DevSecOps: Essential Tooling to Enable Continuous Security

Richard Mills
DevOps Solution Architect, Coveros Inc.
rich.mills@coveros.com
@armillz

Who is this guy?

- Me: Mad-Software-Developer turned Mad-Software-Engineer turned DevOps-Solution-Architect. Pragmatist. Particular focus on tools and automation. CI, CD, DevOps … what's next?
  - PS: Thanks for inventing the term "DevOps" to describe what I like to do.
  - … and then *DevSecOps, DevSecQaEntFinBizOps, etc.*
- Pays my bills: Coveros helps organizations accelerate the delivery of secure, reliable software using agile methods.
  - Agile transformations, development, and testing
  - Dev(Sec)Ops implementations
  - Training courses in Agile, DevOps, Application Security
- Keeps me intrigued: SecureCI
  - Open-source DevOps product
  - Integrated CI/CD stack with security flavor

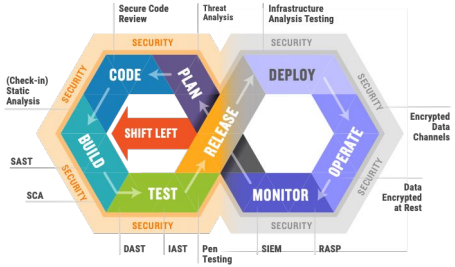
DevSecOps to match agile delivery

Modern Agile/DevOps software delivery is outpacing compliance-driven, late-lifecycle security processes

How do we solve it? → Dev (Sec) Ops

- Integrate security actions into sprint-ly delivery process
- Integrate security team members into development and operations (not police)
- Integrate "Quality Gates" into CI/CD pipeline → Security!

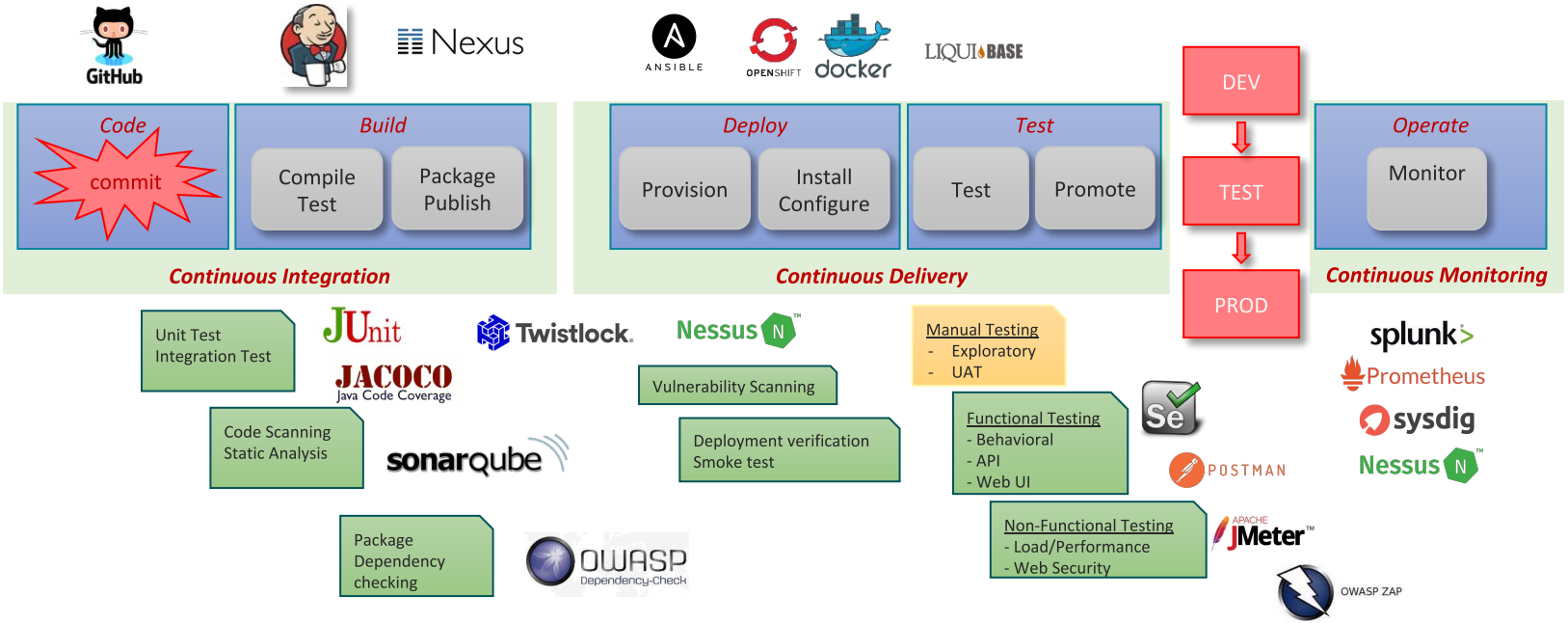
Goal: confidence that software is "secure enough" to defend itself every day



