Welcome!

Intro to DevSecOps

Hello

About me...

- 30 years in the industry
- 25 years in teaching
- Certified Cloud architect
- Passionate about learning
- Also, passionate about Reese's Cups!

Agenda

- Learning objectives
- CI/CD as a capability / differentiator
- DevSecOps – what it is and why it is needed
- Practicing DevSecOps at each phase of the SDLC
- OWASP Top 10

How We’re Going to Work Together

- Slides and words to highlight key concepts
- Demos to bring those concepts “to life”
- Discussion groups and lab work (which will take place in sandboxes provided via AWS WorkSpaces) for hands-on reinforcement
- NOTE: I welcome being interrupted – if you need more info, or clarification, or anything else, just break in and ask. I am here to help you.

Learning Objectives

Learning Objectives

- Understand key concepts and considerations when leveraging DevSecOps in an Agile environment to help “shift security left” during software development
- Construct and prioritize a threat model for an application being developed and use that threat model as a planning tool to guide each phase of the SDLC in a security-minded manner
- Speak to the value of DevSecOps and its consistent application as a set of standards and best practices

Learning Objectives

- Monitor, patch and scan for vulnerabilities in the Operating System (Windows and Linx) and underlying Infrastructure configuration
- Effectively utilize GitLab for Source Code Management and CI/CD, including:
    - Understanding and navigating practical activities related to source code repositories in GitLab
    - Effectively use Software Composition Analysis (SCA), Static Application Security Testing (SAST), and Dynamic Application Security Testing (DAST)

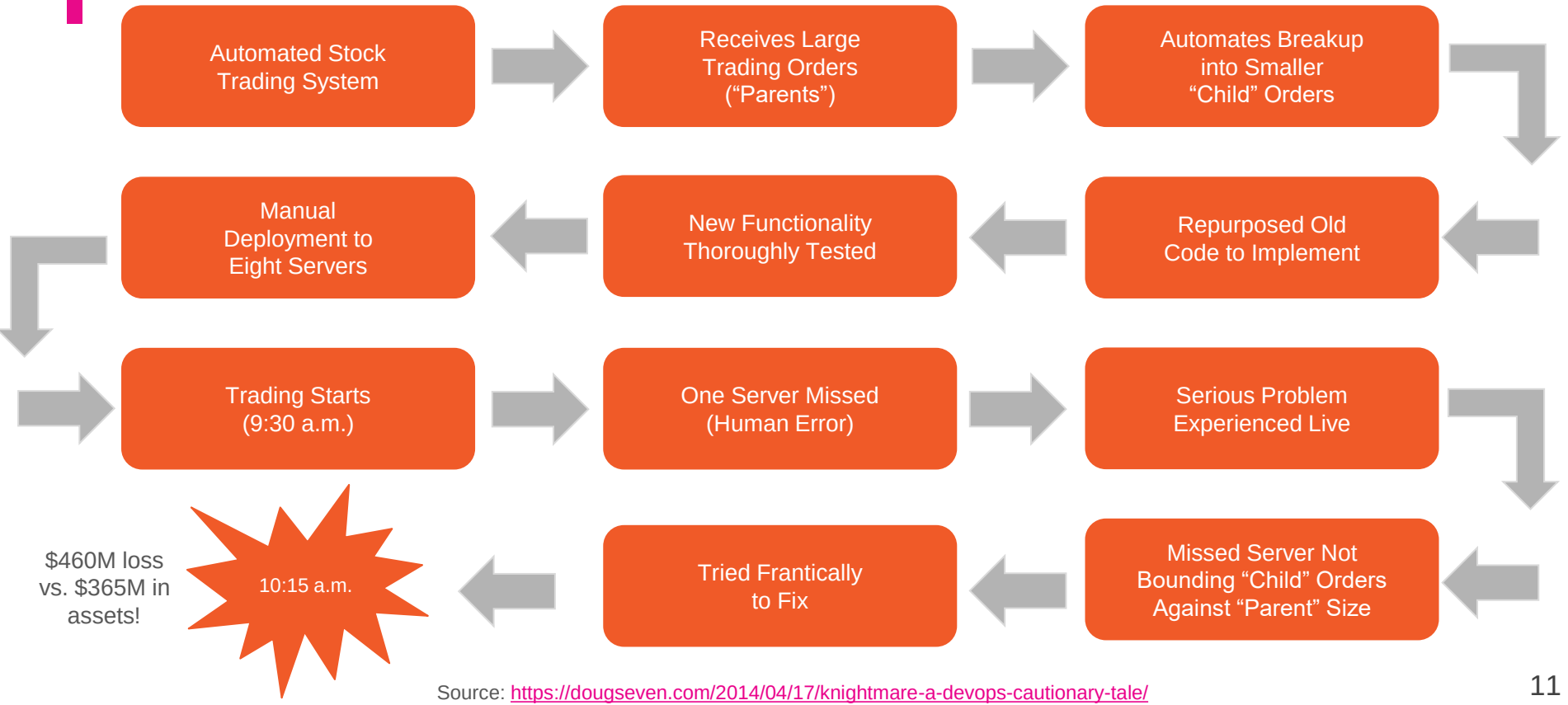
CI/CD as a Capability / Differentiator

What a difference 45 minutes can make!!

How Long Does it Take for a Company to Go Bankrupt?

If the circumstances are right, it can happen in just 45 minutes!

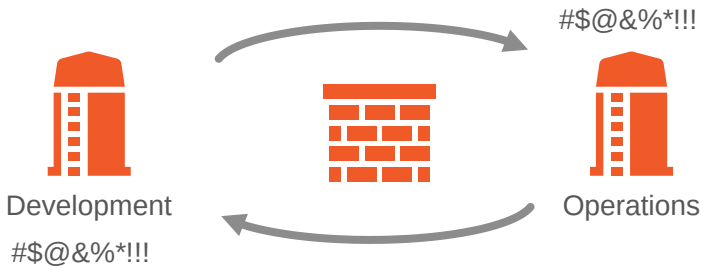
Knight Capital Group – What Happened?



