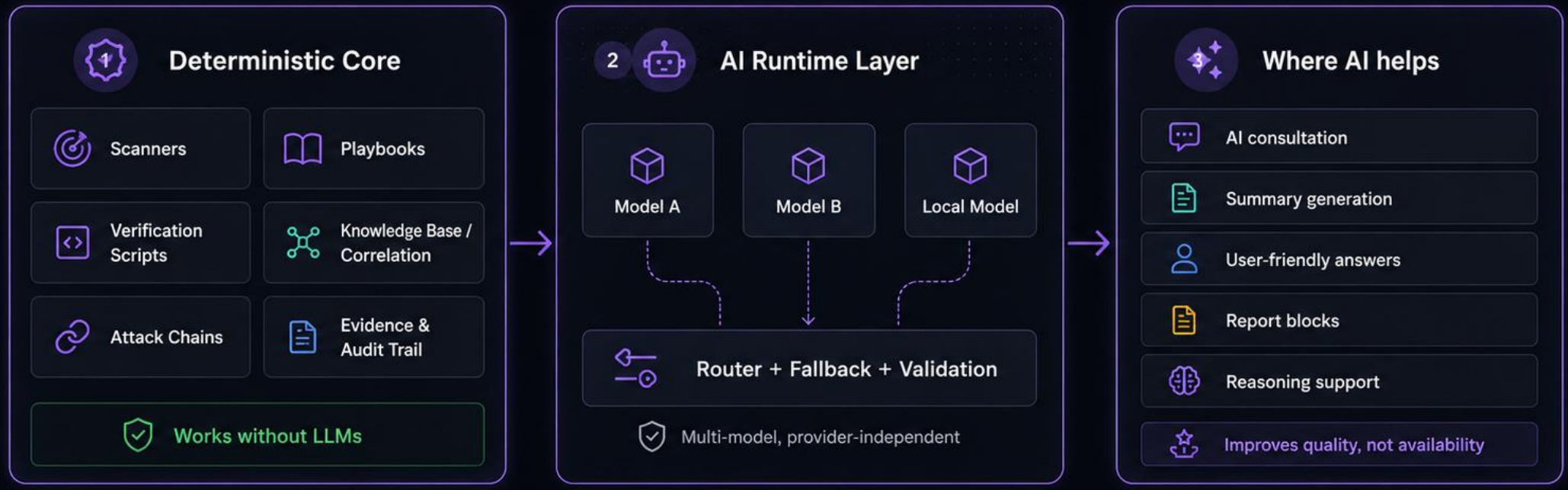
- **Web application** - UI for scans, findings, reports, triage, compliance, and attack chains
- **API server nodes** - auth, RBAC, org isolation, scanner orchestration, reporting, integrations
- **Backend workers** - KB updates, playbooks, correlation, verification, DRSE rules, AI-assisted jobs
- **Database layer** - assets, scan results, findings, evidence, reports, rules, audit trails
- **Message queue** - async communication between API, workers, schedulers, and scanners
- **Scanner nodes** - common, private, internal, and dedicated nodes for scan / pentest execution

Key properties

- **SaaS-ready multi-tenancy** - organizations, workspaces, RBAC, quotas, audit history
- **Distributed scanning** - public, private, internal, and customer-dedicated scanner nodes
- **Asynchronous execution** - long-running scans, playbooks, verification, reporting, rules
- **Extensible logic layer** - KB, playbooks, DRSE rules, verification scripts, AI-assisted analysis
- **Deployment flexibility** - SaaS, private cloud, PaaS, on-prem, and regulated environments

AI-Assisted, Not AI-Dependent

Pentesterra uses LLMs at runtime - but the core product is deterministic, auditable, and works without any AI provider.



Why not AI-first?

