CPSC 4970 Applied Cyber Security

AUBURN UNIVERSITY

Module 1

AUBURN

Why is Cyber Security Important

2020 — 2021 — 2022 →

2020

solarwinds Sunburst

CVE-2021-35211

2021

Kaseya

CVE-2022-21660

LOG4J

CVE-2021-44228

2022

Vue.js

CVE-2021-30116

LOG4J

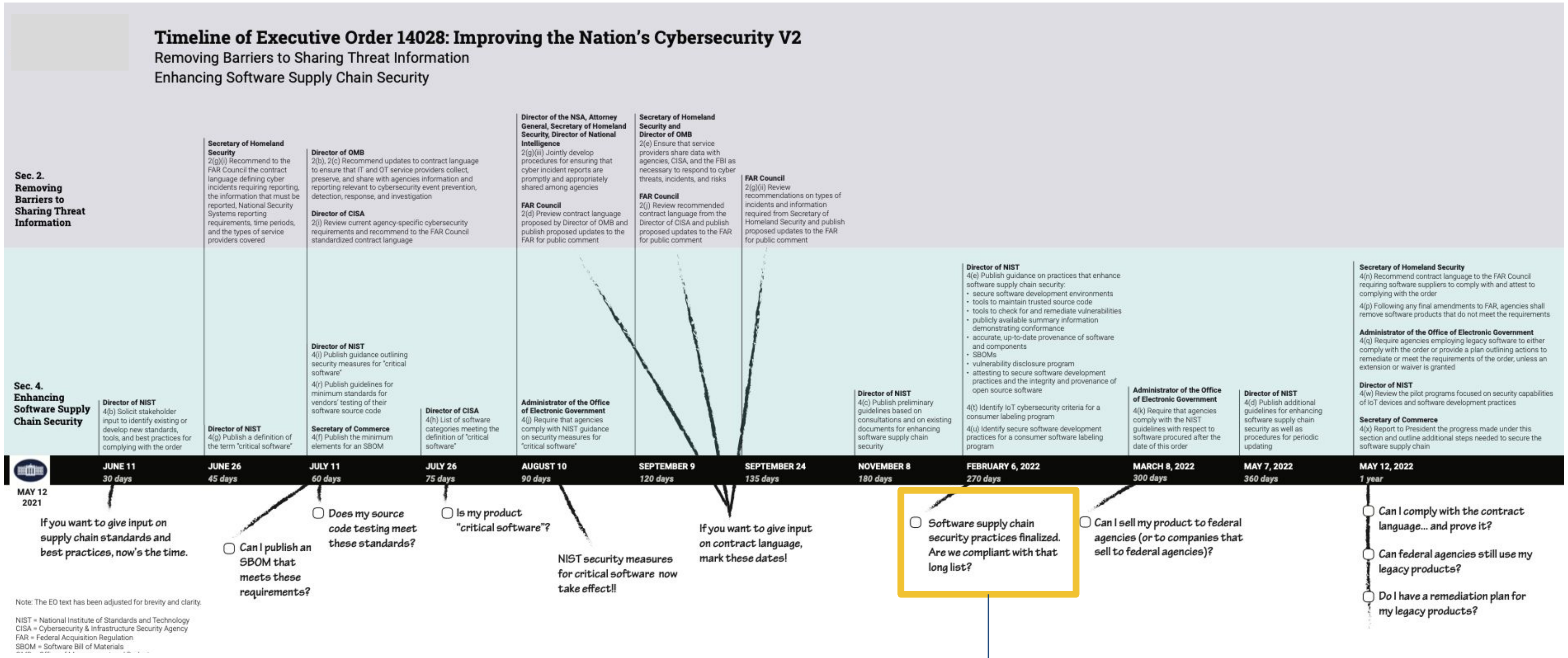
CVE-2021-45046

White House Drives Legislation

AUBURN

- Executive Order (EO) 14028 - May 12, 2021
  - “Improving the Nation’s Cybersecurity” requiring the government to only purchase software that is developed securely.
  - Sec. 4 - *“Enhancing Software Supply Chain Security”* - The development of commercial software often lacks transparency, sufficient focus on the ability of the software to resist attack, and adequate controls to prevent tampering by malicious actors. There is a pressing need to implement more rigorous and predictable mechanisms for ensuring that products function securely, and as intended.
- July 28, 2021 - National Security Memorandum on Improving Cybersecurity for Critical Infrastructure Control Systems
- Memorandum M-21-30 - Aug 10, 2021 - Protecting Critical Software Through Enhanced Security Measures
  - Software that controls access to data, cloud-based and hybrid software, software development tools, such as code repository systems, testing software, integration software, packaging software, and deployment software, software components in operational technology (OT).

White House Drives Legislation



Timeline of Executive Order 14028: Improving the Nation's Cybersecurity V2

Removing Barriers to Sharing Threat Information
Enhancing Software Supply Chain Security

Sec. 2. Removing Barriers to Sharing Threat Information

JUNE 26 *45 days*
Secretary of Homeland Security
2(g)(i) Recommend to the FAR Council the contract language defining cyber incidents requiring reporting, the information that must be reported, National Security Systems reporting requirements, time periods, and the types of service providers covered

JULY 11 *60 days*
Director of OMB
2(b), 2(c) Recommend updates to contract language to ensure that IT and OT service providers collect, preserve, and share with agencies information and reporting relevant to cybersecurity event prevention, detection, response, and investigation

Director of CISA
2(i) Review current agency-specific cybersecurity requirements and recommend to the FAR Council standardized contract language

AUGUST 10 *90 days*
Director of the NSA, Attorney General, Secretary of Homeland Security, Director of National Intelligence
2(g)(iii) Jointly develop procedures for ensuring that cyber incident reports are promptly and appropriately shared among agencies

FAR Council
2(d) Preview contract language proposed by Director of OMB and publish proposed updates to the FAR for public comment

SEPTEMBER 9 *120 days*
Secretary of Homeland Security and Director of OMB
2(e) Ensure that service providers share data with agencies, CISA, and the FBI as necessary to respond to cyber threats, incidents, and risks

FAR Council
2(j) Review recommended contract language from the Director of CISA and publish proposed updates to the FAR for public comment

SEPTEMBER 24 *135 days*
FAR Council
2(g)(ii) Review recommendations on types of incidents and information required from Secretary of Homeland Security and publish proposed updates to the FAR for public comment

Sec. 4. Enhancing Software Supply Chain Security

MAY 12 2021

JUNE 11 *30 days*
Director of NIST
4(b) Solicit stakeholder input to identify existing or develop new standards, tools, and best practices for complying with the order

JUNE 26 *45 days*
Director of NIST
4(g) Publish a definition of the term "critical software"

JULY 11 *60 days*
Director of NIST
4(i) Publish guidance outlining security measures for "critical software"
4(r) Publish guidelines for minimum standards for vendors' testing of their software source code

Secretary of Commerce
4(f) Publish the minimum elements for an SBOM

JULY 26 *75 days*
Director of CISA
4(h) List of software categories meeting the definition of "critical software"

AUGUST 10 *90 days*
Administrator of the Office of Electronic Government
4(j) Require that agencies comply with NIST guidance on security measures for "critical software"

NOVEMBER 8 *180 days*
Director of NIST
4(c) Publish preliminary guidelines based on consultations and on existing documents for enhancing software supply chain security