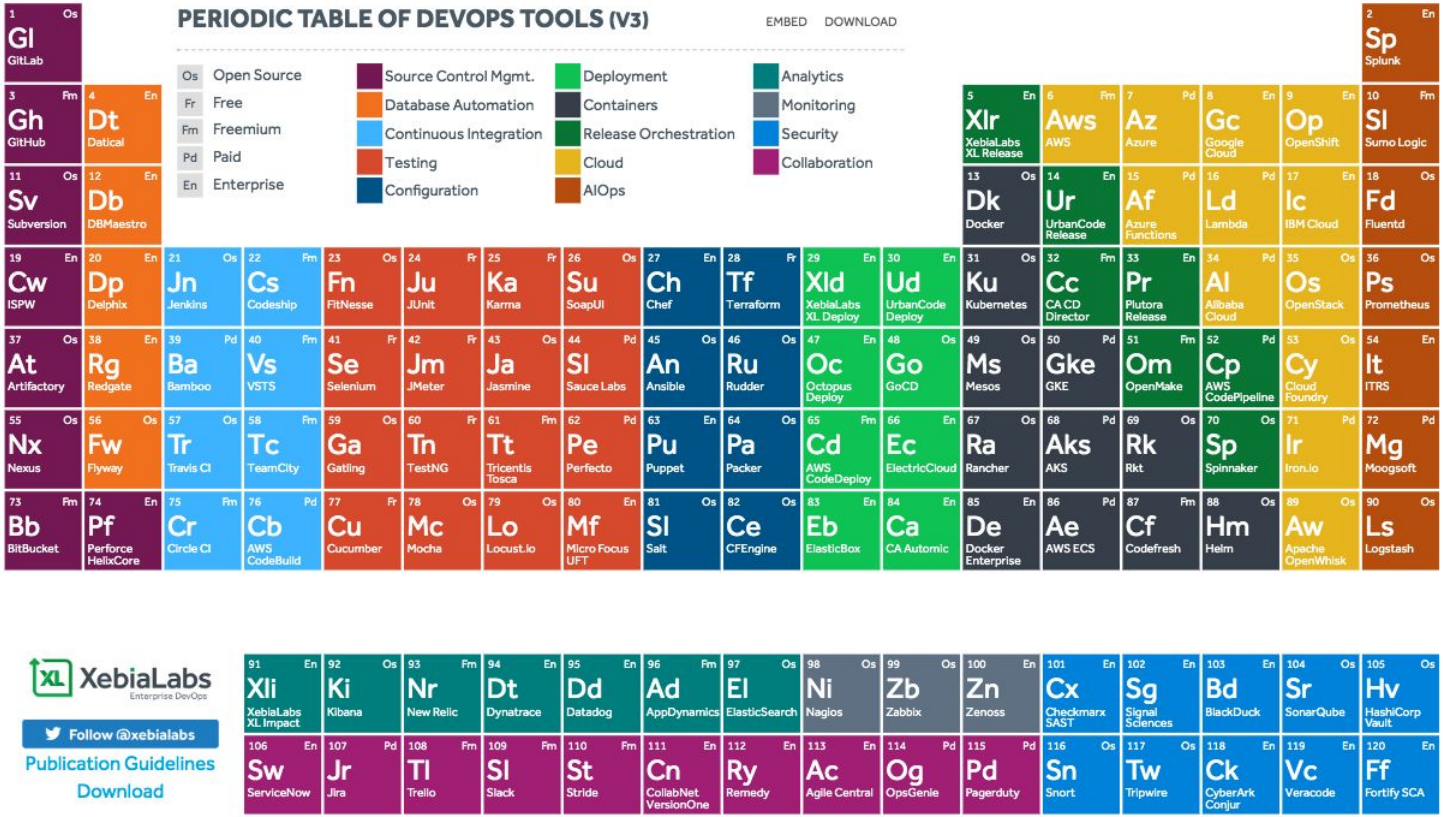
Need "continuous security" integrated into our delivery process

