RAPID7

AppSec Best Practices for Enterprise Companies Webinar

and building a self service portal

Tim Honker & David Howe

3/26/2020

Speakers

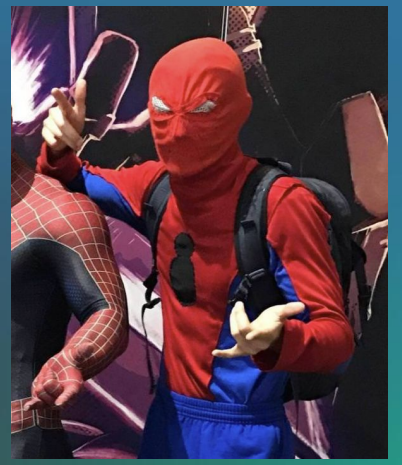
Peter Martin
Sales Manager

Enterprise team - US West

Tim Honker, CISSP
Lead Solutions Engineer

Enterprise Team - US West

former developer



Dave Howe, CISSP ITIL v3
Lead AppSec Specialist

Security Researcher

Purpose & Agenda

Advise maturing AppSec programs on how to move to a mature state

1. State of AppSec
2. How to move past common AppSec program pain points
3. Running a quality and effective Enterprise AppSec program
4. Summary
5. Q&A

State of AppSec

Trends we're seeing in the industry - Q1 2020

1. Continued adoption of modern technologies, tools can't keep up
2. Friction in remediation process with devs
3. AppSec tools delivering high false positive/false negative rates
4. Increasingly complex API auth
5. Security teams underestimate time to onboard new apps
6. Security teams lack accurate list of apps and their tech stacks
7. Half of enterprises claim the application is attacked more frequently than the network

Observations: Immature AppSec programs

- Dev/Security relationship: **Tense**
- **T**rouble proving AppSec value to dev teams - "powerless"
- Focused on scanning production quarterly/annually or just before major releases
- Reporting by individual scan
- Manually emailing scan reports
- Lack AppSec specialist

Observations: Maturing AppSec programs

- Dev/Security relationship: **Serviceable**
- Friction around remediation process
  - Frequent back and forth between security & dev for small things
- Reporting: by app level- combined web app and API findings
- Overlapping knowledge between security/dev: Mild-Moderate
- Security integrated into CI/CD pipeline

Observations: Mature AppSec programs

- **Business owners are invested in security**
- Dev/Security relationship: Symbiotic
- Accurate App Catalog
- Developers know something is serious if security team is talking to them
- Largely automated using AppSec tool APIs
- Releases are mundane
- Reporting on several levels, by business unit, trending
- Self service portal for developers

AppSec Tool Categories Rundown

SAST
Evaluate source code

Benefits

- Fix vulns at their source
- No effect on production environment
- Encourage good code hygiene
- Valuable for secure development

Drawbacks

- Language & framework specific
- False positives
- Challenge of deploying technology at scale/Dev adoption
- Not a substitute replacement for building apps with security in mind
- Doesn't run during runtime

DAST
Scan in run environment

Benefits

- Provides broad coverage with low F+/F- rates
- Not bound to particular server-side technology or language
- Acts like an attacker
- Finds vulns SAST tools cannot
- Integrates into the CI/CD process

Drawbacks

- Devs have trouble retesting
- May not support client-side JS or authentication method
- Extra lift to verify coverage
- Can be slow/inefficient to crawl large sites without additional configuration

IAST
Interactive Testing

Benefits

- Identify vulns down to the line of code
- Improve accuracy of SAST solutions by incorporating results from runtime

Drawbacks

- Not yet widely adopted due to cost
- Requires expertise
- Lack of combined coverage across certain languages and frameworks
- Requires deployment of an agent

WAF
Block malicious traffic

Web Application Firewall

Benefits

- Detects/blocks attackers in production
- Doesn't intrude on development process
- Band-aid to block exploitation while remediating
- Can absorb DDoS attacks

Drawbacks

- Alert fatigue
- Constant need to maintain rule list
- Can be evaded using encoding

RASP
Monitor/Block suspicious behavior inside the app

Real-time Application Self Protection

Benefits

- Detects/blocks attackers in production
- Catches the most advanced attacks that evade other AppSec tools
- Some RASPs can block zero-days
- Fewer false positives
- Hard to evade
- Very strong when combined with DAST

Drawbacks

- Requires agent on production servers
- Some agents have performance impact
- Takes time to tune for each app
- Limited to certain languages