FEBRUARY 6, 2022 *270 days*
Director of NIST
4(e) Publish guidance on practices that enhance software supply chain security:
- secure software development environments
- tools to maintain trusted source code
- tools to check for and remediate vulnerabilities
- publicly available summary information demonstrating conformance
- accurate, up-to-date provenance of software and components
- SBOMs
- vulnerability disclosure program
- attesting to secure software development practices and the integrity and provenance of open source software

4(t) Identify IoT cybersecurity criteria for a consumer labeling program
4(u) Identify secure software development practices for a consumer software labeling program

MARCH 8, 2022 *300 days*
Administrator of the Office of Electronic Government
4(k) Require that agencies comply with the NIST guidelines with respect to software procured after the date of this order

MAY 7, 2022 *360 days*
Director of NIST
4(d) Publish additional guidelines for enhancing software supply chain security as well as procedures for periodic updating

MAY 12, 2022 *1 year*
Secretary of Homeland Security
4(n) Recommend contract language to the FAR Council requiring software suppliers to comply with and attest to complying with the order
4(p) Following any final amendments to FAR, agencies shall remove software products that do not meet the requirements

Administrator of the Office of Electronic Government
4(q) Require agencies employing legacy software to either comply with the order or provide a plan outlining actions to remediate or meet the requirements of the order, unless an extension or waiver is granted

Director of NIST
4(w) Review the pilot programs focused on security capabilities of IoT devices and software development practices

Secretary of Commerce
4(x) Report to President the progress made under this section and outline additional steps needed to secure the software supply chain

If you want to give input on supply chain standards and best practices, now's the time.

- ☐ *Can I publish an SBOM that meets these requirements?*
- ☐ *Does my source code testing meet these standards?*
- ☐ *Is my product "critical software"?*

NIST security measures for critical software now take effect!!

If you want to give input on contract language, mark these dates!

- ☐ *Software supply chain security practices finalized. Are we compliant with that long list?*
- ☐ *Can I sell my product to federal agencies (or to companies that sell to federal agencies)?*
- ☐ *Can I comply with the contract language... and prove it?*
- ☐ *Can federal agencies still use my legacy products?*
- ☐ *Do I have a remediation plan for my legacy products?*

Note: The EO text has been adjusted for brevity and clarity.

NIST = National Institute of Standards and Technology
CISA = Cybersecurity & Infrastructure Security Agency
FAR = Federal Acquisition Regulation
SBOM = Software Bill of Materials

Improve Software Supply Chain Security

Improve Software Supply Chain Security

AUBURN

- Establish baseline security standards for development of software sold to the government, including requiring developers to maintain greater visibility into their software and making security data publicly available.
- Establishes concurrent public-private process to develop new and innovative approaches to secure software development and uses the power of Federal procurement to incentivize the market
- Creates a pilot program to create “energy star” type of label so government and public at large can quickly determine whether software was developed securely.
- Focuses on the using the purchasing power of the Federal Government to drive the market to build security into all software from the ground up.

Securing Software Dev Environments

- Separate build environments with administrative controls
- Regular audits of access controls; implement advanced authentication mechanisms (multi factor).
- Employ data encryption
- Employing automated tools with access to trusted source code supply chains, thereby maintaining code integrity
- Automated tools to check for known and potential vulnerabilities to support quick action for remediation or risk mitigation.
- Provide proof of origin of software code or components and controls on internal and 3rd party software components, tools, and services present during development process.
- Perform audits on effectiveness of controls on a recurring basis.
- Coming: Software Bill of Materials - what does your software contain?

Cyber Security Industry Standards

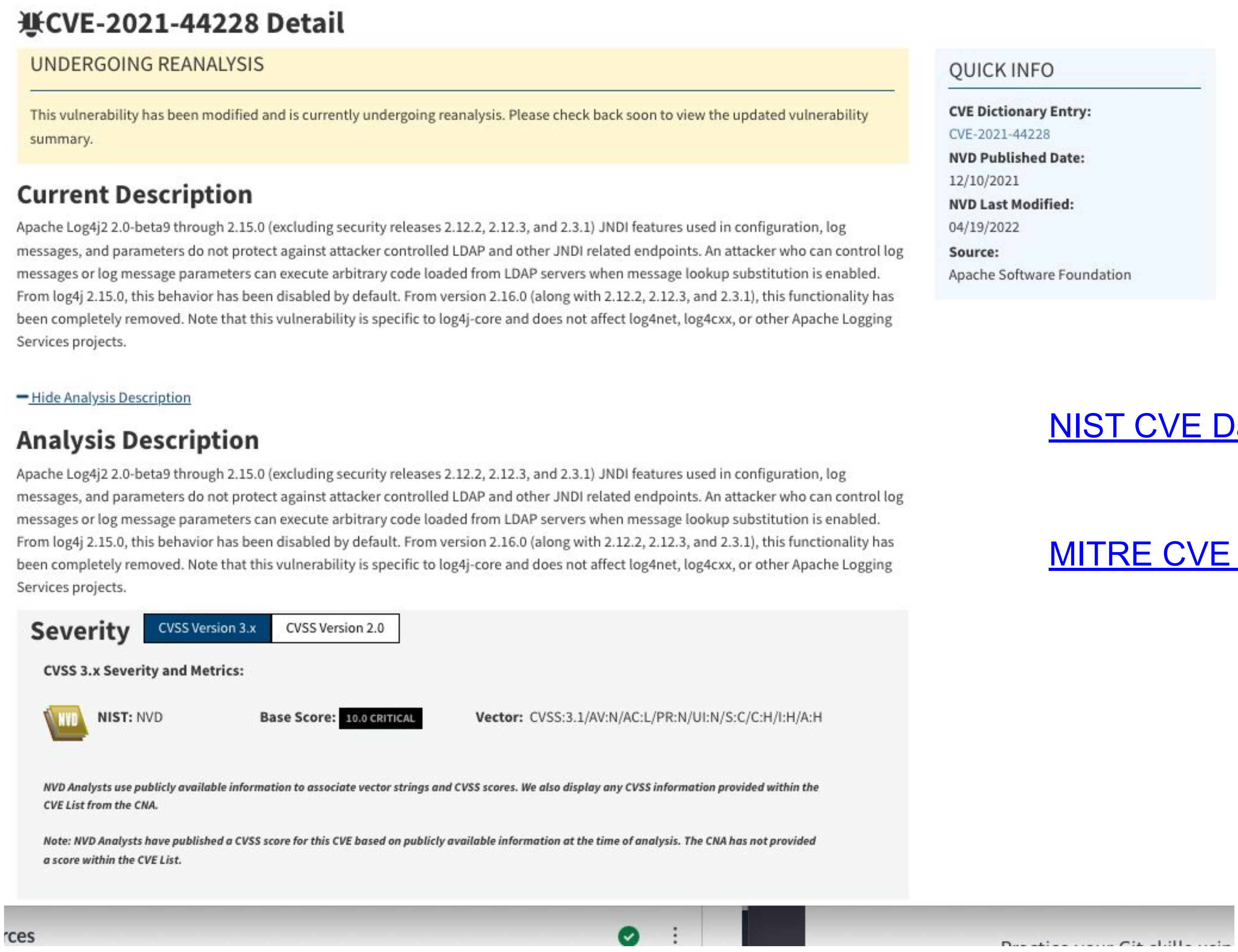
NIST & National Vulnerability Database (NVD)

- US Government standards based vulnerability management data
- Originally created in 1999 (called Internet - Categorization of Attacks Toolkit or ICAT
- Provides common language (taxonomy) for analyzing, scoring, and classifying vulnerabilities.
  - Common Weakness Enumeration (CWE)
    - List of software and hardware weakness types that serves as a baseline for weakness identification, mitigation, and prevention efforts.
  - Common Vulnerabilities and Exposures (CVE)
    - Known vulnerability database for specific code bases, such as software applications or open source libraries
  - Common Weakness Scoring System (CWSS)
    - Provides a mechanism for prioritizing software weaknesses in a consistent, flexible, open manner.