Pipeline defines delivery process

The software delivery process is automated through a CI/CD pipeline to deliver application microservices into various test (and eventually production) environments



Tools, tools, and too many tools



https://xebialabs.com/periodic-table-of-devops-tools/

Essential security tooling categories

- Static application scanning
  - analyze the source code, application structure, or platform as it is built to detect defects or vulnerabilities
  - In security space: SAST, software composition analysis, vulnerability scanning
- *Dynamic functional testing*
  - *variety of sub-categories of functional testing to verify that the software behaves according to its functional requirements.*
- Non-functional testing
  - verify software against sub-categories of cross-cutting, non-functional requirements (security, performance, accessibility, …)
  - In security space: DAST
- Real time monitoring
  - once the software is operating, monitor its operation and look for issues. (not necessarily a "quality gate" but it does ensure that software remains healthy)
  - In security space: may include IAST and RASP

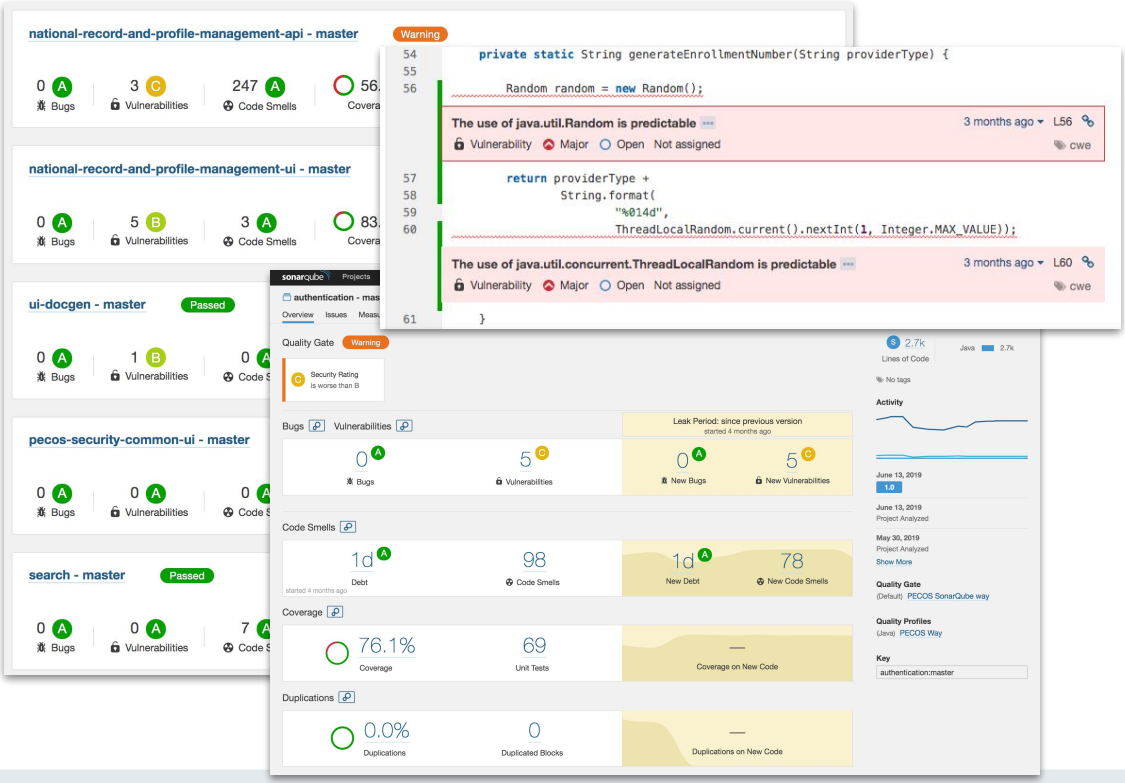
Static application scanning

- Static application scanning - run before we launch/run software
