SESSION ID: DSO-T11

DevSecOps State of the Union

Clint Gibler
Research Director
NCC Group
@clintgibler
Distill tips / insights from talks, blog posts, conversations
About Me

- Technical Director and Research Director at NCC Group
- PhD in Computer Science from UC Davis

Things I ❤

- DevSecOps, security automation, scaling security
- Automated bug finding (static and dynamic analysis, fuzzing, …)
Before We Start - My Assumptions

• You’ve found SAST/DAST not that useful (operational time required & cost)
• You’re willing to invest time now to reap big security wins later
• Your security team has at least a few people, but not dozens
Agenda

- **Big Picture**
  - Mindsets and Principles
  - Choosing How to Invest Your Time
- **Scaling Your Company’s Security**
  - The Fundamentals
  - Scaling Your Efforts
  - Security Endgame
- **Action Plan**
Mindsets & Principles

- **Automate as much as possible**
  - Security teams are always time and person-limited, you need to scale

- **Guardrails not Gatekeepers - minimize “no’s”**
  - Netflix’s *Paved Road*. Scaling Appsec at Netflix by @astha_singhal

- **Prefer high-signal, low-noise tools and alerting**
  - It may be better to miss some issues than drown in triaging alerts that don’t matter
Mindsets & Principles

- **Developers are your customers - UI and UX is important**
  - How can we fit into dev’s existing tools and workflows?
  - Can we make the secure way easier, faster, or otherwise better than the current way?
  - Build in useful features (telemetry, logging, etc.)

- **Self-service security**
  - Provide tools and services devs can use without security team interaction

- See also: [Tech Beacon blog post](#) on mindsets / principles
Choosing How to Invest Your Time

Now

Medium Term

Long Term
Choosing How to Invest Your Time

Ask Yourself

- Of my near / medium term tasks, which will provide the most long-term strategic value?
- Can I do a near term task a little bit differently to make it much more useful later?
- What (sub)problems can I solve with high accuracy, at scale?
Security Tools in Your Tool Belt

- Static analysis
- Dynamic analysis
- Secure wrapper libraries
- Bug bounty
- Fuzzing
- Runtime detection
- Pen tests

Choosing what to ignore

@clintgibler
Targeting Vulns by **Complexity / Class**

- **Easy**
  - Missing TLS
  - No security headers
  - Calling dangerous fxns
  - Missing security controls

- **Medium**
  - Standard OWASP bugs
  - XSS, SQLi
  - XXE, SSRF
  - ...

- **Hard**
  - Complex, multi-step bugs
  - Business logic flaws
  - Abuse
Targeting Vulns by Complexity / Class

- Secure defaults
- Automated tools
- Bug bounty
- Pen tests

Easy

Medium

Hard
Targeting Vulns by Complexity / Class

- Secure defaults
- Automated tools
- Bug bounty
- Pen tests

Easy | Medium | Hard
Targeting Vulns by **Complexity / Class**

- **Easy**
  - Secure defaults: 3 units
  - Automated tools: 1 unit
  - Bug bounty: 1 unit
  - Pen tests: 1 unit

- **Medium**
  - Secure defaults: 3 units
  - Automated tools: 1 unit
  - Bug bounty: 2 units
  - Pen tests: 1 unit

- **Hard**
  - Secure defaults: 3 units
  - Automated tools: 1 unit
  - Bug bounty: 2 units
  - Pen tests: 2 units
Targeting Vulns by **Complexity / Class**

- **Easy**
  - Secure defaults: 100%
  - Automated tools: 0%
  - Bug bounty: 0%
  - Pen tests: 0%
  - Runtime monitoring: 0%

- **Medium**
  - Secure defaults: 70%
  - Automated tools: 30%
  - Bug bounty: 30%
  - Pen tests: 70%
  - Runtime monitoring: 30%

- **Hard**
  - Secure defaults: 50%
  - Automated tools: 50%
  - Bug bounty: 50%
  - Pen tests: 90%
  - Runtime monitoring: 50%
Targeting Vulns by Complexity / Class – Key Takeaways

