Secure software development

An overview of building security into application software systems

Chris Horn

April 2019
About Chris Horn

Principal consultant,
Product strategy & development

Experience
- 18 years in research, software systems, and new product development
- Principal Investigator at Secure Decisions, an R&D division of Applied Visions
- Focused on developing technologies to improve application security
We provide software development, security research, and application security management.

Applied Visions, Inc.
- Software development since 1987
- Primarily develops business applications

dba, Secure Decisions
- Cyber R&D
- Primarily serving DHS and DoD, including DARPA, ONR, AFRL; some intel and commercial projects

Code Dx, Inc.
- Application security spin-out based on SBIR funded by DHS S&T in 2011
Outline of today’s talk

Baseline
- Systems engineering
- Security

DevSecOps
- Overview

for each stage of DevSecOps
- Security practices
Three key takeaways from today’s talk

1. Security is perfectly compatible with DevOps

2. There is no silver bullet to achieve security
   - Results from hundreds of smaller decisions and actions
   - Coordinated application of people, process, and technology

3. There are many great public resources to support learning
   - Citations & links included
Baseline
People apply thought, process, and technology to create application software
Application systems are composed of components that each have their own security needs.
Systems engineering integrates multiple skillsets

The art and science of guiding the end-to-end creation of systems

- Art because it involves extensive people skills and leadership
- Science because it requires rigorous applications of tools & methodologies

Interdisciplinary approach governing the total technical and managerial effort required to transform a set of stakeholder needs, expectations, and constraints into a solution and to support that solution throughout its life.


There are many ways a system can fail
Security feels good

The state of being free from danger or threat

Freedom from those conditions that can cause loss of assets with unacceptable consequences

An asset is an item of value to stakeholders that may be:

- **Tangible** (e.g., a physical item such as hardware, firmware, computing platform, network device, or other technology component)
- **Intangible** (e.g., data, information, software, trademark, copyright, patent, intellectual property, image, or reputation)

Security is an emergent system property

Emergent properties are the opposite of one-and-done feature development

- Result from tens and hundreds of smaller decisions

Other emergent system properties include:

- Availability
- Usability
- Safety
- Security
- Maintainability
- Resilience
- Reliability
- Agility
- Survivability
- Recoverability
- Supportability
- Durability
DevSecOps
DevSecOps is a framework

DevSecOps is a collection of ideas and approaches that represent an evolution in how we develop systems.
DevSecOps is about culture

DevSecOps values, beliefs, attitudes, and behaviors include:

- Iterative value delivery
- Shared responsibility
- Autonomous teams
- Automation
- Measurement
- Learning & experimentation

DevSecOps is about process
DevSecOps is about technical approaches

Continuous integration (CI) & continuous deployment (CD)
- Cloud technologies
- Everything as code
- APIs
- Automated testing
- Rollback / recovery

DevSecOps is about review processes

DevSecOps is a framework

DevSecOps is an evolution to how we develop systems

Most practices from other frameworks carry through

- Systems engineering
- SDLC
- Risk management
- Cybersecurity

DevSecOps is how we apply those practices
Resources for learning about DevSecOps


“Introduction to DevSecOps, Refcard #267.” DZone. Accessed April 1, 2019. 
https://dzone.com/refcardz/introduction-to-devsecops.
Security practices in DevSecOps
DevSecOps

1. Plan & design
2. Code
3. Build
4. Test
5. Release
6. Deploy
7. Operate
8. Monitor
The development pipeline is like a series of screens.

