DI - ASH AI Pentest Platform

BEGINNER GUIDE | FEB 11, 2026

Full Technology Explanation for a New Developer

This presentation explains your platform in plain language: what each technology does, why it was chosen, how secure your setup is, and which open-source alternatives you can consider.

Audience: Project Owner, Senior Management, CISO

BIG PICTURE

What This Platform Does

1) Runs security scans

You submit a target domain/IP. The platform runs Nuclei, Nikto, SQLMap, and Katana with resilient SQLMap preflight/discovery logic.

2) Stores results

Each scan is saved with status, logs, findings, and timestamps in SQLite and report files.

3) Explains findings

A local AI model converts tool output into business-friendly summaries and remediation steps.

Simple workflow

1. User starts scan from dashboard.
2. Backend queues and runs tool process.
3. Raw output/logs are captured.
4. Reports are generated (executive, technical, combined, compliance summary).
5. AI chat explains findings by scan ID or security topic.

Core idea: automate pentest execution and reporting in one place

STACK SUMMARY

Current Technology Stack (What + Why)

Layer	Current Tech	Why This Was Used	What It Helps You Do
Frontend	HTML + CSS + Vanilla JS	Fast to build, no framework overhead, easy browser compatibility.	Dashboard UI, scan trigger, status tracking, report navigation, chat controls.
Backend API	FastAPI + Uvicorn	Modern Python API framework with clean route design and async support.	Authentication, scan orchestration, report generation, AI/chat endpoints.
Database	SQLite (`data/pentest.db`)	Simple, lightweight, no server setup, easy backup for single-node deployment.	Stores users, scan jobs, statuses, outputs, report references.
Pentest Engines	Nuclei, Nikto, SQLMap, Katana	Each tool covers different attack surfaces and complements others.	Vulnerability checks, header/misconfig checks, SQLi tests, endpoint discovery.
AI Analysis	llama.cpp + GGUF model	Runs locally; good for cost control and data privacy vs cloud AI.	Natural-language explanations and risk narrative by scan findings.
Reporting	HTML template pipeline	Readable, shareable, printable, and easy to customize for stakeholders.	Executive, technical, and combined reports with findings/remediation context.

This is a practical and low-cost architecture for an internal security platform

FRONTEND (UI)

Frontend: What Happens in the Browser

Main responsibility

- Collect scan input (target + tool choice).
- Show scan history and live status updates.
- Open reports and chat with AI assistant.
- Render dashboard cards, charts, and search.

Why vanilla JS here?

- No framework learning curve for quick iteration.
- Fewer dependencies means lower maintenance overhead.
- Works well for moderate UI complexity.
- Good for controlled internal app scope.

Open-source alternatives

React + TypeScript

Best if UI becomes much larger and component-heavy.

Vue 3

Good middle ground: easier than React for many teams.

HTMX + server templates

Very simple dynamic UI with minimal JS.

Current choice is fine; move to React/Vue only when complexity grows

BACKEND + DB

Backend API and Data Layer

FastAPI + Uvicorn

- Defines API routes such as scan, history, report, chat, and health.
- Runs asynchronous tasks for long scan/report operations.
- Handles authentication and response formatting.
- Coordinates subprocess execution of security tools.

SQLite database

- Single file database; easy to deploy and backup.
- Stores scan metadata and report references.
- Great for low to medium load, single-node setups.
- Less ideal for high concurrency or distributed scale.

Alternatives

PostgreSQL

Best open-source upgrade when you need higher scale, concurrency, and resilience.

Django

Good if you want batteries-included auth/admin and larger app structure.

Flask

Lighter than FastAPI but less built-in validation and async ergonomics.

For your stage, FastAPI + SQLite is a strong starting point

PENTEST ENGINES

Why You Use Multiple Security Tools

Tool	Primary Role	Typical Output	Value to Business
Katana	Endpoint discovery and crawling	Reachable paths, parameters, linked endpoints	Shows attack surface that may be unknown to teams
Nuclei	Template-based vulnerability checks	Known exposure hits by template/signature	Fast detection of common weaknesses at scale
Nikto	Web server misconfiguration review	Missing headers, risky files, outdated setup indicators	Improves baseline web hardening posture
SQLMap	SQL injection testing	Injection evidence or clean result	Protects critical data from database compromise risk

Alternative scanners (open source)

OWASP ZAP (great for web app active/passive testing) and Greenbone/OpenVAS (network vulnerability management) can be added for deeper coverage depending on scope.

One tool cannot cover everything; layered scanning gives better assurance

AI ANALYSIS

Local AI Engine: Why It Matters

Current implementation

- `llama.cpp` serves/executes local GGUF models.
- Primary model: Qwen 2.5 3B quantized for speed.
- Fallback models available for resilience.
- Chat can explain findings by scan ID and general security topics.