  - **Static code analysis** - quality, maintainability, **security** (frequently referred to as Static Application Security Testing, SAST).
  - **Software Composition Analysis** - performs 3rd party dependency checks
  - **Platform vulnerability scanning** - scan OS, middleware, configuration for known weaknesses
  - **Docker container scanning** - scan container images as they are built to detect whether vulnerable container layers are being used or misconfigured
- Tools:
  - SonarQube, FindBugs, PMD, Fortify, Veracode, …
  - OWASP Dependency Check, RetireJS, …
  - Nessus, OpenVAS, OpenSCAP, …
  - Twistlock, Falco, Aqua, ...

Static analysis: start with SonarQube

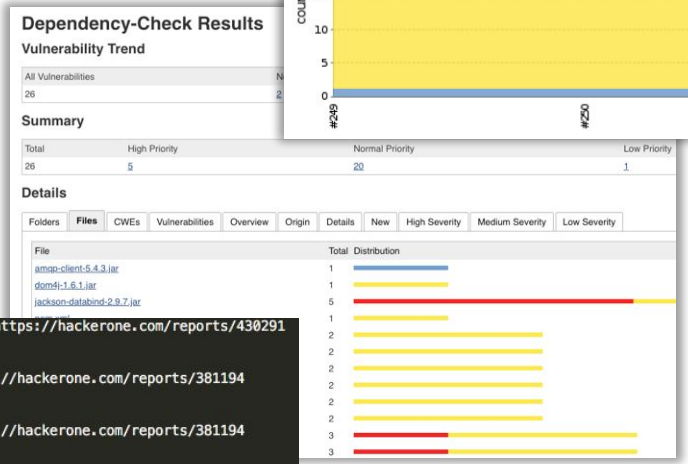
- Code scanning and quality dashboards
- Includes quality, security, and maintainability scans for many languages
- Continuous view of static code health, unit tests, coverage, …
- Inexpensive alternative to commercial tools such as Fortify, Veracode, etc.

sonarqube



Dependency checks for supply chain

- Ensure that you aren't using someone else's vulnerable code
- Software Composition Analysis against NVD with CVE
  - OWASP (Java), RetireJS (JavaScript), ...
  - Sonatype Nexus IQ Server, JFrog Xray, ...

