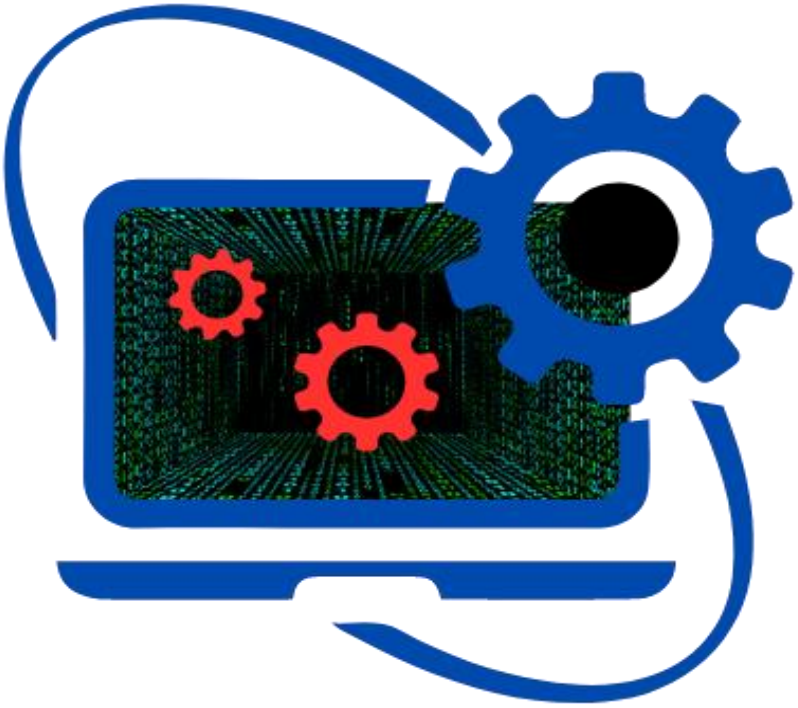


< SECUREWEB />

DEFENCE

Teknisk Sikkerhetsrevisjon av

Boris Lockpicks

Kandidatnummer: 2056

Executive Summary

I denne sikkerhetsrevisjonen presenteres resultater fra en omfattende penetrasjonstest av webapplikasjonen «Boris Lockpicks». Sikkerhetsrevisjonen har avdekket flere alvorlige sårbarheter. Disse sårbarhetene gir potensielle angripere mulighet til å manipulere eller ta fullstendig kontroll over applikasjonen.

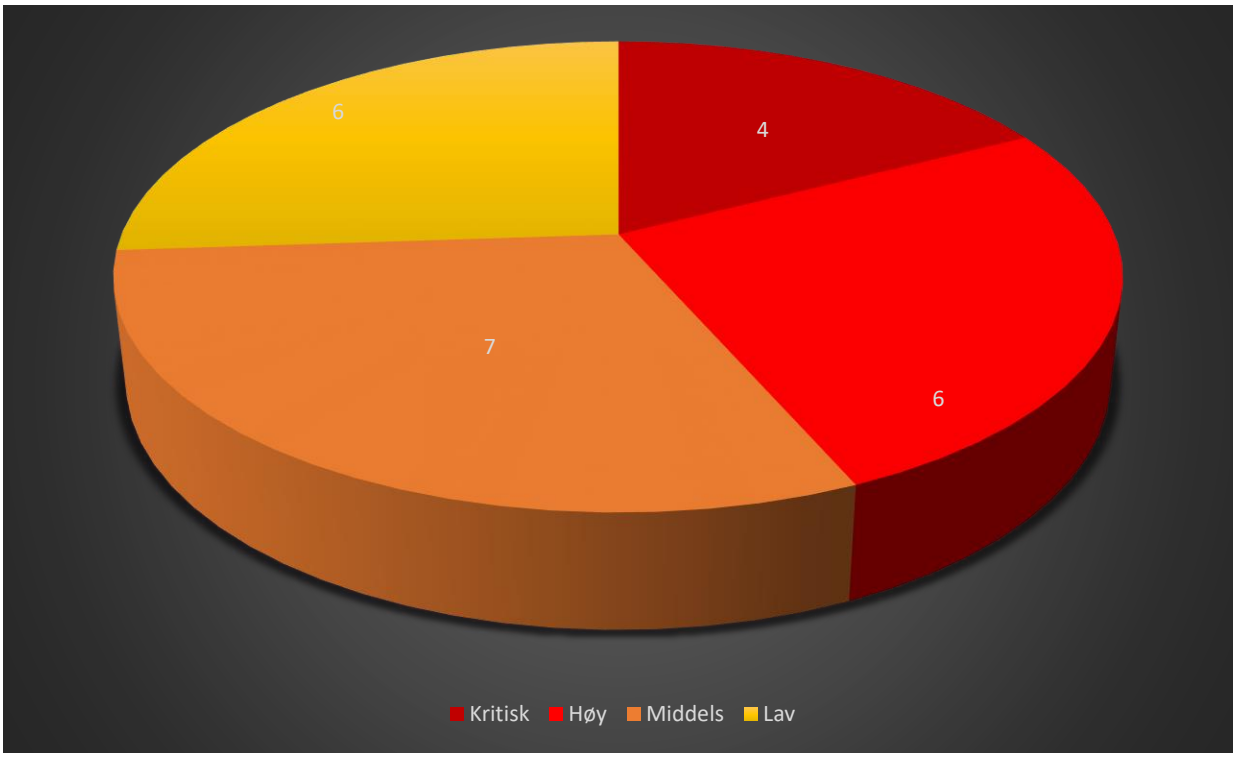
Blant de mest alvorlige tillater FTP protokollen på port 21 anonym innlogging, dette eksponerer sensitive filer, inkludert sensorveiledningen. Dette kan enkelt løses ved å benytte protokollen FTPS istedenfor FTP protokollen, for økt sikkerhet. Både Webservereren og SSH protokollen er sårbare grunnet svake passord, dette kan enkelt løses ved å sette sterkere passord.

En alvorlig sårbarhet er webapplikasjonens mottakelighet for IDOR angrep, som gir uautorisert tilgang til brukerdata uten nødvendige kredentialer, dette må fikses umiddelbart for å beskytte kundedata.

Applikasjonen er også utsatt for cross-site scripting angrep, derav typene reflected og persistent. Det er mulig å kjøre kodeinjesering rett i gjesteboken og koden vil kjøre hver gang noen laster inn siden.

Webapplikasjonen er også sårbar for SQL injection, med verktøyet sqlmap vil en angriper kunne dumpe ut brukerdata, kortinformasjon og passordhasher. Disse hashene er kryptert med den utdaterte md5 krypteringsnøkkelen, en sterk anbefaling er å kryptere passordene med Bcrypt. Videre anbefales kryptering av kortinformasjon, for å forhindre datalekkasjer.

For å få en grundigere oversikt over alle sårbarhetene funnet, vennligst ta en titt på resten av sikkerhetsrevisjonen. Under er det vist en visualisering av antall sårbarheter fordelt utover risikonivå.



Introduksjon til rapport

Denne penetrasjonstesten er utført av meg som penetrasjonstester ved SecureWeb defence. Denne sikkerhetsrevisjonen tar for seg resultatene fra en penetrasjonstest utført for å identifisere og evaluere sikkerhetssårbarheter i Webapplikasjonen «Boris Lockpicks» til selskapet Eastwill Security. Denne applikasjonen er ment for at kunder skal kunne bestille diverse låsdirke utstyr på nettet, se på låsdirkingsvideoer og kommunisere i gjesteboken. Formålet med testen var å etterligne en ondsinnet hacker og avdekke sikkerhetshull som kan utnyttes av ondsinnede aktører. Vi har fokusert på å oppdage flest mulige alvorlige sikkerhetsrisikoer og anbefalt flere tiltak for å forbedre applikasjonens totale sikkerhet. Denne sikkerhetsrevisjonen gir en detaljert oversikt over våre funn.

Innholdsfortegnelse

Executive Summary 2

Introduksjon til rapport 3

Oversikt over sårbarheter 5

Gjennomførelse 6

Type penetrasjonstest 7

Verktøy brukt for testing 7

Sårbarhetsinformasjon 8

Detaljert beskrivelse av sårbarheter 9

Sårbarhet 1 – port 21 eksponerer sensorveiledning 9

Sårbarhet 2 - Port 9999 sårbar for brute-force angrep 11

Sårbarhet 3 – IDOR sårbarhet 14

Sårbarhet 4- Servertilgang gjennom port 22 19

Sårbarhet 5 – Sql injection 21

Sårbarhet 6- Manglende validering av inndata 22

Sårbarhet 7 – Flere åpner porter 25

Sårbarhet 8- Webapplikasjonen er sårbar for XSS 26

Sårbarhet 9- Svake passord 29

Sårbarhet 10 – Path Traversal 30

Sårbarhet 11 – Anti CSRF tokens mangler 34

Sårbarhet 12- Feilmeldinger fra databasen vises 35

Sårbarhet 13- CSP header ikke satt 36

Sårbarhet 14 – Skjult fil funnet 37

Sårbarhet 15 – Port 42420 server ukrypterte data over HTTP 39

Sårbarhet 16 – Anti-clickjacking header mangler 40

Sårbarhet 17 - Port 9999 usikker autentiseringsmetode 41

Sårbarhet 18 – Cookie uten http only flagg 43

Sårbarhet 19 – Cookie uten secure flagg 43

Sårbarhet 20 - Cookie uten SameSite attributt 44

Sårbarhet 21 - Server lekker versjonsnummer 45

Sårbarhet 22 - Strict-Transport-Security-header er ikke satt 46

Sårbarhet 23 - X-Content-Type-Options Header mangler 47

Konklusjon 48

Oversikt over sårbarheter

Kategori	Sårbarhet	Risiko
Sårbarhet 1	Port 21 eksponerer sensorveiledning	Kritisk
Sårbarhet 2	Port 9999 sårbar for brute-force angrep	Kritisk
Sårbarhet 3	IDOR sårbarhet	Kritisk
Sårbarhet 4	Servertilgang gjennom port 22	Kritisk
Sårbarhet 5	Sql Injection	Høy
Sårbarhet 6	Manglende validering av inndata	Høy
Sårbarhet 7	Flere åpne porter	Høy
Sårbarhet 8	Webapplikasjonen er sårbar for XSS	Høy
Sårbarhet 9	Svake passord	Høy
Sårbarhet 10	Path Traversal	Høy
Sårbarhet 11	Anti CSRF tokens mangler	Middels
Sårbarhet 12	Feilmeldinger fra databasen vises	Middels

Sårbarhet 13	CSP header ikke satt	Middels
Sårbarhet 14	Skjult fil funnet	Middels
Sårbarhet 15	Port 42420 server ukrypterte data over HTTP	Middels
Sårbarhet 16	Anti-clickjacking header mangler	Middels
Sårbarhet 17	Port 9999 usikker autentiseringsmetode	Middels
Sårbarhet 18	Cookie uten http only flag	Lav
Sårbarhet 19	Cookie uten secure flag	Lav
Sårbarhet 20	Cookie uten SameSite attributt	Lav
Sårbarhet 21	Server lekker versjonsnummer	Lav
Sårbarhet 22	Strict-Transport-Security-header er ikke satt	Lav
Sårbarhet 23	X-Content-Type-Options Header mangler	Lav

Gjennomførelse

- Testen er rettet mot IP adressen til webapplikasjonen til Boris Lockpicks: 192.168.245.137/…/192.168.245.141, ettersom IP-adressen endret seg gjennom testen.
- Ved bruk av IP-en kan det benyttes ulike verktøy for å lete etter potensielle sårbarheter.

- Denne penetrasjonstesten blir gjennomført som en del av eksamen til faget ETH2100, og oppdraget ble tildelt av Bengt Østby.

Type penetrasjonstest

Denne penetrasjonstesten utføres som en whitebox test. En whitebox test går ut på at man har tilgang til maskinen og/eller kildekoden til applikasjonen man skal teste. I dette tilfellet fikk vi tildelt kildekoden. I en whitebox test har man muligheten til å identifisere potensielle sikkerhetshull og forutsi forventede resultater. Penetrasjonstesting er nødvendig fordi de fleste selskapene ønsker høy sikkerhet, og vil helst at en uautorisert bruker ikke skal få tilgang til sensitiv informasjon. Gjennom en grundig penetrasjonstest vil selskapet få overblikk i sikkerheten til applikasjonen, og finne ut om det er eventuelle feil som må rettes opp i for å forhindre eventuelle cyberangrep. En whitebox test er som oftest den anbefalte måten å teste på, ettersom en ondsinnet hacker vil ha ubegrenset med tid til å finne sårbarheter og misbruke disse, mens en pentester har en frist å forholde seg til. Å gi pentesteren tilgang til kildekoden legger til rette for en grundigere og mer utfyllende jobb enn det en blackbox test gjør.

Verktøy brukt for testing

I denne testen har jeg benyttet meg av ulike verktøy for å identifisere sårbarheter. Verktøyene jeg har benyttet meg av er listet under:

- Owasp Zap

- Burp Suite
- Nmap
- Wireshark
- Zenmap
- SQLmap
- Hydra

Sårbarhetsinformasjon

Kritisk	Kritiske sårbarheter er sårbarheter som kan tillate en angriper å få kontroll over systemet, stjele sensitive data og/eller forårsake betydelig skade.
Høy	Sårbarheter klassifisert som Høy kan lede til betydelig skade. Disse er ikke like alvorlige som kritiske sårbarheter, men kan fortsatt utføre stor skade. Det er viktig at slike sårbarheter fikses raskt.
Middels	Middels sårbarheter er vanskeligere å utnytte enn kritiske og høye sårbarheter, men disse kan fortsatt utgjøre betydelig skade. Det er viktig å fikse middels sårbarheter før de kan utgjøre noen form for skade.

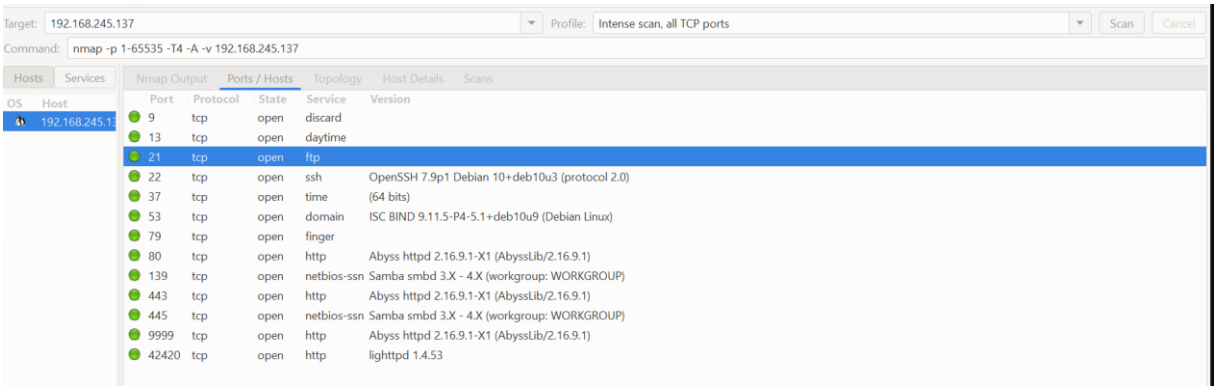
Lav	Lave sårbarheter er sårbarheter som ikke direkte utgjør noen stor sikkerhetsrisiko. Men det er fortsatt viktig å fikse disse, for å forbedre applikasjonens totale sikkerhet.
Info	Dette er ikke sårbarheter, men generell informasjon som kan hjelpe med å forstå systemet bedre.

Detaljert beskrivelse av sårbarheter

Sårbarhet 1 – port 21 eksponerer sensorveiledning

Risiko: Kritisk

En full port-scan med kommandoen «nmap -p 1-65535 -T4 -A -v <ip>» viser at port 21 er åpen. Viser til utklipp under.