Source: https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/

DevOps – What it is & why it’s valuable

Approach Used In the “Olden” Times



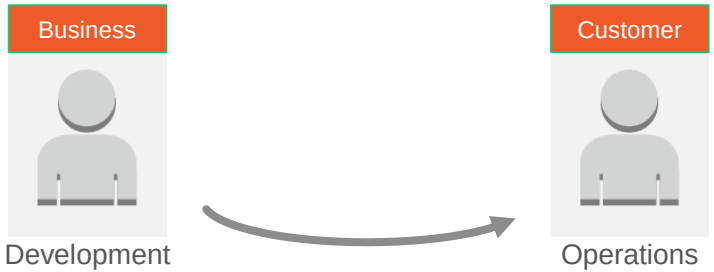
- Development & Operations siloes
- Lack of understanding of the other perspective
- Lack of communication & partnership
- “Fire and forget” engagement

DevOps – The New Way

- Development & Operations work together
- Each understands & appreciates the position of the other
- Good communication, partnership, & collaboration
- Ongoing engagement – continuous improvement

DevOps – The Three Ways

The First Way: Flow/Systems Thinking

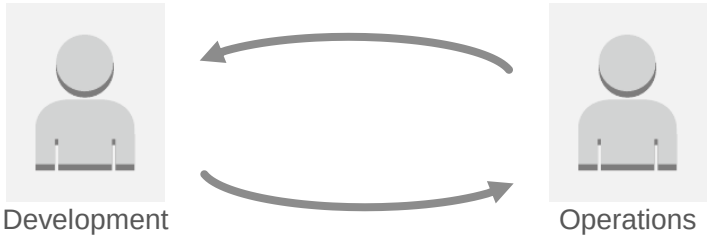


Source: Gene Kim (https://itrevolution.com/the-three-ways-principles-underpinning-devops/)

- Holistic vs. siloed view – success is defined by the performance of the entire system vs. a specific team or individual
- Starts with requirements (defined by the business) and not called “done” until value is delivered to the customer
- Target outcomes:
  - Known defect never passed to downstream teams
  - Favor global performance vs. focusing on local optimization
  - Goal is increased flow (value)
  - Asking ourselves the question: “What don’t I know about our systems?”

DevOps – The Three Ways

The Second Way: Amplify Feedback Loops

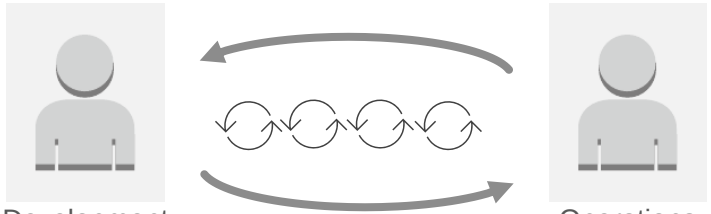


Source: Gene Kim (https://itrevolution.com/the-three-ways-principles-underpinning-devops/)

- Create right-to-left feedback loops
- Shorten & amplify so information needed to make continuous improvements & continuous correction is readily available
- Target outcomes:
  - Understanding all customer perspectives (internal & external)
  - Responding to all customer concerns (internal & external)
  - Knowledge is power!

DevOps – The Three Ways

The Third Way: Culture of Continual Experimentation & Learning



Source: Gene Kim (https://itrevolution.com/the-three-ways-principles-underpinning-devops/)

- Culture that fosters:
  - Continual experimentation, risk-taking, and learning from failure
  - Practice makes perfect!
- Target outcomes:
  - Allocating time for continual improvement (e.g., resolution of technical debt as a standard "category" of work)
  - Taking *calculated* risks and seeing them pay off – No guts no glory!
  - Expecting failure & building resiliency into the process

DevSecOps

The Mindset

DevSecOps

What is it?

Discipline that takes into consideration how People, Processes, and Tools Automation combine to ensure we have:

- Disciplined build of security into all phases of SDLC
- Just-in-time security assessment and testing
- Security as a “shared responsibility”

DevSecOps

Why is it important?

To maintain a competitive advantage:

- We need to deliver software quickly
- We need that software to have high quality
- We need that software to be secure

We want to merge speed & agility with security & quality!

DevSecOps and how can businesses benefit from it?

*DevSecOps integrates software development with IT/SEC operations, leveraging **automation** to **shorten the product life-cycle** and **enable frequent deliveries** aligned with the business needs, while **ensuring quality***

DevSecOps Building Blocks

Continuous integration
Automated testing and single source code mgmt. system

Continuous delivery
Ability to deliver software when ready

Continuous deployment
One-click release of software to live

Continuous monitoring and autonomic operations
Highly automated IT Operations, including security, vulnerability scanning

Standardized and automated infrastructure mgmt. with built in security
Automate environment with infrastructure-as-a-code (IaC),

Impact Levers

Speed
Release in days, not months

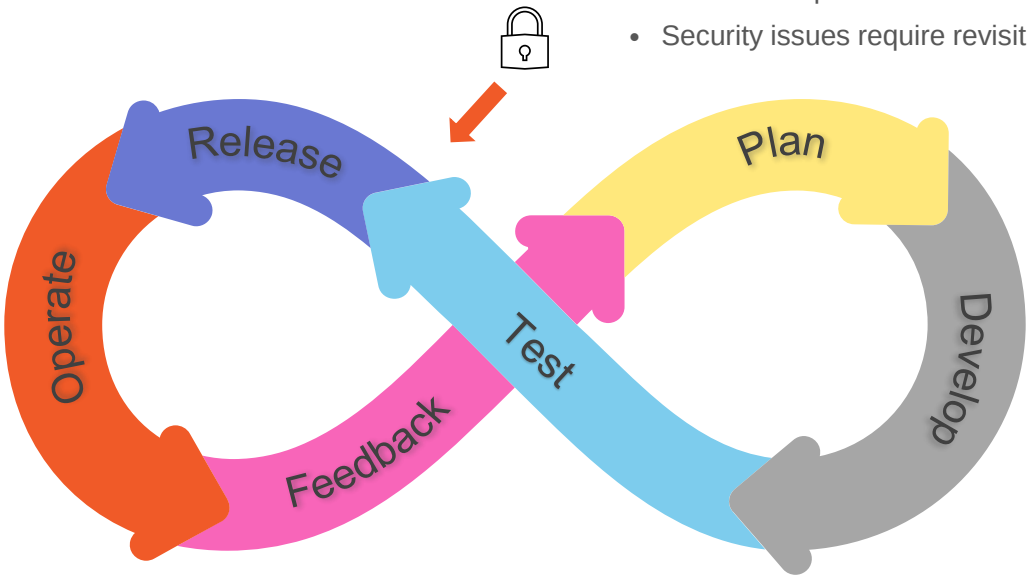
Quality
Enable speed and enhance stability with rigor and confidence

