Secure Code Training

May/June 2023

Introductions - StackHawk Team

Scott Gerlach
CSO, Co-Founder

Holly Hamann
SVP, Marketing

KC Berg
Chief Architect

Casey Cline
Customer Success

Zachary Conger
Solutions Architect

Alexa Sevilla
Product Marketing

Eric Potter*
Solutions Architect

Presenter

Eric Potter
Sr. Solutions Architect

Husband, Father, Hacker

- 20 years experience in information security
- 15+ years penetration testing and security assessments, including classified networks, industrial controls systems, web applications and APIs for USG and Fortune 50 companies
- 2+ years of hands-on cybersecurity teaching experience with UTulsa, CWRU, and UPenn

Agenda

- Intro to StackHawk & Shift Left
- Secure Coding with OWASP Top 10
  - Hands-on Exercise
- StackHawk DAST Overview
- Q&A

Day 2 - Sneak Peek:

Finding/Fixing Auth'd vulns

Day 1: StackHawk Intro

StackHawk

- Created in 2019 to revolutionize automated security testing
- First implementation and industry leader of Shift-left DAST
- ***Only*** DAST solution that runs ***in*** CI/CD
- Only Developer-first, Security-trusted DAST solution
- Industry leader in API security testing:
  - REST
  - GraphQL
  - SOAP
  - gRPC
- Industry leader in DAST speed, depth, and quality*

Security Testing for Teams that Deploy Software Everyday

The only DAST API security testing tool that runs in CI/CD, enabling developers to quickly fix security issues before they hit production

when deployed properly

Maya + StackHawk

maya

Use Case
Operate Efficiently

Industry
Financial Services

Company
Maya

Location
Philippines

HAWKSOME CUSTOMER SUCCESS STORY

Maya partners with StackHawk to automate DAST for web and API security testing

BACKGROUND

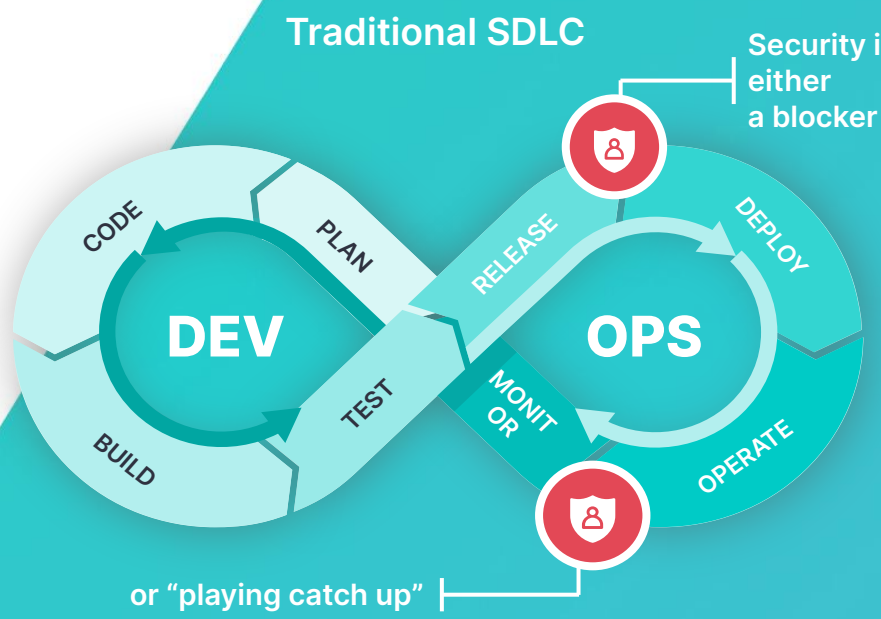
As a heavily security regulated company by the local Filipino government, Maya is invested in the security of their platform and adhering to PCI-DSS compliance policies and procedures.

