Digital Forensics

1)How did the attacker get in?

3	0.000199	172.28.128.4	172.28.128.5	TCP	74 47964 → 80 [SYN] Seq=0 Win=29
4	0.000217	172.28.128.5	172.28.128.4	TCP	74 80 → 47964 [SYN, ACK] Seq=0 A
5	0.000391	172.28.128.4	172.28.128.5	TCP	66 47964 → 80 [ACK] Seq=1 Ack=1
6	0.001023	172.28.128.4	172.28.128.5	HTTP	378 GET / HTTP/1.1
7	0.001050	172.28.128.5	172.28.128.4	TCP	66 80 → 47964 [ACK] Seq=1 Ack=31
8	0.020296	172.28.128.5	172.28.128.4	HTTP	704 HTTP/1.1 200 OK (text/html)

Figure 1, TCP Handshake

Analysing the PCAP file in Wireshark, the first thing which can be noticed is that the Source IP which could potentially be the attacker (IP 172.28.128.4) accessed the Waffle Co website. The TCP handshake has been performed which can be seen in the above figure. The server responded to the client's (Suspected Attacker) http request with a 200 OK.

28	7.452383	172.28.128.4	172.28.128.5	HTTP	418 GET /about.php HTTP/1.1
29	7.452397	172.28.128.5	172.28.128.4	TCP	66 80 → 47968 [ACK] Seq=1 Ack=353 Win=30
30	7.459942	172.28.128.5	172.28.128.4	HTTP	987 HTTP/1.1 200 OK (text/html)
31	7.460251	172.28.128.4	172.28.128.5	TCP	66 47968 → 80 [ACK] Seq=353 Ack=922 Win=
32	11.396473	172.28.128.4	172.28.128.5	HTTP	420 GET /waffles.php HTTP/1.1
33	11.419110	172.28.128.5	172.28.128.4	HTTP	774 HTTP/1.1 200 OK (text/html)
34	11.419380	172.28.128.4	172.28.128.5	TCP	66 47968 → 80 [ACK] Seq=707 Ack=1630 Win
35	14.422221	172.28.128.4	172.28.128.5	HTTP	419 GET /upload.php HTTP/1.1
36	14.428789	172.28.128.5	172.28.128.4	HTTP	660 HTTP/1.1 200 OK (text/html)

Figure 2, Accessing webpages

Further, the suspect visited different pages of the website such as about.php, waffles.php and upload.php which have been captured by tcpdump shown in the figure 2.

In order to check the copy of the web directory, I have hosted it on AWS Ubuntu instance and accessed the webpage from my local. All the pages the suspected attacker had visited are present on the Waffle Co's website as shown in the following screenshots.

