How to Create a Secure Code Pipeline

Robert Haynes
James Brotsos
Your Presenters:

Robert Haynes
Product Marketing Manager

James Brotsos
Product Manager
Code: From Concept to Deployment

Design
Create
Commit
Build
Pipeline
Release

Developer
IDE
SCM

‘The Wild’
Application Security Threats

**MISTAKES**
- Input validation
- SQLi
- XSS

**MALICE**
- Typosquatting
- Malicious Takeover
- Artifact store poisoning

**MISCONFIGURATION**
- Over-privilege
- Broken access control
- Resource Access
Application Security Testing

EARLY
Shift as far left as possible
- Into the IDE
- Into the SCM

OFTEN
Test at multiple stages
Test multiple code types

EASY
Automate!
Automate!
Automate!
Code: From Concept to Deployment

Train
- Developer
- IDE
- SCM

Test

Test

Test

Test

Pipeline

Protect

‘The Wild’
What to Test?

- **CODE YOU WRITE**
  - Most Common

- **CODE YOU USE**
  - Fairly Common

- **INFRASTRUCTURE CODE**
  - Less Common
The Code You Write

Test the code you write for security errors

Sanitize inputs

XSS Errors

Safe deserialization

Secrets management

Parameterize queries

Robust authentication

Use a high-quality Static Code Analysis (SAST) tool
The Code You Write

Describe Attack Vector

Point to Place in Code

Identify Language and Vulnerabilities

Best Fix Location

```
// bookstore/BookDetail.cs
public class BookDetail
{
    public void Describe
    {
        // Code
    }
}
```
The Code You Use

Most codebases contain open source

Open Source is 60-70% of the overall code
Mistakes and Malice
The Code You Use

~ 2 Million packages available just in these selected managers
How to Manage Open Source Risks

SCAN EVERY LIBRARY?
- Impractical

SCAN APPROVED LIBRARIES?
- Restrictive

USE SOFTWARE COMPOSITION ANALYSIS?
- Effective

Use a high-quality Software Composition Analysis (SCA) tool
Things you need to know

- Package Vulnerabilities
- License details
- How to fix the problem
- How difficult a fix?
- Where it's used
The Code that Creates the Infrastructure

"To make error is human. To propagate error to all server in automatic way is #devops."
What kind of misconfigurations?

**Excess Privileges**
- Machines with too wide a scope
- Containers running as root

**Access Control**
- Admin accounts in use
- Open Ports
- Storage buckets open

**Logging**
- Flow logs
- Access logs
The Code that Creates the Infrastructure

Text Based
Set schemas/languages
Great candidate for scanning!

Use a high-quality Infrastructure as Code scanning tool
The Code that Creates the Infrastructure
Things you need to know

**Risk Levels**

**Risk Types**

**Vulnerabilities:**

- **Privilege Escalation Allowed**
  - Platform: Kubernetes
  - Category: Insecure Configurations
  - Containers should not run with allowPrivilegeEscalation in order to prevent them from gaining more privileges than their parent process.
  - [Link](https://kubernetes.io/docs/tasks/configure-pod-container/security-context/)
  - Results: 1

- **CPU Limits Not Set**
  - Platform: Kubernetes
  - Category: Resource Management
  - CPU limits should be set because if the system has CPU time free, a container is guaranteed to be allocated as much CPU as it requests.
  - [Link](https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/)
  - Results: 1
Summary

Use a high-quality Static Code Analysis (SAST) tool

Use a high-quality Software Composition Analysis (SCA) tool

Use a high-quality Infrastructure as Code (IaC) scanning tool

OK, so how?
Point or Platform?
Platform: One trigger, multiple scans

- **Developer IDE**: Initiate scans from CLI
- **SCM**: Integrate w/ repo of choice
- **CI/CD Pipeline**: Run as part of CI/CD pipeline, on every pull request
- **Security Check**: Run prior to deploying to production
- **Ticketing System**: Integrate w/ issue tracking tools

**Flow Diagram**

1. **Design**
   - Developer IDE: Initiate scans from CLI
2. **Code**
   - SCM: Integrate w/ repo of choice
3. **Check-in**
   - CI/CD Pipeline: Run as part of CI/CD pipeline, on every pull request
4. **Build**
   - CxAST Cloud
5. **Test/QA**
   - Security Check: Run prior to deploying to production
   - Ticketing System: Integrate w/ issue tracking tools
Use a high-quality Static Code Analysis (SAST) tool

Use a high-quality Software Composition Analysis (SCA) tool

Use a high-quality Infrastructure as Code (IaC) scanning tool

Use a consolidated Application Security Testing (AST) Platform
What about the first step?

Train

Targeted
Short
Just in time
Summary

Use a high-quality Static Code Analysis (SAST) tool
Use a high-quality Software Composition Analysis (SCA) tool
Use a high-quality Infrastructure as Code (IaC) scanning tool
Use a consolidated Application Security Testing (AST) Platform
Continuously train your development team
Questions?
Thank you.