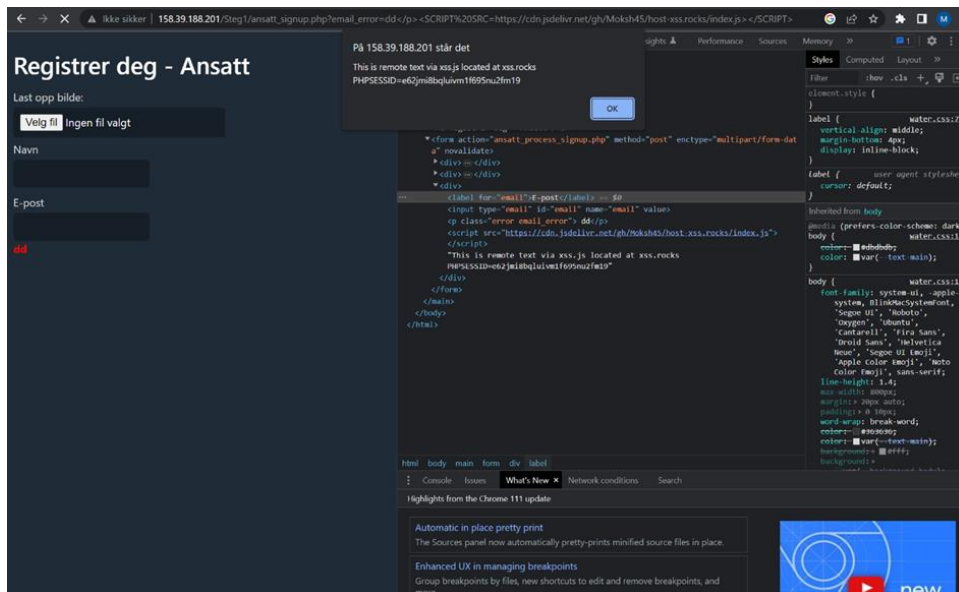


Gruppe 1:

XSS, tilbakemelding til bruker ved inputvalidering



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.



Crawl results - click to expand:



Document type overview - click to expand:

application/xhtml+xml (15)

1. <http://158.39.188.201/> (1073 bytes) [show trace +]
2. http://158.39.188.201/Steg2/gjest_bruker_autentisering.php (814 bytes) [show trace +]
3. http://158.39.188.201/Steg2/student_signup.php (3264 bytes) [show trace +]
4. <http://158.39.188.201/dokumentasjon/> (810 bytes) [show trace +]
5. <http://158.39.188.201/icons/> (199 bytes) [show trace +]
6. <http://158.39.188.201/Steg1/> (536 bytes) [show trace +]
7. <http://158.39.188.201/Steg1/img/> (7974 bytes) [show trace +]
8. <http://158.39.188.201/Steg1/login.php> (1133 bytes) [show trace +]
9. <http://158.39.188.201/Steg2/> (520 bytes) [show trace +]
10. <http://158.39.188.201/Steg2/styles/> (2190 bytes) [show trace +]
11. http://158.39.188.201/Steg2/ansatt_signup.php (2944 bytes) [show trace +]
12. <http://158.39.188.201/Steg2/login.php> (1087 bytes) [show trace +]
13. <http://158.39.188.201/Steg2/reset-request.php> (325 bytes) [show trace +]
14. <http://158.39.188.201/Steg2/reset-request.php> (3363 bytes) [show trace +]
15. http://158.39.188.201/Steg2/signup_choose.php (690 bytes) [show trace +]

image/gif (5)

1. <http://158.39.188.201/icons/a.gif> (246 bytes) [show trace +]
2. <http://158.39.188.201/icons/back.gif> (216 bytes) [show trace +]
3. <http://158.39.188.201/icons/blank.gif> (148 bytes) [show trace +]
4. <http://158.39.188.201/icons/folder.gif> (225 bytes) [show trace +]
5. <http://158.39.188.201/icons/image2.gif> (309 bytes) [show trace +]

image/jpeg (4)

1. <http://158.39.188.201/Steg1/img/1676290994177092602063ea2bb23ca88.jpg> (400000 bytes) [show trace +]
2. <http://158.39.188.201/Steg1/img/1676542719148755397563ee02fac910.jpeg> (9117 bytes) [show trace +]
3. <http://158.39.188.201/Steg1/img/1676545501160486069663ee0ddd14d8e.jpg> (15316 bytes) [show trace +]
4. <http://158.39.188.201/Steg1/img/1676678587124868585763f015bbdc23e.jpg> (44733 bytes) [show trace +]

image/png (7)

1. <http://158.39.188.201/icons/a.png> (189 bytes) [show trace +]
2. <http://158.39.188.201/icons/blank.png> (105 bytes) [show trace +]
3. <http://158.39.188.201/icons/image2.png> (229 bytes) [show trace +]
4. <http://158.39.188.201/icons/link.png> (188 bytes) [show trace +]
5. <http://158.39.188.201/Steg1/img/1676539324129263219063edf5bc5d238.png> (205191 bytes) [show trace +]
6. <http://158.39.188.201/Steg1/img/167991761118922454926421822bb4367.jpg> (377 bytes) [show trace +]
7. http://158.39.188.201/Steg1/img/default_img.png (11495 bytes) [show trace +]

text/css (2)

1. http://158.39.188.201/Steg1/index_style.css (548 bytes) [show trace +]
2. <http://158.39.188.201/Steg2/style.css> (72 bytes) [show trace +]

text/html (4)

text/plain (1)

1. <http://158.39.188.201/Steg1/img/1676542843147068534063ee037b39c2c.jpeg> (24 bytes) [show trace +]

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 fungerte 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
```