Efficiency
Operate in automated environment and focus on high value tasks

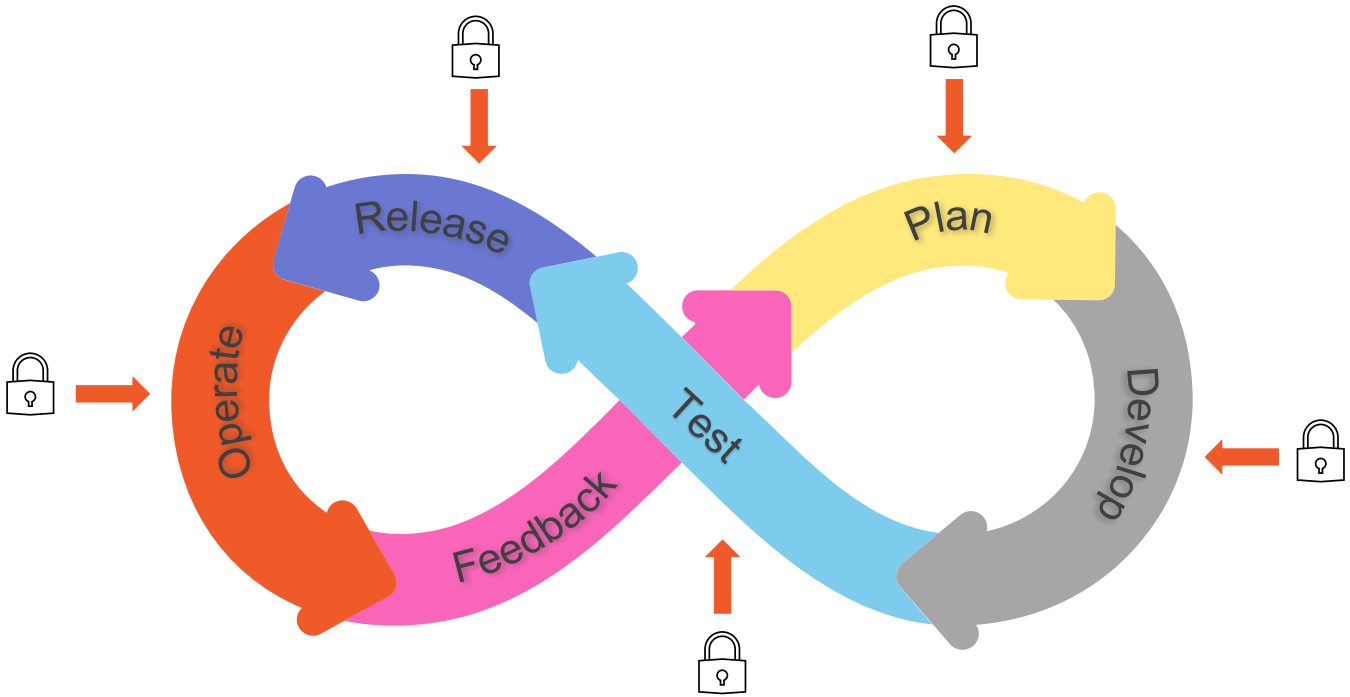
Security
Baked-in and fully integrated into SDLC

Security in Software Engineering – The Traditional Way

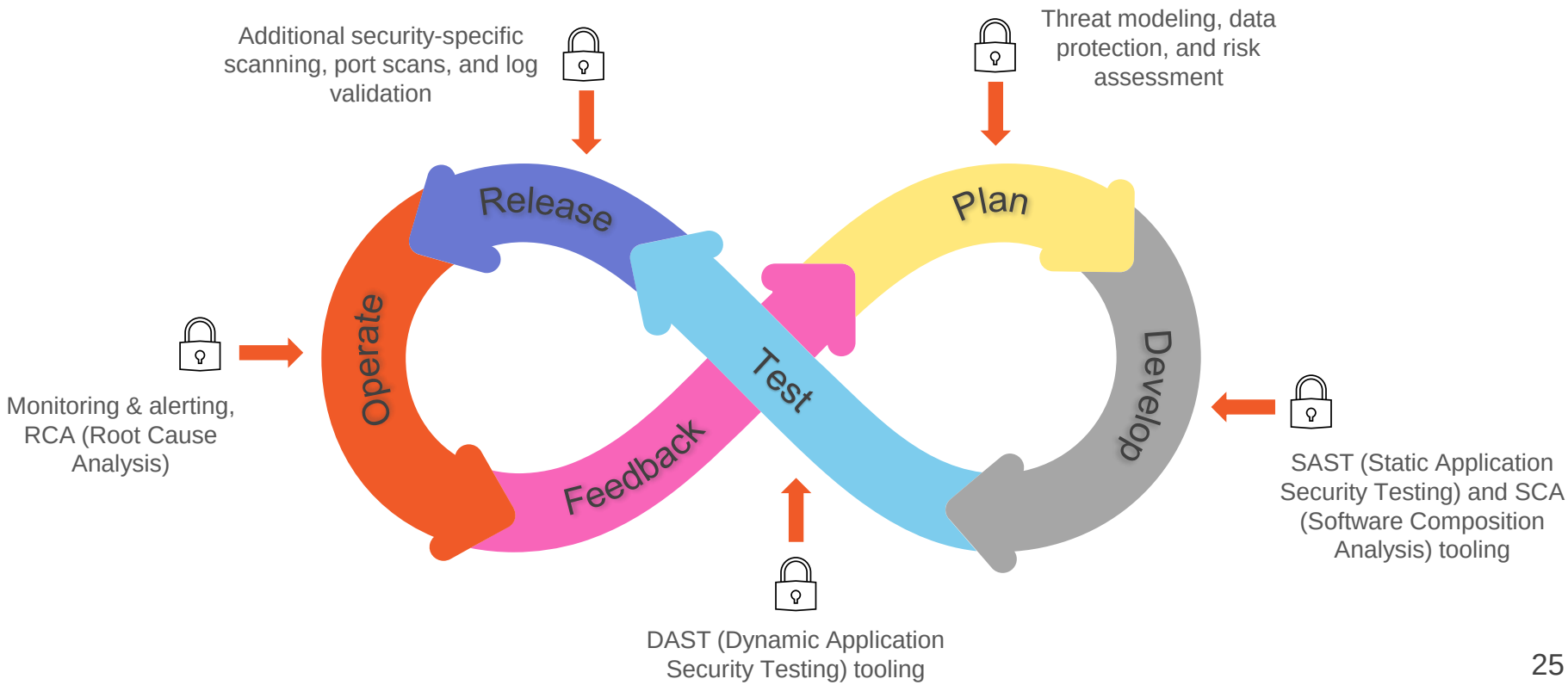
- Not enough time in schedule to absorb changes required to remediate
- Activities required to remediate may be complex
- Security issues require revisit of one or more previous phases