CVE Program

- Maintained by MITRE Corporation, sponsored by Department of Homeland Security (DHS), Cybersecurity and Infrastructure Security Agency (CISA) Division
- CVE IDs are primarily assigned by MITRE
  - Also by authorized CVE Numbering Authorities (CNAs) such as corporations or
- Maintains a centralized, searchable database of known vulnerabilities
  - All information contained in the project is publicly available to any interested party.
- Provides a common means of discussing and researching exploits.
- CVE IDs are used by vendors and cybersecurity personnel for research and the identification of new vulnerabilities.
- The program do not assist in mitigating or patching vulnerabilities on the CVE list
- Format for CVE IDs is: **CVE-[4 Digit Year]-[Sequential Identifier]**

CVE-2021-33228 Log4J JNDI Vulnerability



CVE-2021-44228 Detail

UNDERGOING REANALYSIS

This vulnerability has been modified and is currently undergoing reanalysis. Please check back soon to view the updated vulnerability summary.

Current Description

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

Hide Analysis Description

Analysis Description

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

Severity CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 10.0 CRITICAL **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

QUICK INFO

CVE Dictionary Entry:
CVE-2021-44228
NVD Published Date:
12/10/2021
NVD Last Modified:
04/19/2022
Source:
Apache Software Foundation

rces

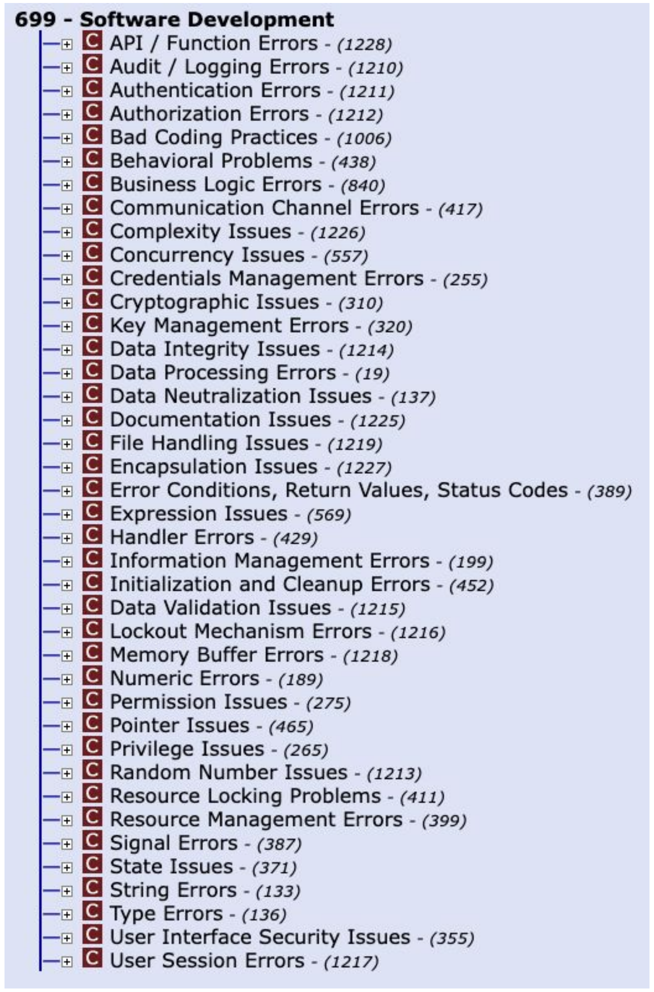
[NIST CVE Database Entry]

[MITRE CVE Database Entry]

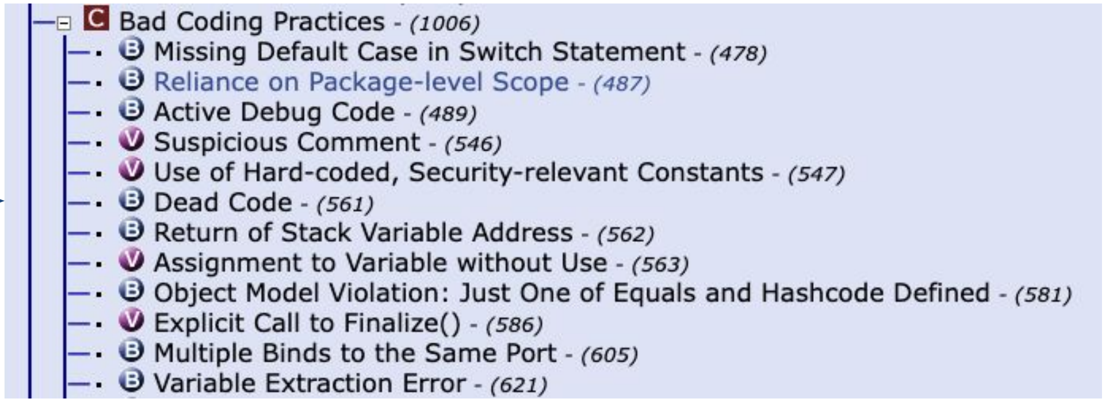
Common Weakness Enumeration (CWE)

- Common language for describing and communicating software and hardware weaknesses types
- Community driven to define concise and specific weakness types
- List is hierarchy in design for both software and hardware. Revised on an ongoing basis as threat landscape and software/hardware architectures evolve.

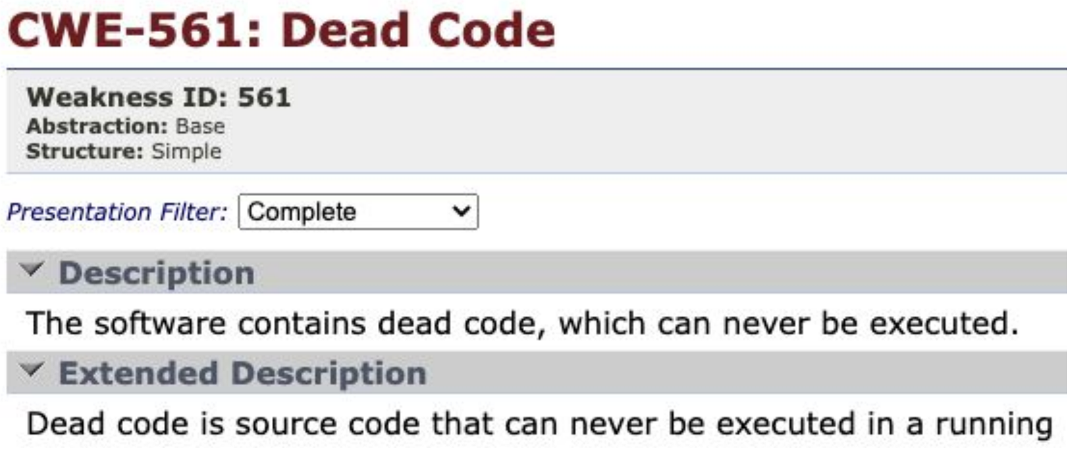
Common Weakness Enumeration (CWE)