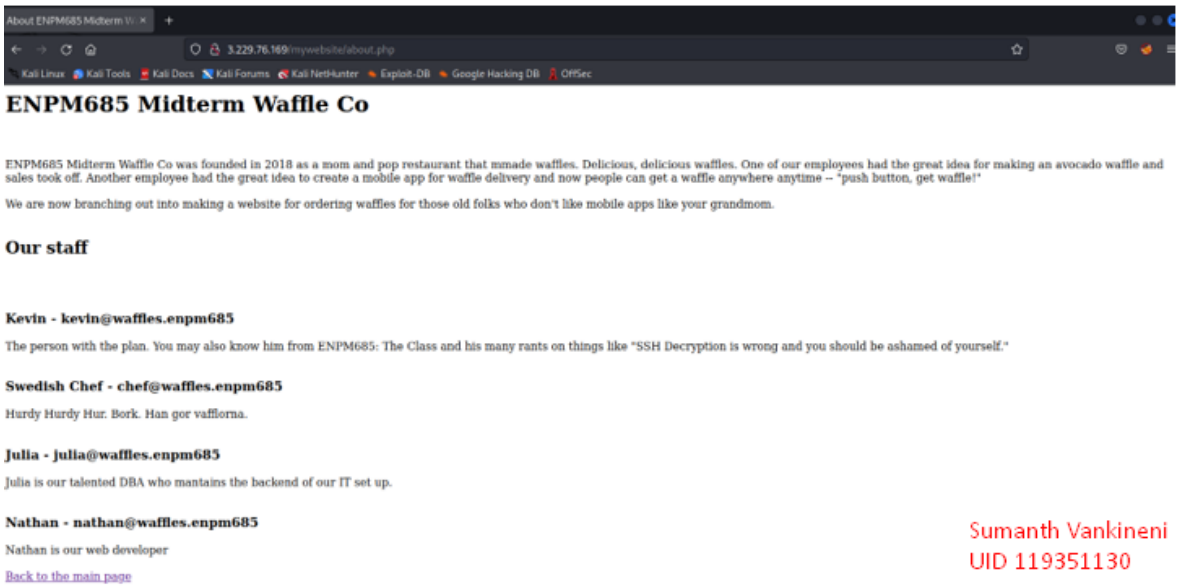

Figure 3,About.php

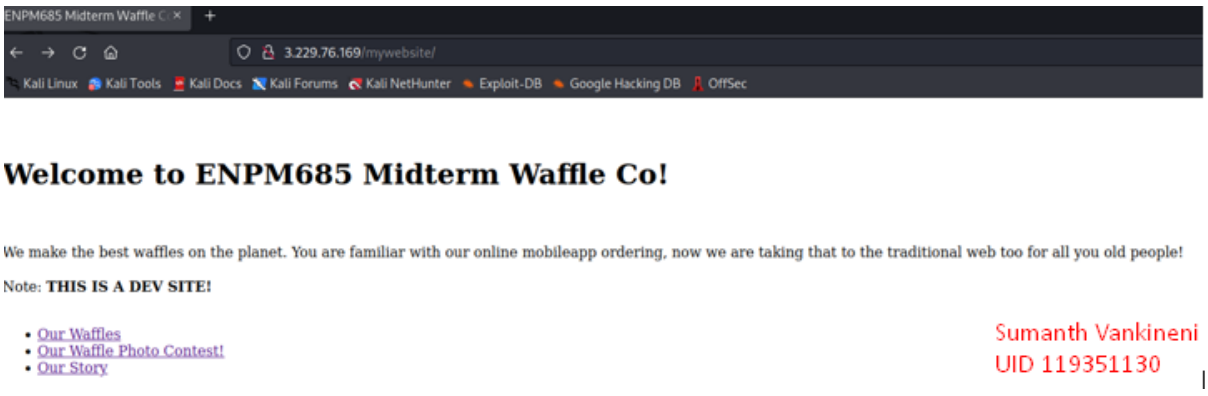

Figure 4,Homepage

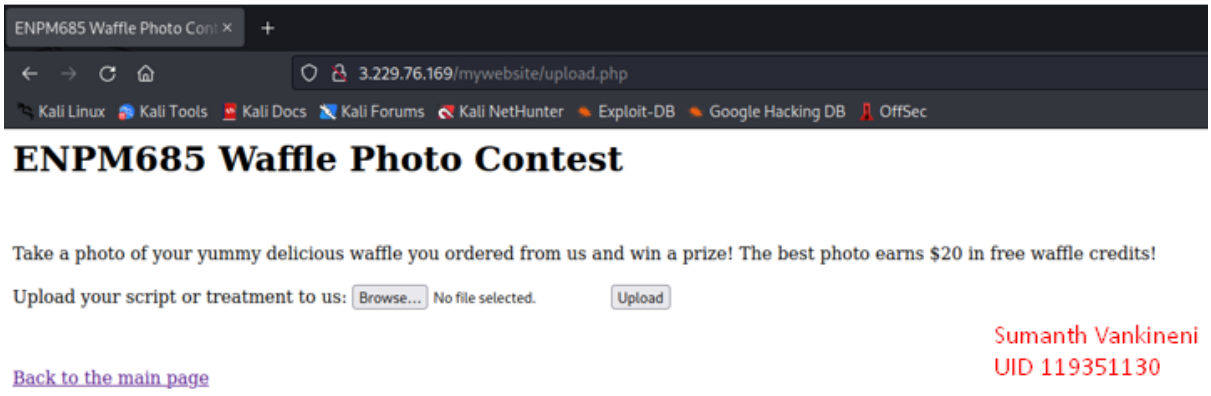

Figure 5,Upload.php

The Upload.php webpage has file upload section, there is a possibility that the attacker could have exploited this.

81	27.128335	172.28.128.4	172.28.128.5	HTTP	518 POST /upload2.php HTTP/1.1 (JPEG JFIF image)
82	27.128337	172.28.128.5	172.28.128.4	TCP	66 80 → 47970 [ACK] Seq=1 Ack=358109 Win=213056 Len=0 T
83	27.137072	172.28.128.5	172.28.128.4	HTTP	465 HTTP/1.1 200 OK (text/html)
84	27.137380	172.28.128.4	172.28.128.5	TCP	66 47970 → 80 [ACK] Seq=358109 Ack=400 Win=30336 Len=0
85	28.816728	172.28.128.4	172.28.128.5	HTTP	441 GET /uploads/TrollFace.jpg HTTP/1.1

Sumanth Vankineni
UID 119351130

Figure 6, Initial File Upload

On further analysing the PCAP file, the suspected attacker had tried uploading an image into the uploads section and then has tried to access it as shown in the above figure 6. So, looks like the uploads folder where all the uploaded content is stored can be accessed by the public which is a big vulnerability.

The following screenshots show the webpage which display the contents of the uploads directory and the image which has been uploaded by suspected attacker.

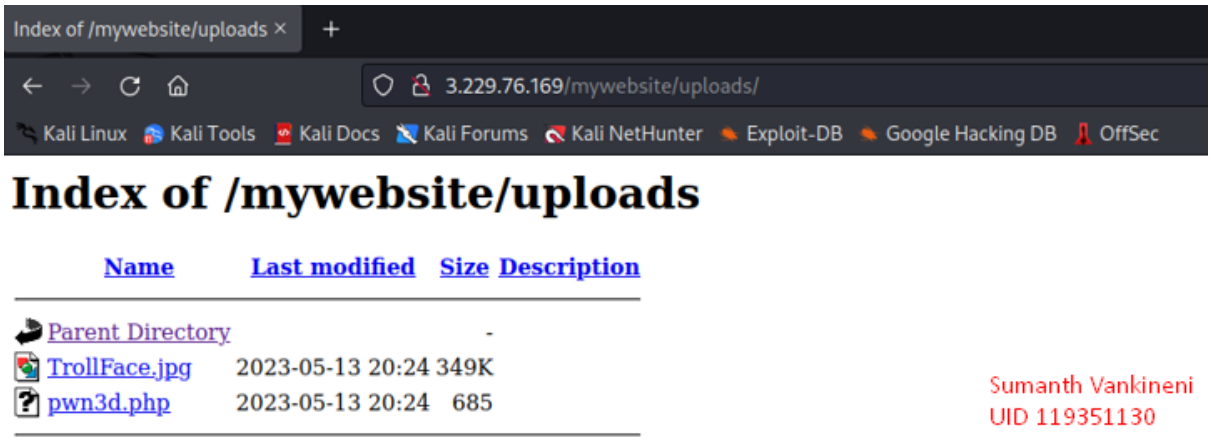

Figure 7, Uploads directory

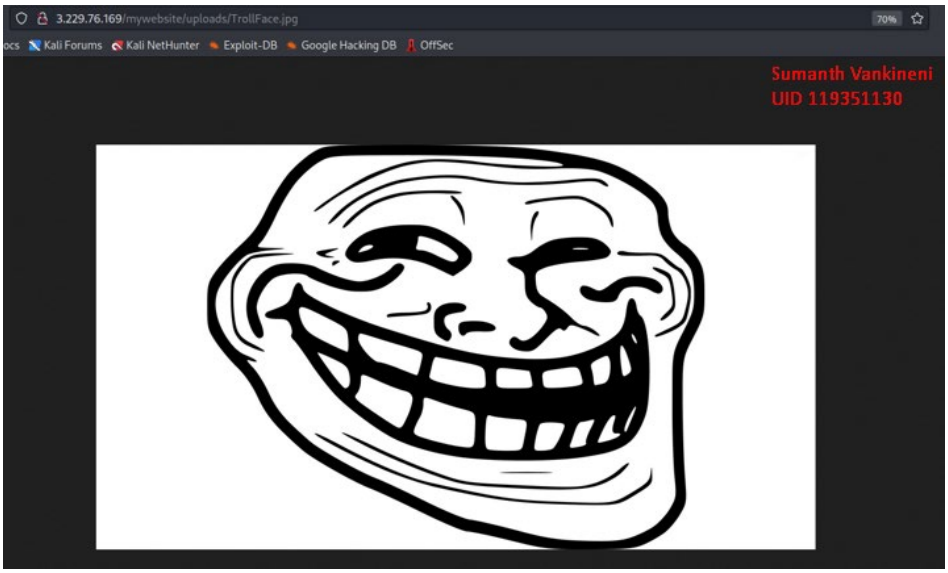

Figure 8, Troll face image uploaded by attacker

I thoroughly examined code of the upload.php webpage to see what conditions are used and what the security misconfigurations could be.

```
┌──(kali㉿kali)-[/var/www/html/mywebsite]
└─$ cat upload.php
<title>ENPM685 Waffle Photo Contest</title>
<h1>ENPM685 Waffle Photo Contest</h1>
<br><br>
Take a photo of your yummy delicious waffle you ordered from us and win a prize!  The best photo earns $
20 in free waffle credits!

<br><br>

<form action="upload2.php" method="post" enctype="multipart/form-data">
Upload your script or treatment to us:
<input type="file" name="fileToUpload" id="fileToUpload">
<input type="submit" value="Upload" name="submit">
</form>
<br><br>
<a href="/index.php">Back to the main page</a>

┌──(kali㉿kali)-[/var/www/html/mywebsite]
└─$ cat upload2.php
<?php

$target_dir = "/var/www/html/uploads/";
$target_file = $target_dir.basename($_FILES["fileToUpload"]["name"]);


if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file))
{
        echo "The file <a href=\"./uploads/".basename($_FILES["fileToUpload"]["name"])."\">".basename($_
FILES["fileToUpload"]["name"])."</a> has been uploaded.";
}
else
{
        echo "Error uploading file.";
}
?>
<br><br>
<a href="/index.php">Back to the main page</a>

┌──(kali㉿kali)-[/var/www/html/mywebsite]
└─$ echo Sumanth Vankineni UID 119351130
```

Figure 9, Upload page source codes.