- Solve as many of your problems as possible with **secure defaults**
- **Automated tools** won’t solve all of your problems
- **Bug bounty** can provide decent coverage of low/medium hanging fruit
  - If you’re building a new AppSec program, start with a private program with few researchers. Consider a pen test first and paying for triage.
- Use **pen testing** for the hard problems, where it provides best value
- **Runtime monitoring** for bugs that are too hard/inefficient to find in other ways
De-emphasized*

- Manual Testing
- Manual Code Review
- Per-App Threat Modeling
- Traditional Vuln Scanning

Used With Reservations*

- Generic Static/Dynamic Scans
- 3rd Party Pentesting
- Training

Heavily Emphasized*

- Automated Visibility & Action
- Org-level Partnerships
- AuthN & AuthZ Everywhere
- Paved Road
- Self-Service
- Killing Bug Classes

* This is the current mix. Wasn’t always this way.
How is the future shaping up for us?

Today
- Secure By Default
- Self Service
- Security Partnership

Mid term
- Secure By Default
- Self Service
- Security Partnership

Long term
- Secure By Default
- Self Service
- Security Partnership
Agenda

- **Big Picture**
  - Mindsets and Principles
  - Choosing How to Invest Your Time
- **Scaling Your Company’s Security**
  - The Fundamentals
  - Scaling Your Efforts
  - Security Endgame
- **Action Plan**
The Fundamentals
Vulnerability Management
Continuous Scanning
Asset Inventory
Vulnerability Management - Basics

Know your current state and if your future efforts actually work

Success Criteria

- Minimal friction for devs and security
- All vulns tracked in the same system as normal bugs
- All vulns processed through the same workflow (bug bounty, pen testing, tools, internal tests)
- Track relevant meta data
**Vulnerability Management - Basics**

**Track Meta Data**

- Relevant code base (and team/org)
- Vuln class - access controls, XSS, SQLi, open redirect, ...
  - OWASP Top 10 is too broad, use a more detailed taxonomy, like [Bugcrowd’s VRT](https://www.bugcrowd.com/vulnerability-taxonomy)
- Risk, Severity, Impact
- How was the vuln found? (Pen test, bug bounty, internal testing, tool A, tool B...)
Vulnerability Management - Leveling Up

- **Automate** as much of the vuln ingestion and triage process as possible
  - Tool -> Triage -> Jira
  - Bug Bounty -> Triage -> Assign to appropriate team
- **Create a vuln/risk dashboard** that’s viewable by project, team, org
  - Where should I invest security engineering efforts?
  - Puts (friendly) pressure on teams / orgs to improve
Vulnerability Management - Leveling Up

The Art of Vulnerability Management | Alex Nassar, AppSec Cali ‘19
Pro tips on creating a vuln management program that works for devs & security

Data Driven Bug Bounty | Arkadiy Tetelman, BSides SF 2018
Use Bug Bounty data to inform where you invest AppSec time
Data Driven Bug Bounty - Open Vulns by Priority

Open Security Vulns (by priority, last 90 days)

- Blocker
- Critical
- Major
- Minor
- Trivial
Data Driven Bug Bounty

Open Vulns By Subteam & Priority
Bug Bounty Cost by Vulnerability Class

- XSS: $20,000
- Access Controls: $15,000
- SSRF: $15,000
- Data Exposure: $15,000
- CSRF: $30,000
- Open Redirect
Domino's Delivery of a Faster Response was No Standard Order
# RSAC

## A Pragmatic Approach to Internal Security Partnerships

AppSec Cali ’19 | Scott Behrens, Esha Kanekar

![Table Image]

<table>
<thead>
<tr>
<th>Updated At</th>
<th>Meechum Policy</th>
<th>Zoltar</th>
<th>Edge Classification</th>
<th>Instance Count Classification</th>
<th>BaseAMI Score</th>
<th>Org Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>about 5 hours ago</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>Yes</td>
<td>High</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 6 hours ago</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>Yes</td>
<td>Low</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 5 hours ago</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Production Engineering</td>
</tr>
<tr>
<td>about 6 hours ago</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 5 hours ago</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 6 hours ago</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 5 hours ago</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 6 hours ago</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 5 hours ago</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 6 hours ago</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 5 hours ago</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 6 hours ago</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
<tr>
<td>about 5 hours ago</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>Yes</td>
<td>Medium</td>
<td>Content Platform Engineering</td>
</tr>
</tbody>
</table>