Denne porten i seg selv er meget sårbar, ettersom FTP protokollen sender informasjon over klartekst, og tilbyr heller ingen form for kryptering.

For å se data sendt gjennom port 21 vil en angriper kunne koble seg til ftp porten med kommandoen «ftp <ip>» fra kali, og logge inn med anonymous som brukernavn og passord for å få anonym tilgang til porten. Viser til utklipp under.

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.245.139
Connected to 192.168.245.139.
220 ProFTPD Server (BorisLockpick) [192.168.245.139]
Name (192.168.245.139:kali): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230-Welcome, archive user anonymous@192.168.245.138 !
230-
230-The local time is: Wed Nov 15 14:06:38 2023
230-
230-This is an experimental FTP server.  If you have any unusual problems,
230-please report them via e-mail to <root@osboxes>.
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -a
229 Entering Extended Passive Mode (|||29407|)
150 Opening ASCII mode data connection for file list
drwxr-xr-x   2 ftp      ftp          4096 Oct  2 00:35 .
drwxr-xr-x   2 ftp      ftp          4096 Oct  2 00:35 ..
-rw-r--r--   1 ftp      ftp        486787 Oct  2 00:35 eksamen_ETH2100_H23.del2_sensorveiledning.pdf
-rw-r--r--   1 ftp      ftp           170 Aug 30  2021 welcome.msg
226 Transfer complete
ftp> get eksamen_ETH2100_H23.del2_sensorveiledning.pdf
local: eksamen_ETH2100_H23.del2_sensorveiledning.pdf remote: eksamen_ETH2100_H23.del2_sensorveiledning.pdf
229 Entering Extended Passive Mode (|||62657|)
150 Opening BINARY mode data connection for eksamen_ETH2100_H23.del2_sensorveiledning.pdf (486787 bytes)
   475 KiB    4.85 MiB/s
226 Transfer complete
486787 bytes received in 00:00 (4.72 MiB/s)
```

Etter at angriperen har fått tilgang til FTP-serveren vil angriperen få tilgang til sensorveiledningen. Denne filen kan nå hentes ut med kommandoen «get eksamen_ETH2100_H23.del2_sensorveiledning.pdf». Dermed blir filen eksportert ut til den lokale maskinen og kan åpnes uten videre. Filen er vist i utklippet under.

Høyskolen Kristiania

Emnekode: ETH2100
Emnenavn: Etisk Hacking
Vurderingskombinasjon: Mappevurdering
Innleveringsdato: 22. desember 2023
Filformat: PDF m/ vedlegg

SENSORVEILEDNING OG FASIT

Neida, det ville vært ganske dumt hvis dere klarte å finne sensorveiledning og fasit til eksamen – inne i eksamens VMen. Det ville vært en ganske stor tabbe av foreleser…

Men denne filen er lagt her på en ikke-standard port for at de som gjør en grundig jobb skal finne den. Dette skal rapporteres i pentest rapporten som en KRITISK sårbarhet, og rapporteres som «Port 42420 eksponerer sensorveiledning».

- Bengt

Side 1 av 1

Husk å oppgi kandidatnummer på din besvarelse, ikke studentnummer.

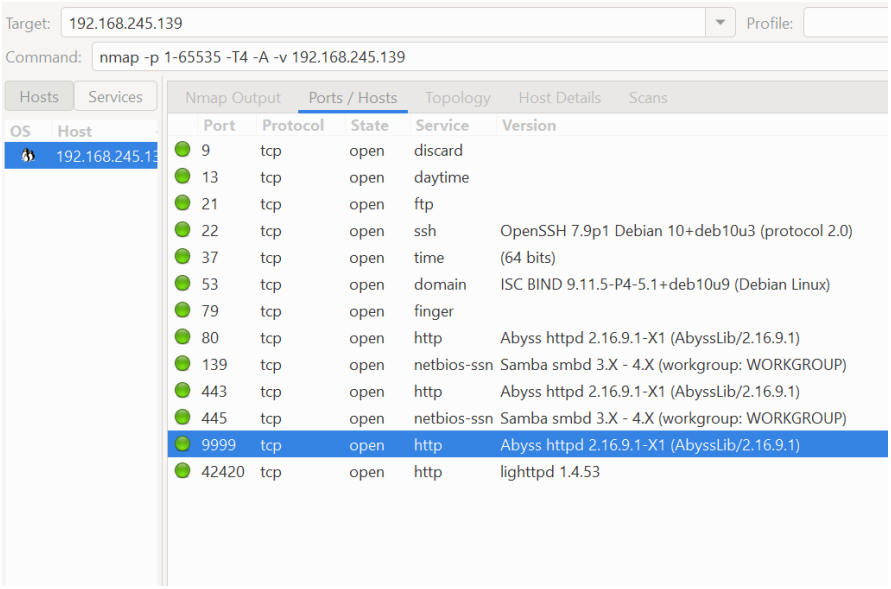
Dette er en fil som ikke burde ligge tilgjengelig for uautoriserte brukere. Istedenfor å bruke FTP protokollen for å overføre data anbefales det at FTPS protokollen blir brukt istedenfor. FTPS protokollen tillater kryptering av dataene og legger til rette for en sikrere filoverføring. Les mer om FTPS her: https://www.thruinc.com/blog/what-is-ftps/.

Sårbarhet 2 - Port 9999 sårbar for brute-force angrep

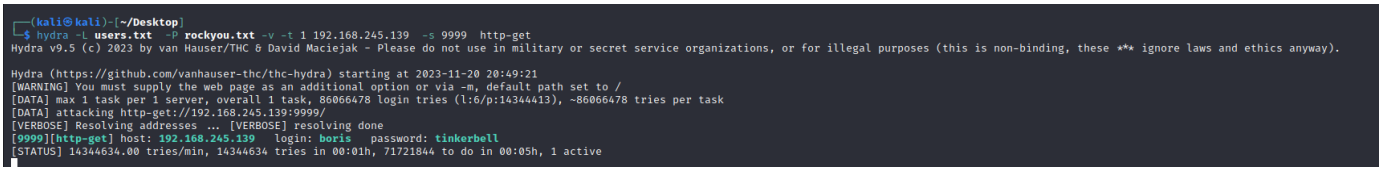
Risiko: Kritisk

Brute-force angrep er en angrepsmetode der angriperen prøver ulike kombinasjoner av brukernavn og passord for å prøve å knekke innloggingen, dette kan gjøres manuelt eller ved hjelp av automatiserte verktøy som blant annet Hydra.

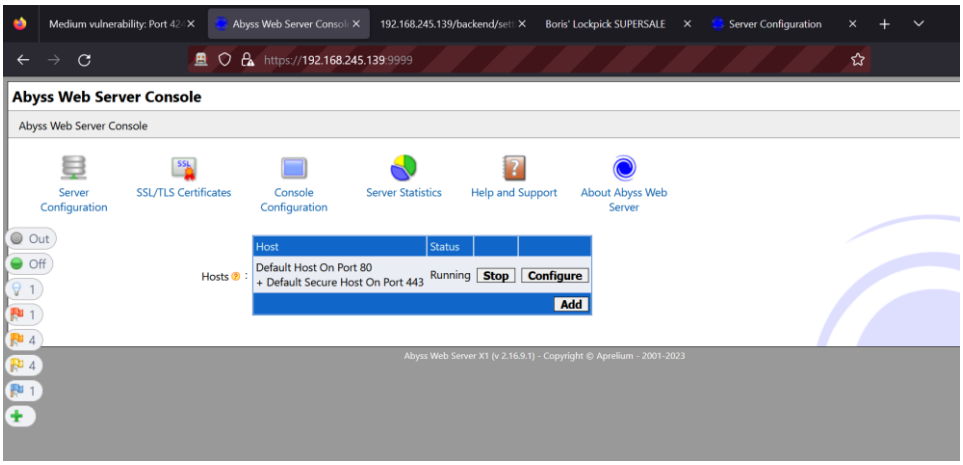
En full port-scan med kommandoen «nmap -p 1-65535 -T4 -A -v <ip>» viser at port 9999 er åpen, og kjører Abyss httpd 2.16.9.1-X1. Viser til utklipp under.



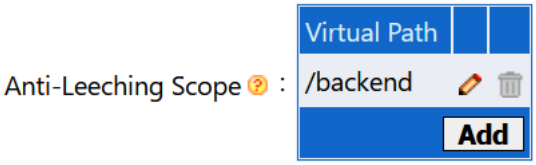
Denne porten kan en bruker få tilgang til gjennom en nettleser, med URL-en «https://<ip>:9999». Deretter vil brukeren få opp et innloggingsskjema. Dette skjemaet har ingen restriksjoner på antall innloggings forsøk, dette gjør den sårbar for brute-force angrep. Et slikt angrep er demonstrert under.



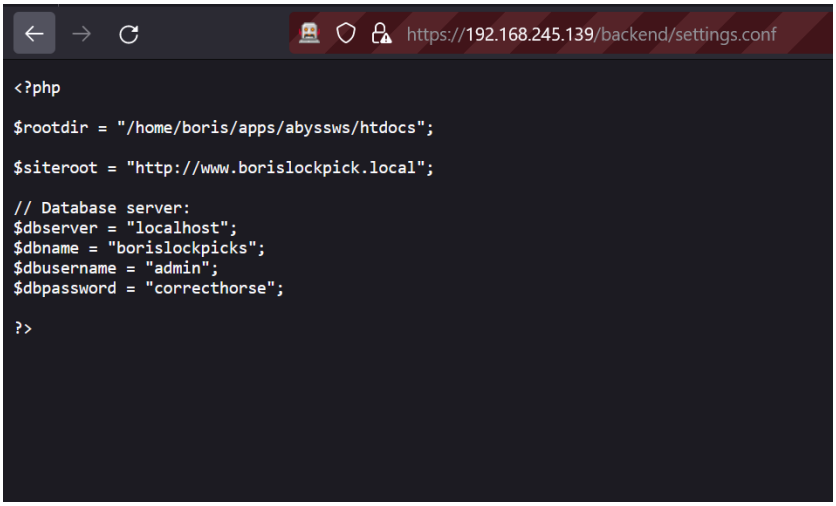
