Case Study: Security Automation Framework for Continuous Vulnerability Management (Anonymized)

Category: Automation & Tooling
Duration: 4 Weeks | **Engagement Type:** End-to-End Security Automation Implementation
Tech Stack: Python, AWS Lambda, S3, DynamoDB, Nmap, ZAP CLI, Amass, Slack API, GitHub Actions

---

Context

A mid-sized SaaS company managing multiple production environments across AWS accounts faced a recurring challenge — **vulnerability visibility gaps**.

While periodic manual scans were performed, they lacked:

- Continuous discovery of new assets (subdomains, IPs, endpoints)
- Automated recon and scan orchestration
- Centralized vulnerability tracking and alerting

Their security operations were reactive — findings surfaced only during audits or external assessments. The goal was to build a **self-sustaining automation framework** that continuously performs reconnaissance, scans, prioritizes issues, and notifies relevant teams all without human intervention.

Approach

I designed and implemented a **serverless Security Automation Framework** orchestrating multiple open-source tools with AWS-native services for scalability and cost efficiency.

1. Asset Discovery Layer

- Integrated **Amass**, **Subfinder**, and **Nmap** to enumerate subdomains and live hosts daily.
- Scheduled recon tasks via **AWS Lambda + EventBridge** with results stored in **DynamoDB** (asset inventory).
- Used tagging to differentiate production vs staging vs sandbox assets automatically.

2. Vulnerability Scan Orchestration

- Automated **OWASP ZAP CLI**, **SQLMap**, and **custom Python scripts** to perform API and web app scans against live targets.
- Implemented **adaptive scanning logic** — endpoints that failed previous scans were skipped unless modified in GitHub (based on commit webhook).
- Each scan generated standardized JSON outputs pushed to **S3 buckets** with metadata (severity, timestamp, source tool).

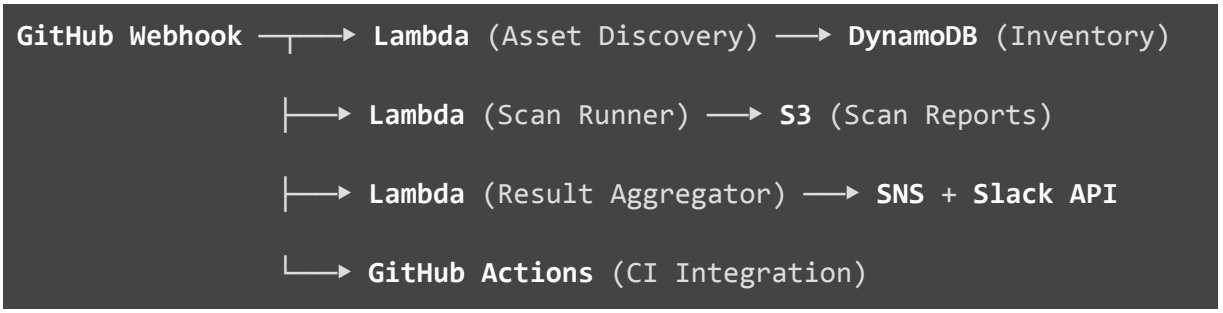
3. Reporting & Alert Automation

- Consolidated results using a **Python aggregation engine** that parsed tool outputs and generated unified CVSS-based summaries.
- Built **Slack integration** using Web API to push:
    - High/Critical findings to the “#sec-alerts” channel.
    - Daily summaries to “#sec-digest”.
- Uploaded detailed HTML reports to S3 with signed URLs for on-demand viewing.

4. Continuous Integration

- Integrated the pipeline with **GitHub Actions** — every new deployment triggered a scan for the changed endpoints.
- Added tagging to correlate vulnerabilities with specific commits and PRs.

Architecture Overview



- **Lambda concurrency:** ensures isolated execution for each target
- **S3 versioning:** preserves historic reports for audit trails
- **DynamoDB TTL:** auto-purges stale targets beyond retention period

---

Key Findings & Metrics

Metric	Before	After Implementation
Scan Frequency	Manual (once a month)	Automated daily
Asset Coverage	~40%	~95% (dynamic discovery)
Detection Lag	3–5 days	< 1 hour
Alert Delivery	Email-based	Real-time Slack alerts
Reporting Time	4–6 hours	10 minutes automated

Security & Compliance Impact

- Reduced **vulnerability detection latency** by **>85%**.
- Introduced **asset-driven automation**, eliminating blind spots.
- Enabled **continuous audit readiness** — exports aligned with ISO and OWASP compliance mapping.
- Provided auditable scan artifacts and evidence storage.

Executive Summary

This engagement transformed a fragmented, manual security testing process into a **continuous, automated vulnerability management ecosystem**.

By combining open-source scanning tools with AWS automation and real-time reporting, the client gained:

- Full visibility of their attack surface

- Proactive detection of critical vulnerabilities
- Near real-time alerting through existing communication channels
- Measurable reduction in both effort and response time

The system continues to run autonomously today — scaling dynamically with their infrastructure and keeping the security posture under constant assessment.

Deliverables

- Infrastructure-as-Code (Terraform + Lambda deployment templates)
- 3x AWS Lambda functions with modular scanning logic
- Slack notification handler + CVSS-based report formatter
- DynamoDB schema for asset tracking
- PDF and HTML vulnerability summary templates