699 - Software Development
- C API / Function Errors - (1228)
- C Audit / Logging Errors - (1210)
- C Authentication Errors - (1211)
- C Authorization Errors - (1212)
- C Bad Coding Practices - (1006)
- C Behavioral Problems - (438)
- C Business Logic Errors - (840)
- C Communication Channel Errors - (417)
- C Complexity Issues - (1226)
- C Concurrency Issues - (557)
- C Credentials Management Errors - (255)
- C Cryptographic Issues - (310)
- C Key Management Errors - (320)
- C Data Integrity Issues - (1214)
- C Data Processing Errors - (19)
- C Data Neutralization Issues - (137)
- C Documentation Issues - (1225)
- C File Handling Issues - (1219)
- C Encapsulation Issues - (1227)
- C Error Conditions, Return Values, Status Codes - (389)
- C Expression Issues - (569)
- C Handler Errors - (429)
- C Information Management Errors - (199)
- C Initialization and Cleanup Errors - (452)
- C Data Validation Issues - (1215)
- C Lockout Mechanism Errors - (1216)
- C Memory Buffer Errors - (1218)
- C Numeric Errors - (189)
- C Permission Issues - (275)
- C Pointer Issues - (465)
- C Privilege Issues - (265)
- C Random Number Issues - (1213)
- C Resource Locking Problems - (411)
- C Resource Management Errors - (399)
- C Signal Errors - (387)
- C State Issues - (371)
- C String Errors - (133)
- C Type Errors - (136)
- C User Interface Security Issues - (355)
- C User Session Errors - (1217)



- C Bad Coding Practices - (1006)
  - B Missing Default Case in Switch Statement - (478)
  - B Reliance on Package-level Scope - (487)
  - B Active Debug Code - (489)
  - V Suspicious Comment - (546)
  - V Use of Hard-coded, Security-relevant Constants - (547)
  - B Dead Code - (561)
  - B Return of Stack Variable Address - (562)
  - V Assignment to Variable without Use - (563)
  - B Object Model Violation: Just One of Equals and Hashcode Defined - (581)
  - V Explicit Call to Finalize() - (586)
  - B Multiple Binds to the Same Port - (605)
  - B Variable Extraction Error - (621)



CWE-561: Dead Code

Weakness ID: 561
Abstraction: Base
Structure: Simple

Presentation Filter: Complete

Description

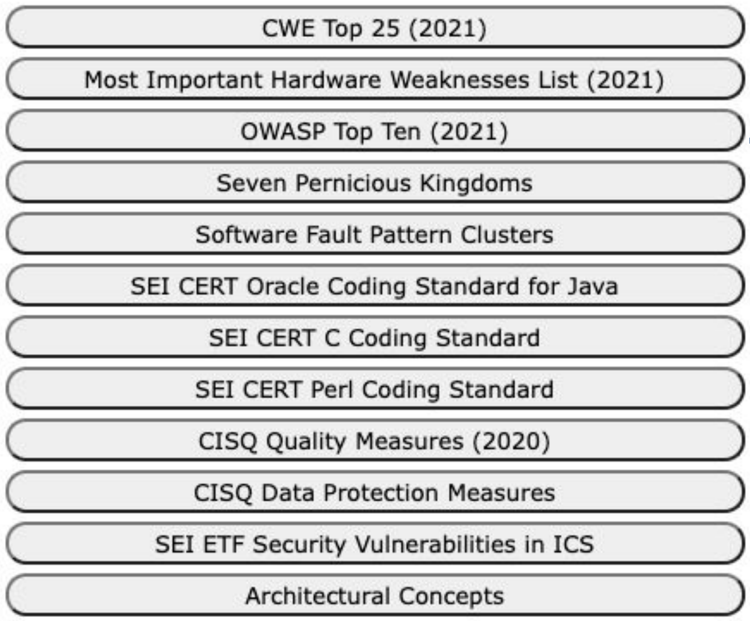
The software contains dead code, which can never be executed.

Extended Description

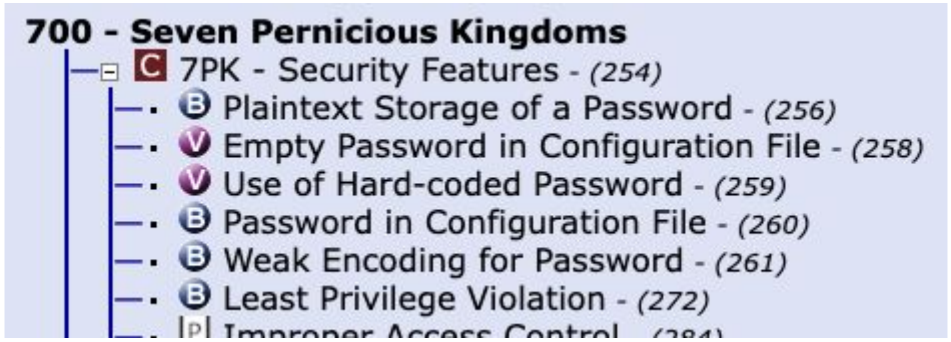
Dead code is source code that can never be executed in a running

Common Weakness Enumeration (CWE)

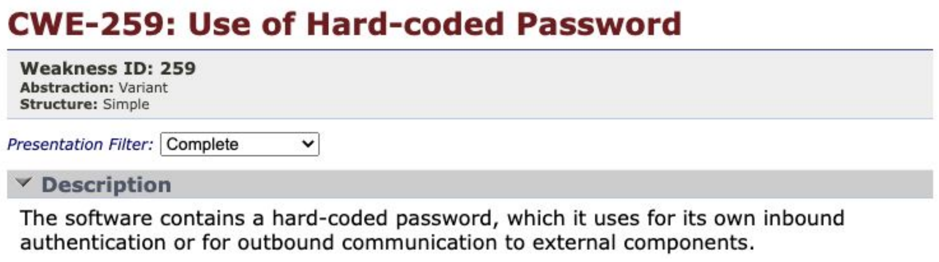
External Mappings & Lists



CWE Top 25 (2021)
Most Important Hardware Weaknesses List (2021)
OWASP Top Ten (2021)
Seven Pernicious Kingdoms
Software Fault Pattern Clusters
SEI CERT Oracle Coding Standard for Java
SEI CERT C Coding Standard
SEI CERT Perl Coding Standard
CISQ Quality Measures (2020)
CISQ Data Protection Measures
SEI ETF Security Vulnerabilities in ICS
Architectural Concepts



700 - Seven Pernicious Kingdoms
- 7PK - Security Features - *(254)*
  - Plaintext Storage of a Password - *(256)*
  - Empty Password in Configuration File - *(258)*
  - Use of Hard-coded Password - *(259)*
  - Password in Configuration File - *(260)*
  - Weak Encoding for Password - *(261)*
  - Least Privilege Violation - *(272)*
  - Improper Access Control - *(284)*



