Cyber Security Industry Standards

NIST & National Vulnerability Database (NVD)

AUBURN

- 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

AUBURN

- 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

AUBURN

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.

QUICK INFO

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

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.

[NIST CVE Database Entry]

[MITRE CVE Database Entry]

Common Weakness Enumeration (CWE)

AUBURN

- 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)

AUBURN

699 - Software Development

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

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

AUBURN

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

- C 7PK - Security Features - *(254)*
  - B Plaintext Storage of a Password - *(256)*
  - V Empty Password in Configuration File - *(258)*
  - V Use of Hard-coded Password - *(259)*
  - B Password in Configuration File - *(260)*
  - B Weak Encoding for Password - *(261)*
  - B Least Privilege Violation - *(272)*
  - P 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)

AUBURN

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

AUBURN

- 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.