- Early in pipeline:
  - Identify & resolve major, overarching issues

- Late in pipeline:
  - Identify & resolve relatively small, detailed issues

Aim to detect issues as early as possible.
Plan & design
Important design principles, patterns, & methods

Principles of secure design

- Compartmentalization
- Minimize attack surface
- Defense in depth
- Economy of mechanism (KISS)
- Authenticate, then authorize
- Least privilege
- Separation of privilege
- Graceful degradation
- Safe defaults
- Audit trails & logging
- Open design
- ...more

Design patterns

- Threat modeling
  - Abuse cases
  - Attack graph/map
  - Fault trees

Attacker mindset

https://www.us-cert.gov/bsi/articles/knowledge/principles/design-principles.


https://drive.google.com/file/d/1qF70eYdWhueNmmdmy2fuYXDOr1fkXZZd/view.
Healthy paranoia requires knowing the threats

Types of problems
- Physical damage
- Loss of essential services
- Technical failure
- Function compromise
- Information compromise

Degrees of intent
- Deliberate
- Negligence
- Accidental
- Environmental

Human adversaries
- National governments
- Terrorists
- Industrial spies
- Organized crime groups
- Hacktivists
- Hackers
- Insiders
  - Disgruntled
  - Financially-motivated


Threat catalogs are useful input to threat modeling


Control catalogs can save on design time

CIS Controls

OWASP ASVS

NIST 800-53 Rev. 4

DoD Instruction 8500.2


Code & build
Checklists and analyzers improve code quality

Apply commonly needed practices

- Sanitize/validate all inputs
- Parametrize queries
- Prevent XML external entities
- Prevent cross-site scripting (XSS)

Use static analyzers

- Enable compiler warnings
- Adhere to a style guide
- Find analyzers that fit your needs with Kompar (https://kompar.tools)


Code reviews are an effective way to catch bugs

Code reviews answer two questions:
- Is it the correct code?
- Is the code correct?

Typical workflow
1. Developer edits code
   - Usually in a feature branch
   - Not for longer than a couple of weeks
2. Initiates a pull request
3. Someone else reviews code
4. Request changes, or approves merge


Secure configuration

Third-party component configuration guidance
- CIS Benchmarks
- DISA Security Technical Implementation Guides (STIGs)

Manage components with known vulnerabilities
- Software composition analysis
  - Dependency-Track, Black Duck, WhiteHat, CA Veracode, etc.
- Apply vendor security updates


Test

There are multiple types of testing to apply

Unit
Integration
Static analysis
Dynamic/interactive application security testing
Abuse cases
Fuzz/robustness
Infrastructure security
Penetration


Selectively apply testing by pipeline stage

- **CODE**
  - Lightweight static analysis (linters, style checkers, etc.)
  - Unit tests

- **COMMIT**
  - Automated
    - Lightweight static analysis (linters, style checkers, etc.)
  - Manual
    - Code review

- **BUILD**
  - Synchronous
    - Unit tests
    - Integration tests
    - Midweight static analysis
    - Software composition analysis (SCA)

- **TEST**
  - Synchronous
    - Configuration compliance
    - Abuse case
  - Asynchronous
    - Dynamic/interactive application security testing
    - Heavyweight static analysis
    - Fuzz/robustness
    - Penetration testing

Synchronous tests should complete in a reasonable amount of time and the results used to decide whether to break the build. Do not go live with a build if it has serious quality errors based on synchronous tests, but many organizations will go live with known security vulnerabilities in their applications.

Application security management systems consolidate testing findings

Application security testing will generate lots of data in different formats and nomenclatures

- Collect, normalize, and deduplicate with a management system
  - E.g., Code Dx, ThreadFix, Defect Dojo, Kenna Security, etc.
Release & deploy
Deployment is a high privilege

Limit who is authorized to deploy builds

Audit all deployment activity

Consider additional technical controls
  ▪ Binary authorization

Operate & monitor
NIST decomposes cybersecurity into five functions

Operations focuses on

- Detect
- Respond
- Recover

Three key takeaways from today’s talk

1. Security is perfectly compatible with DevOps

2. There is no silver bullet to achieve security
   - Results from hundreds of smaller decisions and actions
   - Coordinated application of people, process, and technology

3. There are many great public resources to support learning
   - Citations & links included
Chris Horn
Principal consultant, Product strategy & development
chris.horn@securedecisions.com

https://avi.com
https://securedecisions.com
https://codedx.com