Security in Software Engineering – A Better Way

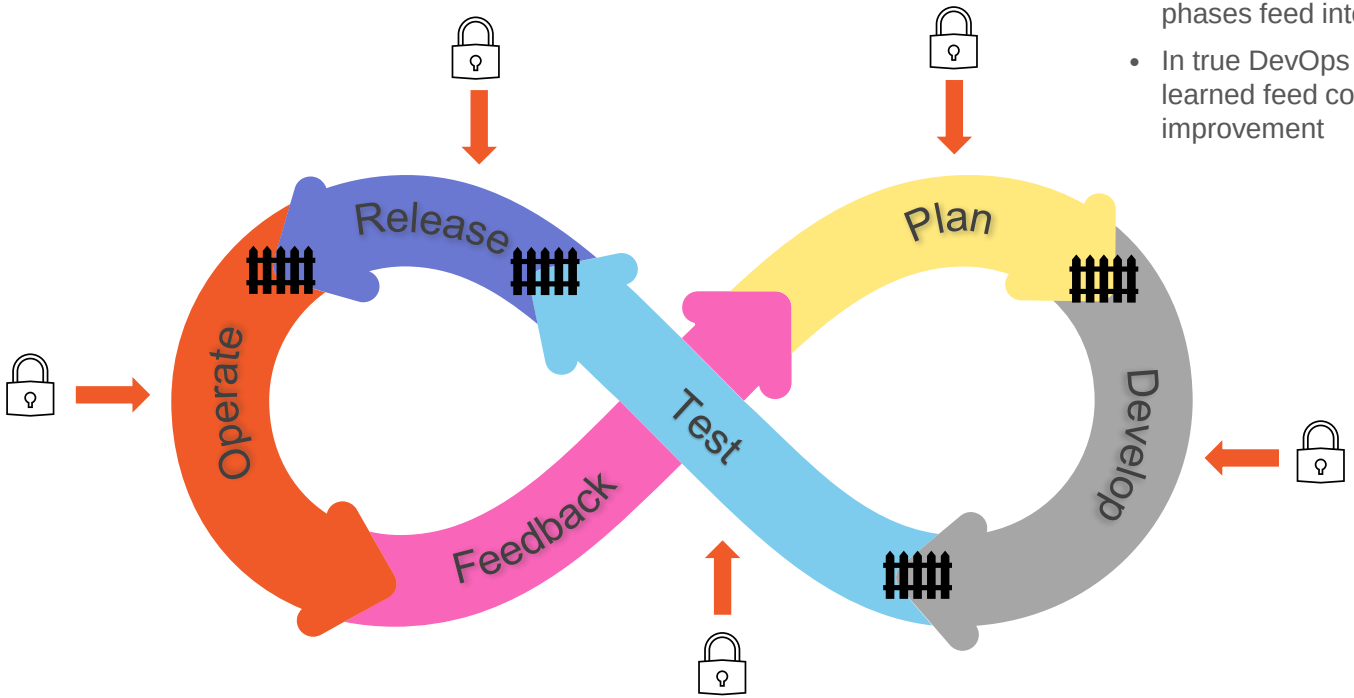


Security in Software Engineering – DevSecOps



Security in Software Engineering – DevSecOps

- Information gathered from early phases feed into later phases
- In true DevOps fashion, lessons learned feed continuous improvement



Quality gates guard against moving security defects forward

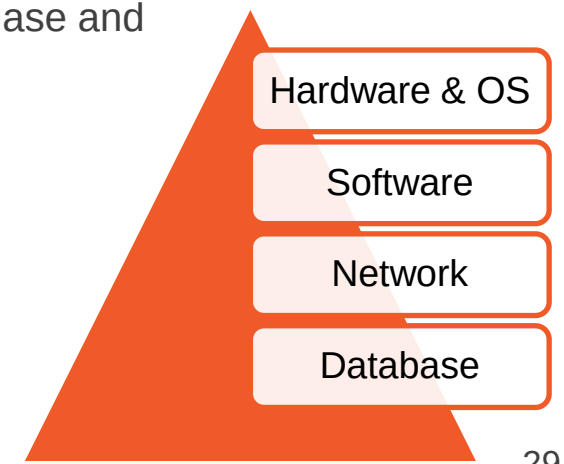
DevSecOps in Plan

Assessing Security Risks (Plan Phase)

- To secure a solution, attack surfaces and potential threats must be identified
- Common practice utilizes something called threat modeling
- Includes modeling and analyzing possible attack vectors based on application profile

Assessing Security Risks

- Risk assessment should account for different “zones” of execution
- Security requirements must be understood in context of specific use cases
- Ideally, threat modeling would be executed during the plan phase and results would be used to inform other phases



Threat Modeling

Can be viewed in two different, but related, contexts:

- Implementation of controls mapped to security requirements & policy (prevention)
- Implementation of countermeasures against possible known attacks (remediation)

Threat Modeling

Multiple approaches:

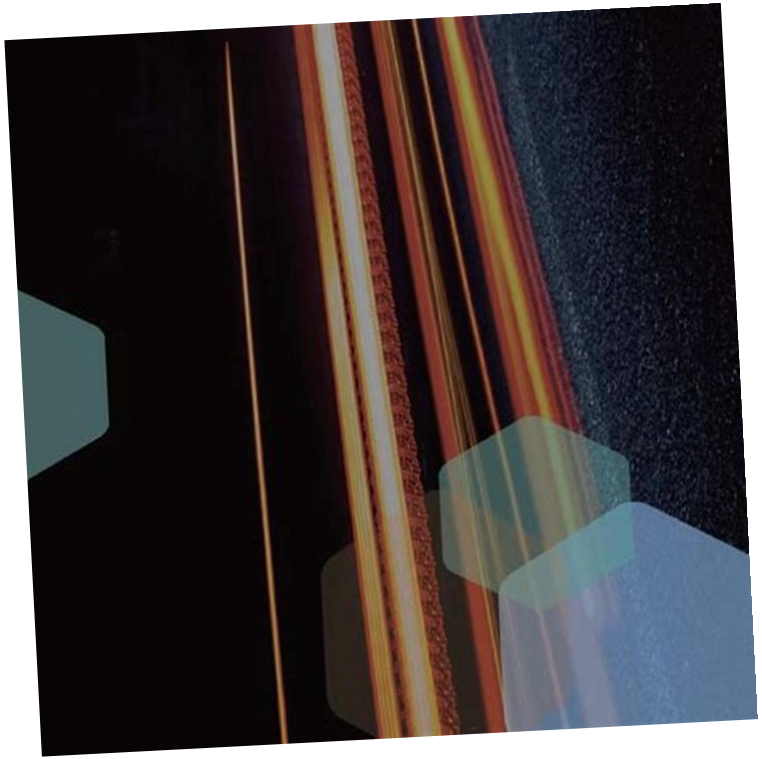
- Attacker-centric (think like an attacker!)
- Asset-centric (what do we have to lose?)
- Application-centric (what are we building & testing?)

Threat Modeling – Key Considerations

Valuable principles:

- Defense in Depth
- Principle of Least Privilege
- Secure by Default

Overview of the STRIDE Methodology for Threat Modeling



- **S** poofing
- **T** ampering
- **R** epudiation
- **I** nformation Disclosure
- **D** enial of Service
- **E** levation of Privilege

Spoofing

- Illegally accessing another user’s identity or authentication information
- Used when interacting with a system to make the system think you’re someone else (with permissions that you don’t actually have)
- Can include things like username and password, IDs, access keys, passkeys, etc.

