Reasons Developers Struggle with AppSec and How to Make it Easier

Hint: None of the reasons are “Developers Don’t Care about Security”
$ whoami

Scott Gerlach

- CSO/Co-Founder StackHawk, Inc
- CISO @SendGrid - 3 years
- Sr. Security Arch @GoDaddy - 9 years
- Husband, Dad, Brewer, Golfer, tinkerer
- @sgerlach
- linkedin.com/in/scott-gerlach-kaakaww
AppSec Problem Overview

**AppSec** = Good; In Theory

Static Code Analysis
- Noisy, often lacks Application Context
- Language Dependant (Don’t get me started on IDE support)

Dynamic Code Analysis
- Better at actual app and context, but still somewhat noisy
- Hard to use

RASP, IAST, WAF
- Wait til someone/something else finds it... in Prod
Problem One:
The Benevolent Security Team
Or Lack Thereof
I wouldn't want a Junior Developer making security decisions.

Trust Issues

Charlie Miller @0xcharlie · Feb 23
Replying to @coleencoolidge and @fredrickl

i’m not sure if a new hire dev is in a position to evaluate risk for a feature, product, or company. i think professional security people can do this better?
Trust and Support

“I wouldn’t want to put a new hire Developer in the position of making an uninformed risk decision”

-Scott Gerlach
Let’s Teach Them AppSec

If they know how attackers think, they’ll be able to test like an attacker - Hack Yourself!

- Here’s 11ty Billion new Acronyms to learn
- Also, let’s talk about risk
- But wait before that, do you know the Internet is a bad place?
- *If you have sent any of your Devs to a Security Training program, who usually gets selected?*
“We Need to Model Out a Price Increase”

Have you ever seen the FP&A team teach the basics of accounting to the Exec Team

ACCOUNTING 101

CHAPTER ONE:
Asset, Liability, Owner’s Equity, Revenue, and Expense Accounts
The Chase to Perfection

- Find 11ty Billion issues -> We have to fix all of these!
- Why? What is the actual risk in the context of the business?
- What if your QA filled 1,000 tickets for bugs that are unlikely to degrade user experience?

“I’ve never had a satisfying conversation on why a security issue is ever more important than a feature. Ever.” - Product VP
If You Don’t Have a Security Team

- Where do I start?
- OMG, forget it. I have other stuff to do.
- Wait, maybe the DevOps team can handle it
Problem Two: AppSec Tools are built for Security Teams
Hey! I broke the crap out of your thing. Cool huh!

I think we’ve got a SQL Injection here
CSRF: Cross Site Request Forgery

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker’s choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth.

XSRF: Cross Site Request Forgery

We couldn’t decide which one to use, so we use both interchangeably. Isn’t that cool? Anyway, I hope some of this text helps you understand what this means because we wrote a lot of it.
There are Good AppSec Dev Tools Out There

Developer native tools (in context, how they work)

- Snyk
- Fossa
- npm audit
- GitHub (package inspection, PR Bot)
- OWASP Dependency Check
Lastly, and worst of all, they all suffer from....
Problem Three: The Production Bias
Examining the Production-Bias: People

**The Security Team**
Production is where they know the app the best

**Pen Tester**
Production is their only point of access

**Primary Value:** These groups are very focused on the “finding” of vulnerabilities/security bugs. MOAR findings = MOAR better.

**Repercussions...**
- More focused on the numbers of things found, than finding and fixing the right things
- Inefficient — the “finders” are not the “fixers”
- Reinforces an adversarial relationship — “Hey look, I broke your stuff”

*Assuming you have a security team*
Examining the Production-Bias: Timing

Security is either a blocker or “playing catch up”
Also, there’s a major problem with appsec tools that favor running in production...

THE BUGS ARE IN PRODUCTION.
Getting Started:

The Right Way
How Test-Driven Security Should Work

When a team writes code, they know the syntax is wrong when it won’t compile.

When a team merges code they know there is a problem when it doesn’t merge.

When a team runs unit tests, they know the code is wrong when it fails the unit test.

When a team runs integration tests, they know the code is wrong when it doesn’t work as designed.

When a team introduces a vulnerability, they know when it fails a security test.
Local Dev & CI/CD

Instrumenting Security Tests into CI/CD gives engineers immediate feedback.

Adding the ability to test locally allows for quick iteration in the fix-test loop if a new bug is identified.
Engineers Are Smart: Let Them Be Smart

- Security Teams tend to want to “Approve” everything: What that means is other people can’t make decisions

- Allow technology to spark collaboration between Development and Security but enable Devs to do their work

- Business Risk is a collaboration, not a one team knows the answer game
Just Start!

- Engage an Engineering Team and their pipeline
- Choose an app or service to start
- Choose a technology (SCA, DAST)
- Iterate and expand
Thanks!

scott.gerlach@stackhawk.com
@sgerlach
https://stackhawk.com