En angriper kan benytte seg av brute-force verktøyet Hydra i kali og bruke kommandoen «hydra -L users.txt -P rockyou.txt -v -t 1 <ip> -s 9999 http-get» for å gjette(brute-force) seg til passordet. Dermed vil angriperen finne ut at brukernavnet på siden er «boris» og passordet «tinkerbell», disse kredentialene kan brukes for å logge inn på konsollen. Dette er vist i utklippet under.



Fra konsollen kan en angriper blant annet stoppe applikasjonen og/eller slette regelen for anti-leeching, som i dette tilfellet er satt opp for å hindre brukere tilgang til backend filstien. Utklipp er vist under.



Ved å slette regelen vil en angriper få tilgang til backend filene deriblant /backend/settings.conf denne filen var ikke tilgjengelig for brukere i utgangspunktet. Utklipp av filen er vist under.



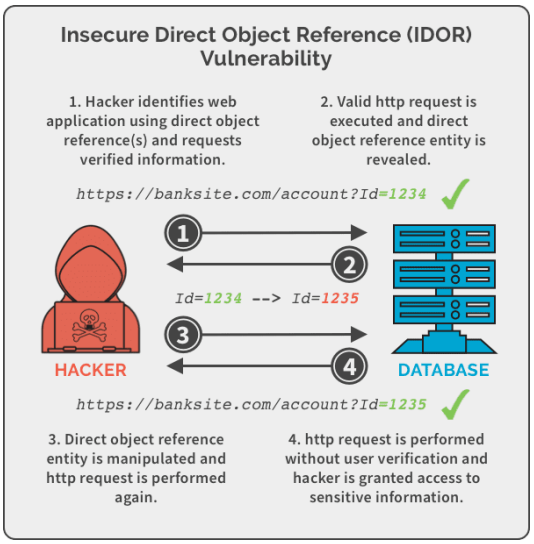
Denne filen eksponerer brukernavn og passord for database-serveren. Disse kredentialene kan en uautorisert bruker benytte for å få tilgang til databasen til applikasjonen. De andre filene i /backend eksponerer kildekoden som en uautorisert bruker kan benytte for å planlegge og utføre et angrep mot applikasjonen.

For å forhindre dette anbefales det å sette et sterkt passord som ikke kan knekkes med ordliste-angrep. Dette kan være et langt passord som består av en kombinasjon av store bokstaver, små bokstaver, tall og spesielle tegn.

Sårbarhet 3 – IDOR sårbarhet

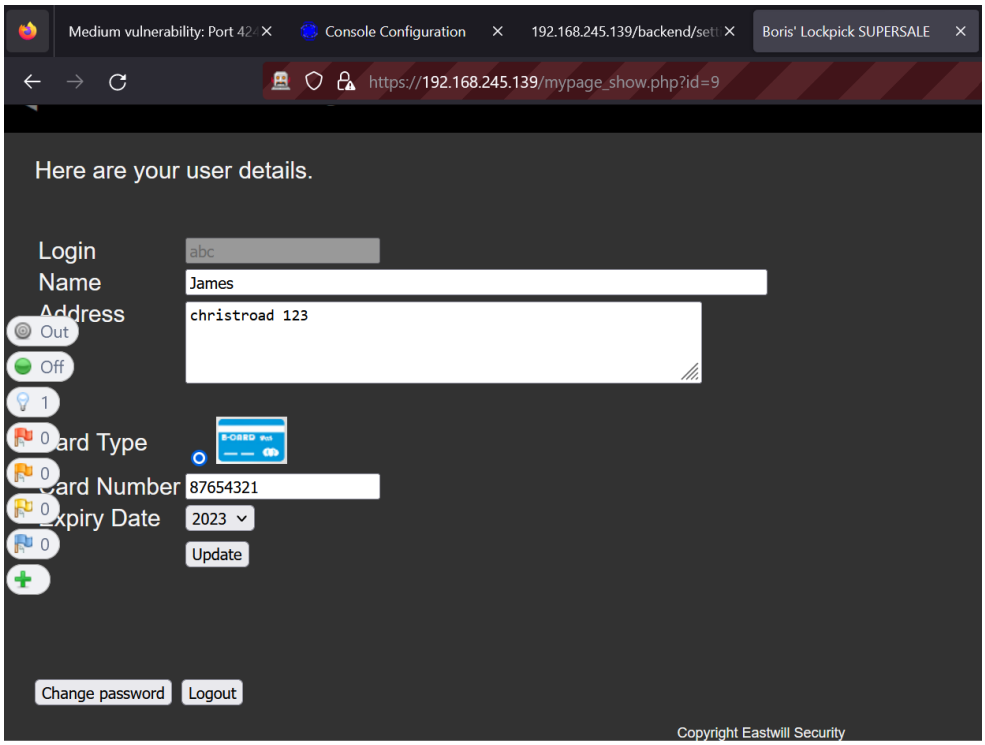
Risiko: Kritisk

Insecure Direct Object Reference er en sårbarhet som tillater brukere å endre parameterverdier i en URL. En uautorisert bruker kan utnytte denne sårbarheten til å få tilgang på informasjon de i utgangspunket ikke er tiltenkt tilgang. Se vedlegg under for detaljert beskrivelse av hvordan en ondsinnet hacker kan utnytte denne sårbarheten, og en demonstrasjon av IDOR på denne applikasjonen.

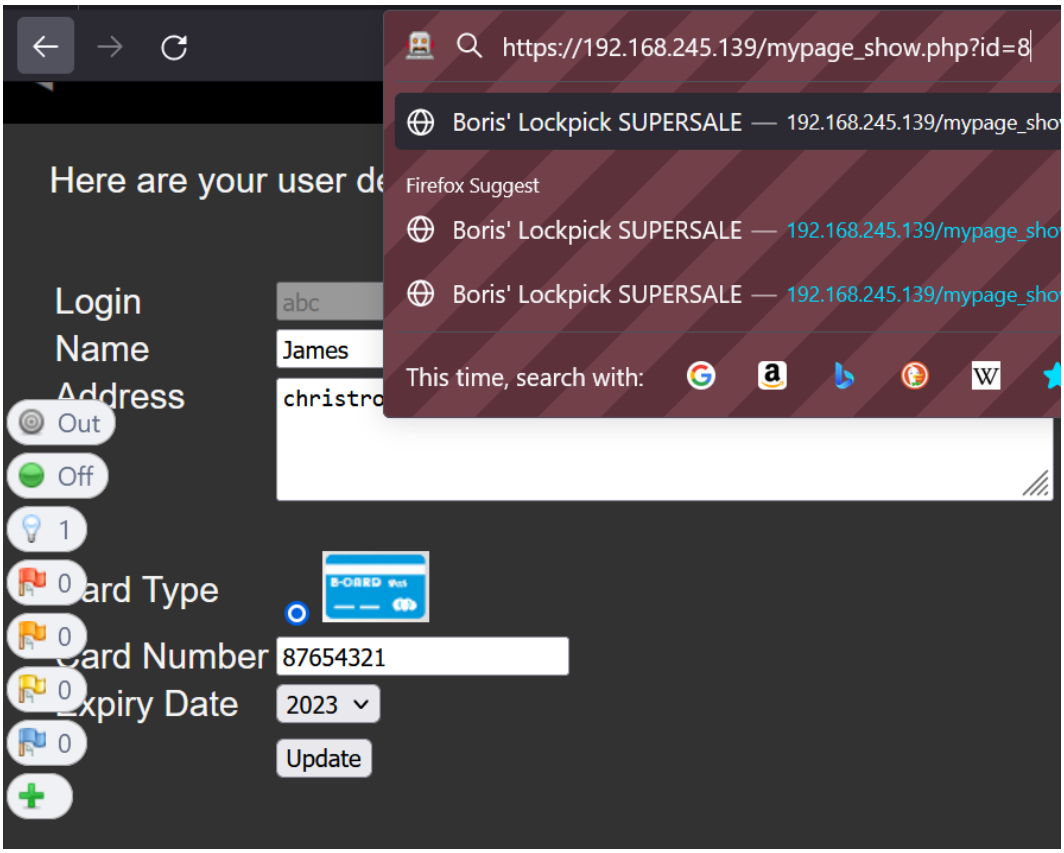


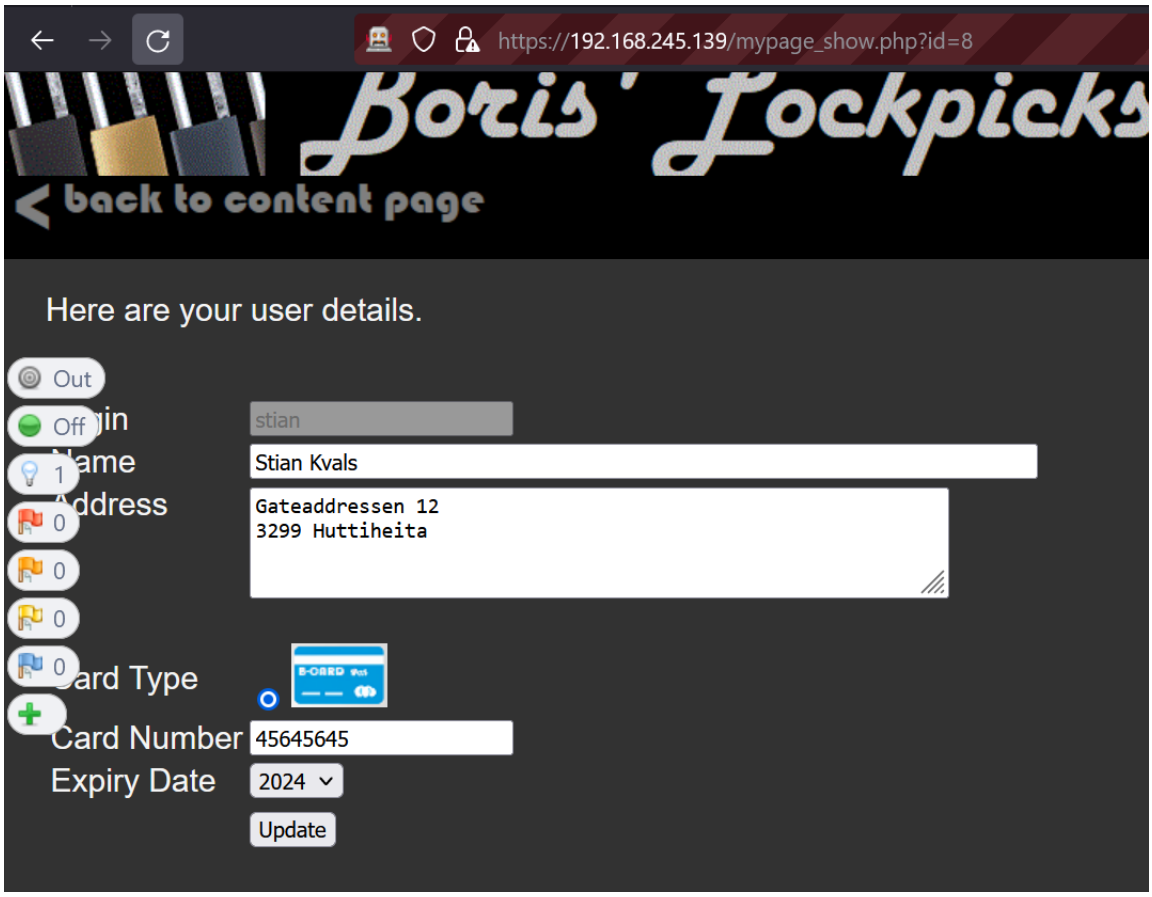
Les mer om IDOR her: https://spanning.com/blog/insecure-direct-object-reference-web-based-application-security-part-6/

Etter å ha registrert en bruker i butikken får brukeren tilgang til min side. Da kommer brukeren til url-en https://<ip>/mypage_show.php?id=9, ved å endre «id»-parameteret i slutten av url-en får brukeren tilgang til brukerinformasjonen og brukeren som tilhører den id-en. Dette er vist i utklippene under.

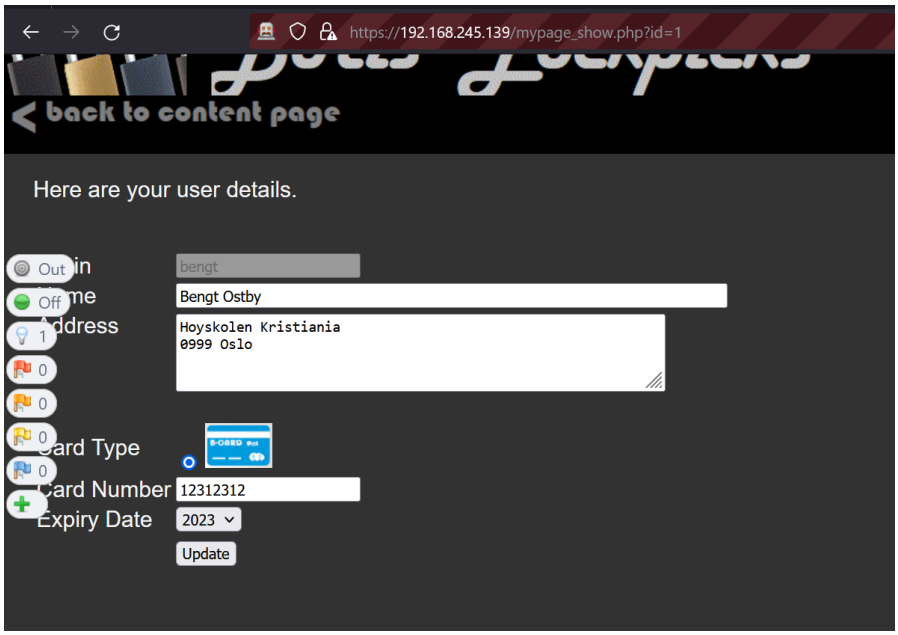


En bruker kan opprette en ny bruker på nettsiden gjennom store, der har brukeren fått tildelt id-en 9, ved å endre id-en til 8 vil en angriper få tilgang til brukeren til Stian.

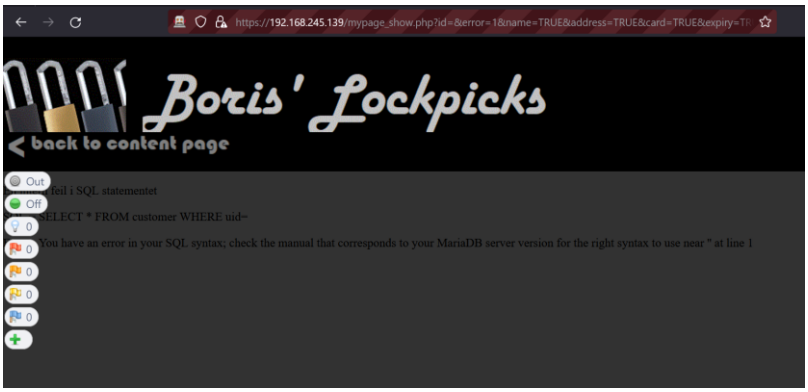

https://192.168.245.139/mypage_show.php?id=8
Boris' Lockpick SUPERSALE — 192.168.245.139/mypage_sho
Firefox Suggest
Boris' Lockpick SUPERSALE — 192.168.245.139/mypage_sho
Boris' Lockpick SUPERSALE — 192.168.245.139/mypage_sho
This time, search with:
Here are your user de
Login
abc
Name
James
Address
christro
Out
Off
1
0
ard Type
0
ard Number
87654321
0
xpiry Date
2023
0
Update



Dersom en angriper endrer id-en til 1 får angriperen også tilgang til Bengt sin bruker.



Det samme kan gjøres i pathen https://<ip>/mypage_update.php. Ved å skrive inn denne url-en blir brukeren sendt videre til filstien som ble brukt for injisering i det forrige eksempelet, men her vises også responsen fra databasen. Dette er demonstrert under.

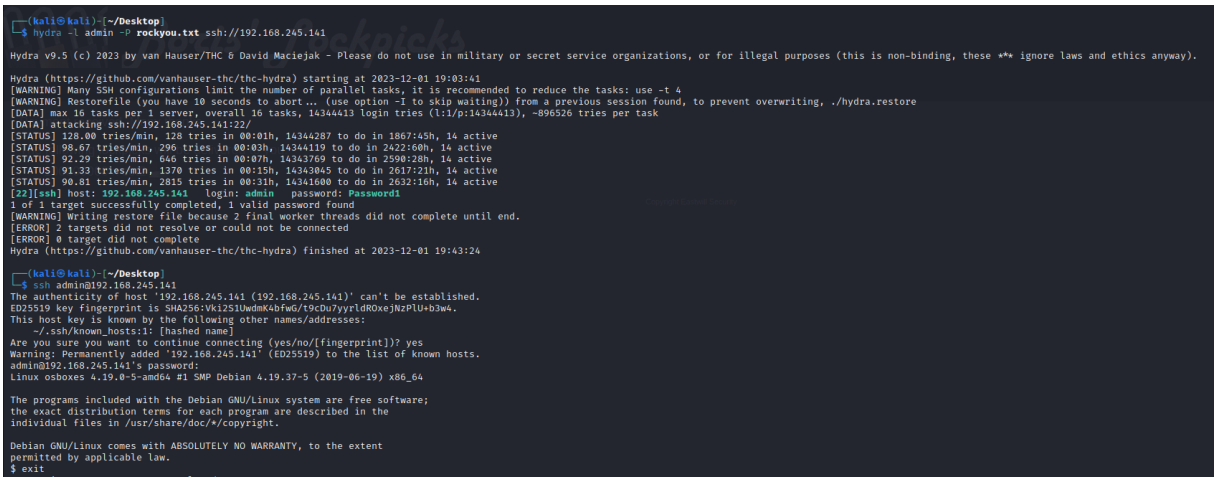


Dette er en alvorlig sårbarhet ettersom alle kundedataene blir eksponert og dette tillater en uautorisert bruker å gjøre kjøp på vegne av andre. For å fikse dette anbefales det å implementere bruker-autentisering, dette kan gjøres ved at brukeren må logge inn for å få tilgang til informasjonen på «min side». Les mer om bruker-autentisering her: https://www.cisco.com/c/en/us/products/security/identity-services-engine/what-is-user-authentication-policy.html.

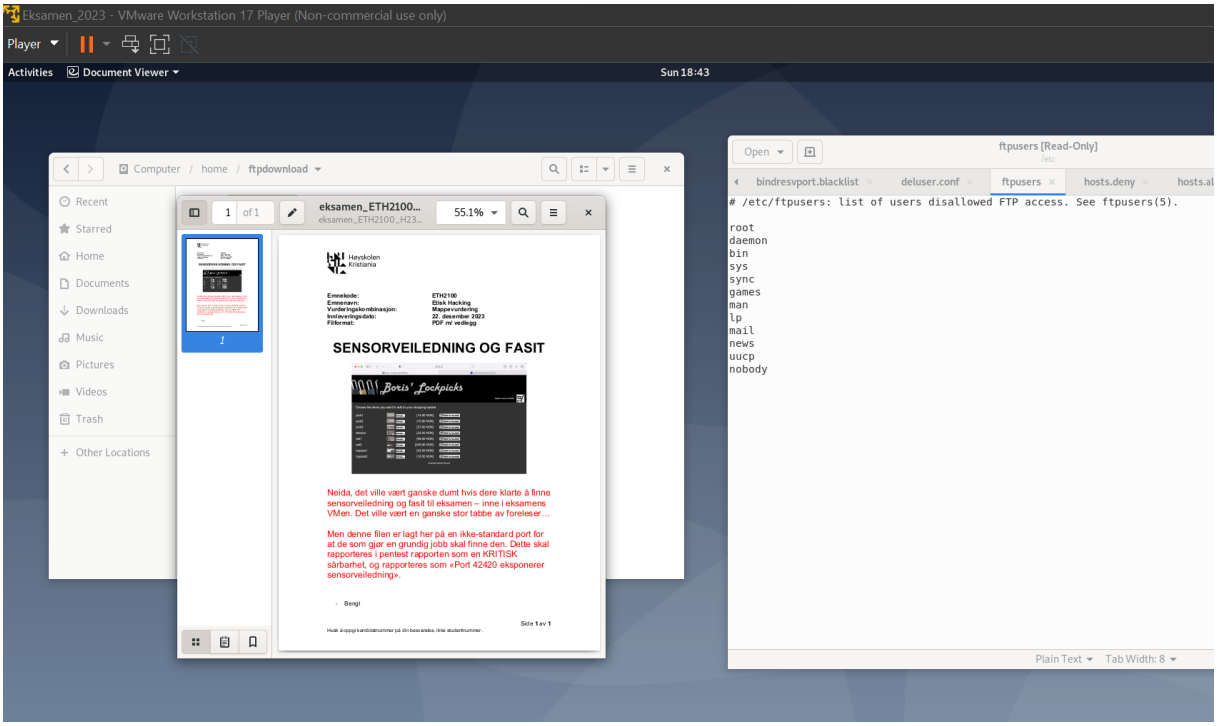
Sårbarhet 4- Servertilgang gjennom port 22

Risiko: Kritisk

Serveren webapplikasjonen kjører på er beskyttet av et svakt passord. En angriper kan i løpet av kort tid få kontroll over serveren ved å kjøre et brute-force angrep rettet mot SSH-protokollen. SSH-protokollen som kjører på port 22 er ment å være en sikker fjernpåloggings tjeneste, som tillater kryptert kommunikasjon med serveren. Gjennom denne protokollen kan en uautorisert bruker kjøre kommandoer på serveren. Derfor er det viktig at denne er beskyttet av et sterkt passord. Utnyttelse av sårbarheten er lagt ved under.