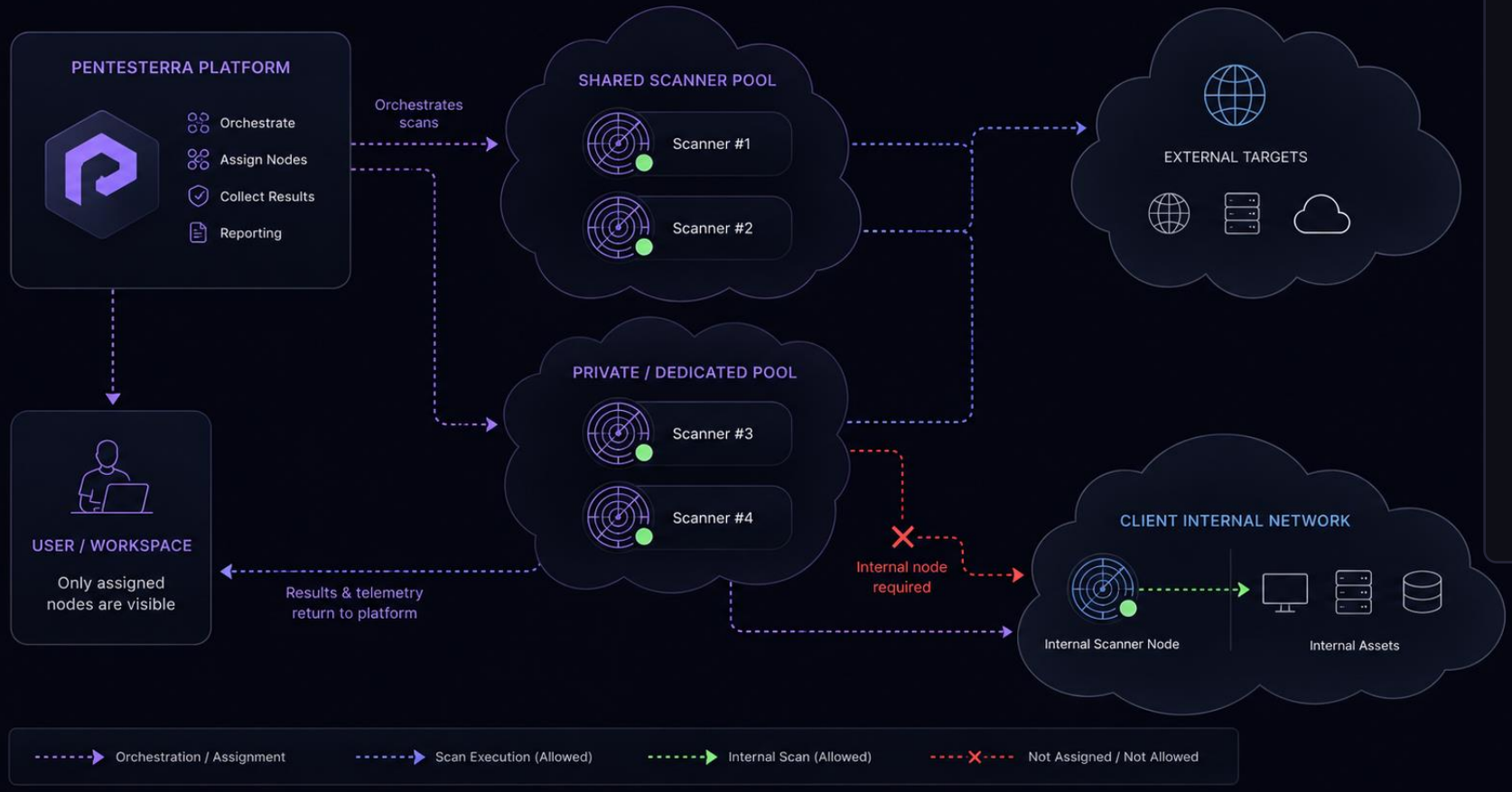
- Latency
- Provider outages
- Inconsistent structured output
- Cybersecurity refusals / blocks
- Compliance & audit constraints

Deployment options

- Deterministic fallback
- No provider lock-in
- Private / GOV local LLM option

Scanner Nodes - How Execution Works

Controlled scan execution across public, private, and internal environments



Execution model

- User selects scan scope and an allowed scanner node
- Nodes can be shared, private, or internal
- External scans run from public or private nodes
- Internal scans run only from customer-side internal nodes
- Pentesterra manages control, access, and reporting
- Results return to the platform for triage, verification, and attack-chain analysis

Pentesterra Web Application Pentest

Automated web and API pentesting with authenticated testing, flexible routing, and modular attack coverage

- Automated pentesting for **web applications and APIs**
- Supports **unauthenticated and authenticated** testing
- Can run through **direct connection, proxy, or Tor**
- Supports **login flows, session-based testing, cookies, headers, and token-based access**
- Includes **bruteforce / credential attack modules** where allowed by scope
- Lets users select from **58+ pentest modules** before the scan starts
- Tests forms, parameters, endpoints, files, auth flows, and exposed application logic
- Can optionally verify findings and produce evidence / PoC-ready output
- Results feed triage, reporting, compliance context, and attack-chain analysis