The source code for the upload page has many security vulnerabilities as listed below:

1) There is no condition used in the source code to limit the input file type to an image with the formats as .jpeg or .png etc. This can allow the attackers to upload a malicious php code.
2) The files uploaded by the customers are stored in directory which is accessible by the public.
3) There is no limit set for the size of the file uploaded. Attackers can upload their malicious content of any length and utilize the servers disk space as their will.

So here the attacker could have misused this upload section to upload a malicious shellcode to trigger remote code execution or any other shellcode and tamper with the Waffle Co's system hosting the web server.

The following frame in the Wireshark shows that a file was uploaded to the server. By Further expanding the POST request frame it can be seen that a php file was uploaded named pwn3ed.php as show in the figure 10 and figure 11. Pawned filename indicates or proves that this was indeed the attacker, and the initial assumption was right.

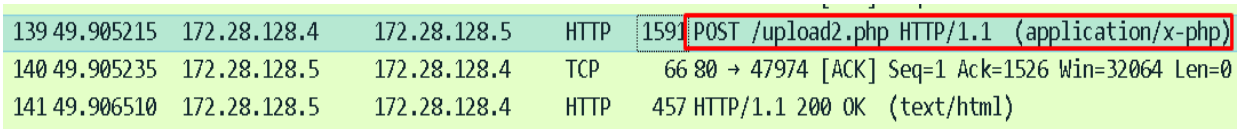

Figure 10, Wireshark Upload pwn3ed

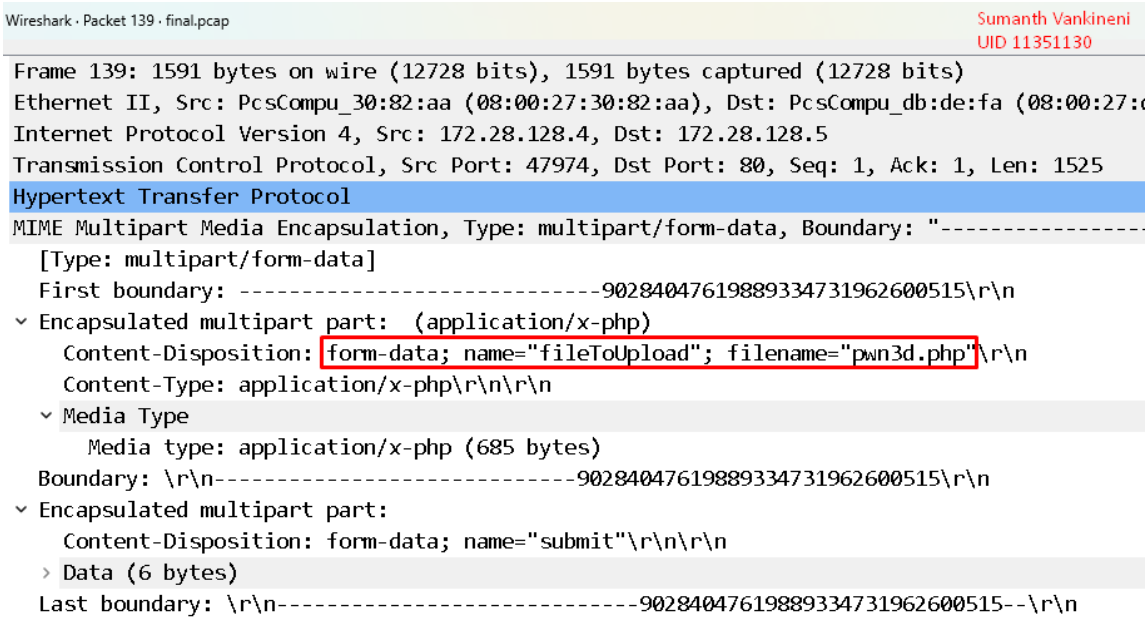

Figure 11, Wireshark Upload pwn3ed

The following consecutive POST requests could indicate the reverse shell or remote code execution's connected communication from the attacker to the Waffles Co's web servers hosted system.