CWE-259: Use of Hard-coded Password

Weakness ID: 259
Abstraction: Variant
Structure: Simple

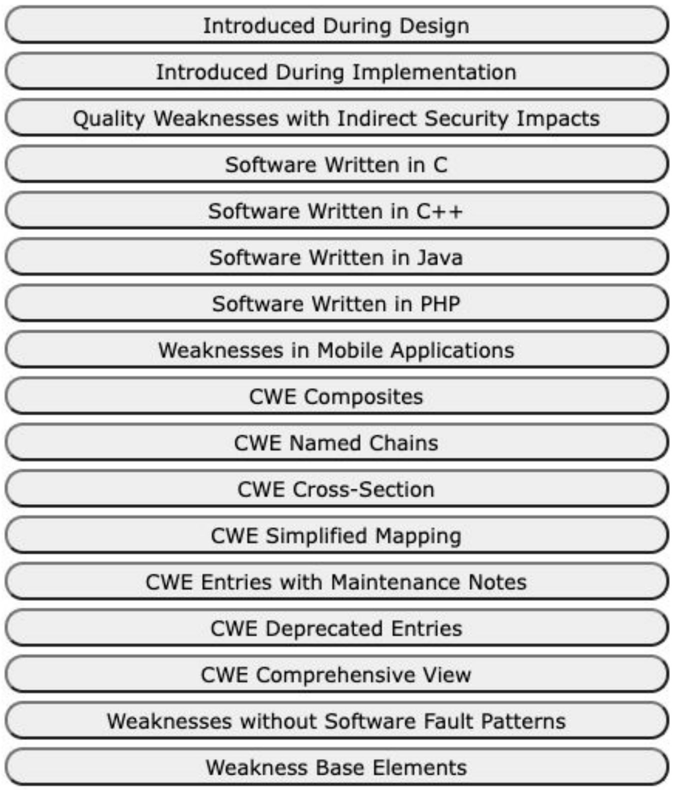
Presentation Filter: Complete

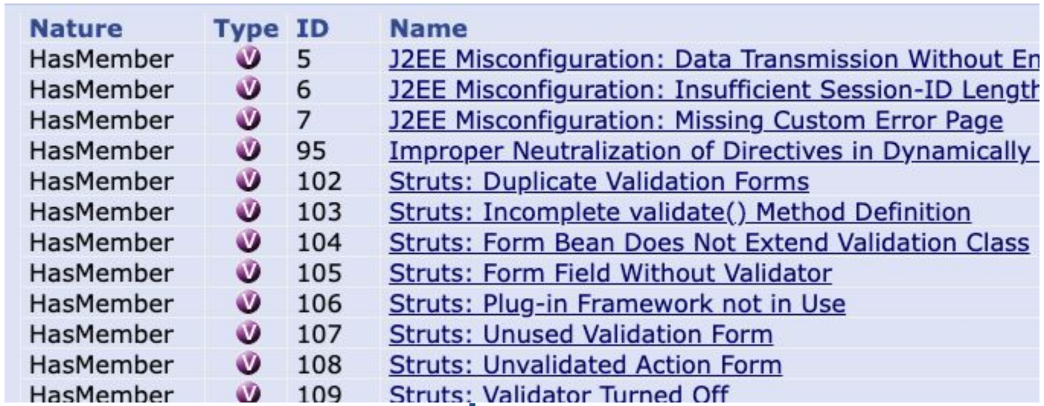
Description

The software contains a hard-coded password, which it uses for its own inbound authentication or for outbound communication to external components.

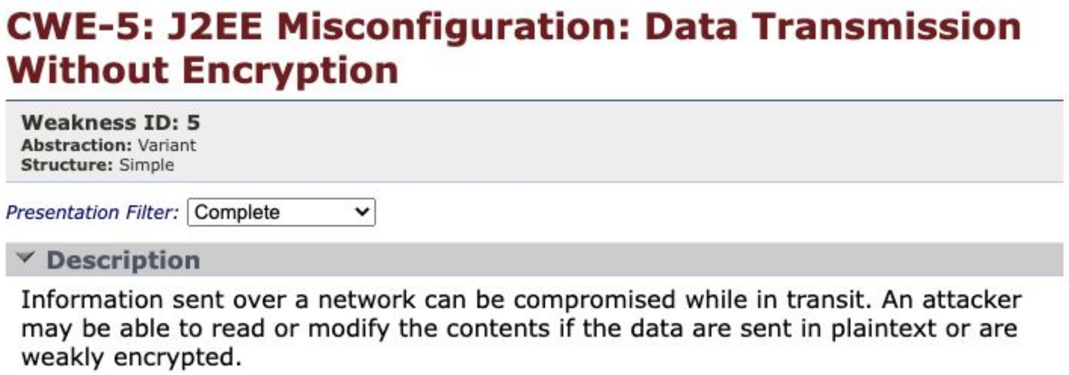
Common Weakness Enumeration (CWE)

Helpful References


- Introduced During Design
- Introduced During Implementation
- Quality Weaknesses with Indirect Security Impacts
- Software Written in C
- Software Written in C++
- Software Written in Java
- Software Written in PHP
- Weaknesses in Mobile Applications
- CWE Composites
- CWE Named Chains
- CWE Cross-Section
- CWE Simplified Mapping
- CWE Entries with Maintenance Notes
- CWE Deprecated Entries
- CWE Comprehensive View
- Weaknesses without Software Fault Patterns
- Weakness Base Elements



Nature	Type	ID	Name
HasMember	Ⓥ	5	J2EE Misconfiguration: Data Transmission Without En
HasMember	Ⓥ	6	J2EE Misconfiguration: Insufficient Session-ID Lengt
HasMember	Ⓥ	7	J2EE Misconfiguration: Missing Custom Error Page
HasMember	Ⓥ	95	Improper Neutralization of Directives in Dynamically
HasMember	Ⓥ	102	Struts: Duplicate Validation Forms
HasMember	Ⓥ	103	Struts: Incomplete validate() Method Definition
HasMember	Ⓥ	104	Struts: Form Bean Does Not Extend Validation Class
HasMember	Ⓥ	105	Struts: Form Field Without Validator
HasMember	Ⓥ	106	Struts: Plug-in Framework not in Use
HasMember	Ⓥ	107	Struts: Unused Validation Form
HasMember	Ⓥ	108	Struts: Unvalidated Action Form
HasMember	Ⓥ	109	Struts: Validator Turned Off