Tampering

- Malicious modification of data
- Can include interception and modification in-transit
- Can also include interception and modification at rest (like in a database)

Repudiation

- Taking advantage of vulnerabilities in a system or workflow that allows a user to take an action without the system being able to confirm, audit, or prove the action
- In a DevSecOps context, allowing a user to take a prohibited action or access secure information anonymously and with deniability

I nformation Disclosure

- Exposure of sensitive information that a user or party should not have access to
- Extremely harmful when juxtaposed against regulations like HIPAA or PCI

Denial of Service

- Involves an attacker sending significant amounts of “bogus” traffic to your system or service
- System becomes overwhelmed and is unable to handle real traffic
- Goes to a site or service’s reliability and availability

Elevation of Privilege

- Allowing an unprivileged user to gain privileged access to a system
- Failing to minimize zones of implicit trust such that once an attacker has invaded, they are granted more access than they should have

Defining Objectives and Scope

01

Critical Asset Identification

Establish a clear inventory of assets requiring protection, ensuring that security measures are focused on the most valuable components of the system

02

Threat Environment Analysis

Assess the external threat landscape, including potential adversaries and attack vectors, to inform the scope and focus of the threat modeling process

03

User Role Specification

Clearly define user roles and their permissions to identify potential security vulnerabilities related to unauthorized access and privilege escalation

Creating a System Architecture Diagram

01

Visual Representation Benefits

A System Architecture Diagram provides a clear visual representation of system components and their interactions, enhancing stakeholder comprehension and facilitating effective communication across diverse teams

02

Security Analysis Framework

The diagram serves as a foundational tool for security analysis, enabling teams to systematically identify and prioritize potential threats by mapping data flows and trust boundaries within the system

Identifying Assets & Entry Points

Comprehensive Asset Inventory	Entry Point Risk Assessment	Continuous Monitoring Practices
• Conduct a thorough inventory of all assets • Categorize according to sensitivity and criticality • Use to drive protection focus	• Evaluate each identified entry point for potential vulnerabilities • Implement security measures tailored specifically to mitigate risks associated with unauthorized access	• Establish ongoing monitoring of assets & entry points • Include change and emerging threat detection • Helps drive timely updates to security protocols and strategies for response

Prioritizing Threats

MITRE ATT&CK Framework

- Can be combined with STRIDE (or other frameworks) to help drive a deeper, more detailed analysis of and plan for potential threats
- Provides a comprehensive set of potential attack vectors to consider in multiple areas of focus
- Also, provides information on examples of the attack, mitigation strategies, and detection strategies

LAB:

Mitre ATT&CK Framework

In your assigned breakout room, visit https://attack.mitre.org and complete a high-level review of the set of security concerns and the associated techniques listed there. Each team will pick 3 of the techniques (pick 3 that your team finds most interesting or considers most relevant). Drill down into each selected technique and, as a team, document the following for “teach back” to the larger group:

- Brief description for the attack technique
- Detail on one mitigation (if any exist for the technique)
- Detail on one detection approach

Nominate someone from your breakout room to gather/document the results of the discussion. That person will present to the larger group when we come back together from our sub-teams.

LAB:

Threat Modeling

In your breakout room, review the assigned scenario. Complete the following:

- Review the scenario
- Identify critical assets in scope
- Create a high-level system architecture diagram
- Assess the internal and external threat landscape – brainstorm, discuss, and document as many potential threats and attack vectors that you see in the defined architecture (use STRIDE and Mitre ATT&CK to assist)
- What kind of monitoring would you recommend?
- Finally, as a team, prioritize each potential threat relative to the others by assessing risk vs. impact

Don't forget to include the following considerations:

- Hardware & OS
- Software
- Data
- Network

Each team will present their architecture diagram and their prioritized list of potential threats/attack vectors to the larger group.

DevSecOps in Development

Shifting Security Left – Develop

- Ideally, security flaws in code would be caught on a developer’s machine
- Can use plugins for IDEs to provide automated static analysis
- In some cases, tools can be configured with custom rules and priorities

Shifting Security Left – Develop

- Rulesets and priorities should be driven by results of threat modeling
- Scanning tools if used should be a standard part of all developer's code tooling, to ensure consistently applied (standard dev build)
- Goal is to catch issues VERY early on in the SDLC

Moment of Reflection

What is the most “secure” programming language in use in the industry today?

Type your answer in chat

DevSecOps in Build

Shifting Security Left – Build

- Changes from a developer are ready to be integrated with others
- Typically driven through submission of Pull Request in GitHub
- Notifies a peer that changes are pending and need review

Shifting Security Left – Build

- In addition to logic/syntax errors, peer reviewer can focus on areas of high risk/high impact identified in threat modeling
- In some cases, a specialist peer reviewer (e.g., one from InfoSec) can be engaged to validate security of code

Shifting Security Left – Build

- Coupled with manual review, SAST (Static Application Security Testing) tools can be integrated into CI/CD to automate security checks
- Tools like Veracode can be integrated into pipeline via plugins (https://docs.veracode.com/r/c_integration_buildservs)
- Some even include ability to capture scan results in resulting build report and use results of scan as quality gate to prevent move forward on failure

Shifting Security Left – Build

- SCA (Software Composition Analysis) tools also provide benefit
- Tools like Mend.io (https://www.mend.io/open-source-security) can be integrated into CI/CD for automated security scans of Open-Source components
- Ideally, governance would be in place to monitor and manage “accepted” set and versions of Open-Source tools approved for usage

Shifting Security Left – Build

- SCA tools can provide a couple of benefits
- Primarily, security scans can be executed, and Open-Source vulnerabilities identified
- Secondarily, can help an organization catalog the Open-Source components (and versions) “in play” already

Shifting Security Left – Build

- Resulting report can be used to identify components that have not been properly vetted through governance
- Results can be merged with overall build output
- Quality gates (manual or automated) can be built around results to prevent move forward if insecure

Moment of Reflection

Answer the following question: Using an open-source library is more secure than building something custom? Yes or no, and why.

Type your answer in chat

LAB:

Develop & Build

Using the threat model your team created for your scenario as part of the previous exercise, discuss in your breakout room how you intend to handle the prioritized threats during the develop/build phase of the SDLC.

Be as specific as possible:

- What manual processes will you use to validate security at this stage?
- What automated processes will you use to validate security at this stage?
- What suggestions would you offer for reporting and responding to findings?
- How would you reduce friction at this phase to ensure we're able to not only move safely but also quickly?
- What concerns you most at this stage?

Each team will present their thoughts & ideas to the larger group.

DevSecOps in Test

Shifting Security Left – Test

- Includes changes ready for move to QA for additional testing
- Dynamic Application Security Testing (DAST) tools provide more sophisticated vulnerability checks
- Dynamically exercise application’s runtime interfaces using injected data

Shifting Security Left – Test

- DAST tools can use complex combinations of invalid input via fuzzing (https://triotechsystems.com/fuzz-testing-in-dast-what-it-is-and-how-to-implement-it)
- Provides multiple layers of protection
- A tool like OWASP Zed Proxy or OWASP ZAP is an example (https://www.zaproxy.org)

Shifting Security Left – Test

- Two types – passive scan and active scan
- Passive scan less aggressive and minimizes use of fuzzing
- As a result, not as robust in its ability to find exposure but also takes less time to run

Shifting Security Left – Test

- Because of that, passive scan can be good fit for CI/CD pipelines, especially those that look to push updates at a greater frequency
- For the C (continuous) part, need flows to complete quickly

Shifting Security Left – Test

- Active scans are more aggressive in their attack approach
- Make extensive use of fuzzing to elevate level of attack sophistication
- Send multiple requests, continually altering request data to simulate attack payloads

Shifting Security Left – Test

- Can also make use of a technique called “spidering”
- Allows the scan to crawl a site or service to simulate sequenced or multi-level attacks (stateful attacks)
- Can be very effective in securing a site or service but also takes much longer to execute (due to added complexity and sophistication)

Shifting Security Left – Test

- Because of its more aggressive nature, should only be executed against owned sites and in a sandboxed environment (risky)
- Often active scans are scheduled to occur out-of-band on a regular basis (weekly, nightly, etc.)