En angriper kan benytte seg av Hydra verktøyet i kali med kommandoen «hydra -l admin -P rockyou.txt ssh://<ip». Dermed vil angriperen få tak i passordet til «admin» brukeren, deretter kan disse kredentialene brukes til å logge seg inn på serveren via terminal eller fysisk, dette er vist under.



For å beskytte mot ordliste angrep anbefales det å sette et sterkt passord for denne protokollen. Les mer om sterke passord lenger nede i rapporten eller her: https://www.netsec.news/how-long-does-it-take-a-hacker-to-brute-force-a-password-in-2023/.

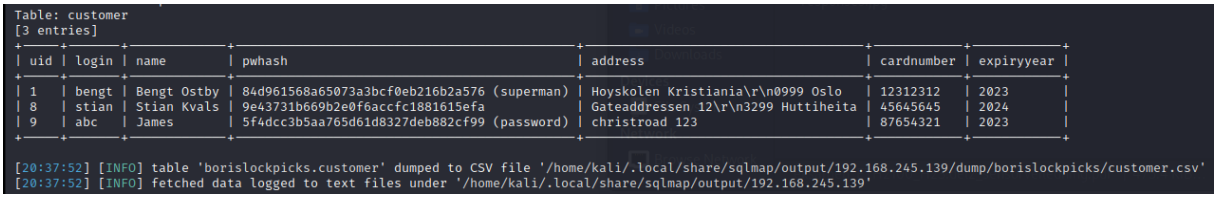
Sårbarhet 5 – Sql injection

Risiko: Høy

Sql injection er en sårbarhet som oppstår når en uautorisert bruker får mulighet til å endre på sql spørringen som sendes til databasen, dermed kan en uautorisert bruker oppnå tilgang til data, få mulighet til å endre på data i databasen og/eller slette data fra databasen.

Sql injection er en meget alvorlig sårbarhet ettersom en uautorisert bruker vil kunne få tilgang til hele databasen ved å kjøre sqlmap mot webapplikasjonen. Ettersom webapplikasjon ikke validerer brukerinformasjon trengs det ikke tilgang til en cookie for å kjøre et sqlmap angrep mot webapplikasjonen. Dette er demonstrert i utklipp under.

En angriper kan kjøre sqlmap med kommandoen ovenfor for å printe ut customer tabellen. Utklipp av tabellen er vist under.

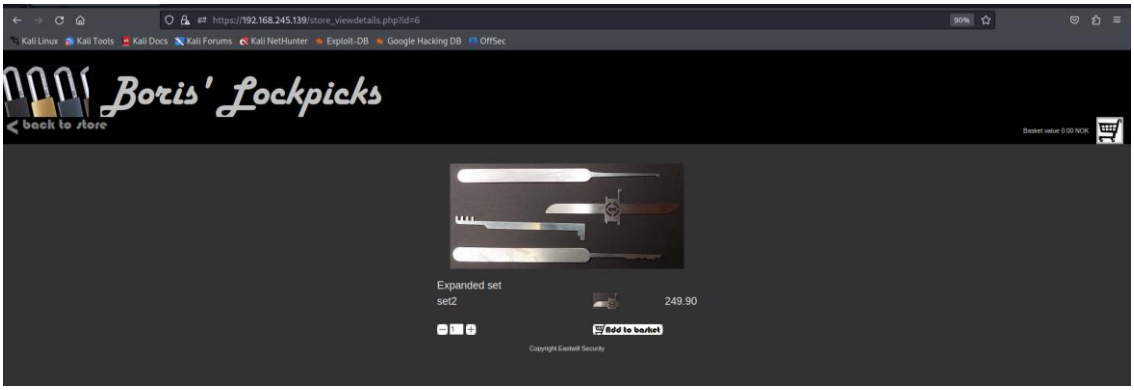


Denne tabellen inneholder oversikt over brukernavn, passord, addresser og kortinformasjon. Selv om Stian har et sterkt passord vil en angriper få tak i kortinformasjonen og adressen hans, som deretter kan misbrukes.

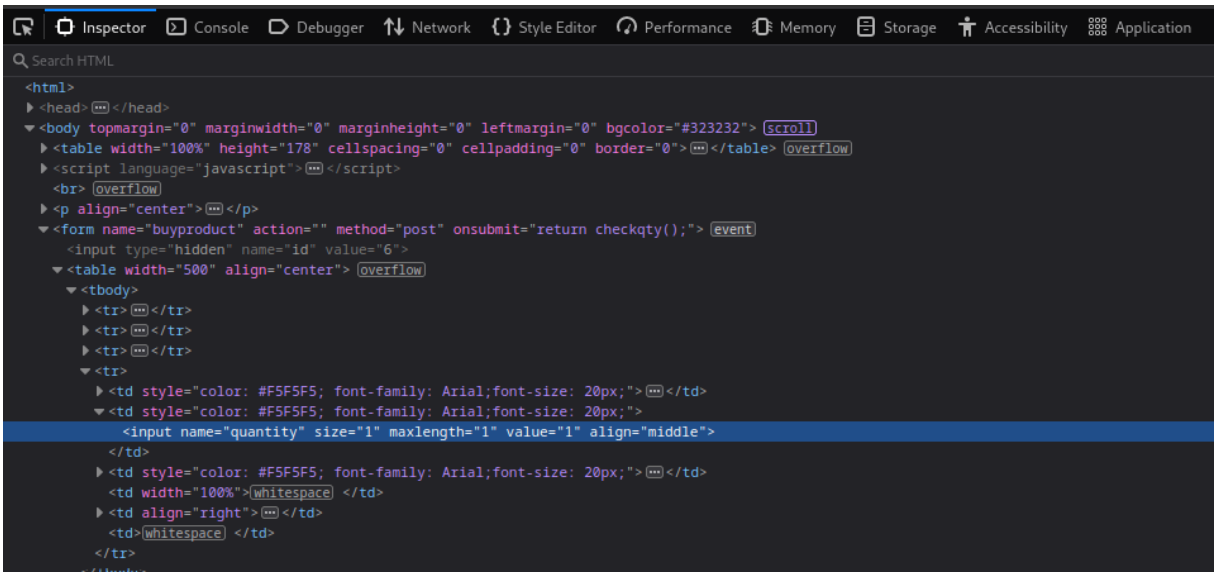
Sårbarhet 6- Manglende validering av inndata

Risiko: Høy

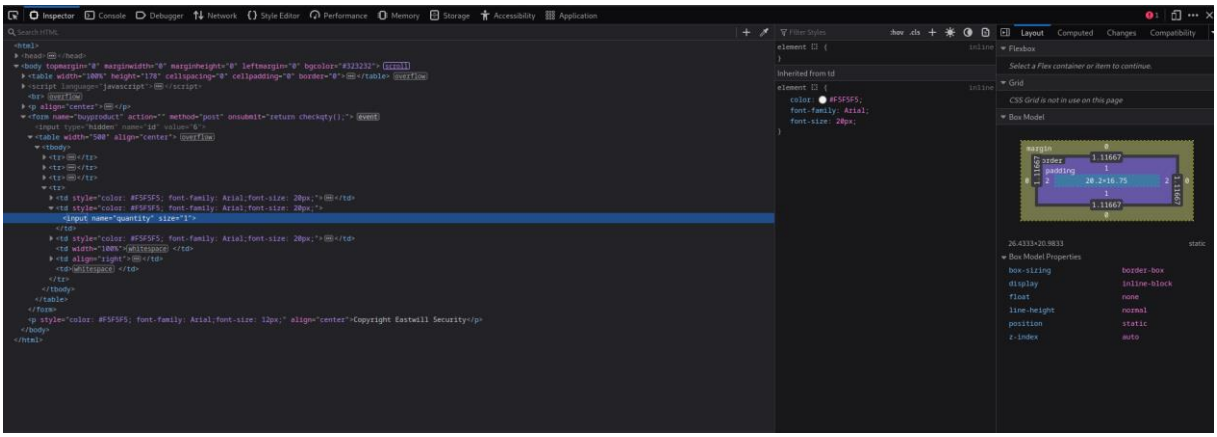
Dette er en sårbarhet som skyldes at applikasjonen ikke validerer eller sanitiserer brukerinput riktig. I dette tilfelle mengden produkter som legges til i handlekurv. Applikasjonen håndterer dette på klient siden av applikasjonen, når dette burde bli håndtert på backend delen av applikasjonen for å forhindre manipulasjon. Utnyttelse av denne sårbarheten vises under.



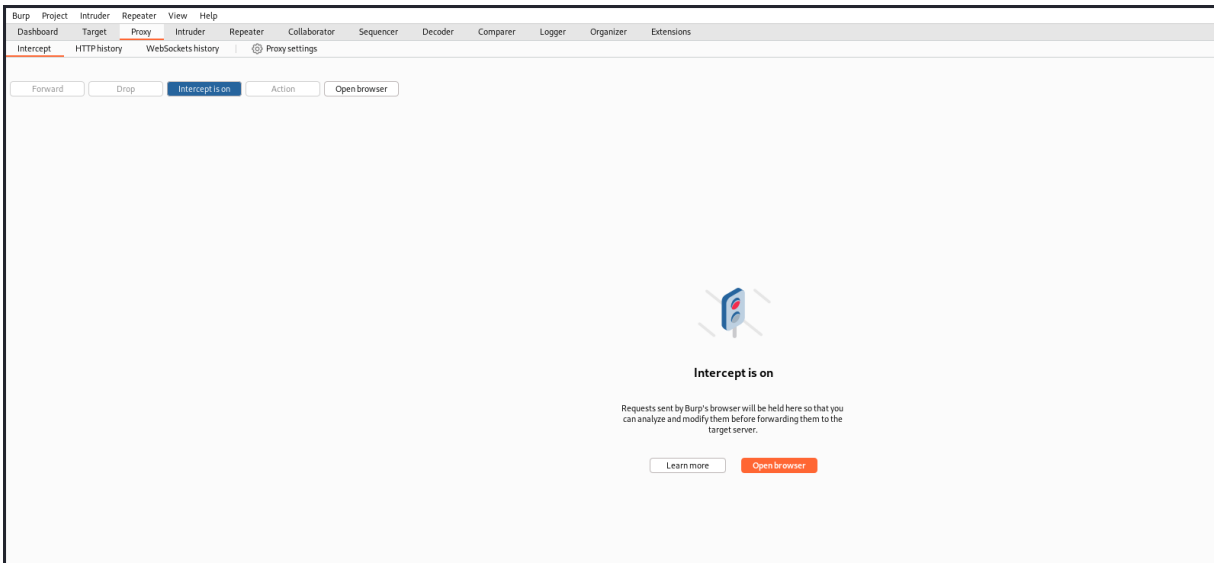
Brukeren kan trykke på detaljer på et produkt for å lese produktbeskrivelse og legge dette i handlekurven,



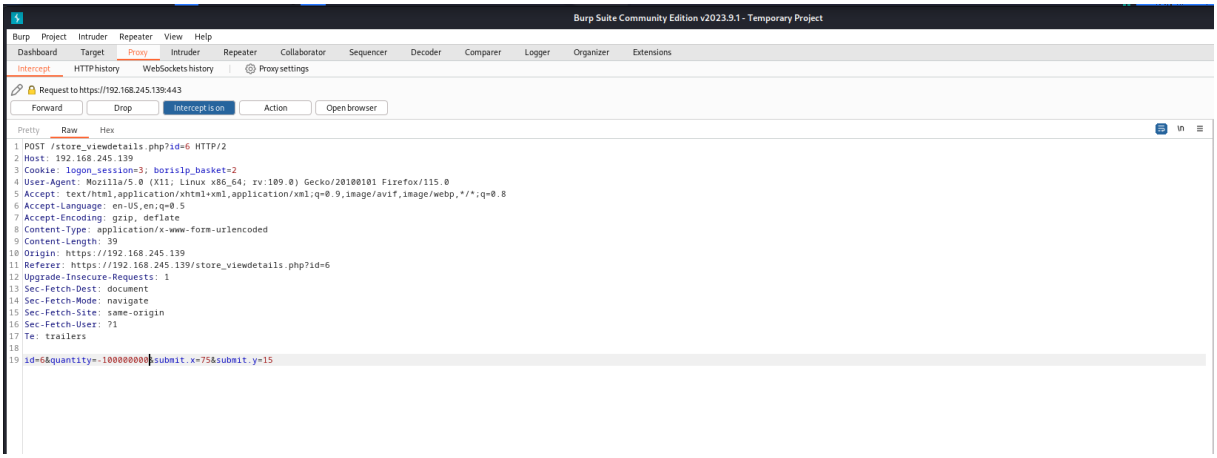
Ved å høyreklikke på siden og inspisere siden kan brukeren lete etter parametere som kan manipuleres.



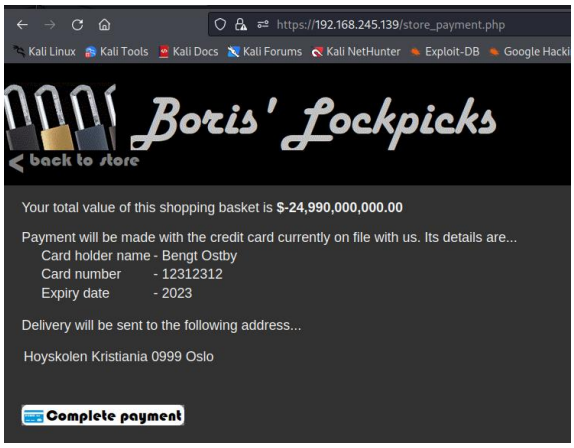
En angriper vil utnytte dette og fjerne all form for validasjon i dette parameteret.

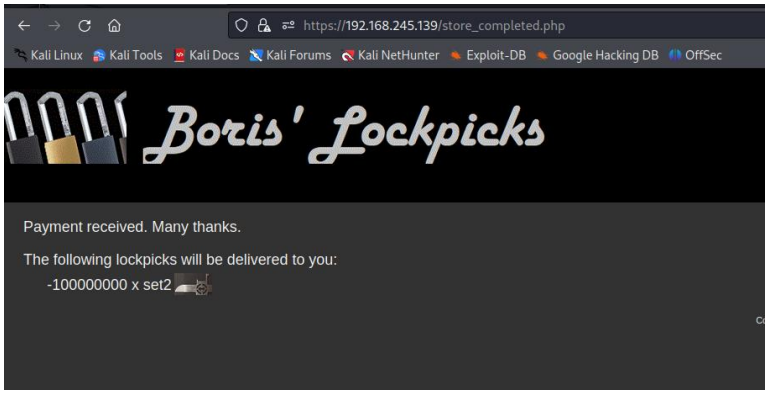


Intercept verktøyet i Burp Suite kan brukes for å fange opp forespørselen når produktet legges i handlekurven.



Her kan angriperen modifisere på quantity verdien i forespørselen til applikasjonen og sende denne istedenfor den orginale. Her kan en angriper legge inn et negativt antall for å få utbetalt penger istedenfor å betale.





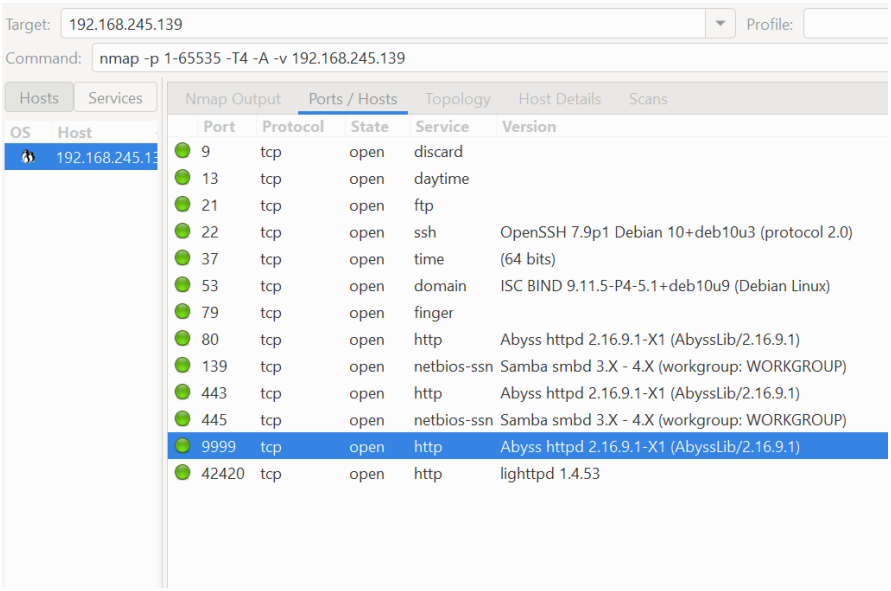
Her har ordren gått gjennom og angriperen vil få utbetalt pengene til kontoen sin.

Sårbarhet 7 – Flere åpner porter

Risiko: Høy

Nettverksporter er digitale kanaler som blir brukt av protokoller for kommunikasjon mellom enheter. Portene kan ha et nummer mellom 1 og 65535, og flere av disse har en standard tjeneste tilknyttet til dem. http kjører på port 80, HTTPS kjører på port 443, FTP kjører på port 21, SSH kjører på port 22 osv. Åpne porter utgjør en mulig sikkerhetsrisiko, dersom de er knyttet til utdaterte eller usikrede/dårlig sikrede tjenester. Dermed kan en mulig angriper prøve å utnytte disse for å få tilgang på data\informasjon som de ikke skal ha tilgang til. En

full port-scan med kommandoen «nmap -p 1-65535 -T4 -A -v <ip>» viser at applikasjonen har 13 åpne porter. Utklipp vises under.



Flere av disse portene er utdaterte og sårbare for hackerangrep. Daytime på port 13 og finger på port 79 er begge utdaterte protokoller. Port 139 og Port 445 kjører begge SMB-protokollen, denne protokollen er ekstremt sårbar og ble blant annet utnyttet av WannaCry(ransomware). For å beskytte mot hackerangrep anbefales det at unødvendige og/eller ubrukte porter stenges for å minske risikoen til og skaden påført av et potensielt angrep. Les mer om sårbare porter her: https://www.all-about-security.de/identifying-secure-and-unsecured-ports-and-how-to-secure-them/.