*Filter:*
- Name
- Tags
- Assignee
- Status
- Task (Search)
- Metadata
  - confidence_analysis:running_instances==true
  - confidence_analysis:internet_accessible==true
  - wp_metadata:organization:kevs="Content Platform Engineering"
The Fundamentals

Vulnerability Management

Continuous Scanning

Asset Inventory
Core idea: continuously scan new code with static and dynamic analysis tools
Continuous Scanning - AppSec Pipeline

Core idea: continuously scan new code with static and dynamic analysis tools

Developer Machines
Static Analysis (SAST) (SAST)

Code Hosting
Static Analysis (SAST)

Test/QA Environment
DAST (scan web apps)
Container scanning
Network vuln scanning
Fuzzing
Continuous Scanning - AppSec Pipeline

**AppSec USA**
- Put Your Robots to Work: Security Automation at Twitter | ’12

**salesforce**
- Providence: rapid vuln prevention (slides, blog, code) | ’15

**CISCO**
- Cleaning Your Applications’ Dirty Laundry with Scumblr (code) | ’16
- Scaling Security Assessment at the Speed of DevOps | ’16

**PayPal**
- SCORE Bot: Shift Left, at Scale! | ’18
Continuous Scanning - AppSec Pipeline

Salus: How Coinbase Sales Security Automation (blog, code) DSC London ’18

Orchestrating Security Tools with AWS Step Functions (slides) DeepSec ‘18

A Case Study of our Journey in Continuous Security (code) DSC London ’19

Dracon- Knative Security Pipelines (code) Global AppSec Amsterdam ‘19
Continuous Scanning - Trends & Best Practices

- Focus on *iteration speed* - adding/removing tools, testing new rules
- Scan unit: *pull requests* - every commit is too noisy, e.g. work in progress
- Scans should be *fast* (a few sec) - give dev feedback while context is fresh
  - Can do longer / more in depth scans daily or weekly
- Tool findings should be shown *within dev systems* (e.g. on PR as a comment)
  - Findings must always be *actionable* – how does the dev fix it?
- Focus on *high signal checks* - +95% true positives
  - Otherwise causes ill will with devs + too much security team operational cost
- Make sure to capture *metrics* - common finding types, FP rate, etc.
- You’ll need to build some deduplication and whitelisting logic

Being able to *map/reduce* over all of your code & live systems is *really* useful
Continuous Scanning - AppSec Pipeline

Good principles, mindsets, and perspectives

Building a Secure DevOps Pipeline | AppSec USA ’17, M Tesauro, A. Weaver
Case study and useful principles behind building a pipeline (code)

*AST In CI/CD – how to make it WORK! | DSC Singapore ‘18
Ofer Maor gives a great overview of SAST, DAST, IAST, ... and their pros/cons
Static Analysis by Complexity

- **Easy**
  - `grep` (regexes)
    - Operates on strings

- **Medium**

- **Hard**
  - Control / Data flow analysis (SAST)
    - Can reason about how data flows through system
Static Analysis by Complexity

Easy

grep (regexes)

• Operates on strings

Pro: Fast
Con: Not expressive

Medium

Hard

Control / Data flow analysis (SAST)

• Can reason about how data flows through system

Pro: Expressive
Con: Slow, noisy (FP)
**Static Analysis by Complexity**

**Easy**
- grep (regexes)
  - Operates on strings
  - Pro: Fast
  - Con: Not expressive

**Medium**
- Linting / Abstract Syntax Tree (AST)
  - Source code aware
  - Middle ground:
    - Fast
    - Match source code structures, some control/data flow

**Hard**
- Control / Data flow analysis (SAST)
  - Can reason about how data flows through system
  - Pro: Expressive
  - Con: Slow, noisy (FP)