Shifting Security Left – Test

- In addition to the automated scans, traditional functional test scripts and manual testing may be used
- Should include security-focused testing as well (or in separate PEN testing)
- Tests should be informed by prioritized set of vulnerabilities identified during threat modeling

Moment of Reflection

Which type of dynamic scan (passive or active) is more effective, and why?

Type your answer in chat

LAB:

Test

Using the threat model your team created for your scenario as part of the previous exercise, discuss in your breakout room how you intend to handle the prioritized threats during the test phase of the SDLC.

Be as specific as possible:

- What manual processes will you use to validate security at this stage?
- What automated processes will you use to validate security at this stage?
- What suggestions would you offer for reporting and responding to findings?
- How would you reduce friction at this phase to ensure we're able to not only move safely but also quickly?
- What concerns you most at this stage?

Each team will present their thoughts & ideas to the larger group.

DevSecOps in Release

Shifting Security Left – Release

- Software confirmed ready for release
- Often deployed to a pre-prod environment (like staging)
- Can include automated “smoke” testing

Shifting Security Left – Release

- This phase may also include security-specific scans as a “last mile” protection
- SAST, SCA, and DAST scans may be repeated in this environment as a double-check
- Port scans can also be used (depending on how closely this env matches prod) to help validate exposure at network communication layer

Shifting Security Left – Release

- Additionally, this stage can utilize application logging as a type of validation
- Provides a unique opportunity to confirm correct sanitization of log detail
- Also, can include verification of specific AuthN/AuthZ controls prior to a prod release

Moment of Reflection

What other types of security checks (besides rerunning SAST, SCA, and DAST scans) are effective as part of the Release phase?

Type your answer in chat

DevSecOps in Operate

Shifting Security Left – Operate

- In this stage, monitoring and alerting become paramount for capturing and notifying support of any security exposure “in the wild”
- System logs, telemetry, and event detail become useful
- Likely not everything will be found prior – ongoing vigilance is a must

Shifting Security Left – Operate

- When (not if) something happens, Root Cause Analysis (RCA) is important
- Helps with understanding the why
- Can also help with identifying short-term workarounds (to stop the bleeding) and long-term fixes (for enhancement prioritization)

Shifting Security Left – Operate

- Preceding detail can feed dashboards or reports to help stakeholders visualize the security posture of the system and org
- System logs can also be used to satisfy auditing requirements (depending on the industry)
- Key Performance Indicators (KPIs) can be used to measure

Shifting Security Left – Operate

Examples

- Number of security-related tickets
- Build or release delays caused by security alerts
- Mean time to compliance
- These can all help support the Continuous Improvement function in the area of security focus

Moment of Reflection

What is one Key Performance Indicator (KPI) or metric you would recommend using (or have used) to evaluate the security posture of a system running in production?

Type your answer in chat

LAB:

Release & Operate

Using the threat model your team created for your scenario as part of the previous exercise, discuss in your breakout room how you intend to handle the prioritized threats during the release & operate phases of the SDLC.

Be as specific as possible:

- What manual processes will you use to validate security at these stages?
- What automated processes will you use to validate security at these stages?
- What suggestions would you offer for reporting and responding to findings?
- How would you reduce friction at these phases to ensure we're able to not only move safely but also quickly?
- What concerns you most at these stages?

Each team will present their thoughts & ideas to the larger group.

Additional Considerations

Other Key Considerations - SAST

- Be aware that SAST tools have the propensity for false positives (depending on tooling employed)
- Tools may err on the side of caution with security scans
- Might require that an organization customize the tool to more accurately report on environment and application
- Otherwise, teams run the risk of wasting time on non-value add activities, impeding “frictionless security”

Other Key Considerations - Containers

- If application profile includes containers or container orchestration (k8s), additional consideration required relative to security of images
- Tools like Aqua (https://www.aquasec.com) and Prisma Cloud (https://www.prismacloud.io/prisma/cloud/cloud-workload-protection-platform) can help with scanning of container images

Other Key Considerations – Sensitive Config

- Sensitive configuration data and app secrets (e.g., connection strings, passwords, etc.) should never be hardcoded in source
- Run the risk of checking into and exposing via source control
- Tools like truffleHog (https://github.com/dxa4481/truffleHog) can help with repo scanning

Key Benefits of DevSecOps (In Summary)

Proactive Security Measures

By integrating security early in the development lifecycle, organizations can identify vulnerabilities sooner, significantly reducing the risk of breaches and enhancing overall application security.

Streamlined Compliance Processes

Continuous compliance through automated checks within CI/CD pipelines ensures that security standards are consistently met, minimizing the risk of regulatory penalties and enhancing operational integrity.

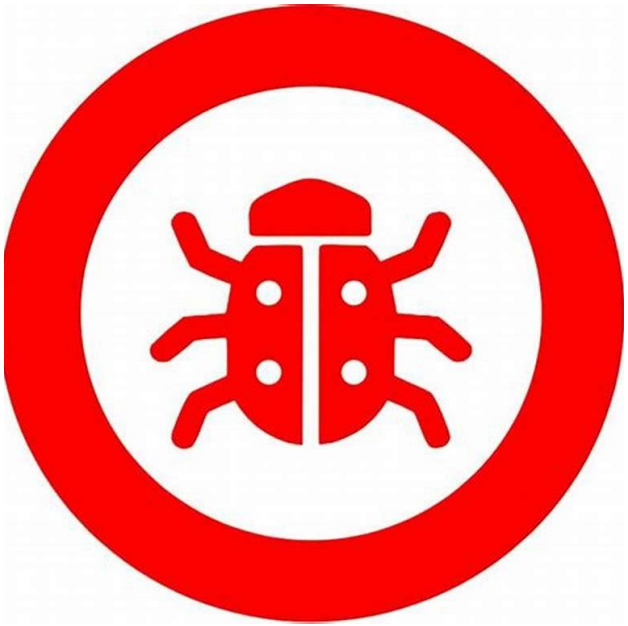
Fostering Team Collaboration

DevSecOps promotes a culture of collaboration among development, security, and operations teams, leading to improved communication, faster problem resolution, and a unified approach to security challenges.