Sårbarhet 8- Webapplikasjonen er sårbar for XSS

Risiko: Høy

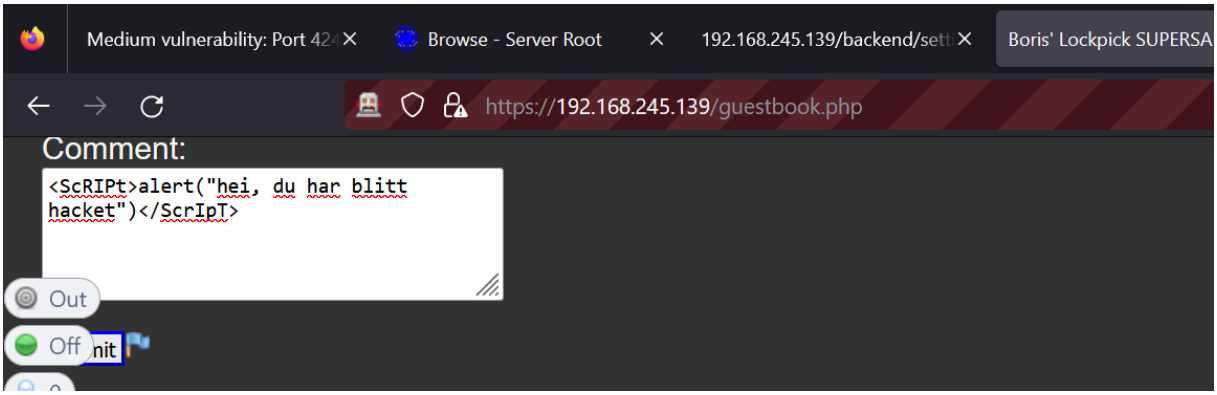
Cross-site Scripting også kalt XSS kan brukes for å gjennomføre en rekke ondsinnede handlinger, som å stjele cookies, sensitive data fra brukere og\eller videresende brukere til

andre nettsteder. Det finnes 3 ulike typer XSS, Reflective XSS, Persistent XSS og DOM-based XSS. Reflective XSS er script som kun blir reflektert 1 gang når scriptet blir kjørt. Persistent XSS er script som lagres på siden og kjører hver gang noen åpner siden. DOM-based XSS er script som legges inn i url-en og kjører når søket utføres. Under vises utklipp fra Owasp Zap som har detektert XSS av typen Reflective og Persistent på denne siden.

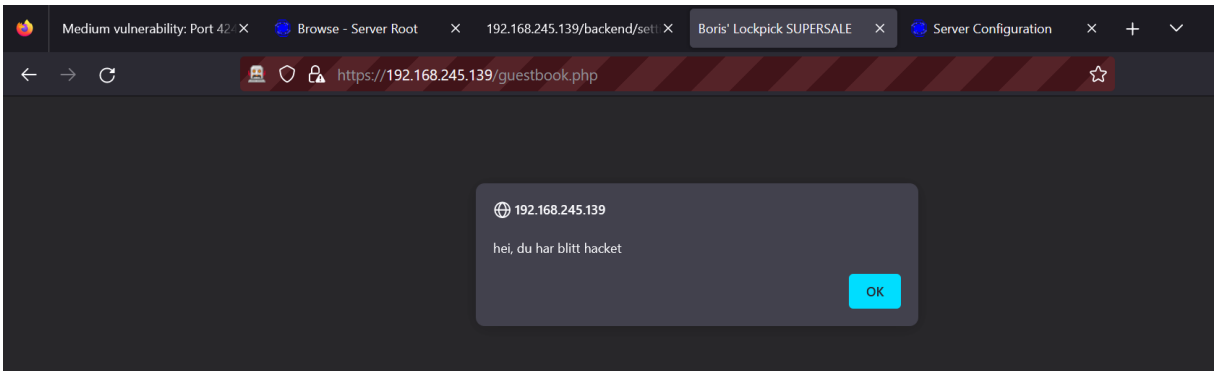
Cross Site Scripting (Reflected)
URL: https://192.168.245.137/guestbook.php
Risk: High
Confidence: Medium
Parameter: name
Attack: <script>alert(1);</scRipt>
Evidence: <script>alert(1);</scRipt>
CWE ID: 79
WASC ID: 8
Source: Active (40012 - Cross Site Scripting (Reflected))
Input Vector: Form Query
Description:
Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself

Cross Site Scripting (Persistent)
URL: https://192.168.245.137/guestbook.php
Risk: High
Confidence: Medium
Parameter: name
Attack: <img src=x onerror=alert(1);>
Evidence: <img src=x onerror=alert(1);>
CWE ID: 79
WASC ID: 8
Source: Active (40014 - Cross Site Scripting (Persistent))
Input Vector: Form Query
Description:
Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself

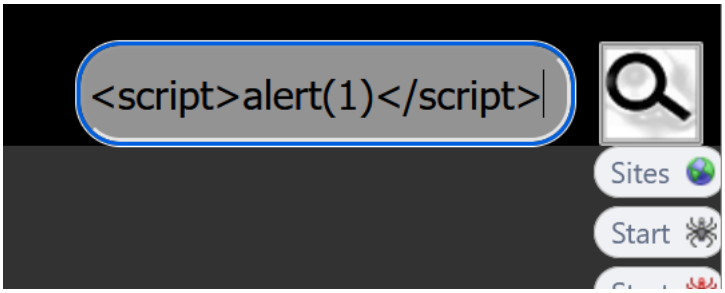
Under er en demonstrasjon av XSS på denne applikasjonen.

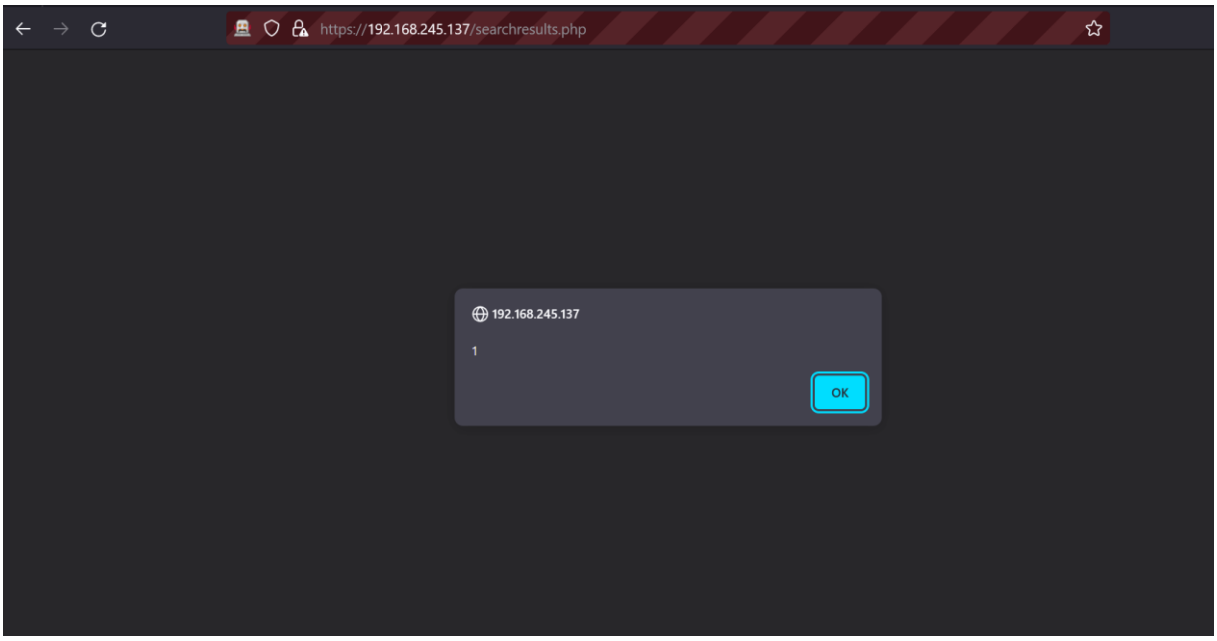


En angriper kan skrive et script i gjesteboken, og scriptet vil kjøre hver gang noen åpner gjesteboken.



Angriperen kan bruke denne metoden til å legge igjen et script i kommentarfeltet som f.eks stjeler cookiene dine hver gang noen laster inn gjestebok siden. Dette er XSS av typen Persistent.





Søkefeltet er også sårbart for Cross-site Scripting, av typen Reflective.

For å rette opp i denne sårbarheten anbefales det å implementere omfattende sanitering av input. Les mer om XSS her: https://www.acunetix.com/websitesecurity/cross-site-scripting/.

Sårbarhet 9- Svake passord

Risiko: Høy

Webapplikasjonen stiller ingen krav til brukeren om å sette et sterkt passord, dette utsetter brukere med svake passord for brute force angrep. Passordene blir også hashet med md5, disse ble knukket av sqlmap i løpet av noen få sekunder. Passordet til webserveren var også svakt og ble brute forcet med en standard passordliste i kali i løpet av minutter. Med dagens ressurser kan en angriper bruke så lite som 5 minutter for å knekke et passord på 8 tegn bestående av tall, store og små bokstaver og spesialtegn. Passordene på webapplikasjonen burde lagres i databasen ved å bruke en sikker hashfunksjon som Bcrypt. Både passordet til boris på webserveren på port 9999 og admin passordet på port 9999 er svake passord som er sårbare for brute-force angrep. Se tabell under for krav som kan stilles brukere for å sette sterke passord.

Les mer her: https://www.netsec.news/how-long-does-it-take-a-hacker-to-brute-force-a-password-in-2023/

Sårbarhet 10 – Path Traversal

Risiko: Høy

Path Traversal er en alvorlig sårbarhet, som tillater angripere å få tilgang til filer og mapper som ligger på serveren. Under vises utklipp fra Owasp Zap som har detektert Path Traversal i

søkefeltet på denne siden.

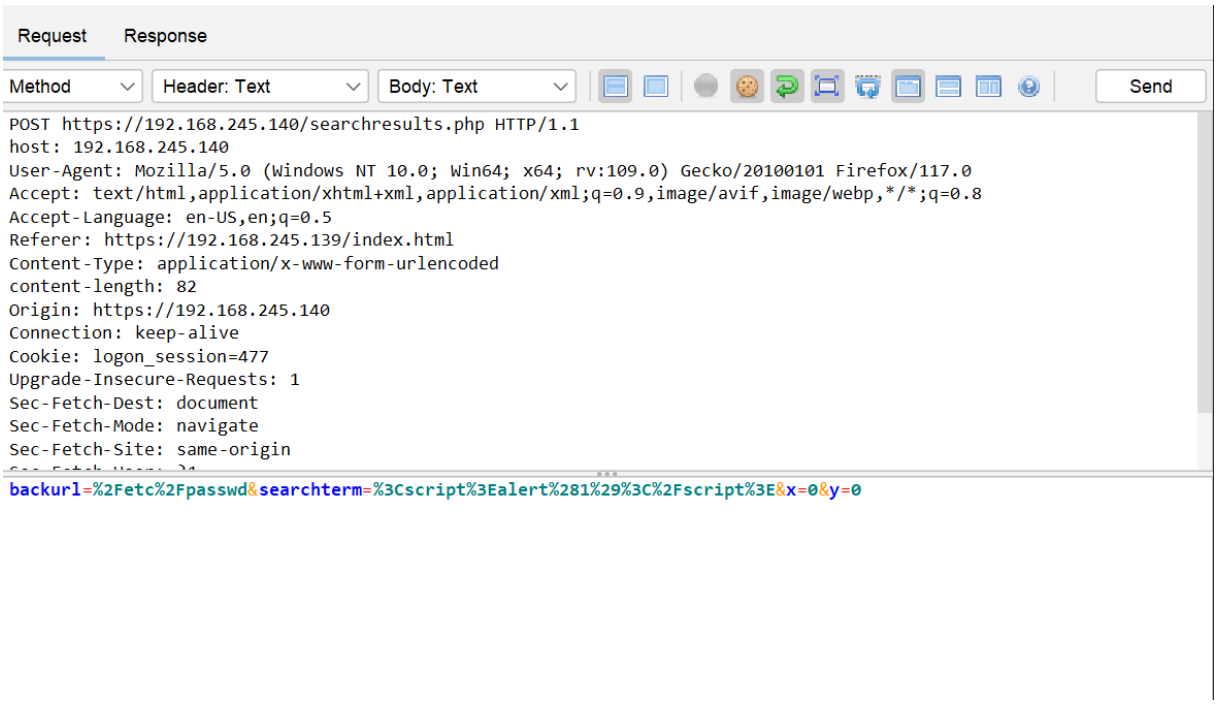
Path Traversal

URL:	https://192.168.245.137/searchresults.php
Risk:	High
Confidence:	Medium
Parameter:	backurl
Attack:	/etc/passwd
Evidence:	root:x:0:0
CWE ID:	22
WASC ID:	33
Source:	Active (6 - Path Traversal)
Alert Reference:	6-2
Input Vector:	Form Query

Description:

The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. Any

Under er en demonstrasjon av hvordan sårbarheten kan utnyttes.



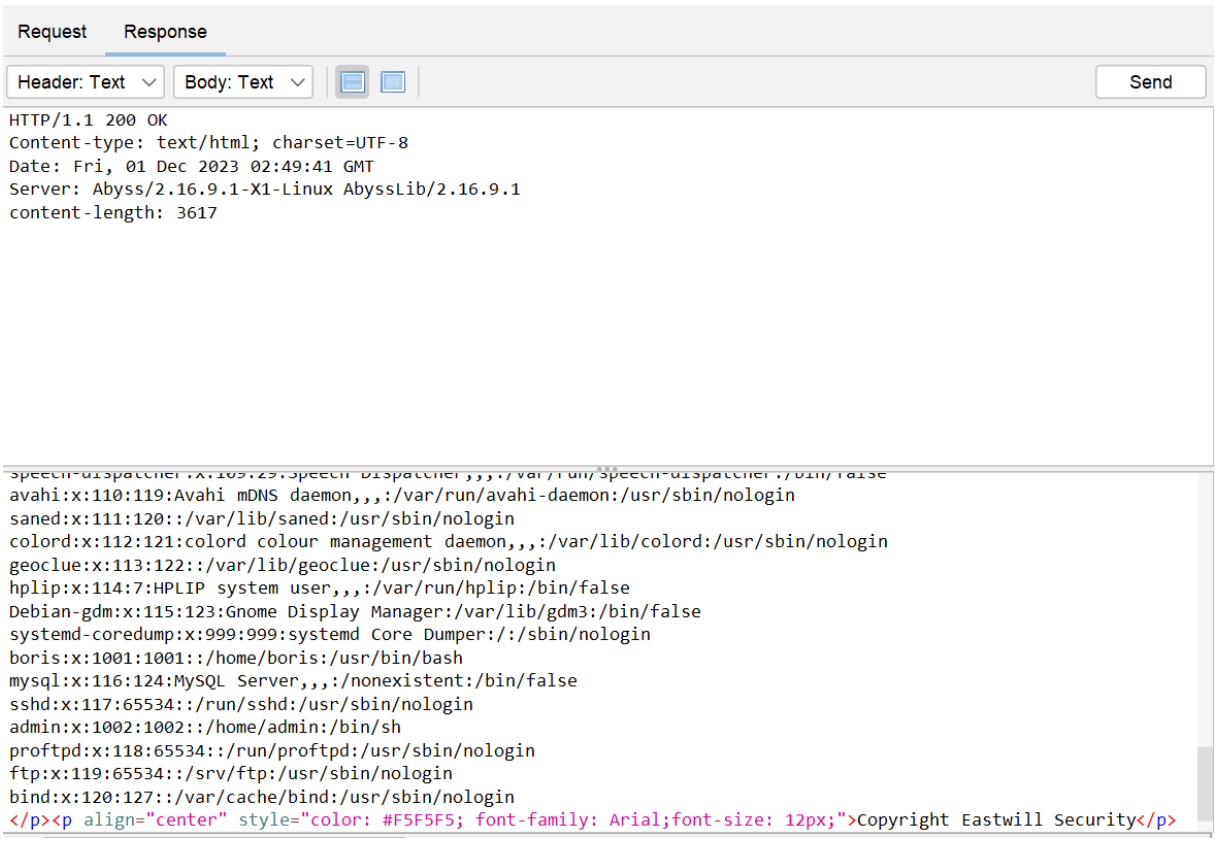
Request Response

Method | Header: Text | Body: Text | Send