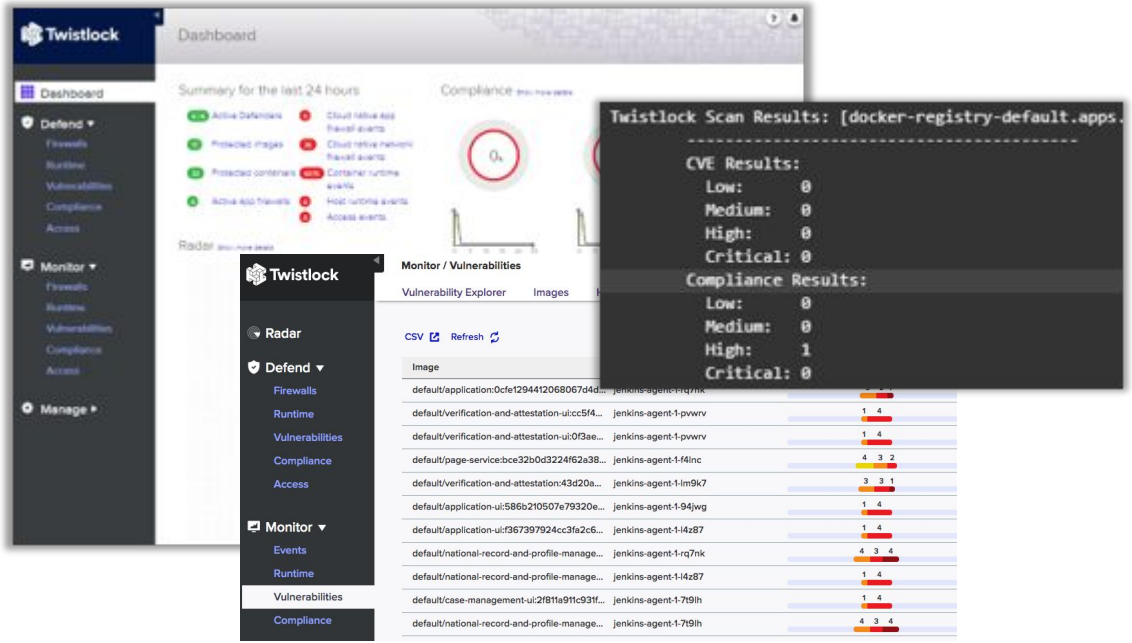



```
just-extend 3.0.0 has known vulnerabilities: severity: low; summary: Denial of Service; https://hackerone.com/reports/430291
@pecos/application 1.0.0
└ just-extend 3.0.0
merge 1.2.0 has known vulnerabilities: severity: high; summary: Denial of Service; https://hackerone.com/reports/381194
@pecos/application 1.0.0
└ merge 1.2.0
merge 1.2.0 has known vulnerabilities: severity: high; summary: Denial of Service; https://hackerone.com/reports/381194
@pecos/application 1.0.0
└ sane 2.5.2
  └ watch 0.18.0
    └ exec-sh 0.2.2
      └ merge 1.2.0
```

Container and platform scanning: Twistlock

- Examine container structure and behavior before and during execution
- Similar to vulnerability scanning of hosts

- Two roles:
  - Scan newly build app container images for vulnerabilities
  - Monitor running containers for compliance
- Others: Falco, Clair, Aqua, …
- Platform: Nessus, OpenVAS, ...



Dynamic functional testing

- Unit testing - verify that code functions properly in isolation during a build (pre-deployment)
- Health Tests - quick API health check endpoint pings to ensure services are running
- API testing - REST tests divided into smoke tests, functional tests, regression tests, etc.
- UI testing - Selenium/selenified tests for UI organized as smoke, functional, etc.

With Security: test your security functions (roles, auditing, encryption, …)

Tools:

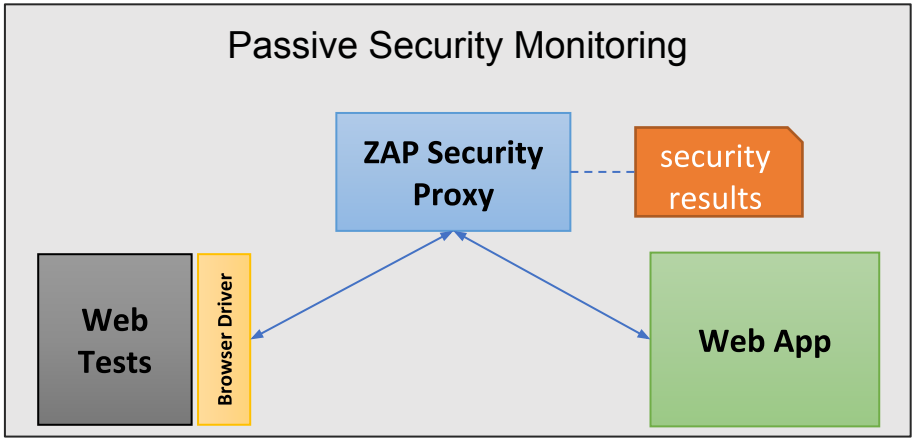
- Junit, Jest, TestNG, ...
- Selenium, Selenified, jBehave, Cucumber, ...
- REST Assured, Postman, JMeter, Taurus, …
- Security proxies: Zed Attack Proxy, Burp Proxy, ...

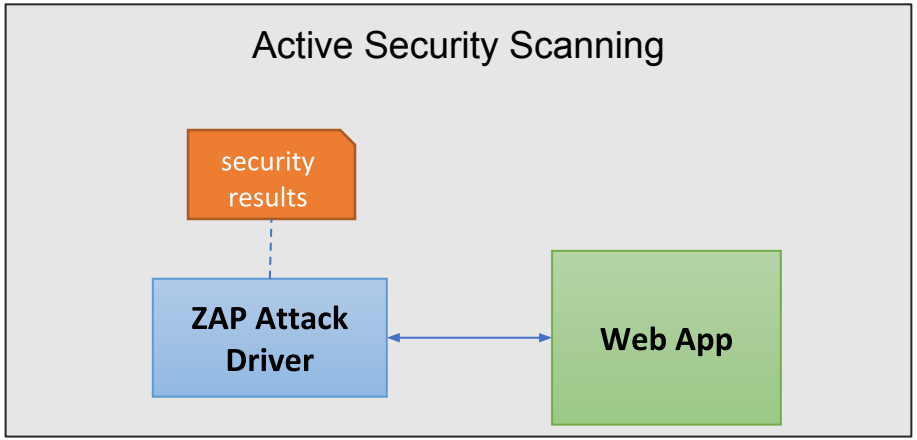
Point: these are good places to integrate dynamic security testing

Security pipeline with ZAP

OWASP Zed Attack Proxy (ZAP) is an easy to use, open-source web scanning and penetration tool