Watch this space!
Static Analysis – Security Linting

Writing custom lightweight static analysis checks (AST matching)

Practical Static Analysis for Continuous Application Security - AppSec USA ‘16
Justin Collins on building custom, lightweight linting rules (Ruby, Python, JS)

How to Write Custom, Lightweight Static Analysis Tools (code) - ShellCon ‘19
Clint Gibler/Daniel DeFreez- AST matching Ruby (explore)/JS (RCE) w/ semantic

Tools

- Useful multi-language parsers: semantic, bblfsh
- Simply match code patterns (code-aware grep): sgrep
Continuous Scanning
What should we look for?
Static Analysis - Code

- High signal vulnerability checks and security anti-patterns
  - E.g. Disabling TLS verification
- Block banned or dangerous functions
  - E.g. Calls to `exec()`, `eval()`
  - `mozilla/eslint-plugin-amo`, `mozilla/eslint-plugin-no-unsanitized` - disallow `innerHTML()`, etc.
- Detect security-relevant code additions
  - “Looks like you’re adding some crypo-related code, let’s chat.”
- Alert on sensitive file changes
  - AuthZ/AuthN, login flow, things that should rarely change
Open Source Static Analysis Tools

- C/C++ - Clang Static Analyzer, Phasar, Cppcheck
- C#/.NET - Puma Scan, Security Code Scan
- Golang - gosec, glasgo
- Java - SpotBugs, Frameworks: Soot, WALA
- JavaScript/Typescript - NodeJsScan, eslint, tslint, eslint-plugin-no-unsanitized
- Python - bandit, dlint, pyre-check (data-flow analysis to find web app bugs)
- Ruby - Brakeman

Massive list: mre/awesome-static-analysis
Static Analysis – Out of Date Dependencies

Automate the Discovery & Eradication of Open-Source Software Vulns | BlackHat USA ‘19

Netflix's Aladdin Almubayed on how to identify and eliminate open-source vulnerabilities across applications you own at scale (slides)

Tools: OWASP DependencyCheck, language-specific tools
Static Analysis – Infrastructure as Code

Static Analysis for Code and Infrastructure | Nick Jones, DevSecCon London ‘16
Scan infra as code (Ansible, Puppet, Chef, ...) for insecure configs

Security Linting Tools
- Terraform: liamg/tfsec, bridgecrewio/checkov, cesar-rodriguez/terrascan
- CloudFormation: skyscanner/cfripper, stelligent/cfn_nag
- AWS IAM policies: Parliament (blog | code) can detect cases like when a role could escalate its privileges
Continuous Scanning - Key Takeaways

- Build the capability to scan every: PR, code base, deployed service
- High signal checks only
- Ensure a security baseline - don’t try to find every bug
- Scan for (missing) security controls, security-relevant changes
The Fundamentals

Vulnerability Management

Continuous Scanning

Asset Inventory
Asset Inventory - What is It?

Depends on who you ask!

A list of the things you own (code, servers, databases, employee devices, ...)

Common Approaches

- Blackbox network-based - OSINT, certificate transparency, ...
- Whitebox network-based - Give tool read access to your cloud env
- Whitebox holistic - Integrations with cloud provider, code hosting, ...
Asset Inventory - Basics

Know what you **own** and how they **connect**

**Success Criteria**

- **Code:** Meta info file in repos containing owning team, team lead, security PoC, ...
- **Cloud:**
  - Live servers/databases/load balancers...
  - Services used
  - Credentials, secrets, API tokens
  - Roles, permissions
  - Network ACLs, segmentation
Asset Inventory - Leveling Up

- Build capabilities to get visibility into other assets
  - Current employees
  - Employee phones and laptops
  - Deployment pipeline - track code from repo -> QA/staging -> production
- Enable querying crossing multiple knowledge domains
Asset Inventory - Talks

**Lyft Cartography: Automating Security Visibility and Democratization** | BSidesSF ‘19
Sacha Faust: Represent your assets as a graph, search across them (code)

**Overcoming old ways of working with DevSecOps - Culture, Data, Graph, & Query**
Erkang Zheng, DevSecCon Seattle ‘19 – Similar approach ^, security policy as code

**Expose Yourself Without Insecurity** | Art Into Science ’20, Rob Ragan, Oscar Salazar
Survey of asset inventory tools and approaches, inside/out vs. outside/in tradeoffs
Lyft Cartography

Represents assets as a graph in Neo4J
Asset Inventory - Examples

- Which RDS instances have encryption turned off?
- Which EC2 instances are directly exposed to the internet?