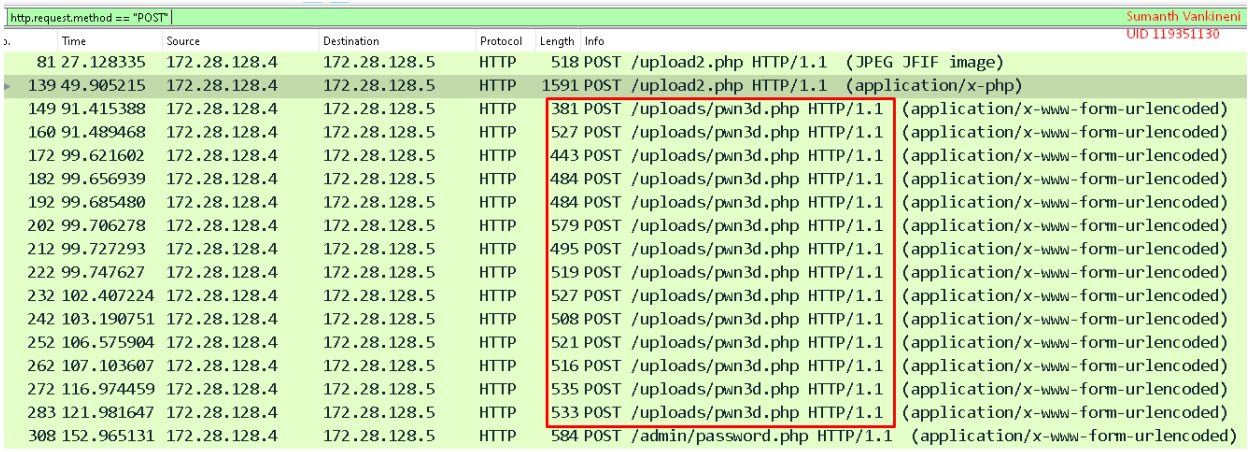

Figure 12, Remote execution connection

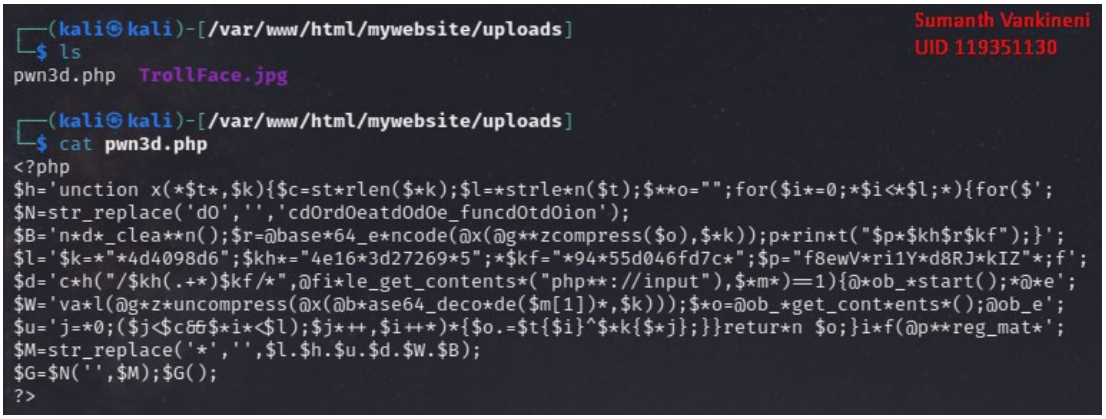

Figure 13, Obfuscated php code

The above figure shows the content of the pwn3ed.php file uploaded by the attacker. I've used the unphp.net to decode the obfuscated code shown above. The eval, gzuncompress and the base64_decode functions are very commonly used in reverse shell payloads to execute system command and obfuscate the data being sent.

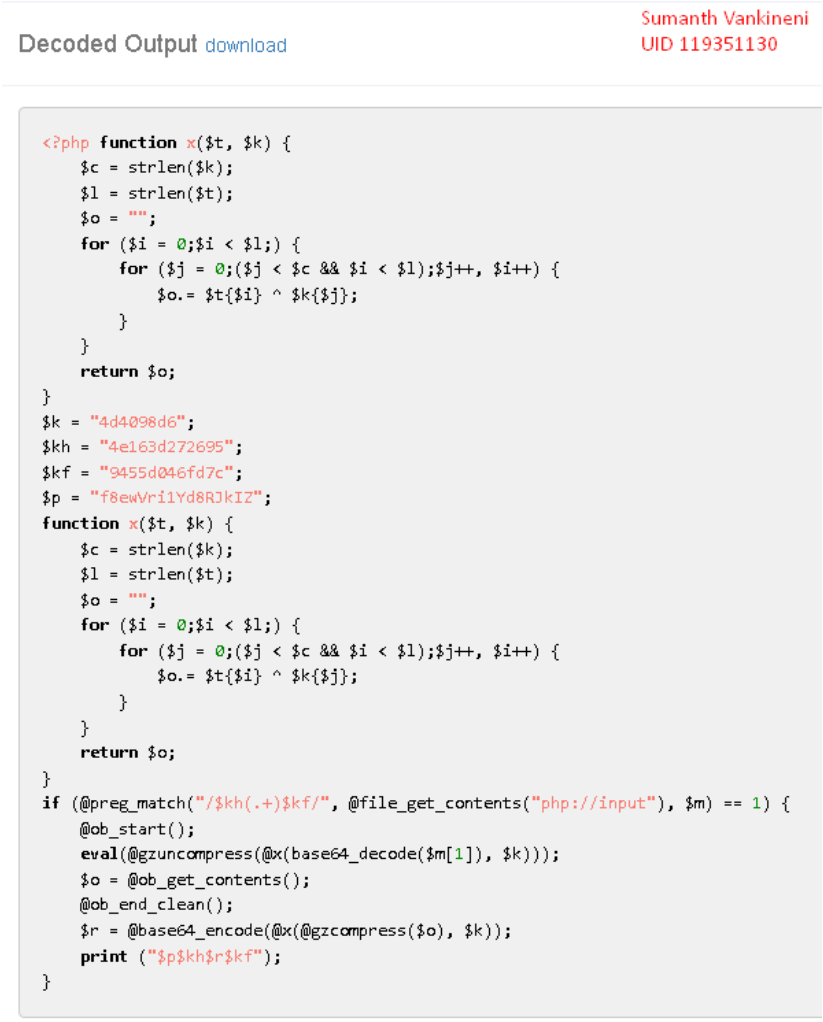

Figure 14, re constructed php code

I've used Splunk to analyse the logs and hence confirmed my previous assumptions.

Figure 15, Splunk analysis of access log file

2)What did the attacker do once they were on the system?

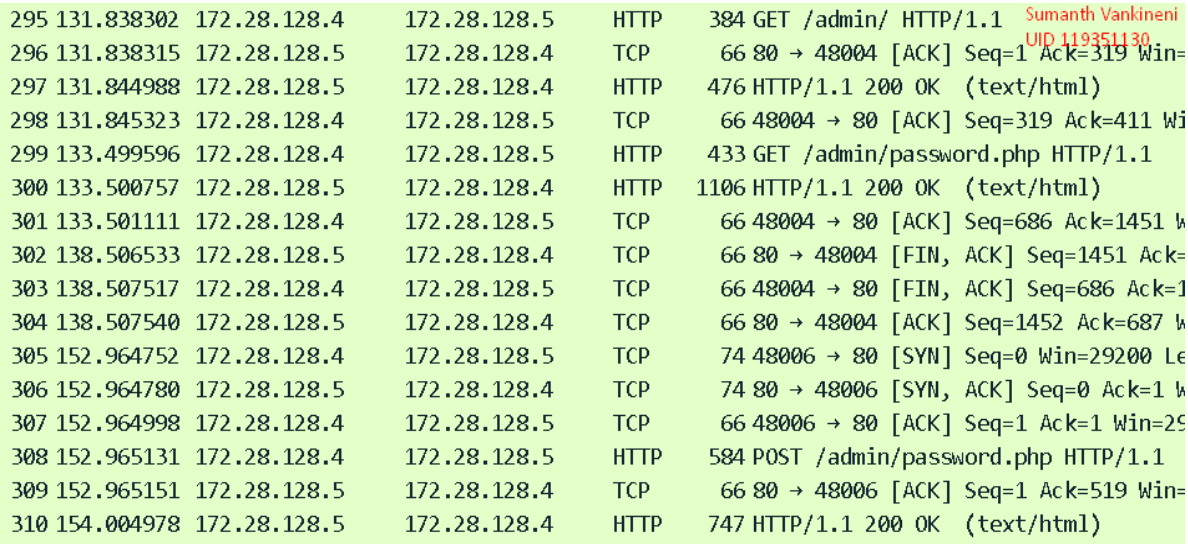

Figure 16, Posting data to password.php

On further anlysis of the PCAP file it can be noticed that the attacker accessed a password.php page on the website and posted data. Further expanding the fragment as show in the figure, the details submitted into the fields of the password.php page can be clearly seen.

So here the attacker has changed password for the user Julia with the password as "hacked". The atatcker specifically changed the password for Julia since the about.php page showed that Julia is the one who maintains the backend of the test app. The attacker attempted to change Julia's password, which he could further utilize in their subsequent plans.