CWE-5: J2EE Misconfiguration: Data Transmission Without Encryption

Weakness ID: 5
Abstraction: Variant
Structure: Simple

Presentation Filter: Complete

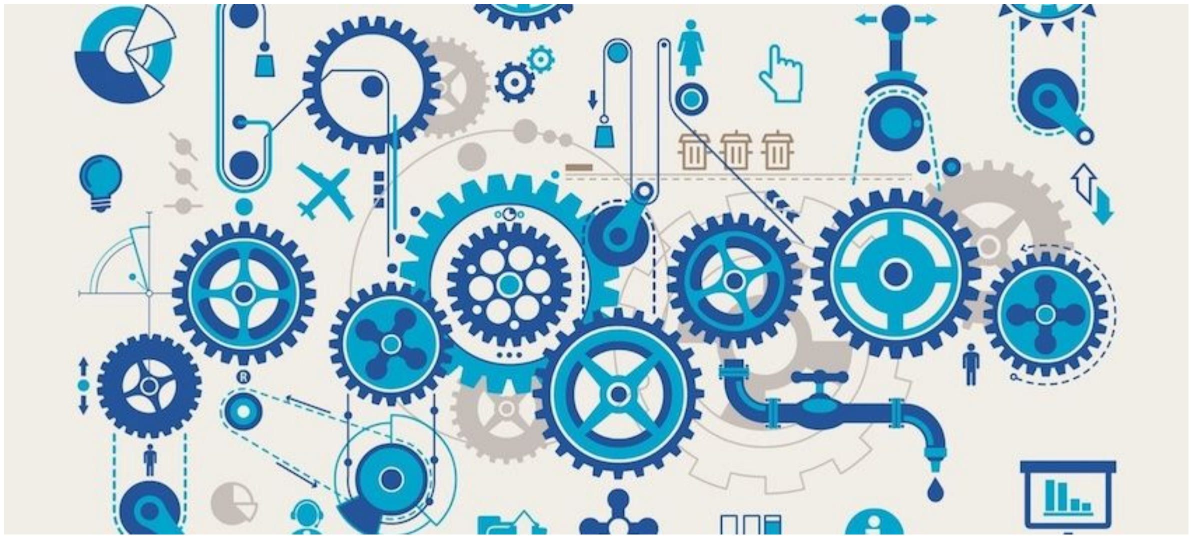
Description

Information sent over a network can be compromised while in transit. An attacker may be able to read or modify the contents if the data are sent in plaintext or are weakly encrypted.

Common Vulnerability Scoring System (CVSS)

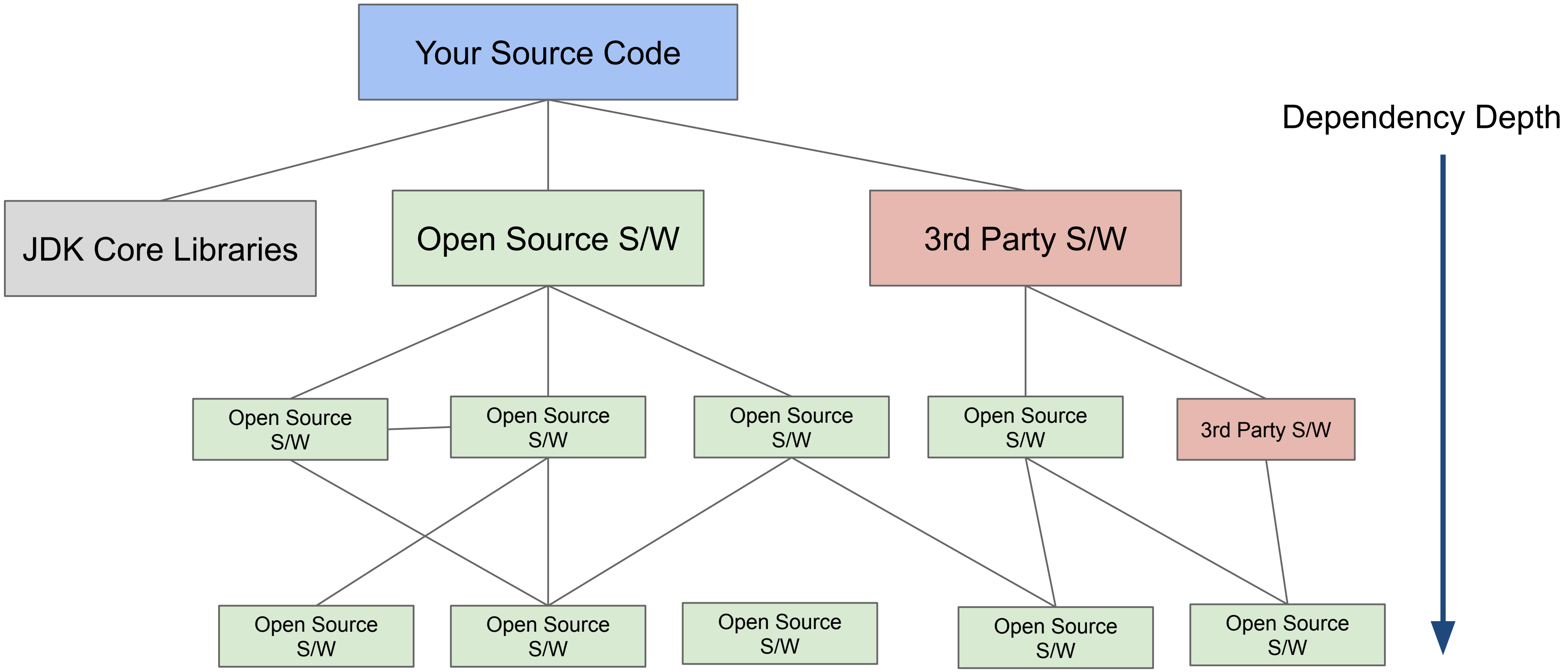
- Open framework for communicating the characteristics and severity of software vulnerabilities
  - Provides a numerical (0-10) representation of the severity of an information security vulnerability
  - Maintained by Forum of Incident Response and Security Teams (FIRST) comprised of 500+ member organizations.
- CVSS Measures Severity, not Risk
- A standardized scoring system provides the ability for software developers to prioritize issues so they can investigate and fix the highest risk items.

Software Composition



Software Composition

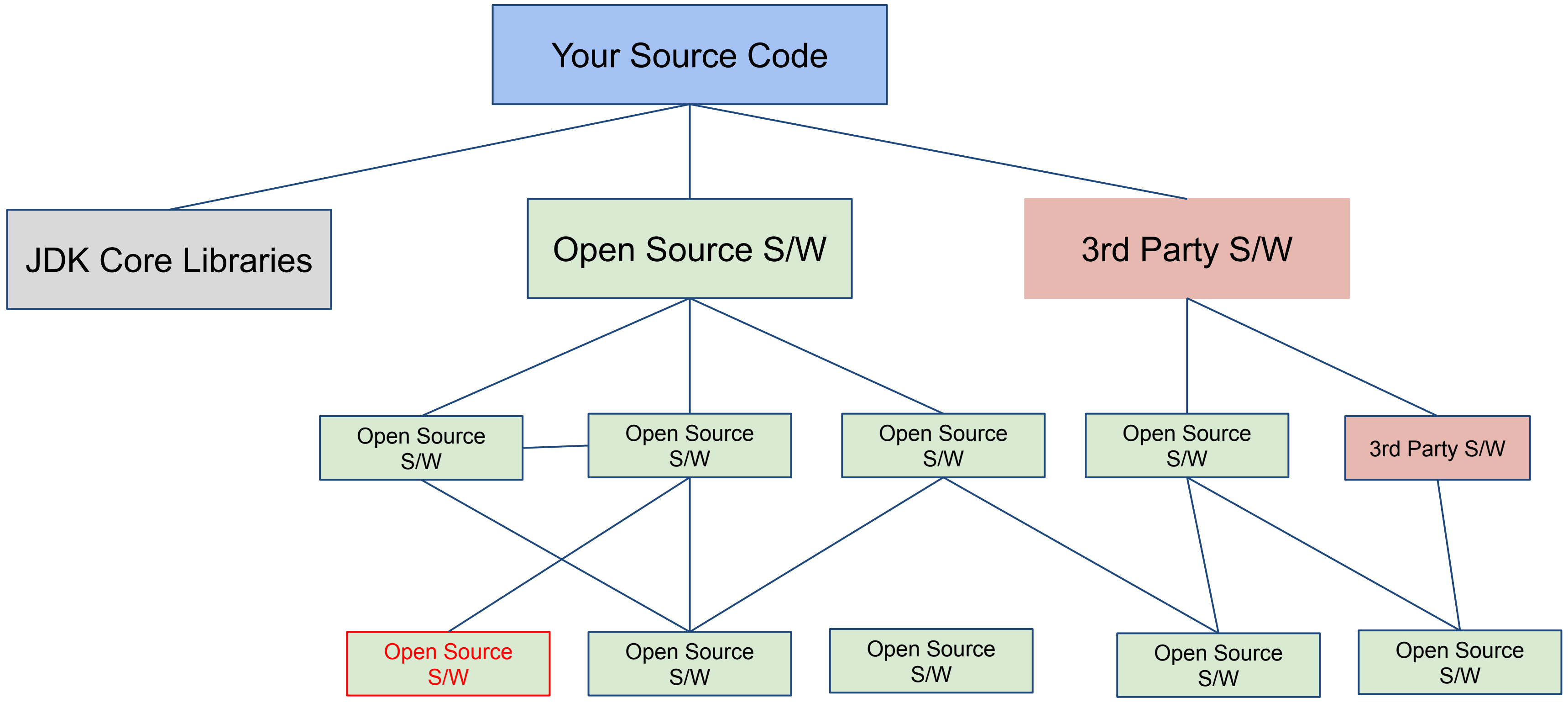
- Most application software utilizes larger amounts of open source and 3rd party software
- Not uncommon for application source code to be <25% of over all source code.