SCA
Find vulns from third party libraries

Software Composition Analysis

Benefits

- Finds additional risk early in development process
- Developer-centric
- Easy to integrate into development process
- Automated remediation

Drawbacks

- Remediation may not be possible
- Only finds known vulns (CVEs)

Pen Test
Humans attacking the live app

Benefits

- Best results
- Finds the most complex attacks
- Covers business logic testing
- Supports all technologies

Drawbacks

- Expensive
- Slow, can't cover many apps per year
- Requires expertise
- May be long booking times

Common Pain Points and Solutions for Enterprise AppSec programs

Lack of ownership of asset program

INDICATORS

- Security seen as bolt-on
- Security can't make other BUs understand why/how it will affect their BU. Can't get buy-in

SOLUTION

- C-Suite must lead and drive AppSec program
- Business Unit led workflows for dev & security, not security leading the charge

HOW

- Security team seeks to truly understand dev and business and sympathize with them
- Security team frames everything as how it is beneficial for the business and dev teams
- Treat dev teams like they are an external customer
- Define clear process for onboarding new apps

Security doesn't know what they need to protect

INDICATORS

- Incomplete list of apps, app architecture, tech stack

SOLUTION

- Create App Catalog
- Perform internal and external app discovery
- Risk rank apps for prioritization

HOW

- App onboarding questionnaire tracks and stores in App Catalog:
  - Business & Tech owners
  - Tech stacks
  - Auth methods
- Learn app architectures
  -

Scaling AppSec in the Enterprise Space

INDICATORS

- Understaffed security team driving security for hundreds/thousand of apps - NOT scalable
- Apps built by dozens/hundreds of dev teams, using disparate tech stacks

SOLUTION

- Shift security ownership to dev teams and business owners
- Build a self-service portal for developers

HOW

- Designate security champions for each development team
- Integrate tool findings with developer's issue trackers
- Target remediation to most severe findings on critical apps
- Standardize development process and tech stack

Integrating with the SDLC

INDICATORS

- Devs already struggle to meet deadlines
- Devs don't want another tool to login to

SOLUTION

- Frame it as helping the dev team, be the helpful parent, not the enforcing cop

HOW

- Focus on real-world risks that could lead to runtime errors or crashes, breaches of sensitive data, fraud or compromise of critical systems.
- Motivation: 35x cheaper to catch early
- Get to the point where releases are mundane

Proving efficacy of AppSec program to leadership

INDICATORS

- Executive reports often use technical metrics that non-technical leaders don't understand
- Hard to quantify risk for CWEs
- Difficulty getting budget to improve your program

SOLUTION

- Measure remediation success with KPIs - track progress over time
- Link metrics to business goals, risk exposure of critical operations, regulatory compliance, legal risks

HOW

- Executive Metrics: tie to uptime, reduction of operations costs, SOC incidents
- Boil everything down to costs, speed, risk, and efficiency
- Import findings into GRC, quantify risk if possible. Manually review findings, adjust severity if needed
- Use BI tool to provide trending if not available in tools

Lack of expertise

INDICATORS

- Security team doesn't understand development process/terminology
- Inefficient - developers rely on lengthy rescans or security team to validate remediation

SOLUTION

- Use AppSec tools that provide convincing proof of vulnerability, quality remediation guidance
- Provide secure code training to developers on an ongoing basis

HOW

- Enroll security team in training: basic web app pen testing course, learn Burp, cloud architectures
- Consider managed services, staff augmentation
- Optimize validation scans to minimize rescan time, or use Attack Replay