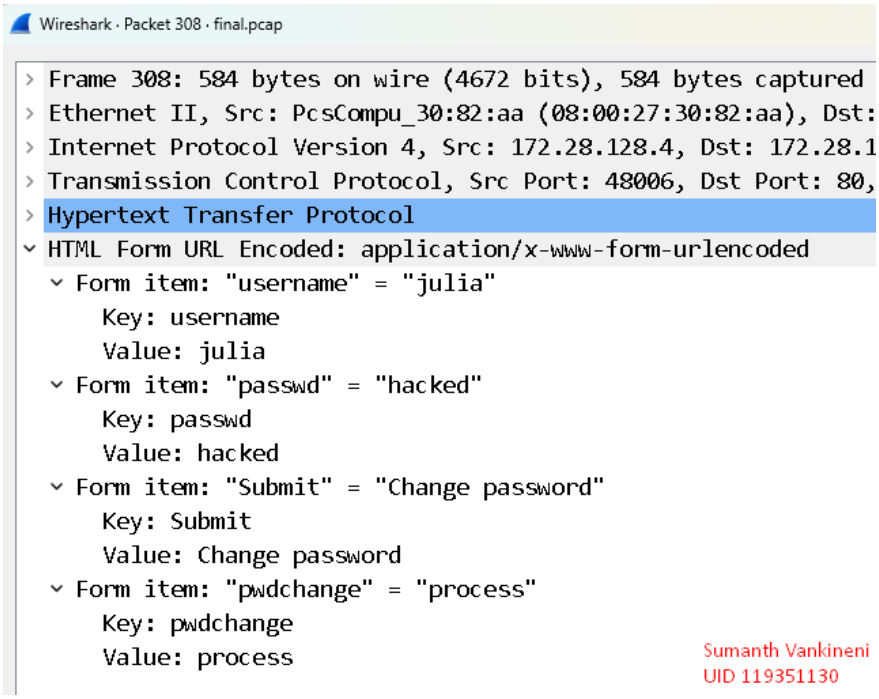

Figure 17, Changed password details.

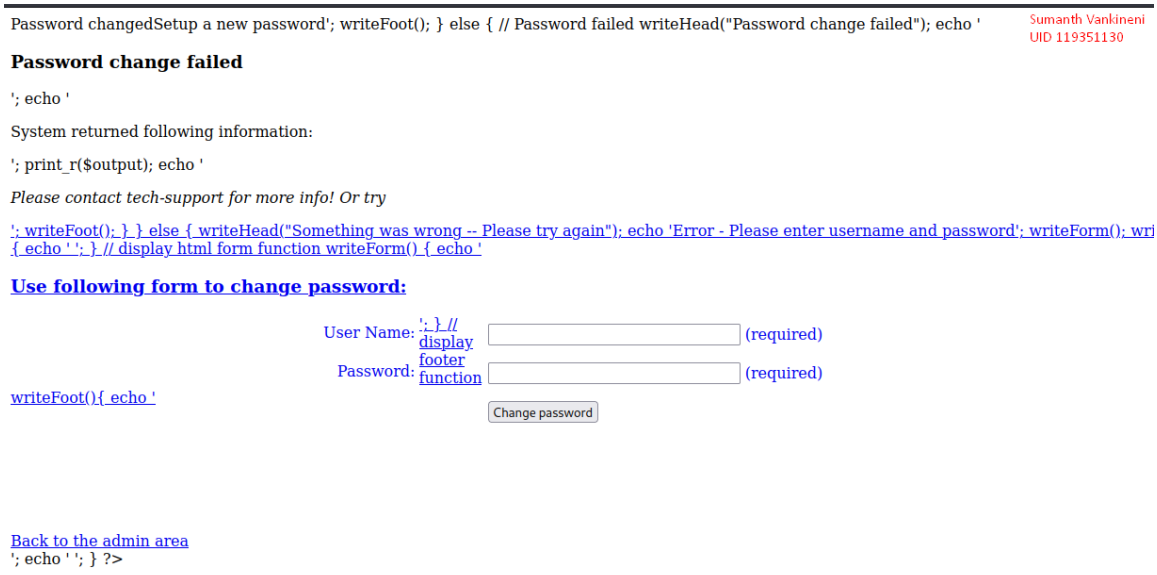

Figure 18, Change password webpage.

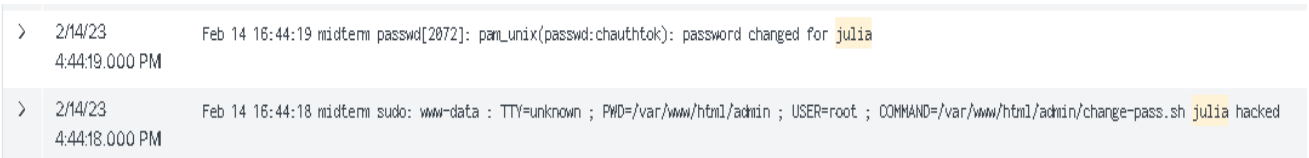

Figure 18, Splunk logs of changing password

The above screenshot is taken from Splunk which shows that password has been changed.

```
┌──(kali㉿kali)-[/var/www/html/mywebsite/admin]
└─$ cat password.php
<?php
// change .. me! - shell script name
$shellscript = "sudo /var/www/html/admin/change-pass.sh";

// Make sure form is submitted by user
if(!(isset($_POST['pwdchange']))) {
 // if not display them form
 writeHead("Change password");
 writeForm();
 writeFoot();
}
else {
 // try to change the password
 $callshell=true;
 // get username and password
 $_POST['username'] = stripslashes(trim($_POST['username']));
 $_POST['passwd'] = stripslashes(trim($_POST['passwd']));

// if user skip our javascript ...
// make sure we can only change password if we have both username and password
 if(empty($_POST['username'])) {
   $callshell=false;
```

Sumanth Vankineni
UID 119351130

Figure 19, Password.php source code

```
┌──(kali㉿kali)-[/var/www/html/mywebsite/admin]
└─$ cat change-pass.sh
#!/bin/sh
# \
exec expect -f "$0" ${1+"$@"}
set password [lindex $argv 1]
spawn passwd [lindex $argv 0]
sleep 1
expect "assword:"
send "$password\r"
expect "assword:"
send "$password\r"
expect eof
```

Sumanth Vankineni
UID 119351130

Figure 20, pass.sh code

```
┌──(kali㉿kali)-[~/…/Final/julia/home/julia]
└─$ ls -al
total 32
drwxr-xr-x 3 kali kali 4096 Feb 14  2019 .
drwxr-xr-x 3 kali kali 4096 May 13 15:56 ..
-rw------- 1 kali kali   35 Feb  9  2019 .bash_history
-rw-r--r-- 1 kali kali  220 Feb  5  2019 .bash_logout
-rw-r--r-- 1 kali kali 3637 Feb  5  2019 .bashrc
drwx------ 2 kali kali 4096 Feb  5  2019 .cache
-rw------- 1 kali kali   16 Feb 14  2019 .mysql_history
-rw-r--r-- 1 kali kali  675 Feb  5  2019 .profile

┌──(kali㉿kali)-[~/…/Final/julia/home/julia]
└─$ cat .bash_history
exit
passwd
clear
exit
passwd
exit
```

Sumanth Vankineni
UID 119351130

Figure 21, Bash history

The figures 19,20 and 21 show the content from Julia's directory, the .bash_history shows the proof that the password had been changed for the user Julia. The screen was also cleared before exiting the session.