OWASP & OWASP Top 10

OWASP & Its Importance

Introduction to OWASP



OWASP’s role in security:

- Provides essential resources & frameworks
- Empowers organizations to identify, manage, and mitigate security vulnerabilities
- Seeks to build a culture or security awareness and proactive risk management

The Role of OWASP in Security

Educational Resources and Training

OWASP provides comprehensive educational materials and training programs, equipping developers with the knowledge to address and mitigate security threats effectively

Advocacy for Security Standards

OWASP promotes essential security standards and practices, guiding organizations to integrate security into their software development processes from the outset

Open-Source Security Tools

OWASP develops and maintains open-source tools, such as ZAP, enabling organizations to identify vulnerabilities without incurring significant costs, thus enhancing overall cybersecurity

Overview of the OWASP Top 10

Critical Security Risks Identified, Tracked, and Documented

Informs Risk Management Strategies, Helps Prevent Financial Losses, and Protects Brand Reputation

LAB:

OWASP Top 10 - Intro

In your assigned breakout room, visit https://owasp.org/www-project-top-ten and complete a high-level review of the application security risks your team was assigned. Drill down into each and, as a team, document the following for “teach back” to the larger group:

- Brief description for the risk (2 – 3 bullets)
- Summary of how to prevent (2 – 3 bullets)
- Information on an example of an attack scenario

Nominate someone (different) from your breakout room to gather/document the results of the discussion. That person will present to the larger group when we come back together from our sub-teams.

OWASP Top 10 Vulnerabilities

Types & Examples

01

SQL Injection Risks

SQL Injection can lead to unauthorized data access, manipulation, and potential system compromise, emphasizing the need for robust input validation and sanitization.

02

Command Injection Exploits

Command Injection allows attackers to execute arbitrary commands on the server, highlighting vulnerabilities in input handling and the importance of strict validation measures.

03

XSS Attack Variants

Cross-Site Scripting (XSS) can steal sensitive user information through malicious scripts, necessitating effective input sanitization and user trust management in web applications.

SQL Injection

- Attacker injects SQL (Structured Query Language) queries into app flow (e.g., UI)
- In some cases, injected SQL used to retrieve additional sensitive detail
- In other cases, injected SQL used to alter or damage a company’s critical data

Types of SQL Injection Attack

- In-band
- Blind

In-band SQL Injection

- Uses existing channel of communication for an attack
- Error-based – attacker performs actions that cause errors in order to gather “intel” about database structure
- Union-based – attacker takes advantage of UNION SQL operator to fuse multiple SELECT statements into single response

Blind SQL Injection

- Attacker sends separate, independent data queries to a server
- Called blind because results of query are not sent back to attacker
- Instead, attacker observes results to infer vulnerabilities

Blind SQL Injection

- Relies on response and behavior patterns of server so slower to execute
- Boolean – attacker sends SQL query to database and determines if attack valid based on response received (true or false)
- Time-based – attacker sends SQL query to database and determines if attack valid based on amount of time taken to process

Defending Against SQL Injection

- Use well-defined contracts to explicitly map expected results from application
- Sanitize inputs and use parameterized queries in code
- Use a Web Application Firewall that includes protections at the application layer (including protection against SQL Injection)

Risks & Mitigations

Unauthorized Access Prevention

Implementing strong authentication measures reduces the risk of unauthorized access to sensitive user data and protects against identity theft.

Session Management Best Practices

Effective session management, including secure token handling and session expiration, is crucial to prevent session hijacking and unauthorized account access.

Regular Security Assessments

Conducting frequent security audits and vulnerability assessments helps identify weaknesses in authentication processes, enabling timely remediation and enhanced security posture.

Prevention Strategies

Data Classification Importance

- Proper classification of sensitive data
- Enables organizations to apply security measures appropriately tailored
- Helps prioritize protection efforts around the most critical data assets

Robust Incident Response

- Comprehensive incident response planning is key to minimizing damage from data exposure
- Promotes timely communication with relevant stakeholders
- Helps maintain compliance with regulatory requirements

Security Misconfiguration

Impact of Default Settings

- Default configurations often lack necessary security measures
- Vigilance and active management is required to ensure attack vectors are not exposed

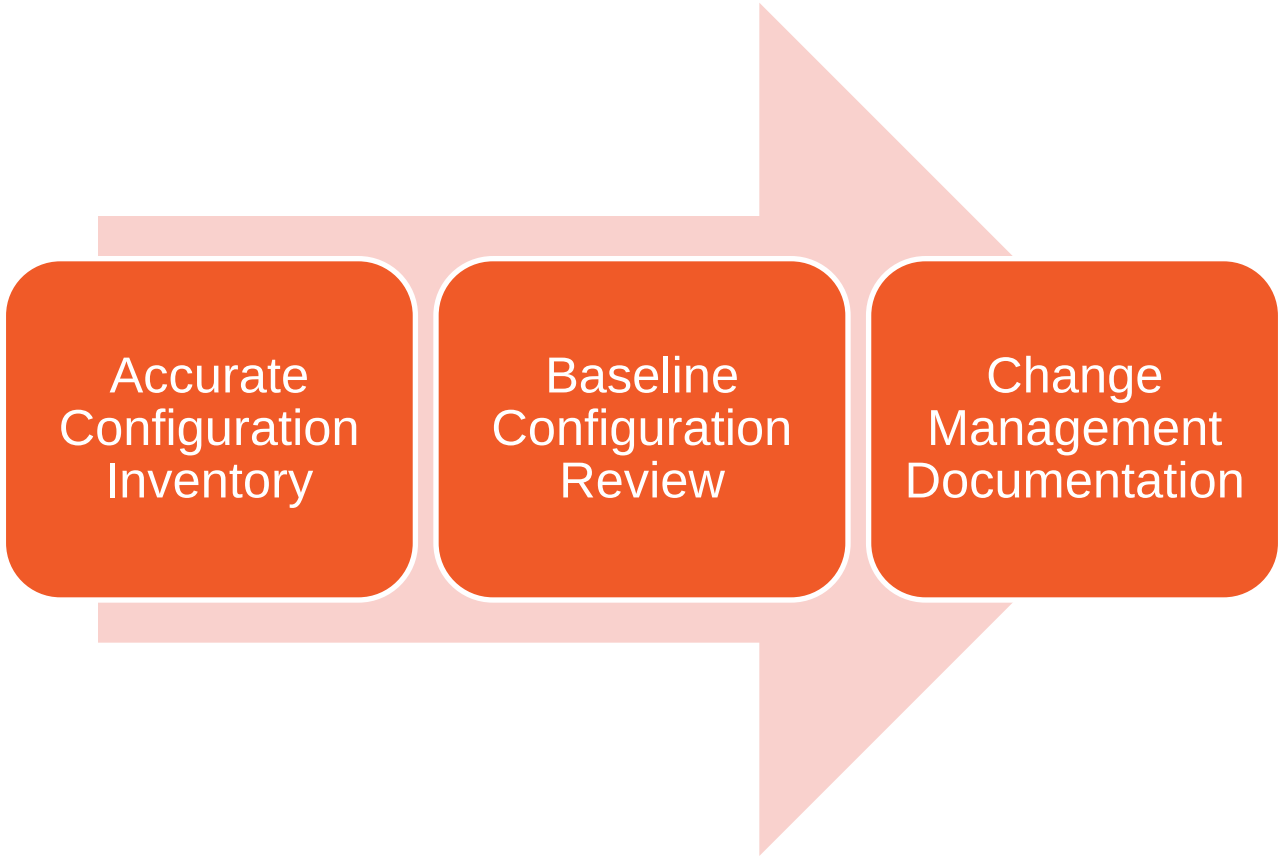
Importance of Regular Updates

- Failure to apply security patches to the infrastructure on which systems are running can leave systems exposed
- There should be a well-defined and regimented set of procedures in place for applying