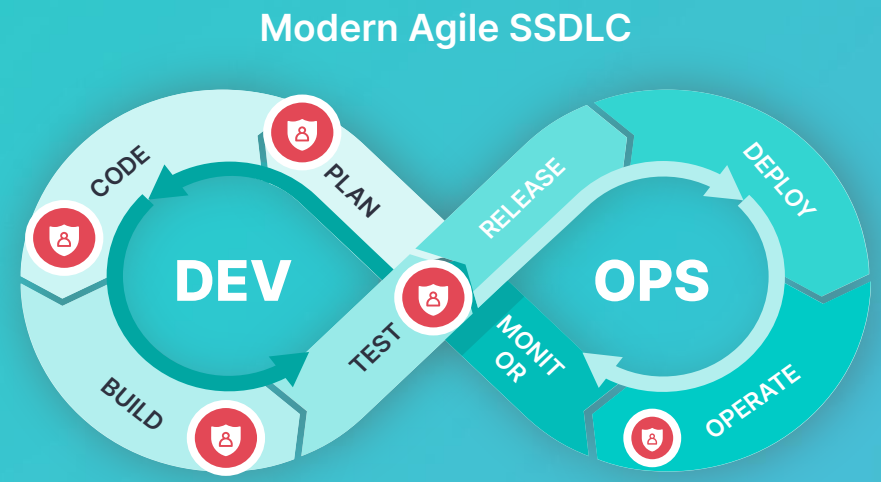
With their previous DAST solutions, the team ran into operational inefficiencies such as long scan times, high false positives, manual testing, and overall business delays, which also increased the potential for risk of bug/vulnerabilities undetected for periods of time. "Our scan times range from 20-30 minutes with StackHawk, compared to 20 minutes to 2 hours with our previous DAST solutions."

StackHawk and “Shift Left”

What/Why Is “Shift Left”?

- Secure Software Development Lifecycle (SSDLC)
  - Classic/Agile SDLC with Security woven into and through every stage
  - Minimizes cost/time/risk of security vulns by finding and fixing during planning and implementation vs post-deploy
- Empowers and requires Engineering/Development to take a leading role in application security
  - 100:1 industry average devs to appsec
  - Developers become force multipliers for security
  - CI/CD automation becomes force multiplier
- Benefits
  - Ship features faster
  - Ship features safer
  - Maintain a predictable roadmap
  - Minimize friction between engineering and security





Major Types of Automated Security Testing

Static Analysis (SAST)

Surfaces anti-patterns in *your* codebase

- Top Shift-left Vendors:
  - Snyk Code
  - GitHub CodeQL
- Pros:
  - Specific language awareness
  - Often in-depth analysis
- Cons:
  - Hard to prioritize vs volume
  - Must support your language/framework
  - No connection between findings and runtime risk

Software Composition Analysis (SCA)