```
POST https://192.168.245.140/searchresults.php HTTP/1.1
host: 192.168.245.140
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://192.168.245.139/index.html
Content-Type: application/x-www-form-urlencoded
content-length: 82
Origin: https://192.168.245.140
Connection: keep-alive
Cookie: logon_session=477
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
```

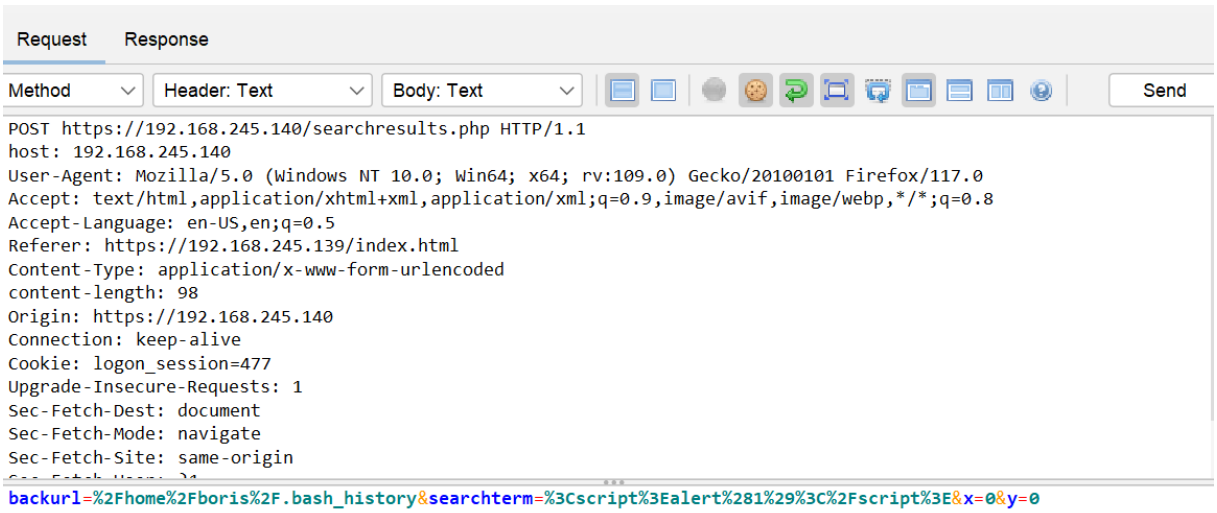
```
backurl=%2Fetc%2Fpasswd&searchterm=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&x=0&y=0
```

Denne forespørselen i zap henter ut passwd filen som ligger i filstien etc/passwd.

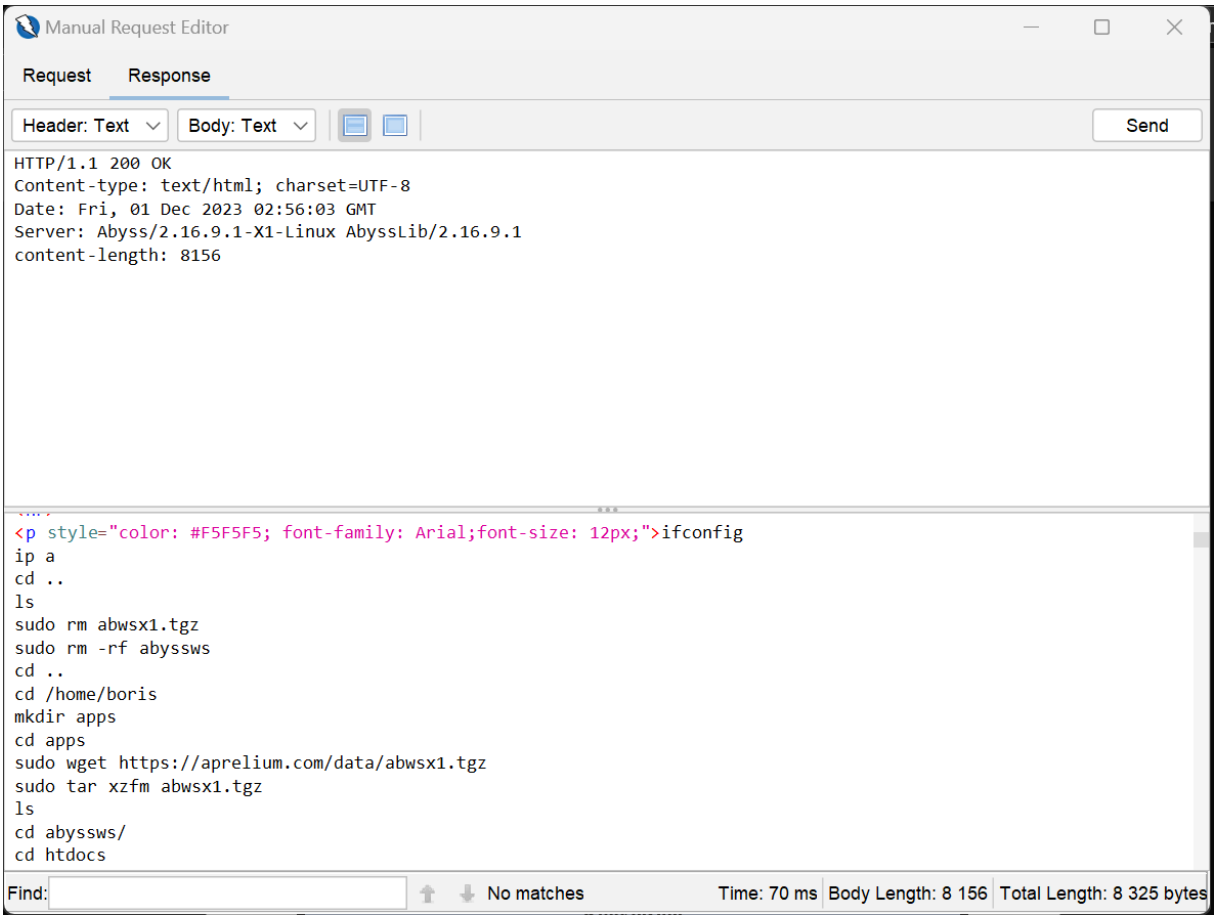


Denne filen inneholder alle brukernavnene.

Ved å endre på parameterene i backurl kan en angriper se ønskede filer.



Ved å endre filstien til home\boris\.bash_history kan vi se alle kommandoene kjørt i terminalen fra denne brukeren. Dette er vist under.



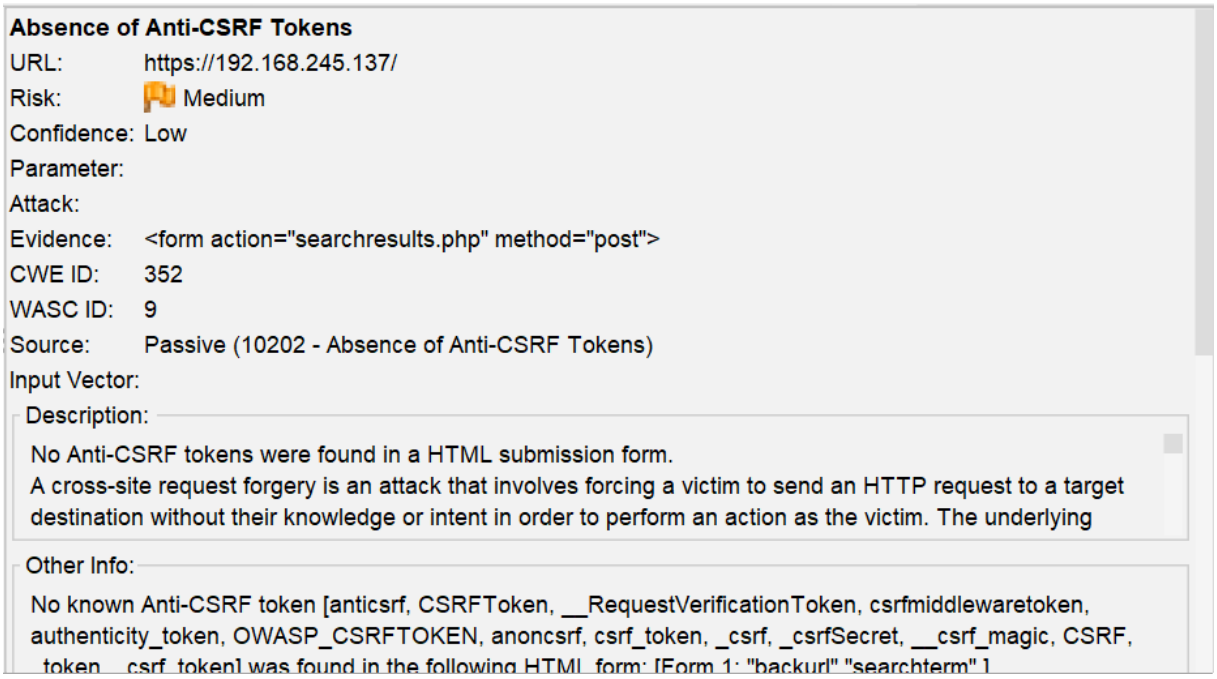
En angriper vil utnytte denne sårbarheten for å lese filer på serveren som ikke skal være tilgjengelig for dem. For å forhindre Path Traversal angrep anbefales det å implementere input sanitering, dette vil hindre en angriper å få tilgang til filer som ligger utenfor filstien tiltenkt.

Sårbarhet 11 – Anti CSRF tokens mangler

Risiko: Middels

Et CSRF (Cross-Site Request Forgery)-angrep utnytter en sårbarhet i webapplikasjonen dersom den ikke kan skille mellom en forespørsel gjennomført av en legitim bruker og en forespørsel gjennomført av en uautorisert bruker. Dette betyr at en angriper kan lure en legitim bruker til å utføre uautoriserte handlinger i webapplikasjonen, ettersom applikasjonen

ikke kan validere om forespørselen er utført av brukeren eller ikke. Les mer her: https://www.synopsys.com/glossary/what-is-csrf.html . Under er et utklipp fra Owasp Zap som viser at dette er en sårbarhet på nettsiden.



Sårbarhet 12- Feilmeldinger fra databasen vises

Risiko: Middels

Dette er en sårbarhet som viser responsen fra databasen i form av feilmeldingen «En intern feil i SQL statementet». Denne feilmeldingen indikerer at webapplikasjonen er sårbar for SQL injection. Dersom slike feilmeldinger ikke skjules fra brukere, vil det føre til at en angriper kan planlegge og utføre SQL injection angrep mot denne applikasjonen. Deteksjon i Owasp ZAP, og utklipp fra nettsiden vises under.

Application Error Disclosure

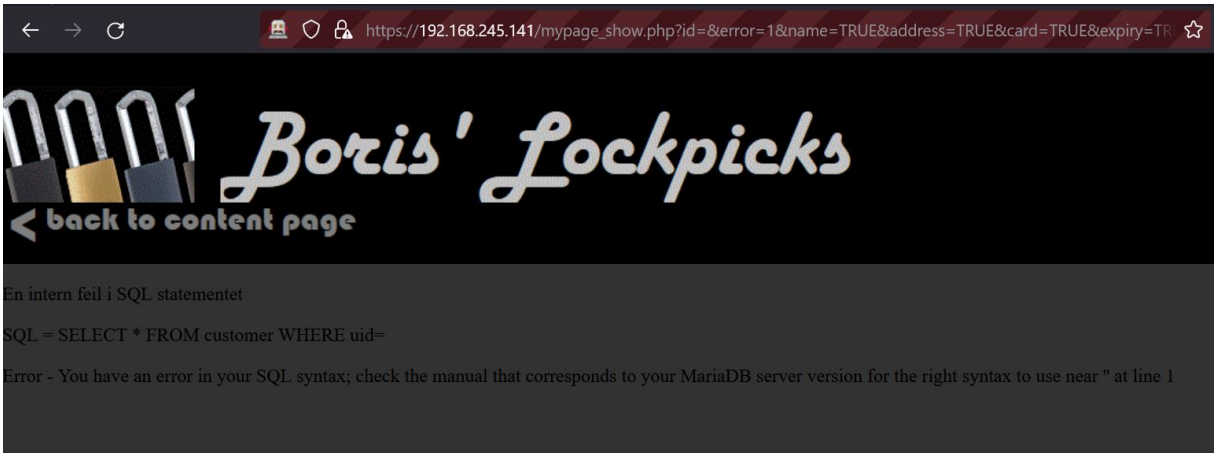
URL: https://192.168.245.137/store_addtobasket.php
Risk: Medium
Confidence: Medium
Parameter:
Attack:
Evidence: You have an error in your SQL syntax
CWE ID: 200
WASC ID: 13
Source: Passive (90022 - Application Error Disclosure)
Input Vector:

Description:

This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation

Other Info:

Ved å gå inn på https://<ip>/mypage_update.php vil følgende side dukke opp.

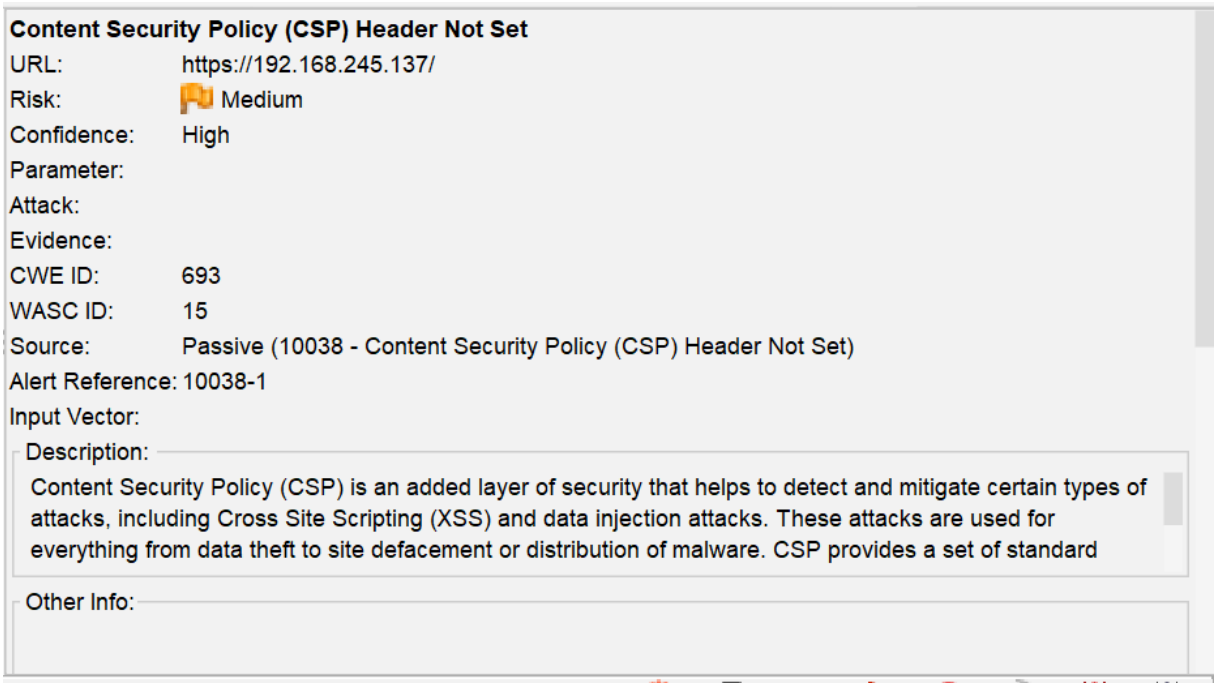


Denne siden viser en feilmelding fra databasen. En annen side i applikasjonen som også inneholder feilmeldingen er https://192.168.245.141/store_addtobasket.php.

Sårbarhet 13- CSP header ikke satt

Risiko: Middels

Content-Security-Policy er en innholdssikkerhetspolicy som hindrer XSS, clickjacking og andre angrep som begås ved å injisere en kode på nettsiden. En CSP header lar deg begrense hvilke ressurser som kan kjøres på nettsiden, eksempelsvis fra hvilke nettaddresser javascript kan kjøres. I tillegg kan man begrense eller deaktivere funksjoner som blir kjørt av angripere. les mer om CSP her: https://content-security-policy.com. Det anbefales at det settes opp CSP header i applikasjonen. Under er et utklipp fra Owasp Zap som påviser at CSP header ikke er satt.