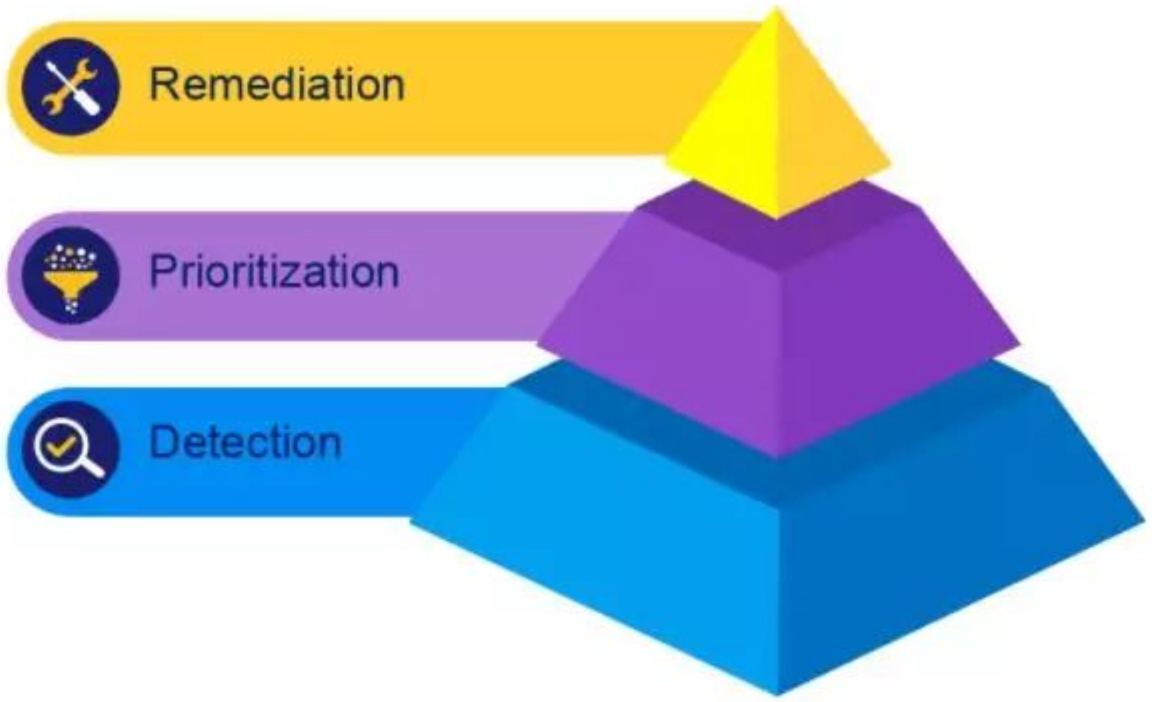
Software Composition Challenges

AUBURN

- How do you validate your developers are writing secure source code?
- How do you control the introduction of open source and 3rd party software?
- How are you notified when a vulnerability is uncovered in a software dependency?



Risk and Remediation



Confidentiality, Integrity, Availability (CIA)

3 goals of secure software development to insure information security:

Confidentiality - Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information

Integrity - Guarding against improper information modifications or destruction, and includes ensuring information non-repudiation and authenticity

Availability - Ensuring timely and reliable access to and use of information.