Identifies vulnerable 3rd-party library dependencies

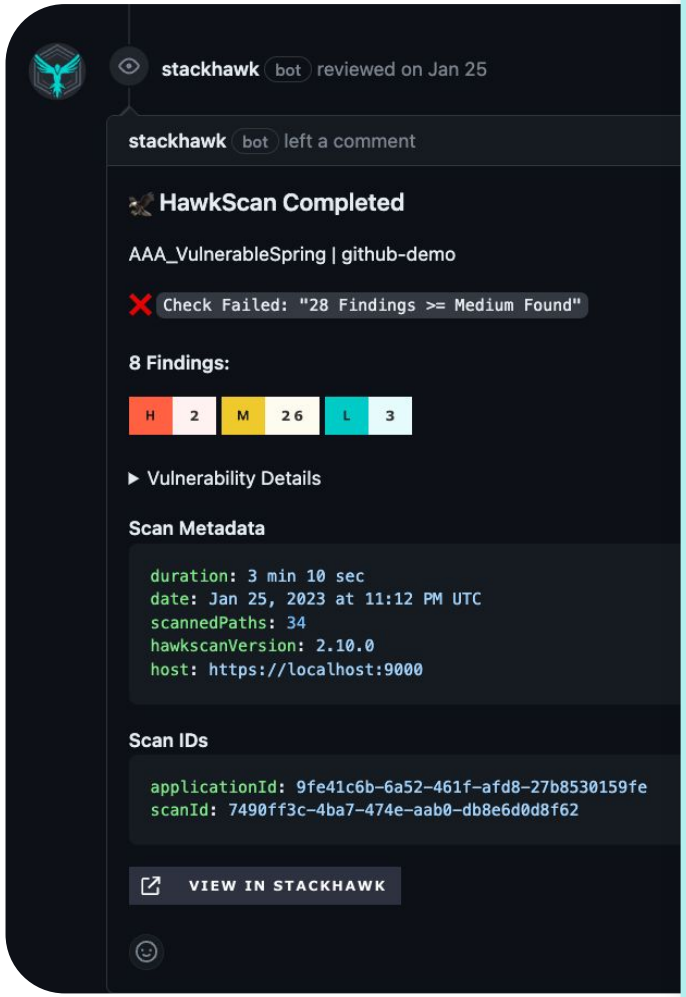
- Top Shift-left Vendors:
  - Snyk Open Source
  - GitHub Dependabot
- Pros:
  - Provides visibility into supply chain maturity
  - Can sometimes inform about licensing risks
- Cons:
  - Endless findings
  - No local control over fixes
  - No connection between findings and runtime risk

Dynamic Analysis (DAST)

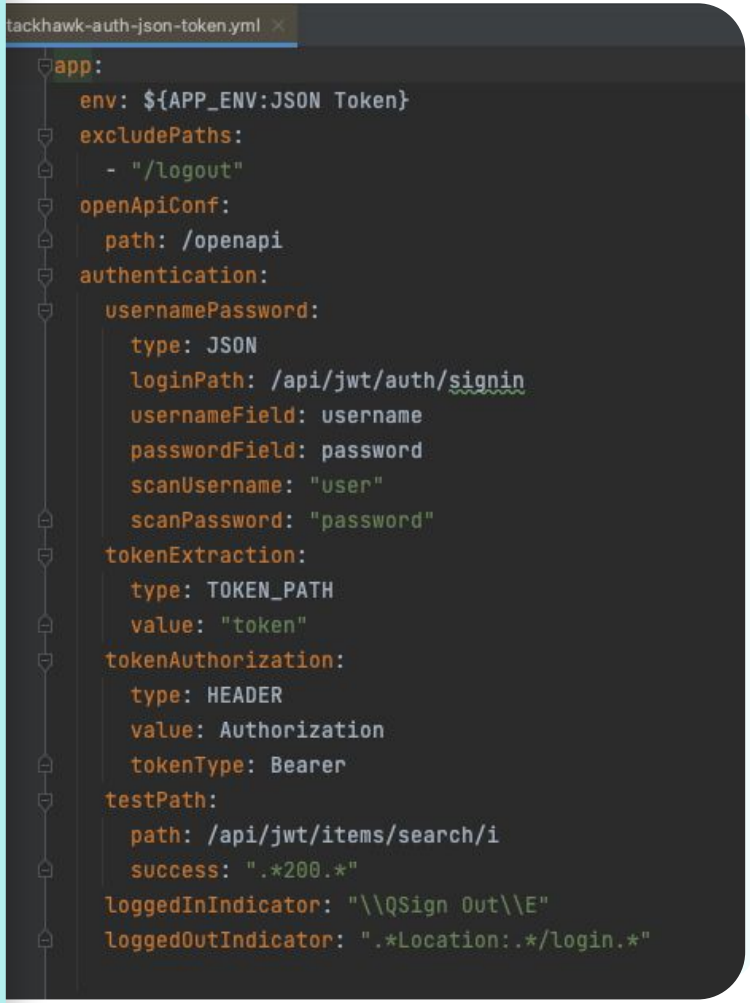
Exposes exploitable vulnerabilities and security misconfiguration in the runtime, penetration-testing style