I further checked the source code of the password.php which does not have any authentication or authorization measures taken which allow an user accessing the page to change the password of any user.

Next, the attacker connected to the Waffle Co's system using SSH protocol.

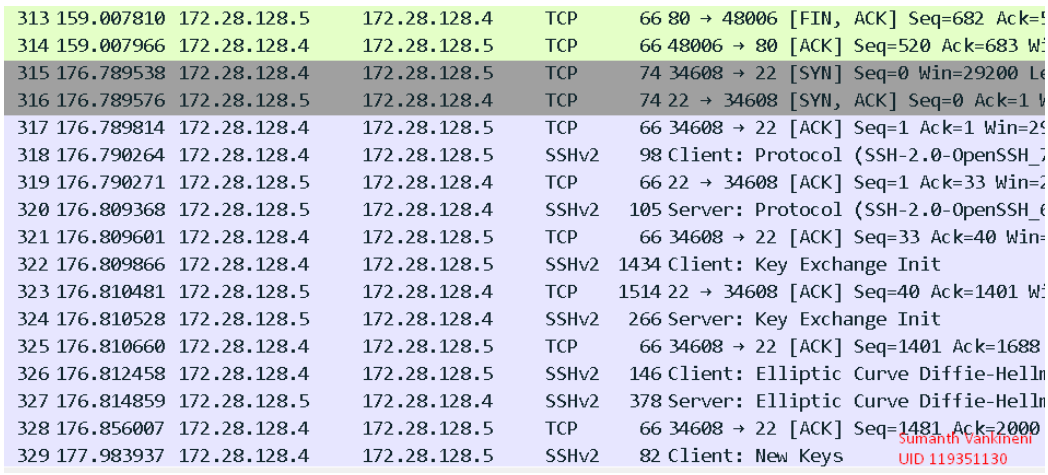

Figure 22, Wireshark showing ssh connection

"ps OR sudo" is the filter I used for the following Splunk analysis.

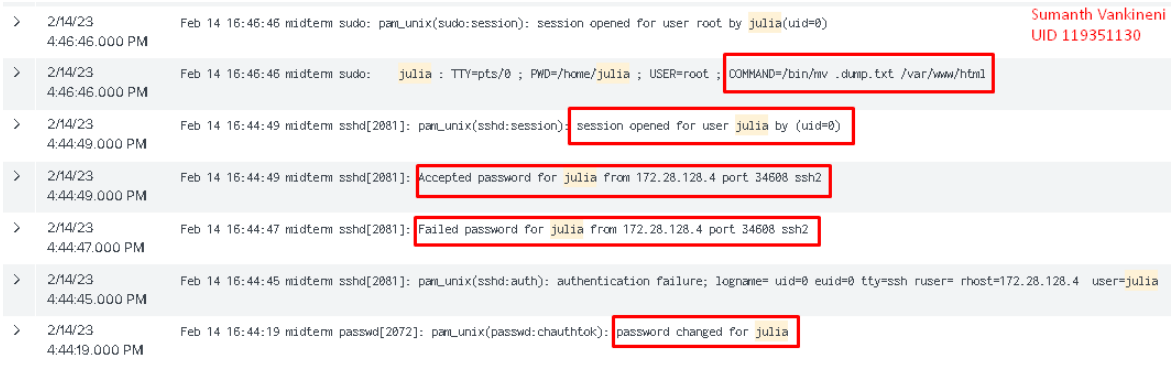

Figure 23, Splunk analysis of attacker's commands on system.

The attacker already tried to ssh into the system but failed. The second attempt he was successful in connecting to the system using ssh as shown in the figure 23 above.

The above screenshot from Splunk of the logs file shows that the attacker connected to the system with the user Julia, performed a move operation of the.dump.txt to the /var/www/html directory using Sudo permission.

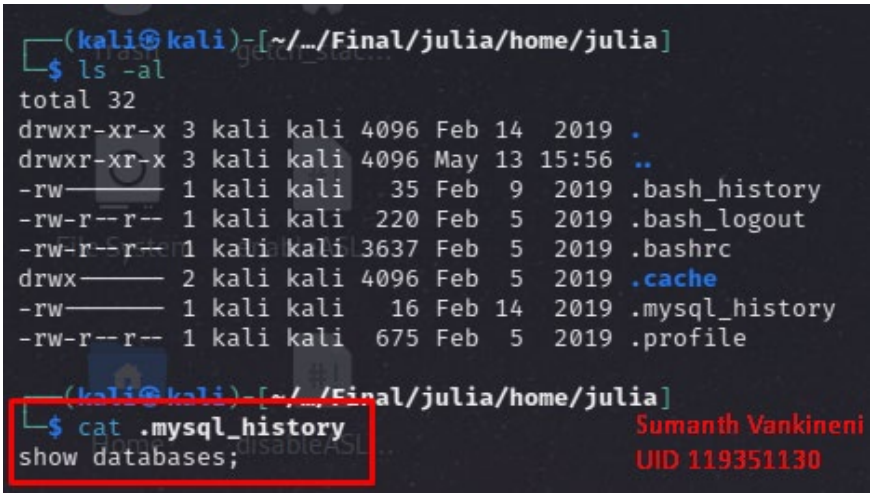

Figure 24, sql query history

The .mysql_history files shows that the command show databases was used. So, after viewing the content of the tables, the attacker must have moved the .dump file.

3.Was sensitive data accessed? How can you tell if it was/was not accessed?

The PCAP file shows that the attacker accessed the .dmp.txt file from the website, as he had earlier moved to it be accessible from the Waffle Co's website.