Building a Self-Service DAST Portal

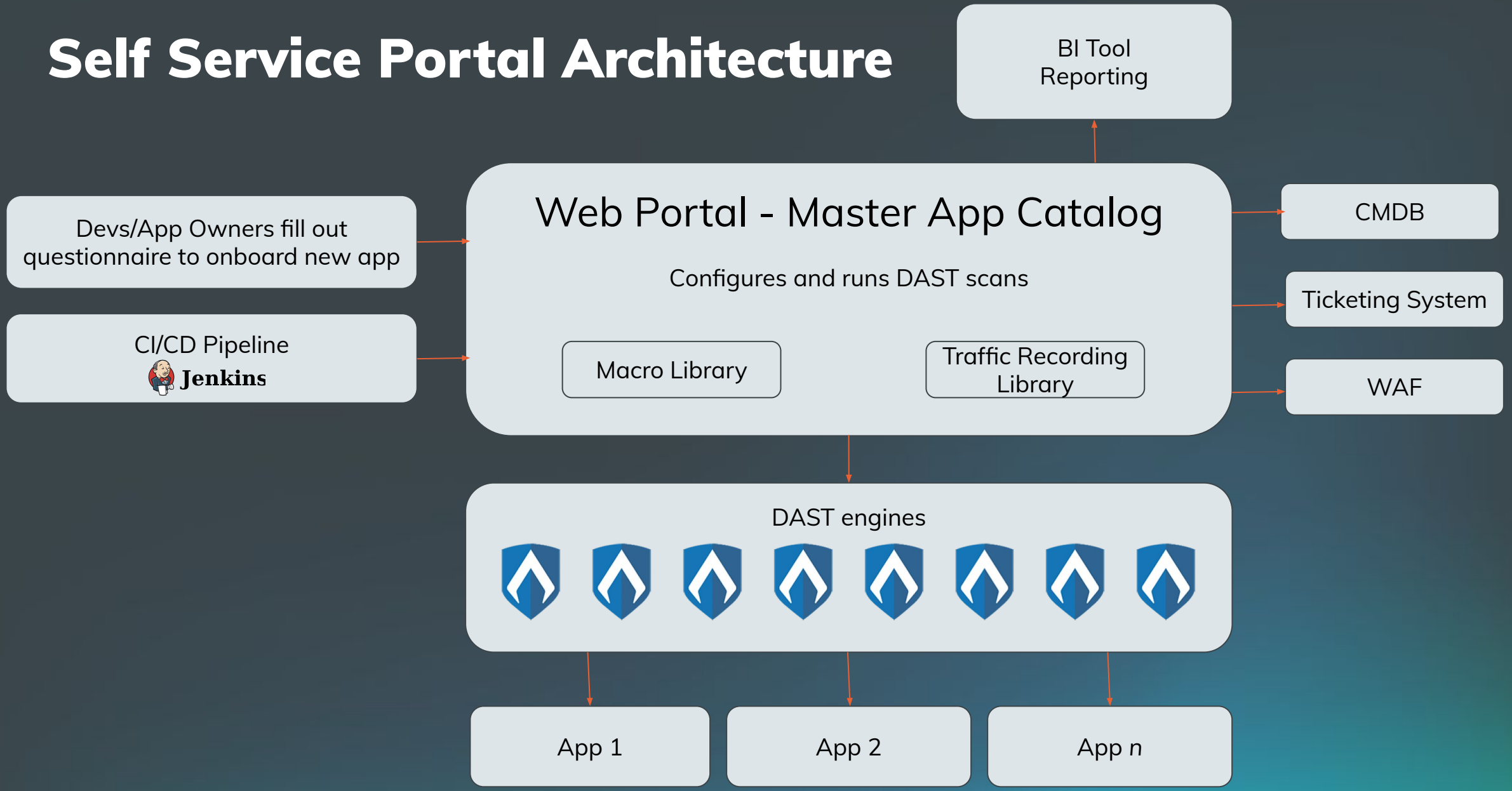
Building a Self Service DAST Portal - Process

1. Takes a couple years- won't happen overnight
2. Make it extremely simple for devs to use- pass/fail scan status
3. Largely rely on unauthenticated scans
4. Work closely with your AppSec tool vendors - influence roadmap, get access to their devs
5. Form coalitions with other enterprise companies

Building a Self Service DAST Portal - Technical

1. Keep it as simple as possible - limit config options
2. Ease troubleshooting by giving logs and scan status
3. Provide quick and easy authentication testing
4. Reuse QE's test harnesses (Selenium scripts)
5. Produce crawl maps so devs can confirm coverage
6. Display live telemetry data during scan

Self Service Portal Architecture



Summary

Key Takeaways

1. Must get C-levels onboard with security
2. Security must understand how the business works and work to support that
3. Create an formal app onboarding process & form that feeds into App Catalog
4. Frame all security's suggestions as how they will help devs and the business
5. Security can only protect what they understand - learn new tech, meet the devs where they are

Take-Home Exercises

Great for teaching people new to HTTP and API communication

Starts with zero knowledge, works up to DAST troubleshooting for auth and crawling SPA problems

21 walk-through exercises with screenshots

@TimHonker

github.com/thonker-r7

Q&A

Thank you