New Critical RCE!!1! #StrutsBleed
Agenda

- Big Picture
  - Mindsets and Principles
  - Choosing How to Invest Your Time
- Scaling Your Company’s Security
  - The Fundamentals
  - Scaling Your Efforts
  - Security Endgame
- Action Plan
**Threat Modeling**

**Challenge:** Security team can’t threat model every story. What do you focus on?

**Approaches:**

1. Self-service security questionnaires
2. Add lightweight threat modeling to SDLC
3. Threat model as code

Then security engineers get involved in highest risk services and new features
Scaling Your Efforts

Threat Modeling
Security Engineering
Detection & Response
Security Engineering - “Paved Road”

**Core idea**: Build libraries / tools that are secure by default for dev teams

Framework/tech choices matter
- Mitigate classes of vulns

**Areas to consider**
- Managing secrets
- Anything related to crypto
- Authentication / Authorization
- SQL, file system access
- Shell `exec()`
Security Engineering - Examples

- Port front end to React - **XSS**
- Wrote data model wrapper library - **SQLi**

**Key takeaways:**

- “**<X> is hard to do securely, have to be aware of threats 1, 2, and ...**”
  - Build a secure by default implementation
- “Hitch your security wagon to developer productivity” - **Asthा Singhal**, Netflix
- The secure version should have an even better dev UX than the old way
- Integrate security at the right points (e.g. new project starter templates) to get automatic, widespread adoption with minimal effort
Scaling Your Efforts

Threat Modeling
Security Engineering
Detection & Response
Slack IR Bot

Core idea: When a fishy event occurs, prompt originating user with Slack bot question + 2FA. Only escalate if user did not initiate action.

Motivation: Push validation to devs to free up security engineer time.
Slack IR Bot

**Core idea:** When a fishy event occurs, prompt originating user with Slack bot question + 2FA. Only escalate if user did not initiate action.

**Motivation:** Push validation to devs to free up security engineer time.

- Ryan Huber, Slack - [Distributed Security Alerting ‘16](#)
- Dropbox - [Meet Securitybot: Open Sourcing Automated Security at Scale 2017 (code)](#)
- [Empowering the Employee: Incident Response with a Security Bot](#)
  - Pinterest - Jeremy Krach, AppSec USA 2018
How Dropbox Builds Tools for Threat Detection & IR

Something sketchy happened

Alerts via Kafka

Grabs context from disparate systems
- Users
- Hosts
- Processes
- ...

Investigation happens in Jupyter notebook
- Repeatable
- Documented

Returns & runs a Jupyter notebook corresponding to the alert
Automating SOC Security Runbooks

- Write security runbooks that define how to respond to a given event
  - AWS Lambdas spin up to call the relevant security products, custom scripts, etc

Blog Posts

- **Twilio's SOCless: Automated Security Runbooks**
  - Source code | Docs

- **Auth0: Guardians of the Cloud: Automating the Response to Security Events**
Detection & Response - Key Takeaways

- Push first line triage to originating user (as appropriate)
- Any context needed for human analyst to proceed - gather automatically
- Document runbooks for how you respond to different events - automate
Agenda

- **Big Picture**
  - Mindsets and Principles
  - Choosing How to Invest Your Time
- **Scaling Your Company’s Security**
  - The Fundamentals
  - Scaling Your Efforts
  - Security Endgame
- **Action Plan**
Security EndGame

Automating Least Privilege
Targeting Vuln Classes: Case Study
Enforce Invariants
Automating Least Privilege

- **Least Privilege: Security Gain without Developer Pain | Enigma ‘18 (code)**
- New apps at Netflix are granted a base set of AWS permissions
- RepoKid gathers data about app behavior and automatically removes AWS permissions, rolls back if failure is detected
- Apps converge to least privilege with minimal security team interaction
- Unused apps converge to zero.
Automating Least Privilege

- **Least Privilege: Security Gain without Developer Pain** | Enigma ‘18 ([code](#))
- New apps at Netflix are granted a base set of AWS permissions
- RepoKid gathers data about app behavior and automatically removes AWS permissions, rolls back if failure is detected
- Apps converge to least privilege with minimal security team interaction
- Unused apps converge to zero.