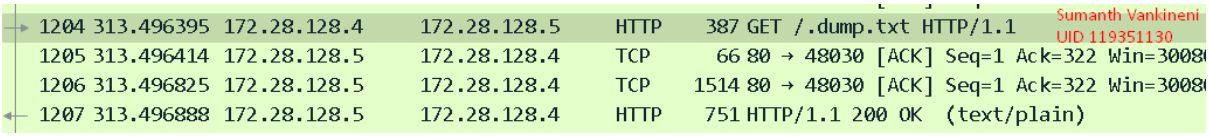

Figure 25, Downloading the .dump file

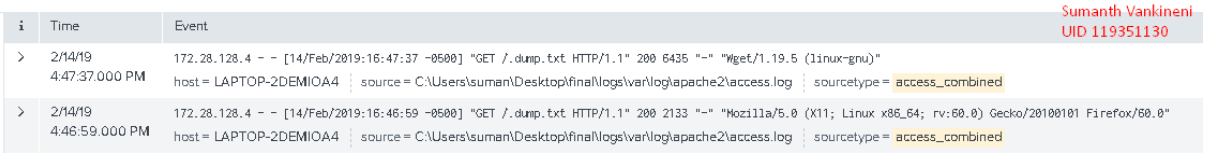

Figure 26, Downloading the .dump file

Using the following filter on Splunk for the above figure 26.

sourcetype="*access_*" | rex field=_raw "cmd=(?<command>[^&]+)"

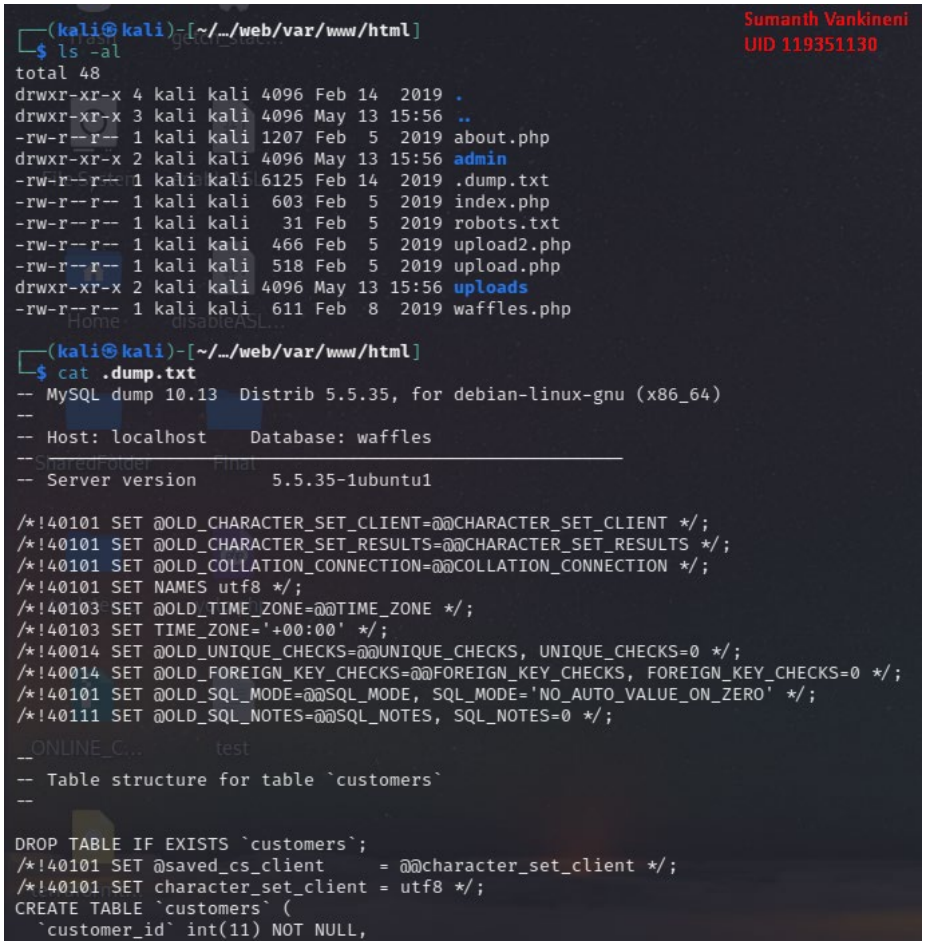

Figure 27, Dump file output

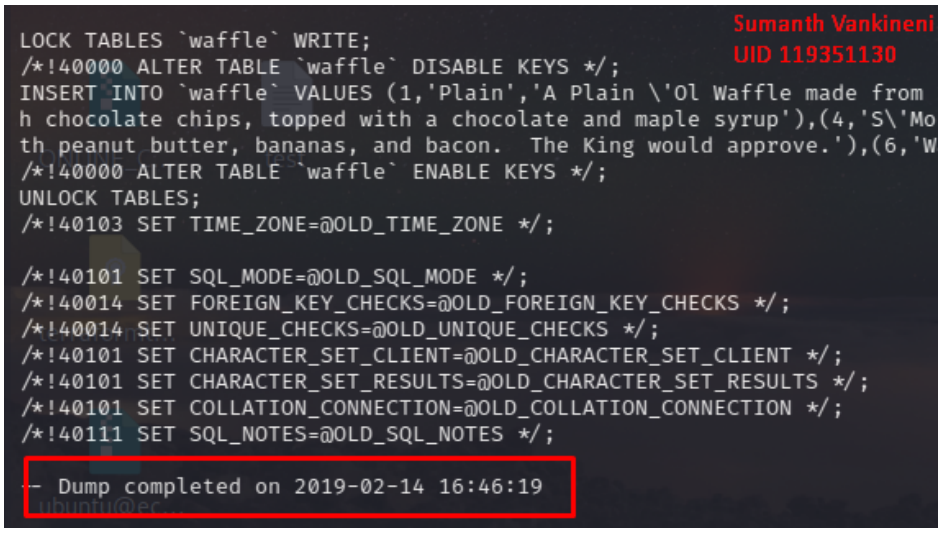

Figure 28, Dump file ouput

The above figures 27,28 show the content of the .dump.txt file. The .dump.txt file is usually created as a memory snapshot. The attacker downloaded the file twice. The following screenshots illustrate the database contents. I have utilized the snapshot to generate a table for enhanced data visualization.

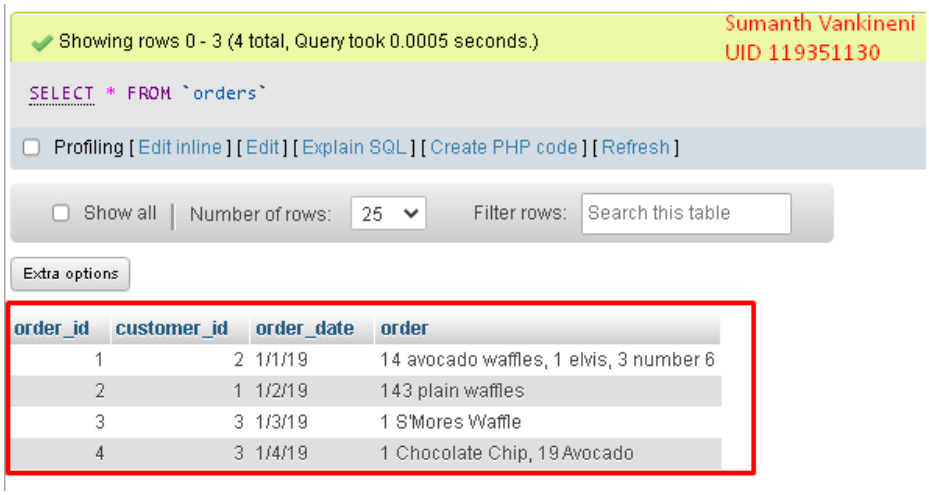

Figure 29, Orders table

This table contains the order history details. Though there is not any sensitive information here it could be used for social engineering.