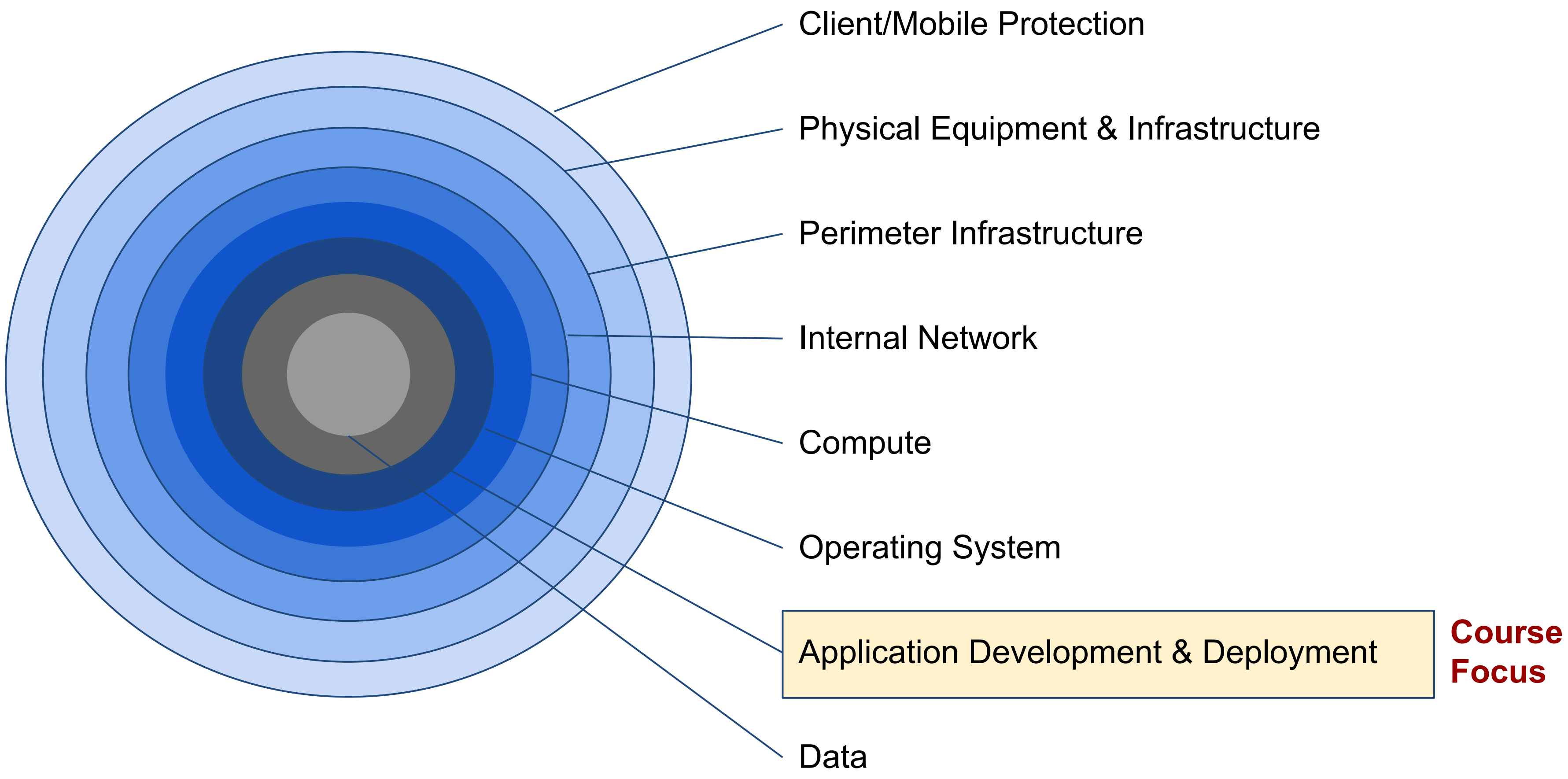
Cyber Security Challenges

- Software security does not equal software quality.
- No point solution will provide a single solution for software security.
- A holistic defense-in-depth approach is required
- A blend of people, process, and technology. The most important part being people.
- This course will apply techniques utilizing people, process and technology to build a secure software development process

Defense in Depth Approach

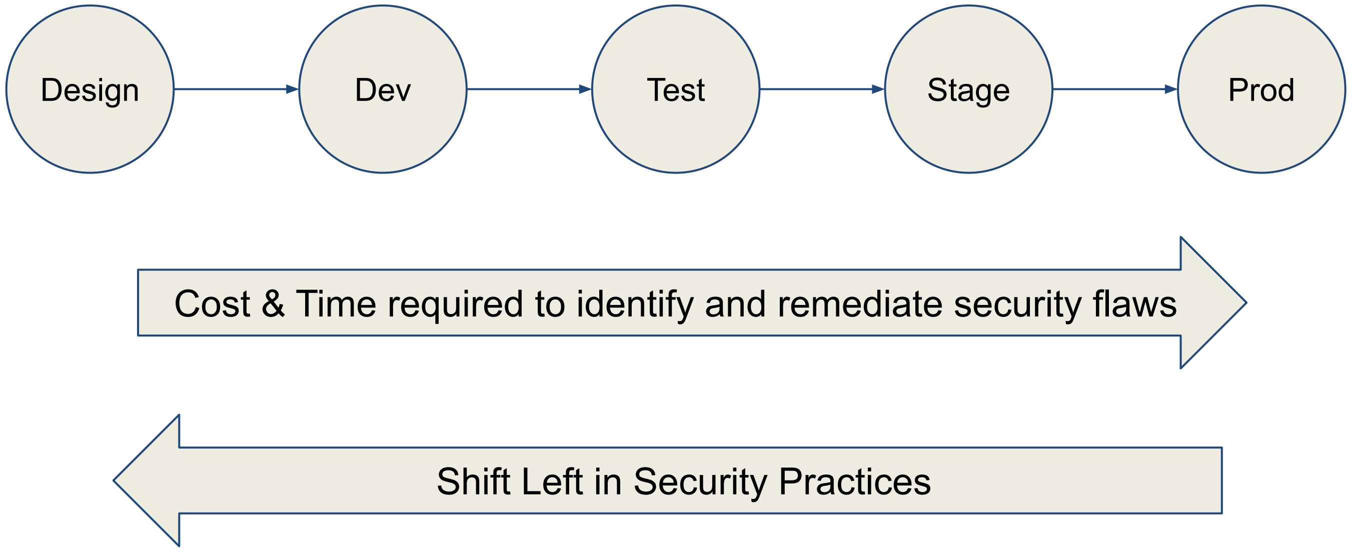
- Historically information security industry has focused on network and application security, assuming software is secure and their security controls were sufficient protection.
- Security practices during the software development cycle is now considered a first step in securing software systems and reducing risk.
- **Defense-in-Depth** security approaches see the software development process as a essential part of policies and standards of a comprehensive cyber security system.
  - Defense-in-depth is a strategy that provides multiple security controls in case other security control fails or a vulnerability is exploited.
  - Originated from a military strategy by the same name, which seeks to delay the advance of an attack by constructing multiple defensive lines.
  - Defense-in-depth cybersecurity use cases include end-user security, product development, network security, etc.

Zero Trust Defense-in-depth approach



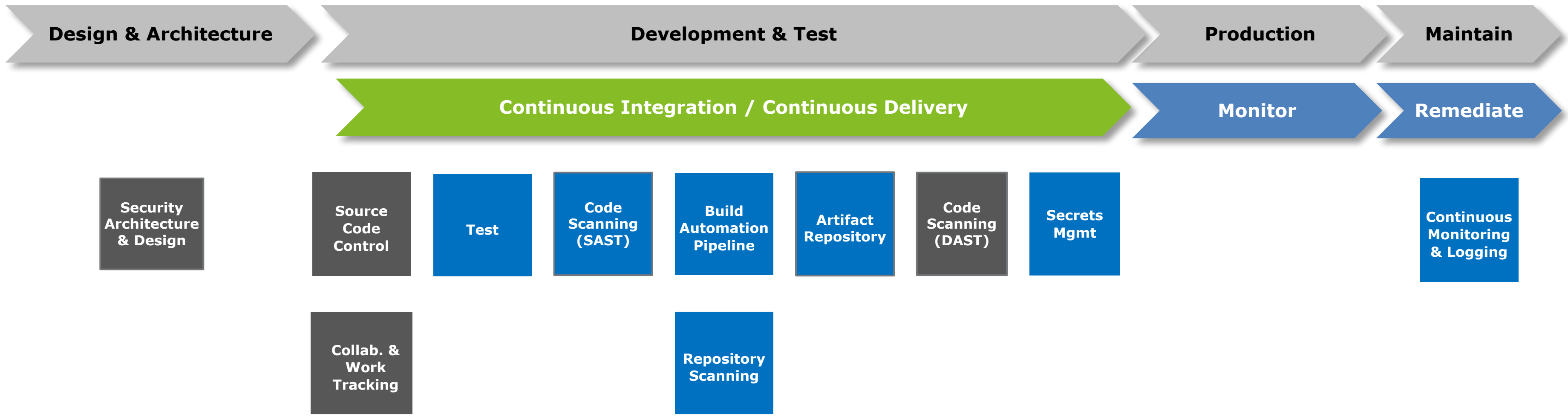
Shift Left on Software Security

- By Integrating information security practices into daily work, software teams can achieve higher levels of software delivery performance and build more secure systems.
- Security concerns, are addressed earlier in the software development lifecycle thus shifting left with regards to later stages of the development lifecycle.


Design
Dev
Test
Stage
Prod
Cost & Time required to identify and remediate security flaws
Shift Left in Security Practices

Course Project Overview

Course Objectives & Project



Security Automation

The following are commonly identified security gaps in a DevOps Pipeline:

- Static code scanning to support code quality and secure coding standards
- Dynamic application security testing for applications is performed
- Continuous logging and monitoring of production environment

Manual Security Methods

Checkpoints and gates must be implemented so security controls are in place:

- Security and architecture based on attack surfaces and data sensitivity.
- Ongoing source code change management controls based on code inspections and reviews.
- Periodic penetration tests are performed (e.g., OWASP Zap)