Output

- Discovered URLs, endpoints, parameters, forms, and input vectors
- Web and API findings linked to severity, evidence, and affected surface
- Authentication, session, and access-control weaknesses Injection and application-security findings across selected modules
- Verified findings and PoC-ready evidence where supported
- Attack-chain entry points connected to business and compliance context

Key properties

- **Flexible routing**

Direct, proxy-based, or Tor-routed scan execution depending on the target and engagement model

- **Authenticated coverage**

Session cookies, headers, tokens, login flows, and authenticated application areas

- **Modular testing**

Users select from 58+ modules before scan execution, depending on scope and allowed actions

- **Web + API coverage**

Traditional web flows, modern APIs, auth surfaces, forms, parameters, and business logic paths

- **Verification-first results**

Findings can be validated and turned into evidence before escalation or reporting

Web App Pentest findings become structured application-layer signals for triage, compliance, and attack-chain analysis.

Web Application Pentest - Execution flow

From scoped application testing to verified exploitable evidence

Every web/API finding becomes structured evidence for verification, reporting, and attack-chain analysis.

Key properties

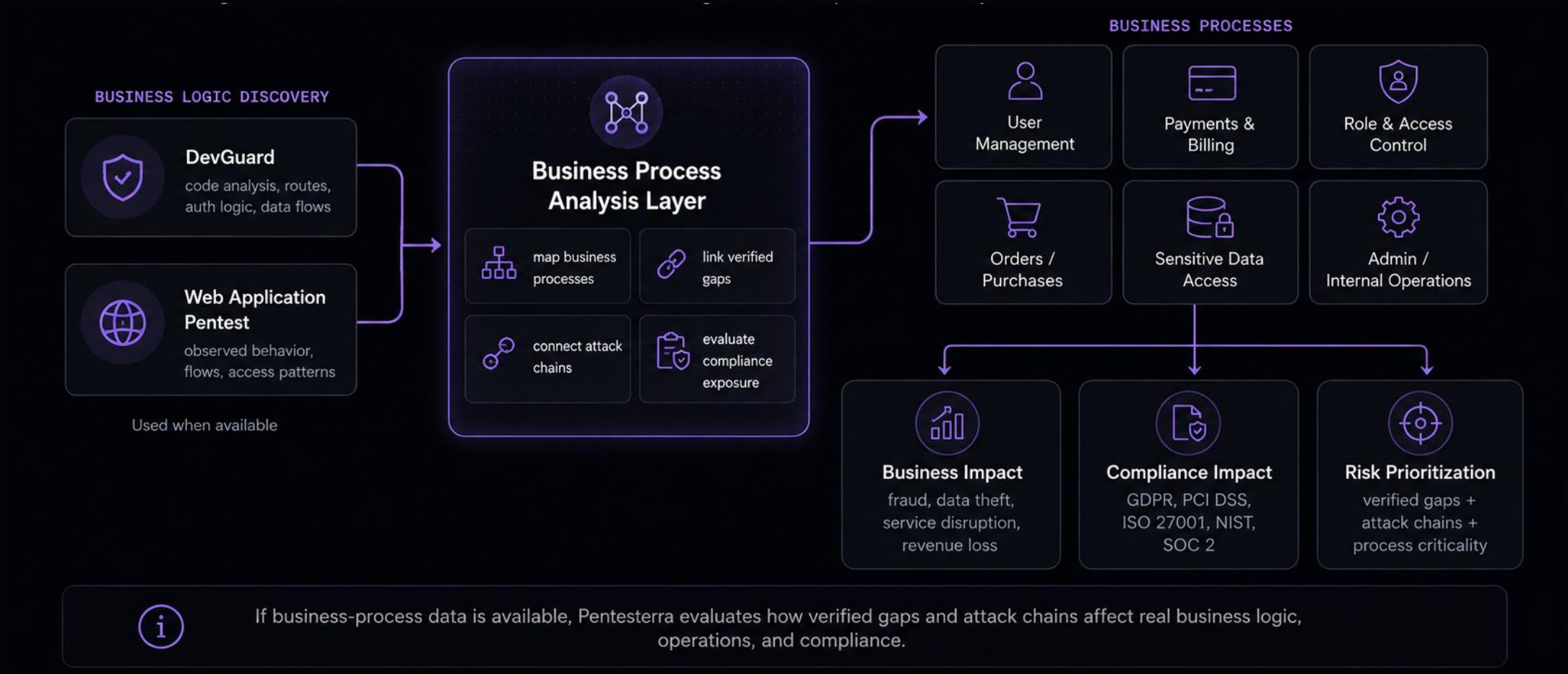
- Define target, scope, limits, and allowed actions
- Choose unauthenticated or authenticated testing
- Route traffic through direct connection, proxy, or Tor
- Select from **58+ web/API pentest modules**
- Execute scan or schedule
- Capture results & evidence: URL, payload, response, proof, severity
- Send results to triage, reports, compliance, and attack chains

