**CURRICULUM**

*Get the right training to build secure applications.*

**PLATFORM SECURITY**

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV531</td>
<td>Defending Mobile Applications Security Essentials</td>
</tr>
<tr>
<td>DEV541</td>
<td>Secure Coding in Java/JEE GSSP-JAVA</td>
</tr>
<tr>
<td>DEV544</td>
<td>Secure Coding in .NET GSSP-NET</td>
</tr>
</tbody>
</table>

**CORE**

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>STH.DEVELOPER</td>
<td>Application Security Awareness Modules</td>
</tr>
<tr>
<td>DEV522</td>
<td>Defending Web Applications Security Essentials GWEB</td>
</tr>
<tr>
<td>DEV534</td>
<td>Secure DevOps: A Practical Introduction</td>
</tr>
<tr>
<td>SEC540</td>
<td>Secure DevOps and Cloud Application Security</td>
</tr>
</tbody>
</table>

**SPECIALIZATION**

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEC542</td>
<td>Web App Penetration Testing and Ethical Hacking GWAPT</td>
</tr>
<tr>
<td>SEC642</td>
<td>Advanced Web App Penetration Testing and Ethical Hacking</td>
</tr>
</tbody>
</table>

**ASSESSMENT**

AppSec CyberTalent Assessment

[sans.org/appsec-assessment](sans.org/appsec-assessment)
Ben Allen

• Security Engineer at SANS Institute
• Operations Engineer, Developer at SANS prior to Security
• Network Security Analyst ... Architect at UMN
• GCIA, GPEN, GWEB, GWAPT, GMON

• Contact information
  ben.allen@mrsecure.org   @mr_secure
Agenda

• Continuous Opportunity: DevOps & Security

CONTINUOUS OPPORTUNITY

1. The DevOps Movement
2. Shifting Security Left
3. Examples
A LONG TIME AGO IN A GALAXY FAR, FAR AWAY

- Waterfall development
  Phased delivery in large projects
- Slow, gated deployment cycles
  Several months between releases
- Numerous handoffs between teams
  Dev -> QA -> Users -> Ops -> Sec
HOW’S THAT WORKING FOR YOU?

- Customers provide feedback too late in the process
- Delays between handoffs
- Security is left until the very end
- High risk / failed deployments
- Slow deployment cycles cause:
  - Projects are delayed and over budget
  - Long zero-day vulnerability windows
HOW’S THAT WORKING FOR YOU?

WORKED FINE IN DEV

OPS PROBLEM NOW
• Agile
  Break down walls between development and the business / customer

• DevOps
  Break down walls between development and operations

• SecDevOps
  Break down walls between security and development, operations, business
Much of the Security DNA in DevOps comes from a few leaders aka “unicorns”:

- **Etsy**
  Security in continuous delivery, “a Just Culture”

- **Netflix**
  Security in AWS, Simian Army

- **Facebook**
  Security at scale, OSQuery

- **Twitter**
  Self-service security for developers
Online crafts marketplace (PCI regulated), established in 2005. Over 1 million sellers, 21 million buyers.

In the beginning (2008):
• Difficulty scaling up engineering, ops teams
• Reliability, downtime problems during deployments
• Production releases 2 times per week
• Each release takes 4 hours
• Deployment process of a large enterprise
Fast forward to 2012:

• Continuous Deployment (CD)
  50 changes to production per day

• Dark launching (aka feature flags)

• A Just Culture
  Blameless post-mortems (and Morgue)
  It is safe to make mistakes – as long as you participate in solving them
  Record what happened and learn from it

• Dev and Ops all take on-call rotations

• Measure and track everything
DevOps is about CAMS:

- **Culture** - People and process first. If you don’t have culture, all automation attempts will be fruitless.
- **Automation** - This is where you start once you understand your culture. At this point, the tools can start to stitch together an automation fabric for DevOps.
- **Measurement** - If you can’t measure, you can’t improve.
- **Sharing** - Sharing is the feedback loop in the CAMS cycle.

*John Willis*
What Devops Means to Me, July 2010
This faster delivery cycle lets teams experiment, creating a feedback loop with customers. The result? The entire organization benefits, as measured by profitability, productivity, and market share.

2017 State of DevOps Report
Puppet / DORA 2017 State of DevOps Report for high-performing organizations:

• Deploy changes 46 times more often
• Lead times are >440 times shorter
• Change failure rate is 5 times lower
• Failure recovery is 96 times faster
• Spend 50% less time remediating security issues

• [https://puppet.com/2017-devops-report](https://puppet.com/2017-devops-report)
Agenda

• Continuous Opportunity: DevOps & Security
50 DEPLOYMENTS A DAY!

How does security keep up?

No pen testing?

No control gates?

No time for source code assessments?

No security sprints?
• Configuration Management
  aka Infrastructure as Code
  Puppet, Chef, Ansible, Salt, CFEngine

• Continuous Integration
  Jenkins, Travis, Bamboo, TeamCity

• Continuous Delivery
  Jenkins, Chef Delivery, Atlassian Bamboo,
  Amazon AWS Code Pipeline

• Continuous Deployment

• Continuous Monitoring
Keep up with the pace of continuous delivery by:

- Identify risks using threat modeling during planning
- Automate unit testing for security stories
- Iterative, incremental scans during code, test, and release

**SHIFTING SECURITY LEFT**
CAST STUDY | ETSY’S SECURITY PROGRAM | STEP 1 - 3

1. Don’t be an InfoSec jerk. Build security into the frameworks.

2. If it moves, graph it! Real-time monitoring for building attack-driven security defenses.

3. Just ship it! Every engineer can push to prod at any time, including security.
CAST STUDY | ETSY’S SECURITY PROGRAM | STEP 4 - 7


