App security in current era

Ajit Dhumale
Director of Engineering, Qualys

OWASP Pune Chapter Meet - 8 June 2019
Daddy, what are clouds made of?

Linux servers, mostly

What are clouds made of?
Web App Access Eco System

Browser

Network

Internet Infra

Web Server/App

Under user’s control

Hostile

Under app owner’s control

Images for various sources from the internet
Changes over time: technology

- Mainframe
- Client-Server
- Web
- Virtualization
- Hosted/cloud
- Micro Services
- Containers
- Serverless

Dedicated resources
Static/rigid/manual provisioning

Shared resources
Automated/elastic provisioning
Change over time: app availability

- **9 to 5**
- **Maintain Downtime**
- **Upgrade Downtime**
- **Long release cycles**

- **Always On (24/7)**
- **No Downtime**
- **Rolling Upgrades**
- **Frequent Releases**
Orchestration: Dynamic, auto, elastic provisioning

Logical resource pool:
Compute, storage, IO, network, …

Physical shared resource pool:
Compute, storage, IO, network, …
Changes over time: service model

Pets vs Cattles
Changes over time: people/roles

- Engineering
- Operations
- Security
- DevOps
- Security
- DevOpSec
Changes over time: app composition
Changes over time: app packaging

**Writable layer**
- Web App
- Spring
- Tomcat
- Apache

**Base layer: Alpine**
Overview of Security Changes over time:

- **Old way: Bolt-on Security**
  - Secure coding, secure libraries
  - Pre-commit static assessment
  - Declarative request/response specification

- **New way: Built-in/continuous Security**
  - Secure coding, secure libraries
  - Post-commit static assessment
  - Declarative request/response specification

### Development Cycle

- **Develop**
  - Secure coding, secure libraries (static assessment in IDE)
  - Declarative request/response specification

- **Check-in**
  - Secure coding, secure libraries
  - Pre-commit static assessment

- **Build**
  - Build time static assessment
  - Post-build automated DAST
  - Built complete app (e.g., containers)

- **Test**
  - Automated/repeated DAST

- **Deploy**
  - App-centric firewall
  - Access Token Rotation
  - Minimal privileges
  - RASP
  - Log monitoring

- **Monitor**
  - Declarative access control
  - Declarative resource usage limits
  - Shared resources with microsegmentation

**Bolt-on vs built-in/continuous**

- Firewall/IDS/IPS

---

*Note: Diagram and text content are aligned to provide a comprehensive overview of security changes and practices.*
Pre-requisites for shiftleft and continuous security

- Automation
- Speed
- Accuracy
- Declarative specification
• IDE
  • Continuous source code scanning (may be as you type)
  • Monitor 3\textsuperscript{rd} party components at inclusion time

• CI/CD
  • Continuous monitoring of vetted/approved dependencies
  • Commit time static scanning
  • Build time security
    • Static assessment
    • SCA
    • Dynamic scanning

• Runtime
  • Dynamic secrete management
  • App centric firewall
  • RASP
  • Runtime instrumentation
    • Monitoring
    • Profiling
SCA: Software Composition Analysis

• Why its important lately
  • More then 90% of the code in modern apps is 3rd party open source libraries/frameworks
  • Securities issues in 3rd party open source components are known to the world
  • Exploitation often does not need app specific knowledge
    • Exploits become available on internet
  • Easy to launch attack on large number of targets
  • Explosion in CVEs declaration in widely used software components
    • STRUTS RCEs
    • Wordpress RCEs
    • WebLogic RCEs
    • …
SCA: Software Composition Analysis

- Create inventory/BoM of all 3rd party components used by the app
- Check which components have known vulnerability

Issues/concerns
  - Coverage: Not all vulnerability have CVEs
  - Noise/FPs: App may be using vulnerable library but may never be calling vulnerable function

- Shiftleft: Integrate SCA in IDE, Artifactory, CI/CD
- Continuous: Detect production apps affected by new vulns
RASP: Runtime Security

• Runtime Instrumentation
  • Agent
  • Built-in the app
  • Instrumented runtime

• Monitor code flow, function calls, system access

• Log/block undesired behavior
  • App specific tuning

• Concerns: Performance overhead, undesired side effects (DoS by FP)
Dynamic secret management

- In pets era:
  - Manual provision of secrets

- In cattle era:
  - Challenge:
    - How to securely make secrets (DB password, API tokens, private key, ...) to dynamically provisioned ephemeral app processes/containers/micro services
  - Risk
    - Secrets sprawl
    - Secrets leakage (via github, ...)
  - Solution
    - Vault, secure introduction (SI) and dynamic tokens
Declarative security □ automated security

- Declarative network topology
  - Auto generate network access rules

- Declarative request/response
  - Auto generate app firewall rules/filters

- Declarative quota limits
  - Prevent resource exhaustion

- Micro compartments
  - Containment in case of compromise (warrants minimal privileges)
I have concerns about this cybersecurity candidate.

Anything in particular?

He's available.
We are hiring

https://www.qualys.com/careers/
Dev, QA, Support, Ops, Security