Two primary modes: Passive and Active



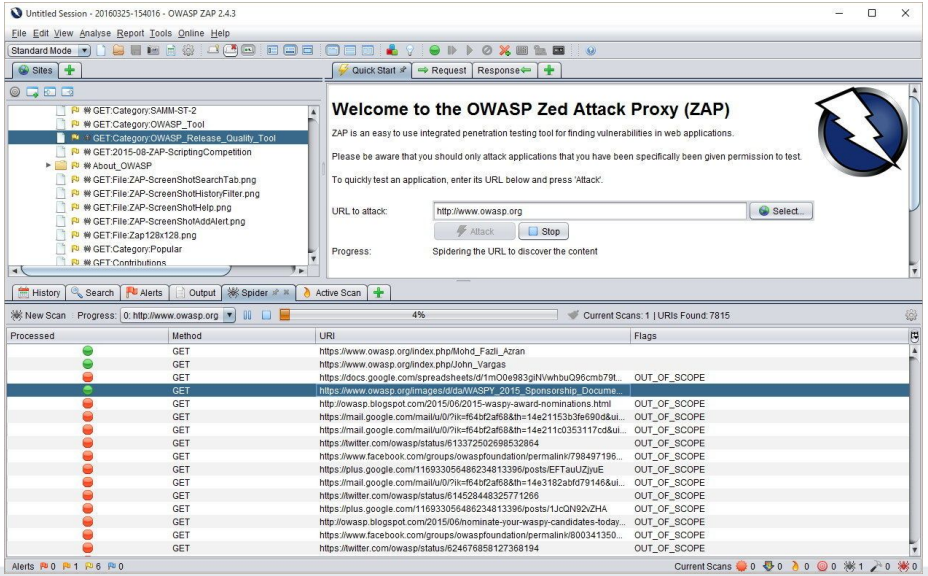


Non-functional testing

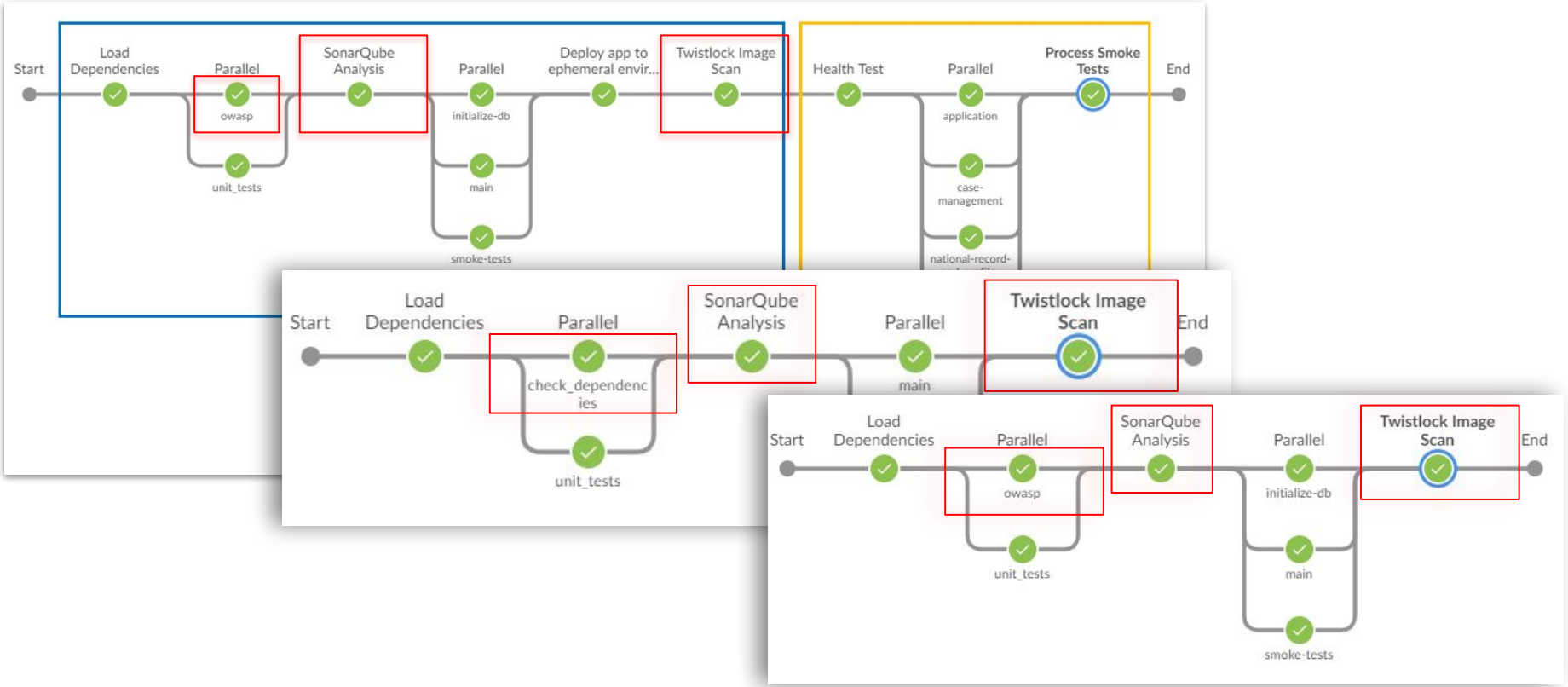
- ***Dynamic Application Security Testing (DAST) - automated web scanning, penetration testing, database testing, ...***
- Performance testing - automated performance tests run manually with JMeter by QA Team
- 508 Accessibility testing - executed periodically to validate that the application is usable for all people
- Other compliance testing…