- Top Shift-left Vendors:
  - StackHawk
- Pros:
  - Surfaces genuine runtime risk
  - High confidence / low false positives
  - Reproducible findings
- Cons:
  - Poor visibility into source code/root cause
  - Traditionally slow, but StackHawk fixes this

StackHawk Unique “Shift-Left” Superpowers



- Runs locally on-demand
  - CLI Application
  - Docker Container
  - DevOps speed
  - Results in terminal/IDE
- Runs ***in*** CI/CD, on every PR
  - Results in CI
  - Can break the build
- Configuration-as-Code
  - YAML-based
  - Maintain in git repository
  - Template w/env variables
  - Can be layered
- Findings validation/repro via cURL
- Developer-centric triage



OWASP Top 10

Who/What is OWASP?

Why should you care?

OWASP provides free online documentation and training material to assist organizations with their development of secure code.

- Secure Coding Training
- Security Testing Guidance
- Open Source Reference Applications
- Zed Attack Proxy
- Most famously, OWASP maintains the Top 10 Web Application Security Risks.

"The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members,

Per OWASP (https://owasp.org/)

OWASP Top 10

Awareness / Training / Common language

- Risks/Root Causes

History / Process

- First launched in 2003
- Data Call via social media to companies hosting applications
- 2021: ~250K applications w/ exploited vulnerabilities represented
- Hybrid model (Data + Survey)
- Data selects 8 of 10
- Survey selects 2 (emerging trends)
- For 18 of last 20 years, Injection has been #1

Testing Guidance

- Application Security Verification Standard (ASVS)
- Web Security Testing Guides (WSTGs)
- CheatSheet series

Docs:

https://owasp.org/www-project-application-security-verification-standard/

https://github.com/OWASP/ASVS

https://cheatsheetseries.owasp.org/index.html

- Application Top 10:

  https://owasp.org/www-project-top-ten
- API Top 10:

  https://owasp.org/www-project-api-security
- Mobile Top 10:

  https://owasp.org/www-project-mobile-top-ten

Current Top 10 Risks & Root Causes

Application Top 10

(2021)

- **A01**: Broken Access Control
- **A02**: Cryptographic Failures
- **A03**: Injection
- **A04**: Insecure Design
- **A05**: Security Misconfiguration
- **A06**: Vulnerable and Outdated Components
- **A07**: Identification and Authentication
- **A08**: Software and Data Integrity Failures
- **A09**: Security Logging and Monitoring Failures
- **A10**: Server-Side Request Forgery

API Top 10

(2019)

- **API01**: Broken Object Level Authorization
- **API02**: Broken User Authentication
- **API03**: Excessive Data Exposure
- **API04**: Lack of Resources & Rate Limiting
- **API05**: Broken Function Level Authorization
- **API06**: Mass Assignment
- **API07**: Security Misconfiguration
- **API08**: Injection
- **API09**: Improper Assets Management
- **API10**: Insufficient Logging & Monitoring

Mobile Top 10

(2016, 2023 Update in progress)

- **M1**: Improper Platform Usage
- **M2**: Insecure Data Storage
- **M3**: Insecure Communication
- **M4**: Insecure Authentication
- **M5**: Insufficient Cryptography
- **M6**: Insecure Authorization
- **M7**: Client Code Quality
- **M8**: Code Tampering
- **M9**: Reverse Engineering
- **M10**: Extraneous Functionality

Using OWASP: Walkthrough

Demo Summary:

- Drill into Injection Top 10 entry
- Examine artifacts available for secure coding/testing guidance

Activity Summary:

- Pick an entry from Top 10 to research
- Review some available docs
  - How is it tested?
  - How can it be avoided?
- Review

Hands-On

StackHawk DAST

Using StackHawk: Guided Tour

Demo summary:

- Live scan
- Output in CI
- Reproducing, Remediating, Triaging
- Hawkdocs (docs.stackhawk.com)
- API Docs (apidocs.stackhawk.com)

Day 2 Prep

Clone

github.com/kaakaww/live-training

And

docker-compose build

Q&A

Secure Code Training

May 2023

Day 2

- Getting Started with HawkScan
  - Hands-On Exercise
- Testing Beyond Access Controls with StackHawk
  - Hands-On Exercise
- Fixing Injection & Rescan
  - Group Exercise
- Q&A

Getting Started with HawkScan

Using StackHawk: HawkScan and JSV

Demo summary:

- Walk through the curriculum repo
- Review procedures for launching JSV
- Look briefly at JSV features
- Review procedures for launching HawkScan
- Review config personalization (env:)

Activity Summary:

- Verify installed training material
- Verify installed hawkscan and API key
- Personalize configuration with **env** string
- Verify baseline (4-line) scan runs locally
- Explore documentation or SH API
- Review

Hands-On

AuthN/Z With StackHawk

2 Perspectives

Testing Authorization and Access Controls Directly

- IDOR/BOLA
  - Insecure Direct Object Reference
  - Broken Object Level Authorization
  - These are the same thing
  - #1 API Vulnerability in 2021+
- Highly application & business-logic specific
  - Difficult to test for in a generic/automated way
  - StackHawk provides the functionality to write custom logic tests using Javascript or Kotlin

Docs:

https://docs.stackhawk.com/hawkscan/configuration/custom-test-scripts.html#custom-test-scripts-configuration

Blog:

https://www.stackhawk.com/blog/scanning-with-custom-test-scripts/

Testing through Authorization to Access-Controlled Endpoints

- Authentication type
  - HTTP/NTLM
  - FORM/JSON
  - External
  - Script (custom)
- Authorization type
  - Cookies
  - Header
  - Script (custom)
- Test automation controls
  - testPath
  - loggedIn/Out Indicators
  - antiCsrfParam
  - excludePaths

Docs:

https://docs.stackhawk.com/hawkscan/authenticated-scanning/

Principles of Automated Authenticated Testing

1. **Test in lower environments / Simplify Auth**

StackHawk has been designed to allow deployment of the scanner _close_ to the running application environment

- Local laptop, Cloud Server, CI Pipeline
- Avoid infrastructure (WAF, API Gateways, VPNs, Firewalls)
- Lower environments don't have the same requirements as production - Use that to your advantage

2. **Understand testPath (iterate quickly)**

- Unit/Functional test for AuthN/Z config
- Allows fast failure and rapid iteration
- Supports multiple test combinations
  GET/POST, HEADER/BODY, SUCCESS/FAIL, REGEX

3. **Leverage loggedIn/Out Indicators**

- Java regex triggers for automated re-auth if necessary

4. **Use excludePaths to avoid logout**

```
8   authentication:
9     loggedInIndicator: "\\QSign Out\\E"
10    loggedOutIndicator: ".*Location:.*/login.*"
11    usernamePassword:
12      type: FORM
13      loginPath: /login
14      loginPagePath: /login
15      usernameField: username
16      passwordField: password
17      scanUsername: user
18      scanPassword: password
19    cookieAuthorization:
20      cookieNames:
21        - "JSESSIONID"
22    testPath:
23      path: /search
24      success: ".*200.*"
```

Using StackHawk: AuthN/Z

Demo summary:

- Walk through documentation for auth
- Look at some YAML examples in JSV repo
- Walk through setup of an auth layer
- Scan
- Quick look at kaakaww/examples

Activity Summary:

- Select JSV authentication type
- Add authentication layer to scanner config
- Complete auth’d scan
- Review Findings

Hands-On

Using StackHawk: ReScan

Group walk-through summary:

- Find/Fix Injection vulnerability
- Conduct reScan
- Examine delta in StackHawk UI

Q&A