Attack Chains - From Signals to Paths

From DevGuard, Vulnerability Management, and Web App Pentest to prioritized attack paths

Business Processes & Compliance Impact

When business logic is discovered, Pentesterra shifts from surface findings to business-process risk analysis.



Pentesterra - From Findings to Verified Business Risk

One platform connecting code, infrastructure, web/API exposure, attack chains, compliance, and business impact

DEMO

Contacts

Pentesterra is already discoverable through AI search - ask ChatGPT, Claude, Gemini, or Perplexity to explore the platform, positioning, and technical depth.

Verified exposure. *Not ~~more~~* noise.

All-in-One Automated Offensive Security Platform

Verify vulnerabilities, triage exploitable risk, build attack chains, and prioritize remediation by business process impact and compliance exposure across APIs, web apps, cloud, and internal networks.

Learn more: pentesterra.com
Contact: os@pentesterra.com
SaaS: app.pentesterra.com

APPENDIX: “AI free Pentest tools”?

Three Ways to Pentest

Three fundamentally different approaches - and only one that works at enterprise scale.
scale.

01

”Prompt-and-Pray AI Wrappers”

`AI WRAPPER · SCRIPT KIDDIE TIER`

Good for demos. Impressive to watch. Useless in production.

You give it a target. It does... something. Results are non-deterministic, vary every run, and can't be audited. Nobody knows exactly what actions were taken or why a finding was flagged.

✗ Unpredictable
✗ Uncontrollable
✗ Unauditable
✗ Not enterprise

02

”Scoped Pentest”

`PENTESTERRA ANPTT MODULE`

Autonomous Network Pentest with with Playbooks

Security team defines the scope (IP ranges, targets, allowed actions). The platform runs an automated pentest campaign using structured playbooks combined with AI reasoning.

✓ Strictly limited scope
✓ Playbook-driven
✓ AI cannot perform arbitrary actions
✓ Full audit trail
✓ Human-in-the-loop: approval gate

03

”Targeted verification”

`LIVE NOW: CORE PRODUCT`

Point Checks on Specific Vulnerabilities & Attack Vectors

The platform identifies a specific finding (CVE, misconfiguration, injection point) and runs a targeted, non-destructive verification script against that exact surface.

✓ Deterministic results
✓ Fully reproducible
✓ Non-destructive by default
✓ Full audit trail
✓ Scope enforcement

Why now

Three converging forces created this market - right now.

01

AI-generated code explodes the attack surface

77% of developers now use AI coding assistants daily (GitHub, 2025). AI-generated code introduces vulnerabilities at scale - SQL injection in web handlers, hardcoded credentials, insecure configurations - often without the developer knowing. No existing scanner covers the AI agent attack surface: MCP hijacking, vibe-coding risks, prompt injection in data files, AI CI/CD backdoors.

02

Supply chain is one of the hottest attack vectors in Q2 2026

Open-source supply chain attacks grew 742% over three years (Sonatype, 2023). npm and PyPI package compromises are up 300%+ YoY. DevGuard is the only pre-push scanner that catches malicious packages before they enter the codebase - and feeds those signals into the attack chain engine.

03

Regulatory pressure turns security from optional to mandatory

NIS2 (EU, 2024), SEC cybersecurity disclosure rules (US, 2024), DORA (EU financial sector, 2025) - all require continuous monitoring and verified evidence of security posture. Pentesterra generates exactly that audit trail: who found what, when, verified how, remediated by whom.

Why LLM Wrappers Don’t Replace Pentesterra

We use AI where it wins. We use deterministic logic where it doesn’ t. The product works without any specific AI provider - and gets better as models improve.

01

Demo ≠ Production

Every "Claude found a vulnerability" headline is a research paper - run on disclosed CVEs with full source code access. In production, you scan black-box targets: live networks, unknown assets, no source. That's a fundamentally different problem.