Security Tools:

- ZAP, Burp Suite, IBM App Scan, Metasploit, Nmap, SQLmap, ...

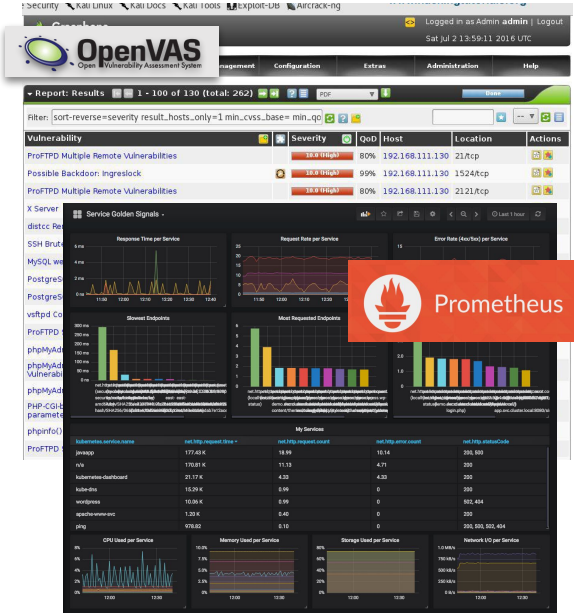


Tying it together: pipeline flow



Eventually: real-time monitoring

- Various aspects
  - **Log aggregation and scanning** - use processing rules to detect anomalous behavior (information leakage, high error rates, attack detection)
  - **Real-time container and host monitoring** - security monitoring of running docker containers running in test environments for behavior, configuration
  - **Container and host scanning** - scan hosts against configuration benchmarks
  - **Performance monitoring** - monitor system resources, response times, etc.
- Wraps into *Security Information & Event Management (SIEM)*
- Tools
  - Kibana/Logstash (ELK), Splunk, Tripwire, …
  - Nessus, OpenVAS, Twistlock, …
  - Prometheus, Graphana, Hawkular, New Relic, ...



kibana
logstash
splunk>
tripwire

Takeaways for continuous security

- Develop a *product* with security built in
- Find tools that fit each major category
  - Static analysis
  - Software Composition Analysis
  - Vulnerability scanning (platform, containers)
  - Dynamic testing
  - Monitoring
- Start with simple (free!) tools until you understand their value and cost
- Strive for continuous assessment
- Develop a *culture* of security

Thank You!

Questions?

rich.mills@coveros.com
@armillz
https://www.coveros.com/services/devops/

Join us on Slack! https://hub.techwell.com

Bonus Round: Integrating Teams

Integrating Dev, Sec, QA, Ops

Integrate your development, security, quality, and ops teams to streamline your delivery process and enable success

- Use team structures that encourage collaboration of security engineers with developers
  - Need engineers who understand code, build, deployment, testing, automation
  - Can't succeed with only compliance box checkers (yes, you need them too)
- Half the battle: getting teams to work together, not against each other
  - Security consultants, not security police
  - Contributors, not naysayers

Build a culture of security. Expect every build to be secure.

Horizontal Technical Guilds

- Group of specialized professionals working together to solve cross-team problems
- Guild members in-team are focused on team-specific problems
- Dedicated guild members support cross-team needs
- Guild establishes cross-team standards and shared success
- Important: share knowledge across team members

Cross-team function (vs. cross-functional team)

Challenge: You will never have enough security engineers for every team