Role of Access Controls

- Properly configured access controls are essential
- Misconfigured permissions can lead to data leaks and unauthorized actions
- Requires rigor in process around access management

Best Practices for Configuration Management



Types of XSS (Cross Site Scripting) Attacks

Reflected XSS Mechanism

Relies on immediate execution of scripts reflected from user input, often requiring user interaction through crafted links to trigger the attack

Stored XSS Characteristics

Involves persistent storage of malicious scripts, affecting multiple users who access the compromised content, leading to widespread exploitation

DOM-Based XSS Dynamics

Occurs entirely on the client side, manipulating the DOM without server interaction, highlighting the importance of secure client-side coding practices

Cross Site Scripting (XSS)

- Without inspection for malicious content, <script> can be returned (and executed) in user's browser
- Malicious content can include JavaScript, HTML, Flash, etc.
- Really, any code that browser can execute

Cross Site Scripting (XSS)

Common forms of attack:

- Stealing cookie or session information
- Redirects to web content controlled by an attacker
- Executing malicious operations on user’s machine
- Leveraging impersonation for elevated privilege

Cross Site Scripting (XSS)

Stored XSS attacks:

- Injected script permanently stored on target servers
- Could include storage in database, forum, comment field, etc.
- Malicious script returned to browser as part of retrieval from storage

Cross Site Scripting (XSS)

Reflected XSS attacks:

- Malicious script is indirectly transferred back to browser
- When user clicks on malicious link, code gets injected into vulnerable site
- Malicious code then reflected back to user under the cover of “valid” site interaction for immediate execution

Cross Site Scripting (XSS)

DOM-based XSS attacks:

- Takes advantage of sites that copy input to DOM without validation
- Similar to reflected in that victim is tricked into sending malicious code to vulnerable site
- However, input lands in DOM in the browser for execution instead of being reflected back

Impact of XSS on Users, Applications, and Enterprises

User Trust Erosion

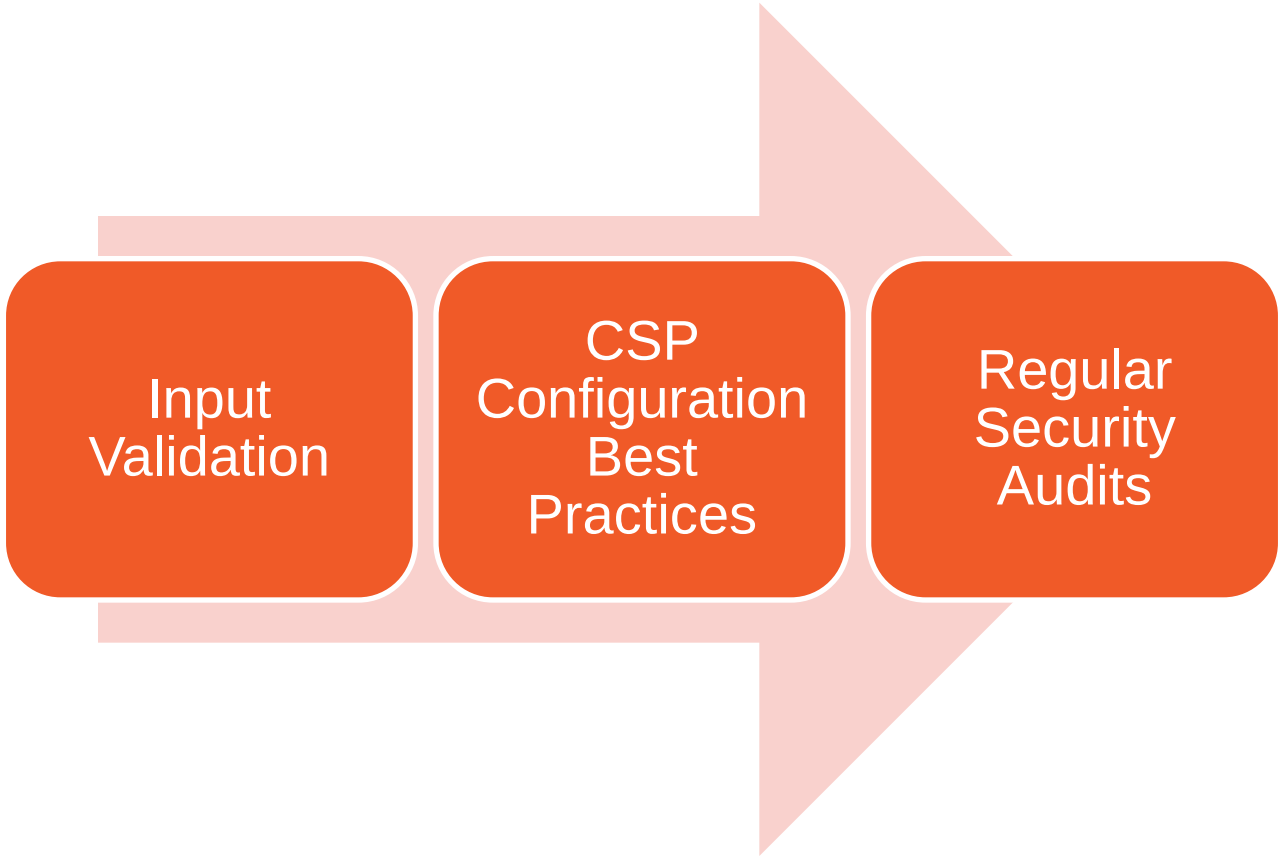
- Can severely undermine user trust in web applications
- Victims may feel vulnerable to ongoing threats
- Can lead to decreased user engagement and potential abandonment of services

Increased Security Costs

- Organizations may face escalating mitigation costs
- When not handled proactively, can lead to required investment in security tools, training, and incident response

https://brightsec.com/blog/xss-attack

Prevention Techniques for XSS



LAB:

OWASP Top 10

Follow along with the instructor to setup an account on your VM to access https://tryhackme.com/r/room/owasptop102021. This provides a series of interactive labs for exploring some of the common top 10 vulnerabilities and better understanding each.

Thank you!

If you have additional questions, please reach out to me at:
asanders@gamuttechnologysvcs.com

PLURALSIGHT