02

Context window can't cover enterprise attack surface

A medium enterprise has 500+ assets, thousands of endpoints, hundreds of config files. You can't fit the attack surface into any context window - and even if you could, signal drowns in noise. Before an LLM can reason about anything, you need structured discovery, enumeration, and prioritization. That's what Pentesterra does.

03

Pentesting is execution, not prediction

Finding SQL injection means sending 40 payload variants, measuring response timing, observing error codes, comparing against a baseline - on a live system. LLMs predict the next token. They cannot run a payload against a real target and verify whether it worked. You need a deterministic execution engine. We built one.

04

The economics don't work

Continuous monitoring via LLM API: 1,000 assets × 100K tokens × daily scans = $300–1,500 per scan cycle. Enterprise security runs 24/7 across hundreds of systems. That's $100K+/month in tokens alone - before false positives, retries, and human review. Pentesterra uses LLMs surgically, where deterministic logic underperforms. Not as the engine.

05

The AI said so fails compliance

When a CISO presents to the board or responds to a regulator, they need: what was tested, what method, what payload, what response, when, who authorized it. Pentesterra generates that audit trail. An LLM wrapper generates a chat log. NIS2, DORA, and SOC 2 auditors don't accept chat logs as evidence.

Why We Win

Not assembled from tools. Built from first principles.

Not a wrapper

Proprietary DAST engine, triage system, and attack-chain correlator - built from scratch since 2021. Not Kali Linux wrapped in a UI. Not OpenAI calls over nmap console output. Every detection module is owned code.

Cross-layer correlation

Network scanner + automated web/API pentest + DevGuard pre-DevGuard pre-push signals, fused into one chain engine. No other No other platform connects all three layers. Competitors cover one cover one layer at a time - you still have to correlate manually. manually.

GOV-grade from day one

Full on-premise and air-gapped deployment available now. Not a roadmap item - live in production with government pilots. SaaS, private cloud, and air-gapped installations run on the same codebase.