Sårbarhet 14 – Skjult fil funnet

Risiko: Middels

Hidden File Found

URL: https://192.168.245.137/phpinfo.php
Risk: Medium
Confidence: High
Parameter:
Attack:
Evidence: HTTP/1.1 200 OK
CWE ID: 538
WASC ID: 13
Source: Active (40035 - Hidden File Finder)
Input Vector:

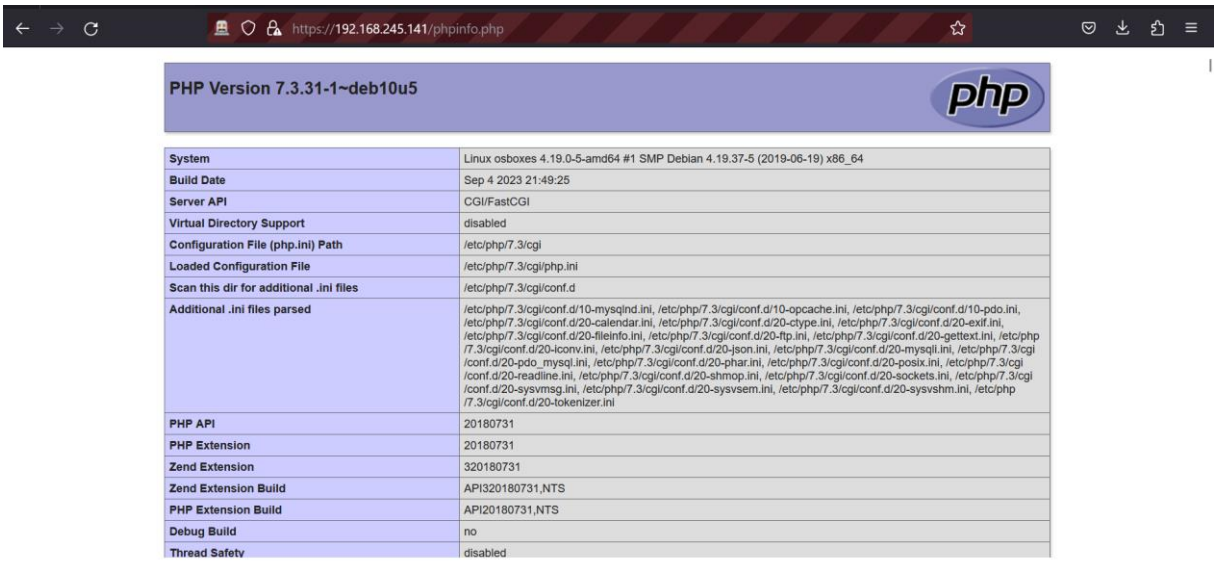
Description:

A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.

Other Info:

phpinfo

Over er det vist et utklipp fra Owasp Zap som detekterer phpinfo.php som en sensitiv fil. Ved å gå inn på «https://<ip>/phpinfo.php» får man opp følgende fil.



Denne filen inneholder mye informasjon om webapplikasjonen, slik som operativ system, PHP versjon, server informasjon mm. Denne filen bør ikke være tilgjengelig gjennom webapplikasjonen. Dette er en sårbarhet som kan brukes av en angriper i planleggingsfasen,

for å finne informasjon om sårbarheter som kan misbrukes, slik som f.eks utdaterte servere. Les mer om phpinfo filen her: https://www.php.net/manual/en/function.phpinfo.php.

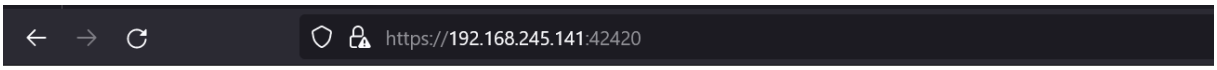
Sårbarhet 15 – Port 42420 server ukrypterte data over HTTP

Risiko: Middels

Et nmap scan i zenmap med følgende kommando «nmap -p 1-65535 -T4 -A -v <ip>» viser at port 42420 utgjør en middels sårbarhet. Viser til utklipp under.

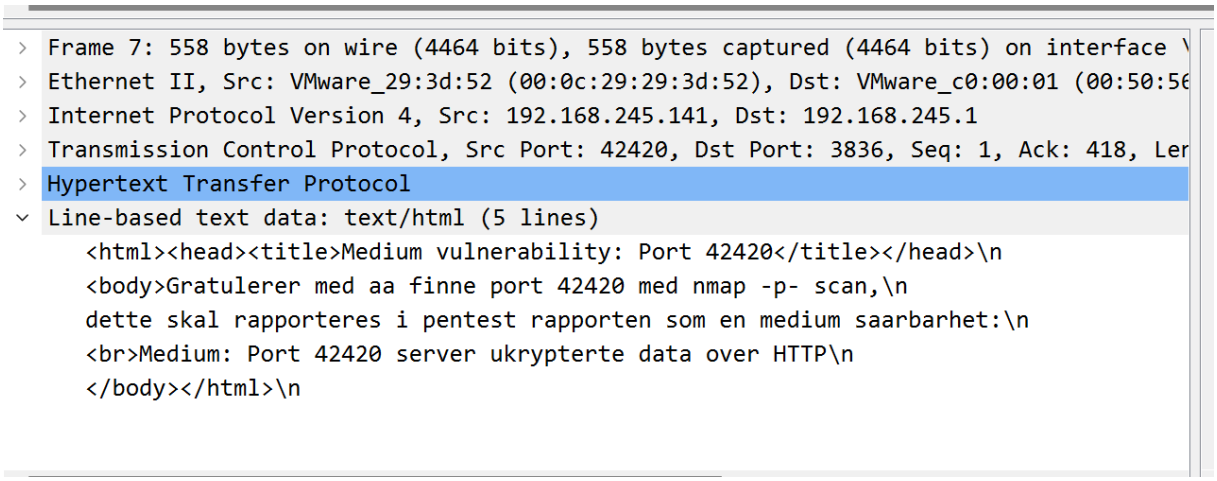
```
42420/tcp open  http        lighttpd 1.4.53
|_http-server-header: lighttpd/1.4.53
| http-methods:
|_  Supported Methods: OPTIONS GET HEAD POST
|_http-title: Medium vulnerability: Port 42420
|_ms-sql-info: ERROR: Script execution failed (use -d to debug)
|_ms-sql-ntlm-info: ERROR: Script execution failed (use -d to debug)
MAC Address: 00:0C:29:29:3D:52 (VMware)
```

Ved å søke etter «http://<ip>:42420» får vi opp følgende side.



Gratulerer med aa finne port 42420 med nmap -p- scan, dette skal rapporteres i pentest rapporten som en medium saarbarhet:
Medium: Port 42420 server ukrypterte data over HTTP

Port 42420 bruker http protokollen for å overføre data, dette er en sårbarhet som tillater hvem som helst med tilgang til nettverket å lese mulig sårbar informasjon sendt over port 42420. Dette vises i utklippet under.



Her vil en angriper kunne lese all data sendt gjennom port 42420. Det anbefales å bruke HTTPS protokollen over port 443 for å overføre data på en sikker måte, istedenfor å sende data over http protokollen over port 80 slik som i dette tilfellet. Data sendt gjennom HTTPS blir kryptert, dette vil sikre webapplikasjonen mot man in the middle angrep. Les mer om hvorfor HTTPS er sikrere enn http her: https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/.

Sårbarhet 16 – Anti-clickjacking header mangler

Risiko: Middels

Clickjacking er et angrep som lurer brukeren til å trykke på et webelement som er skjult eller fordekt som et annet element. Dette kan føre til at brukere uvitende laster ned malware, besøker ondsinnede sider, gir fra seg sensitiv informasjon eller overfører penger. Les mer her: https://www.imperva.com/learn/application-security/clickjacking/.

Missing Anti-clickjacking Header

URL:	https://192.168.245.141/
Risk:	Medium
Confidence:	Medium
Parameter:	x-frame-options
Attack:	
Evidence:	
CWE ID:	1021
WASC ID:	15
Source:	Passive (10020 - Anti-clickjacking Header)
Alert Reference:	10020-1
Input Vector:	

Description:

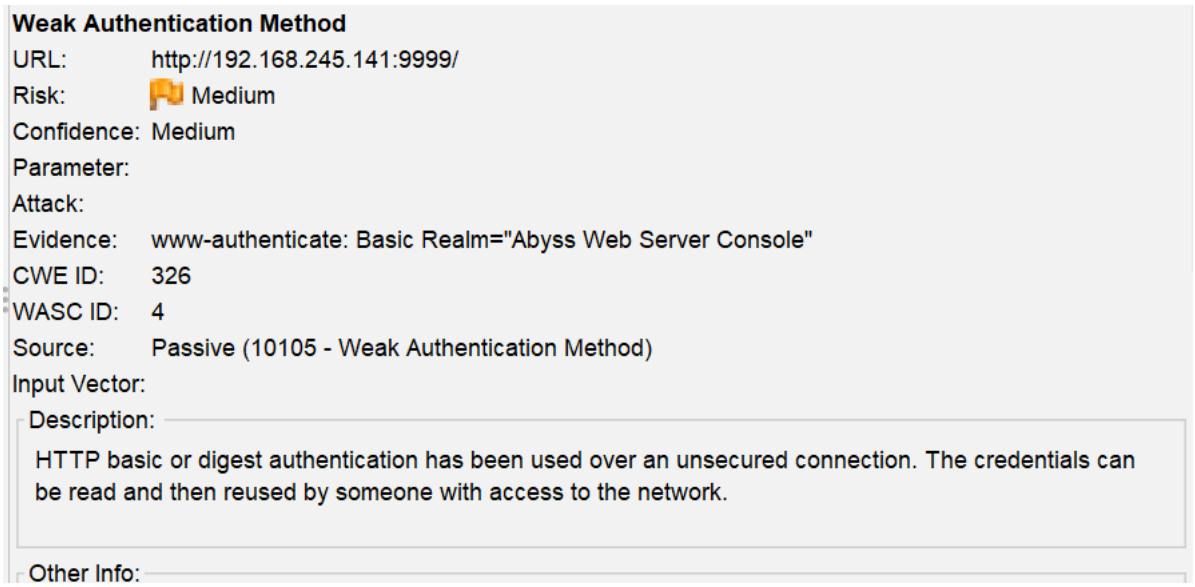
The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

Over vises et utklipp fra Owasp Zap, som viser at webapplikasjonen ikke har CSP eller XFO fot satt for å beskytte mot clickjacking angrep. Dermed er denne webapplikasjonen sårbar for slike angrep.