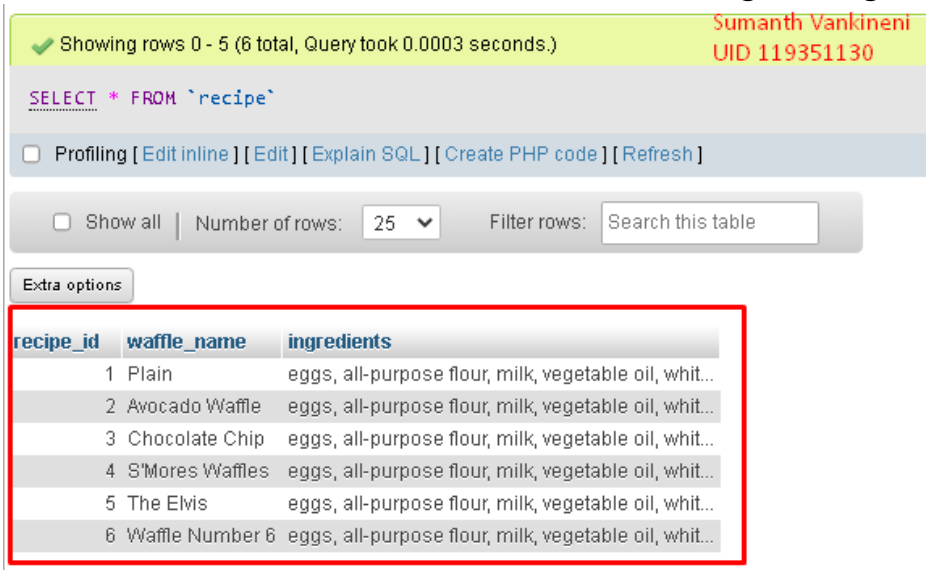

Figure 30, Recipe table

The above table contains the recipes for different dishes by Waffle Co. This is very sensitive information as the recipe for any food industry is their trade secret which is invaluable to them.

Figure 31, Waffle table

The above table contains the details of the descriptions of each waffle making process. That is not sensitive but is still a valuable source of data.

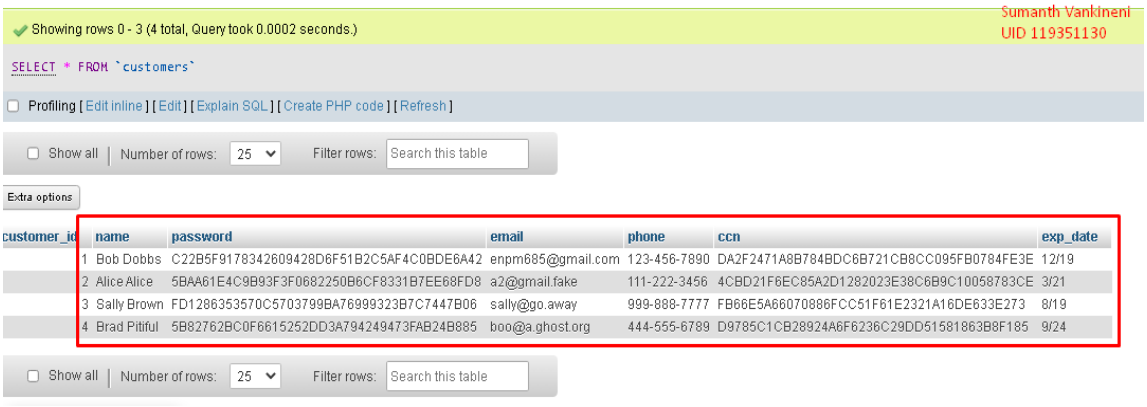

Figure 32, Customer details

The above table contains the Customer details including their name, email-id, phone number and their credit card numbers with the expiry date respectively.

4. Were you able to learn anything about the attacker? (What were their attack tools, tactics, techniques, and procedures?)

Upon my analysis the attacker could have been a customer of the waffle co who found the website vulnerable. The attacker misused the lack of security practices implemented in the website and exploited them.
After examining the logs, tcpdump PCAP files, and other traces left by the attacker, the attacker employed a methodology or techniques which I've listed below.

Reconnaissance: Here the attacker gathered information regarding the target website, here which is owned by the Waffle Co. Here the attacker first visited the website, accessed the different available webpages.

Resource Development: Based on the data the attacker collected after scanning through the website, the attacker crafted the required payloads to exploit the system.

Initial Access: To gain the Initial access, the attacker used his crafted payload written in PHP to exploit the file upload vulnerability on the Waffle photo contest page. This gave the attacker the remote code execution access to the Waffle Co's system.

Execution: Using the remote code execution connection to the target system which here is owned by the Waffle Co, the attacker executed some malicious commands.

Privilege escalation: Since the attacker gained access to the user Julia, he was able to execute some command with sudo permissions which therefore provide higher privileges.

Défense Evasion: The php malicious code the attack injected was obfuscated in order to avoid any detection by antivirus systems or malware detections software's if installed in the victim's system.

Credential Access: The attacker found a big vulnerability where any user who has access to the password.php page could change the user's password without any authentication or validation. So here the attacker exploited this vulnerability and changed the password for the user Julia who is the backed developer for the website of Waffle Co.

Persistence: Since the remote code execution or the reverse shell isn't a very stable connection the victims' system. So, the attacker connected to the Waffle Co's system using ssh with the acquired credentials as stated in the previous step.

Discovery: Here the attacker found a major vulnerability where any person with access to the website can changed the password of users.

Lateral Movement: The attacker after connection to the Waffle Co's system using ssh traversed the directories to look for potential sensitive fields containing sensitive information.

Collection: The attacker found the .dump.txt file which is basically a snapshot of the database. This was a major finding for the attacker.

Exfiltration: The attacker here didn't use any secure copy protocol or other protocols to transfer the sensitive information (The Database snapshot). Instead, the attacker moved the file to the /var/www/html folder which can be accessed from the internet and can be remotely downloaded via accessing the website.

Impact: The attacker extracted the database snapshot to his system. The database consisted of very sensitive information such as the personal information of the customers along with their credit card details. The attacker also got the recipes of the waffle Co' which could be their trade secret. Here in this case the attacker sent an email to Nathan claiming to have all of the company's data and offering to sell it to the highest bidder.

Conclusion: Since no proper security measures ere taken during the build phase of the website for the Waffle Co, the attacker was able to exploit its vulnerabilities and gained access to sensitive information. As a security investigator, I will try my best to track down the attacker. Until then my suggestion to Waffle Co is to temporarily inform all the customers that the website has been compromised and not to access it. The customers whose sensitive information have been leaked should also be made aware of that so that they can take the precautionary measures such as blocking their credit card from the bank