**policy_sentry**

IAM Least Privilege Policy Generator, auditor, and analysis database.

By Kinnaird McQuade of Salesforce ([blog](#)) ([code](#))
Automating Least Privilege - Key Takeaways

- Ongoing time requirements from security team: none (some maintenance)
- Security benefit / risk reduction: huge

Any time you can find opportunities like this, you should take them.

=> Maximize ratio of security ROI to ongoing time requirements for sec team
Security EndGame

Automating Least Privilege
Targeting Vuln Classes: Case Study
Enforce Invariants
Data Driven Bug Bounty

Vulns by Category
Help!

“We’re not getting much value out of <popular SAST tool>, can you review how we’re doing things and see what makes sense for us?”
Eliminating Bug Classes: Scoping the Problem

“How do we get more value from <current SAST tool>?”

“How can we make our SAST tool find more bugs of higher criticality with less manual time?”

“How can we find bugs more efficiently regardless of approach?”

At the end of the day:

Given limited AppSec engineer time and budget, what’s the best way for us to reduce overall risk?

Note: Not tool or approach-specific
How to Do This at Your Company

**Aggregate your vulns over the past N quarters**

- Group by vuln class (access controls, XSS, SQLi, open redirect, ...)
- Group by how you found them (pen test, bug bounty, internal testing, ...)
- Weighted by severity/risk/impact
How to Do This at Your Company

Aggregate your vulns over the past N quarters
- Group by vuln class (access controls, XSS, SQLi, open redirect, ...)
- Group by how you found them (pen test, bug bounty, internal testing, ...)
- Weighted by severity/risk/impact

Review the vulns, what can find / prevent them at scale?
- **Goal:**
  - Minimize cost (bug bounty, pen test, tool licensing)
  - AppSec time (finding bugs manually, triaging tool results)
- Is one method finding most of your high severity vulns?
- What’s worked well for your org or team in the past?
  - Any success stories for other vuln classes you can leverage?
Data Driven Bug Bounty

Vuln Source To Resolution
Your AppSec Time

- Threat Modeling
- Running security tools
- XXE
- Secrets in Code
- Triaging bug bounty
- Security training
Your AppSec Time - XXE

Current:

- Developer training: 2 hours / month
- Triaging XXE-related bug bounty submissions: 2 hours / week
- Working with devs to fix XXE issues: 3 hours / week
- Validating XXE fixes: 1 hour / week
- Configuring SAST scans for XXE, triaging results: 1 hour / week
- ... DAST ... : 1 hour / week
- ...
- ...
Your AppSec Time - XXE

**Current:**
- Developer training: 2 hours / month
- Triaging XXE-related bug bounty submissions: 2 hours / week
- Working with devs to fix XXE issues: 3 hours / week
- Validating XXE fixes: 1 hour / week
- Configuring SAST scans for XXE, triaging results: 1 hour / week
- ... DAST ... : 1 hour / week
- ...

**Future:**
- Build a safe by default wrapper library for parsing XML that disables DTDs (External Entities)
- Teach devs about the new library and roll it out everywhere
- Scan every PR for XML parsing that doesn’t use this library
- **Done**
Ban Footguns: How to standardize how devs use dangerous aspects of your framework | ShellCon ‘19

Morgan Roman on how DocuSign eliminated Regex DoS, XXE, open redirects, SSRF via secure-by default libraries + ensuring their use.
Your AppSec Time

- Threat Modeling
- Running security tools
- XXE
- Secrets in Code
- Triaging bug bounty
- Security training
Security Engineering is the compound interest of security.