Why this is good

- No per-request cloud AI cost.
- Better control over sensitive scan data.
- Works offline in restricted environments.
- Predictable latency on your own hardware.

Backup model segregation recommendation

Primary Qwen 2.5 3B (best quality/speed balance for your setup) Backup TinyLlama or Mistral-7B-Q4 (fallback when memory/load changes). Use automatic failover when primary is unavailable.

REPORTING

Report Types and When to Use Them

Executive Report

- High-level risk summary.
- Business impact and priority.
- Best for senior management.

Technical Report

- Evidence and detailed findings.
- Coverage, methodology, remediation.
- Best for security engineers.

Combined Report

- Single document for all stakeholders.
- Useful for external sharing and audits.
- Good for board-to-technical alignment.

Compliance Summary

- Maps findings to ISO 27001, SOC 2, NIST, OWASP, CIS, and UAE IAS themes.
- Includes heat map, posture snapshot, and remediation appendix.
- Best for audit and governance stakeholders.

Why HTML pipeline works well

HTML is easy to style, inspect, print to PDF, and version-control. It also lets you tailor output for different audiences without changing scanner logic.

SECURITY STATUS

How Secure Is Your Current Setup?

Good controls already present

Auth · Password hashing · Security headers · Scan timeout handling · Operational logging

- Role-aware auth and step-up action password are enforced for sensitive actions.
- Password hashing implemented with bcrypt.
- Headers like CSP, nosniff, frame deny, HSTS are set.

Important hardening still recommended

CORS currently permissive · Credential governance · Single-node DB limits

- Restrict CORS to trusted frontend origins only.
- Keep strict password rotation and admin secret governance.
- Use reverse proxy TLS and network segmentation.
- Move to PostgreSQL when scaling users/automation.

Security position today

This is a strong internal prototype and pilot platform. With targeted hardening, it can move to production-grade operations.

OPEN-SOURCE ALTERNATIVES

Could Anything Be Better Than Current Choices?

Layer	Current	Alternative	When Alternative Is Better
UI	Vanilla JS	React / Vue	Better for very large, complex, component-heavy frontend apps.
DB	SQLite	PostgreSQL	Better for multi-user concurrency, scaling, and reliability controls.
AI Runtime	llama.cpp	Ollama / vLLM	Ollama is simpler model management; vLLM is stronger for high-throughput GPU serving.
Scanning	Nuclei + Nikto + SQLMap + Katana	Add ZAP / OpenVAS	Better when you need deeper app testing or broader network VM workflows.
Reporting	HTML templates	Sphinx / MkDocs / Pandoc pipeline	Better for very formal document governance and documentation versioning.

Current stack is strong; alternatives are mainly scale and governance upgrades

WHAT TO TELL LEADERSHIP

Presentation Script for Senior Management and CISO

Suggested 60-second talk track

"We built an AI-assisted pentesting platform that runs multiple open-source scanners, stores evidence, and generates both executive and technical reports. It already improves speed and consistency of risk visibility. The platform runs a local AI model, so sensitive findings stay in our environment with no per-call AI cost. Current build is strong for controlled internal operation. We have a defined hardening roadmap to reach production readiness: tighten CORS, enforce credential controls, and scale data services when user volume grows."

Business value points

- Faster detection-to-report cycle.
- Better remediation clarity for teams.
- Lower external tooling and AI cost.

Risk governance points

- Traceable findings and evidence outputs.
- Alignment with OWASP, NIST testing style, ISO risk treatment.
- Roadmap for production hardening and control maturity.

Lead with business impact, then show technical confidence and control plan

NEXT STEPS FOR YOU

Beginner Action Plan (Practical)

Technical priorities

- Finalize CORS policy and remove wildcard origins.
- Confirm admin password policy and secret handling.
- Run scan/report regression test on each release.
- Keep tools and templates updated monthly.

Business priorities

- Define service scope for internal and outsourced clients.
- Set SLA for scan completion and report turnaround.
- Create pricing tiers if positioning as micro SaaS.
- Track monthly KPIs: scans, findings, remediation closure rate.

Easy growth path

Keep the same architecture, harden controls first, then upgrade only the layers that hit scale limits (database, UI framework, AI serving mode).

Do not over-engineer early; scale intentionally based on usage

FINAL SLIDE

Conclusion

Your current stack is practical, cost-aware, and already valuable. The right strategy is to keep this base, harden key controls, and add backup AI + scale upgrades only when usage requires it.

If you want, this same presentation can be converted to PDF and a simplified 1-page version for board-level communication.