5. Designated hackers assigned to a handful (~ 5) projects.

6. Engineering / Security job rotations

7. Bug bounties, both internal and external
Make security a first class citizen during development workflow:
• Static Application Security Testing (SAST) is built into the IDE
• Commits trigger automated security scans (out of band)
• Light-weight, **accurate** static analysis scans (in the pipeline)
• Alerts when high-risk code is changed
• Automated unit testing for security features
• Fast accurate feedback which returns pass / fail results
Security tools for static analysis:

- **Free / open source:**
  - Find security bugs, Phan, CAT.NET, Brakeman, Bandit, Flawfinder, QARK
- **Commercial:**
  - HP Fortify, Checkmarx, Coverity, IBM AppScan Source, Klocwork, Veracode, Brakeman Pro

Security tools for vulnerable dependencies:

- **Free / open source:**
  - OWASP Dependency Check, SafeNuGet, Retire.js
- **Commercial:**
  - Sonatype, Black Duck, Palamida, Source Clear
Automate various dynamic tests throughout the delivery pipeline:

- Functional security testing
  Automate tests against authentication, authorization, password management using Selenium or similar tool

- Dynamic Application Security Testing (DAST)
  Black box scanners looking for known classes of weakness
  Library of past flaws to scan for
DYNAMIC ANALYSIS TOOLS

Security tools for dynamic analysis

• Free / open source:
  ZAP, Arachni, w3af, Skipfish, Nikto

• Commercial:
  Burp Suite, HP WebInspect, IMB AppScan, Nessus, Veracode, WhiteHat Sentinel

• CI Scanning frameworks:
  Gauntlt, F-Secure, BDD-Security, Mozilla Minion, Yahoo Gryffin
Leverage monitoring tools and approaches for security monitoring:

- Look for attack signatures
  Authentication failures, 4XX/5XX errors, database syntax errors, login failures, access control exceptions
- Correlate with traffic information (source, type)
- Feed trends and anomalies back to monitoring tools

Must watch: Christopher Rimondi “Using DevOps Monitoring Tools to Increase Security Visibility”
- [https://www.youtube.com/watch?v=TNCVv9itQf4](https://www.youtube.com/watch?v=TNCVv9itQf4)
CONTINUOUS MONITORING – DASHBOARD - Etsy
• Hygieia – Capitol One (https://github.com/capitalone/Hygieia)
Agenda

- Continuous Opportunity: DevOps & Security

CONTINUOUS OPPORTUNITY

1. The DevOps Movement
2. Shifting Security Left
3. Examples
SHIFTING SECURITY LEFT

- AWS CodePipeline used to build Java App
- AWS CodeBuild “Build” phase creates docker container
- AWS CodeBuild “Test” phase runs SAST, Dependency checks
- Data published into Jenkins

- Integrate security testing into the build process
FindBugs Result

Warnings Trend

<table>
<thead>
<tr>
<th>All Warnings</th>
<th>New this build</th>
<th>Fixed Warnings</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Summary

<table>
<thead>
<tr>
<th>Priority</th>
<th>Total</th>
<th>High Priority</th>
<th>Normal Priority</th>
<th>Low Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Details

<table>
<thead>
<tr>
<th>Type</th>
<th>Total</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_INJECTION_JDBC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SQL_NONCONSTANT_STRING_PASSED_TO_EXECUTE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
# Dependency-Check Results

## Vulnerability Trend

<table>
<thead>
<tr>
<th>All Vulnerabilities</th>
<th>New Vulnerabilities</th>
<th>Fixed Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Summary

<table>
<thead>
<tr>
<th>Total</th>
<th>High Priority</th>
<th>Normal Priority</th>
<th>Low Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

## Details

<table>
<thead>
<tr>
<th>Folders</th>
<th>Files</th>
<th>CWEs</th>
<th>Vulnerabilities</th>
<th>Overview</th>
<th>Details</th>
<th>High Severity</th>
<th>Medium Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Folder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app/app.jar/BOOT-INF/lib</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total: 7
MONITORING FOR SECURITY

• Backstory:
  Basic OS-level monitoring on hosts
  Using collectd to gather disk/cpu/network stats & ship to graphite

• What can we identify?

• Note: These data sets are based on simulations, not actual incidents
Data Exfiltration?

Disk Bytes - read up, write down

eth0 - bytes tx/rx

CPU Utilization
Ransomware?
Opportunities that SecDevOps presents:

• Trade inefficient, ineffective point-in-time compliance snapshots for continuous, real-time verification

• Build security testing, scanning, and reviews into the pipeline to find low hanging fruit & prevent regressions

• Reduce time spent on security remediation

• Ensure the entire project team understands the hostile environment their applications face via continuous feedback from production
Questions ?
Get the right training to build secure applications.

**Platform Security**
- **DEV531**
  Defending Mobile Applications Security Essentials
- **DEV541**
  Secure Coding in Java/JEE
  GSSP-JAVA
- **DEV544**
  Secure Coding in .NET
  GSSP-NET

**Core**
- **STH.DEVELOPER**
  Application Security Awareness Modules
- **DEV522**
  Defending Web Applications Security Essentials
  GWEB
- **DEV534**
  Secure DevOps: A Practical Introduction
- **SEC540**
  Secure DevOps and Cloud Application Security

**Specialization**
- **SEC542**
  Web App Penetration Testing and Ethical Hacking
  GWAPT
- **SEC642**
  Advanced Web App Penetration Testing and Ethical Hacking

**Assessment**
- AppSec CyberTalent Assessment
  sans.org/appsec-assessment