@clintgibler
Targeting Vuln Classes - Key Takeaways

- Use vulnerability history to determine where to invest effort
- Consider: vuln classes, what’s finding them, impact
- What’s the most effective way (AppSec time / $) to reduce risk?
- Solving bug classes amplifies your effectiveness
Security EndGame

Automating Least Privilege
Targeting Vuln Classes: Case Study
Enforce Invariants
Enforce Invariants

Core idea: Enforce/Alert on things that should always or never be true

AWS instances should never be accessible on all ports to the whole Internet

```
Improving Cloud Security Visibility with ChatOps - Lambda auto shuts it down
```

Manual changes should never be made through the AWS Console

```
Detecting Manual AWS Console Actions (CloudTrail) Arkadiy Tetelman
```

We should never use these <regions> or <cloud services>

- Instances in other regions or use of other services are red flags
- Can go further and do this per-app / service
- Netflix’s Layered Approach to Reducing Risk of Cred Compromise
Enforce Invariants - Key Takeaways

What are things in your environment that should **always** or **never** be true?
- Cloud, security controls, code, AuthN/AuthZ, users, ...

Which of these can you programmatically **enforce** or **alert** on?

No context to make the decision -> **No operational time** for security team
Enforce Invariants - Key Takeaways

What are things in your environment that should always or never be true?
- Cloud, security controls, code, AuthN/AuthZ, users, ...

Which of these can you programmatically enforce or alert on?

No context to make the decision -> No operational time for security team

Enforce Invariants - Key Takeaways
How is what we’re doing now making us more effective in the future?

If it’s not, consider deprioritizing.
Agenda

- **Big Picture**
  - Mindsets and Principles
  - Choosing How to Invest Your Time
- **Scaling Your Company’s Security**
  - The Fundamentals
  - Scaling Your Efforts
  - Security Endgame
- **Action Plan**
Action Plan: Apply

Next Week 3 – 9 Months Future
Apply – Next Week – Assess

- Evaluate your fundamentals
  - Vulnerability Management, Continuous Scanning, Asset Inventory
- Brainstorm with security leadership and SMEs
  - What vulnerability and asset inventory info would we like?
  - Promising projects?
Apply – 3 to 9 Months – Build

- Nail the fundamentals
  - Build a solid foundation

- Do a bite-sized project
  - Based on historical vuln data and your company’s business factors / risk
Apply – Future – Scale

• Be more highly leveraged with your time (Target vuln classes)
• Invest in projects with high security ROI & minimal ongoing time requirements (Automating Least Priv, Invariants)
• Focus on areas that meaningfully raise your security bar
Wrapping Up
What I Learned Watching All 44 AppSec Cali 2019 Talks

2 Days | 4 Rooms

~32 Hours of Talks
As a busy exec who’s heart is still deep in tech. I have found it almost impossible anymore to keep up with latest good tools/talks in infosec. I have to give a shoutout to Clint Gibler’s newsletter tldr;sec which gives me a weekly email that is a curated view of the best stuff. It is absolute gold - keep up the good work Clint. I highly recommend people sign up:
DevSecOps State of the Union

  @clintgibler
- https://www.linkedin.com/in/clintgibler/
- Uplevel your security knowledge: https://tldrsec.com

- **Big Picture**: Mindsets & principles, using the right tool for the job
- **Fundamentals**: Vuln management, continuous scanning, asset inventory
- **Scaling Security**: Threat modeling, security engineering, detection & response
- **Security Endgame**: Automating least priv, targeting vuln classes, invariants

Next Week
Assess

3-9 Months
Build

Future
Scale
Thanks

- Big thanks to all of the people and companies behind the talks, blog posts, and tools referenced - you’re pushing the industry forward!
- Special thanks to Barbara Schachner, Emil Vaagland, John Melton, Aladdin Almubayed, Julian Berton, Scott Behrens, Travis McPeak, Doug DePerry, Leif Dreizler, David Scrobonia, Marco Lancini, Chris Horn, and many others for feedback