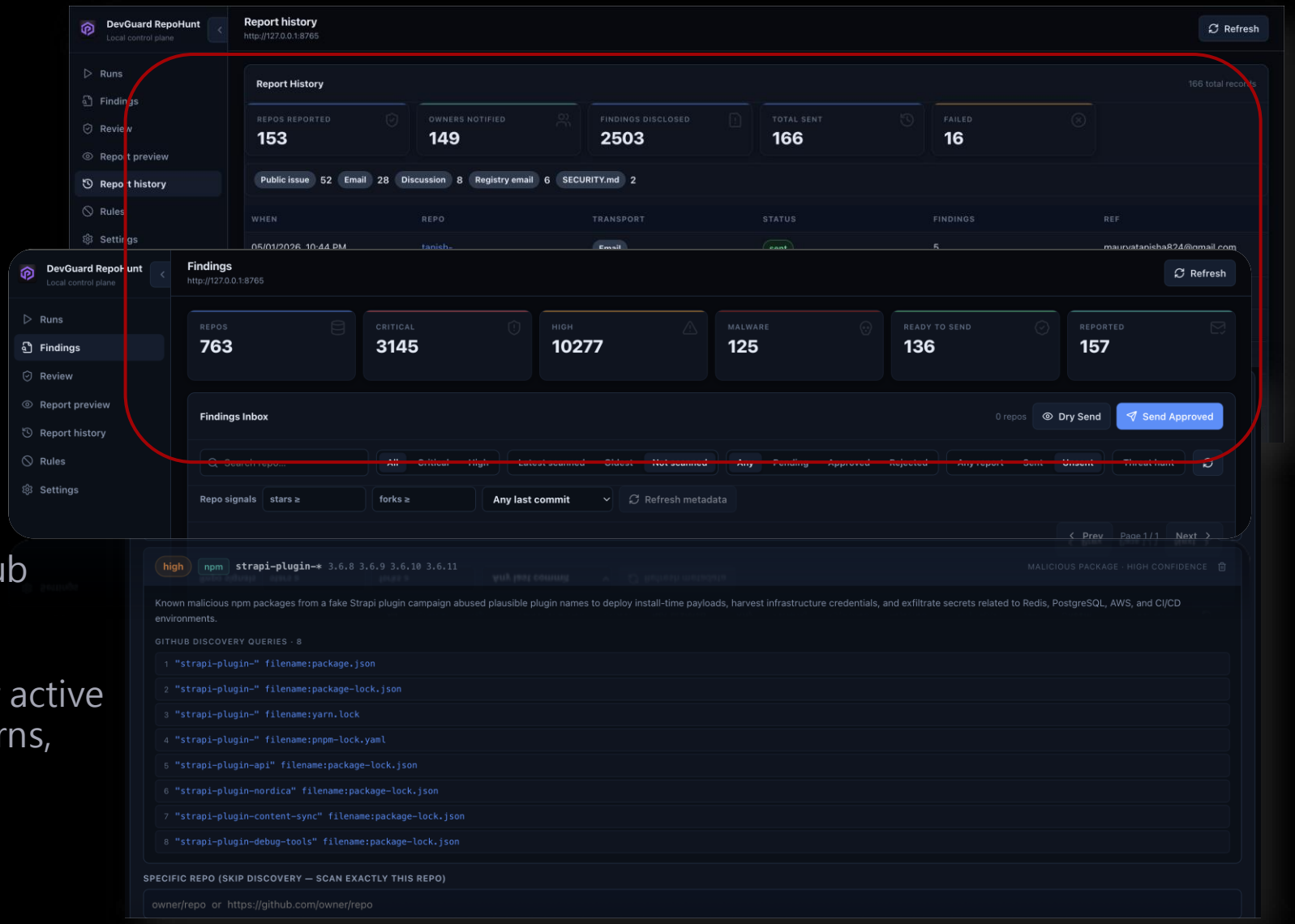
Verification-first architecture

Competitors report findings. We verify them. Findings move from Detected → Verified → Exploited *safe before the analyst sees them. 82% noise reduction is the default, not a premium feature.

APPENDIX: CASE STUDY

Case Study: DevGuard RepoHunt.

From product capability to ecosystem-level distribution



A separate internal control plane for GitHub threat hunting and responsible disclosure.

RepoHunt searches public repositories for active supply-chain and mass-exploitation patterns, clones candidate projects, runs DevGuard analysis, and prepares human-reviewed disclosure reports.

Case Study: The forgotten external domain that opened the internal network

From product capability to ecosystem-level distribution

During an external assessment, Pentesterra followed an outbound link from the target's web application to a forgotten third-party file-hosting domain.

The domain was no longer in the client's asset inventory. It had lapsed SSL, an exposed admin panel, and a directory traversal vulnerability. Exploitation exposed a path to a VPN gateway reachable from that host.

Pentesterra connected the full chain in one graph:

Web app link → forgotten domain → exposed admin panel → directory traversal → VPN gateway → internal network pivot

This is the core Pentesterra value: not another scanner finding, but a verified exposure path showing what an attacker can actually reach.

Summary

Access to internal network

Asset in inventory: No

Initial signal: External web link

Finding type: Forgotten third-party asset

Chain impact: Internal network pivot confirmed

Case Study: Public Google Drive folder exposed 200+ internal documents

From product capability to ecosystem-level distribution

A government client's public website linked to a Google Drive presentation. The link was configured as a folder-level share, not a single-file share.

Pentesterra followed the external link, expanded the shared folder scope, extracted the file listing, and flagged the exposure as critical.

The folder contained 200+ internal documents, including strategic plans, budgets, personnel records, third-party contracts, and project briefings - readable by anyone with the link, with no authentication required.

This is why exposure management cannot stop at scanners. Real business risk often sits in forgotten links, cloud shares, and unmanaged external dependencies.

Summary

TOP SECRET documents

- 200+ files exposed
- Auth required: None
- Exposure age: Years
- Contents: budgets · personnel · contracts · strategic plans

Case Study: Malicious IDE extension injecting reverse shell

From product capability to ecosystem-level distribution

A VS Code extension used for “ code formatting improvements” was silently injecting a reverse shell payload into JavaScript files on save.

The extension came from the VS Code Marketplace and had a 4.2-star rating. It did not trigger traditional SAST or dependency alerts because the malicious behavior lived in the developer environment, not in the application dependency graph.

DevGuard detected the extension by analyzing installed IDE plugins for filesystem write behavior, outbound network signatures, and code injection patterns.

The chain was clear:

Install extension → modify JS files on save → inject reverse shell → developer machine compromise

This is the AI-era development risk Pentesterra is built to catch before code reaches CI/CD or production.

Summary

Malicious IDE extension

- Vector: IDE extension
- Trigger: *.js file save
- Payload: Reverse shell
- Impact: Developer machine compromise
- Detected by: DevGuard extension analysis