Sårbarhet 17 - Port 9999 usikker autentiseringsmetode

Risiko: Middels

Autentisering bør ikke skje over http ettersom det gjør webapplikasjonen sårbar for man in the middle angrep, og tillater en angriper med tilgang til nettverket å lese kredentialene til brukere. Under vises et utklipp fra Owasp Zap, som viser at webapplikasjonen benytter seg av en svak autentiseringsmetode.



En angriper kan benytte seg av verktøyet Wireshark for å analysere trafikken sendt til og fra «http://<IP>:9999». Dermed vil en angriper fange opp kredentialene til autoriserte brukere, og kan logge seg inn med privilegiene til disse. Utklippet under fra Wireshark viser autentisering som eksponerer kredentialene i klartekst.

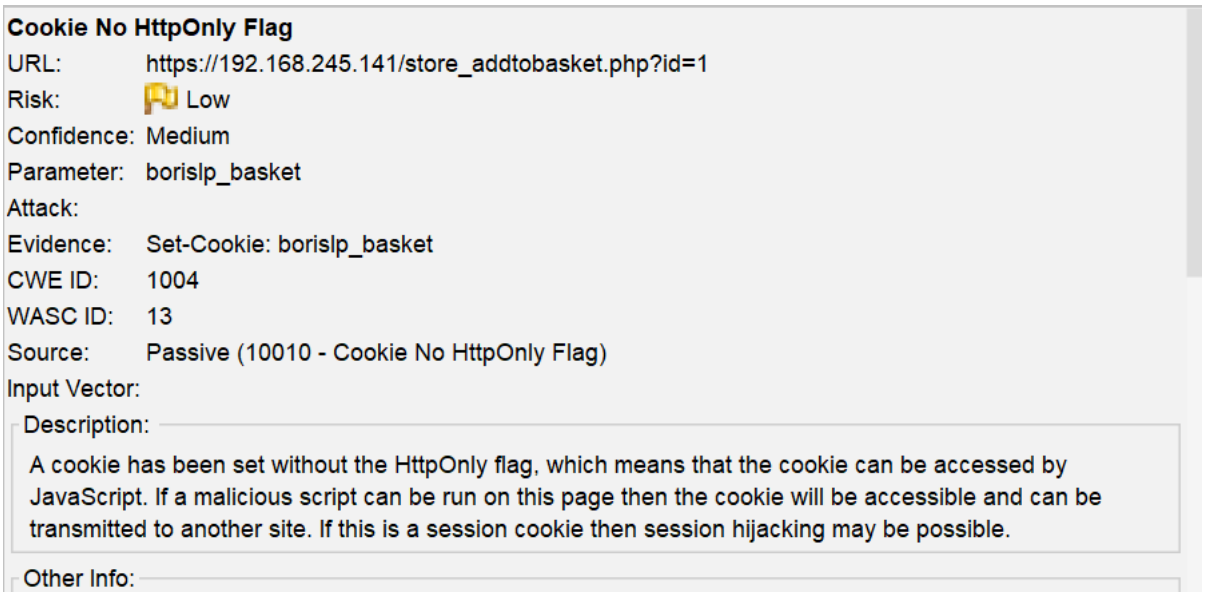
```
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Sec-Fetch-Dest: document\r\n
    Sec-Fetch-Mode: navigate\r\n
    Sec-Fetch-Site: none\r\n
    Sec-Fetch-User: ?1\r\n
  ⌄ Authorization: Basic Ym9yaXM6dGlua2VyYmVsbA==\r\n
      Credentials: boris:tinkerbell
    \r\n
    [Full request URI: http://192.168.245.141:9999/]
    [HTTP request 1/1]
    [Response in frame: 13482]
```

Det anbefales å benytte HTTPS som en sikker autentiseringsmetode, dette vil hindre uautoriserte brukere fra å få tak i kredentialer. Les mer om hvordan man sikrer autentisering her: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

Sårbarhet 18 – Cookie uten http only flagg

Risiko: Lav

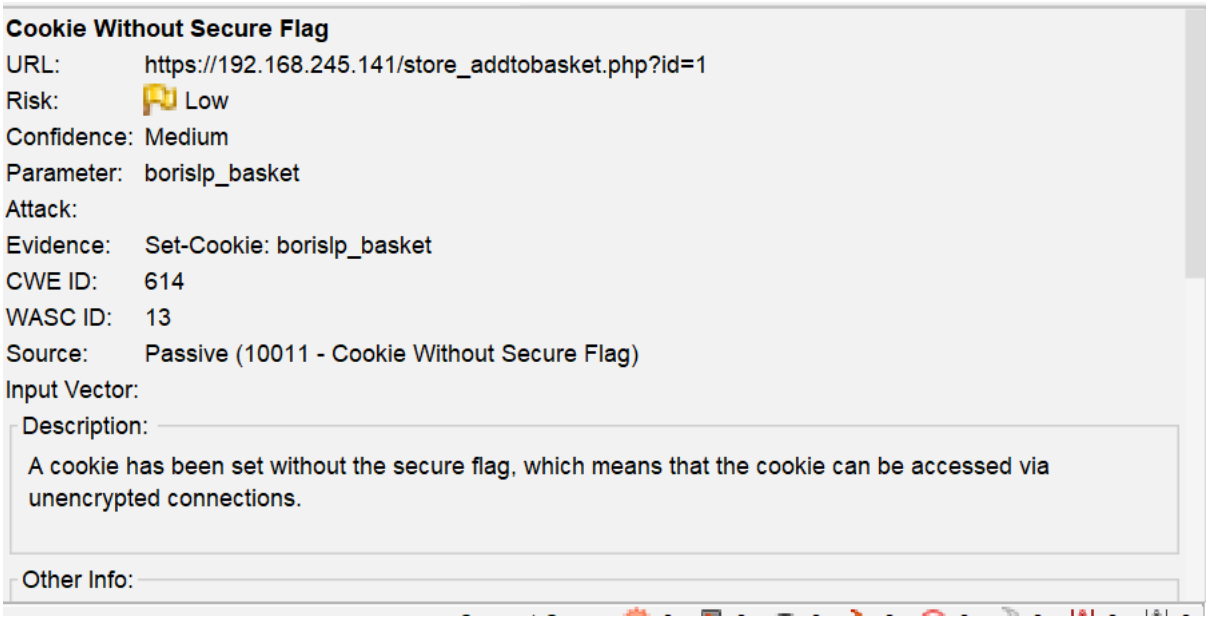
En cookie som er satt uten et HttpOnly flag kan bli tilgjengelig gjennom javascript. Dette gir en angriper muligheten til å kjøre et script på siden og deretter få tilgang til informasjonskapselen og kan misbrukes enten ved å flytte informasjonskapselen til en annen side eller ved å ta over en kjørende sesjon. Utklippet under viser dette i Owasp Zap.



Sårbarhet 19 – Cookie uten secure flagg

Risiko: Lav

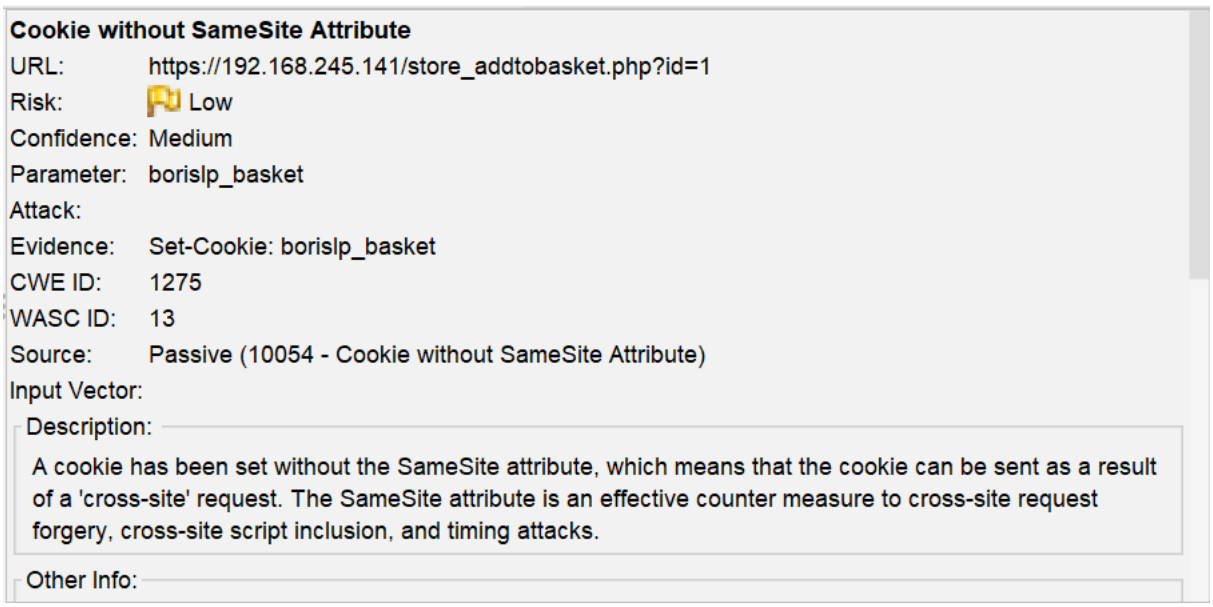
Når en cookie ikke har et secure flagg satt vil cookien bli sendt med enhver forespørsel over både http og HTTPS. Selv om webapplikasjonen kjører på HTTPS kan en angriper hijacke en session ved å tvinge brukeren til å kjøre en http forespørsel og dermed få tilgang til en kjørende session . Les mer her: https://support.detectify.com/support/solutions/articles/48001048982-cookie-lack-secure-flag. Utklippet under viser dette i Owasp Zap.



Sårbarhet 20 - Cookie uten SameSite attributt

Risiko: Lav

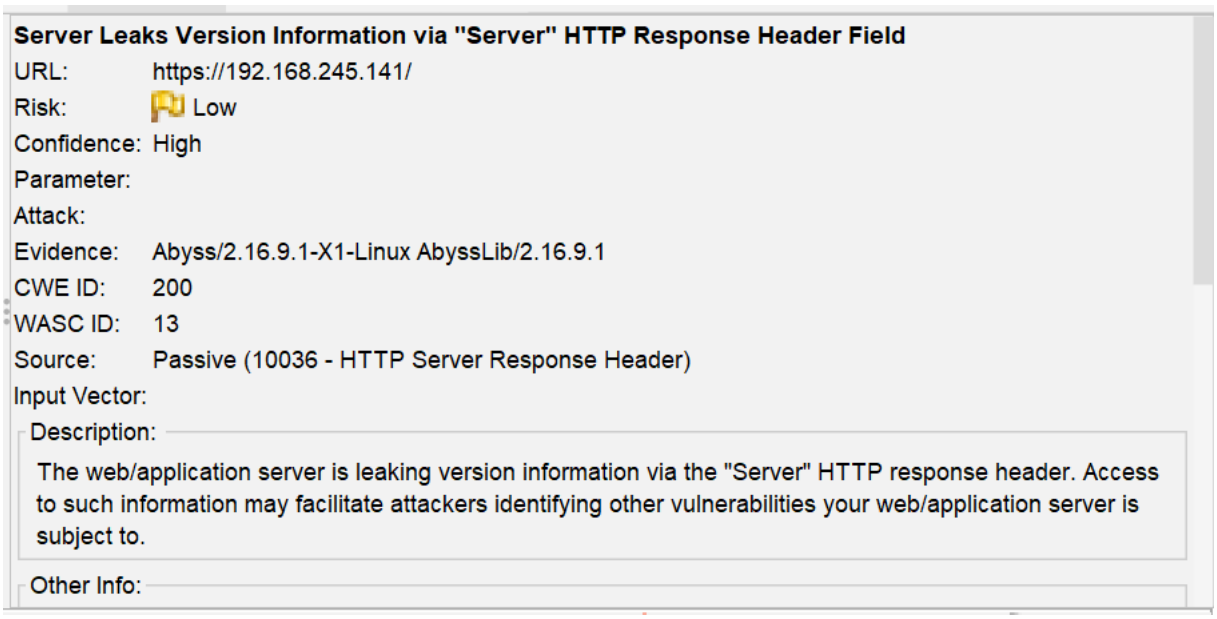
En cookie satt uten SameSite attributt kan la en angriper få tilgang til informasjonskapselen som et resultat av et cross-site request forgery angrep. Dermed kan angriperen late som de er brukeren og dermed få tilgang til informasjon de ikke skal ha tilgang til. Les mer her: https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site. Under er et utklipp fra Owasp Zap vist.



Sårbarhet 21 - Server lekker versjonsnummer

Risiko: Lav

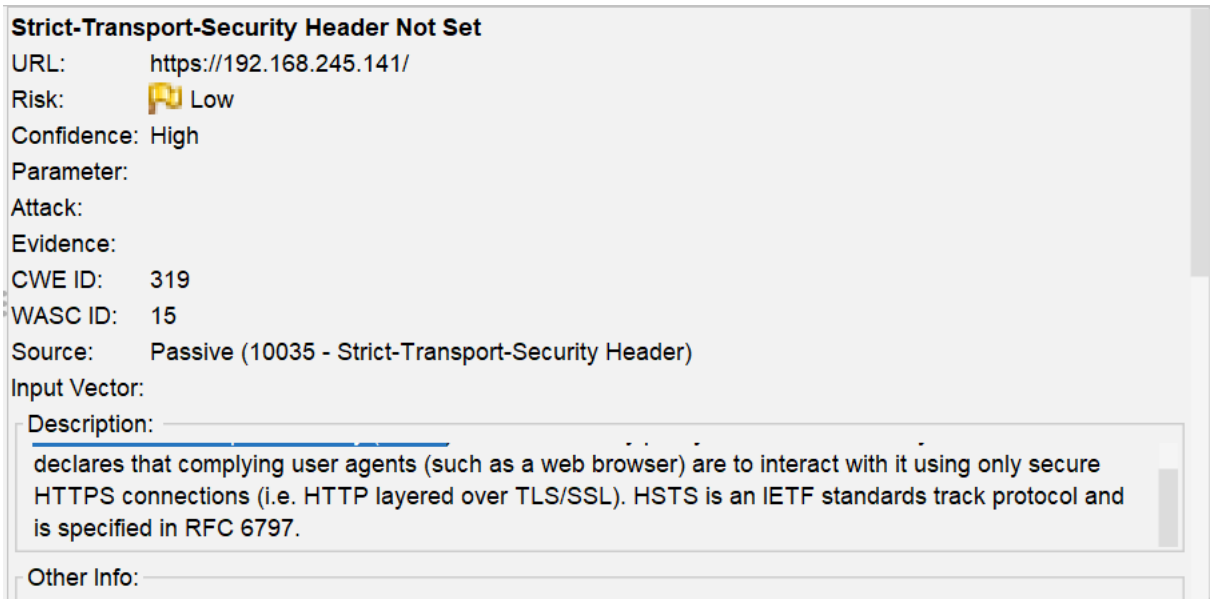
Denne sårbarheten lekker informasjon om hvilken server versjon webapplikasjonen kjører på. I seg selv er ikke dette en alvorlig sårbarhet, men dersom versjonen har kjente sårbarheter kan en angriper ved hjelp av et enkelt søk kartlegge og planlegge et angrep på webapplikasjonen. Under er et utklipp fra Owasp Zap vist.



Sårbarhet 22 - Strict-Transport-Security-header er ikke satt

Risiko: Lav

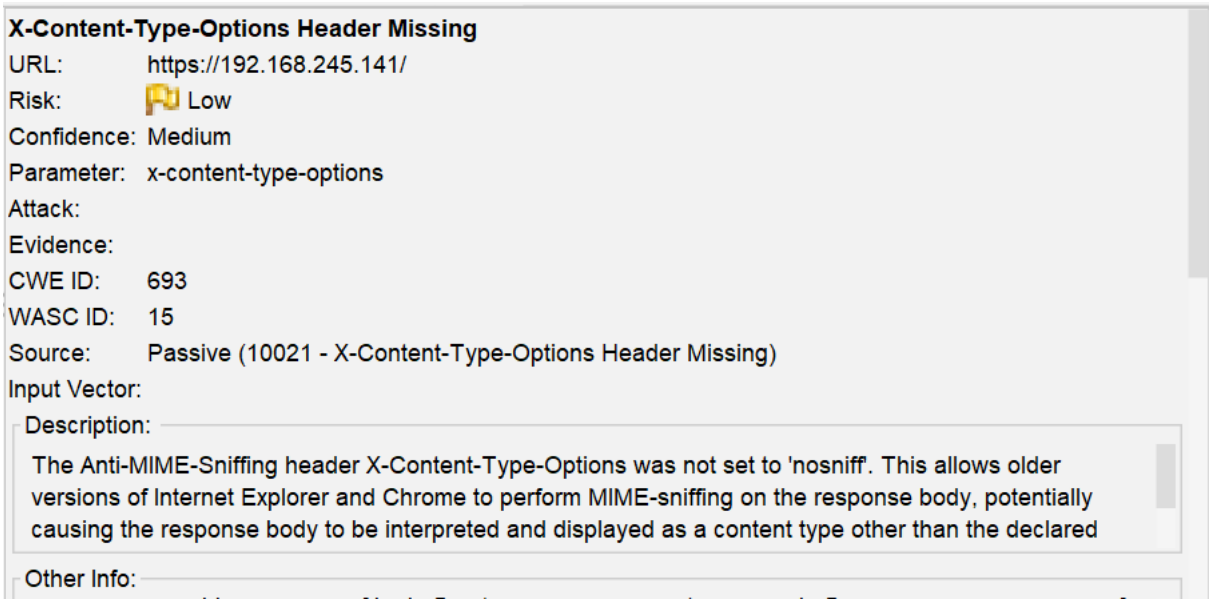
HTTP Strict Transport Security (HSTS) er en sikkerhetsfunksjon som tvinger webapplikasjoner til å sende data over HTTPS istedenfor http. Dermed vil en Strict-Transport-Security-header beskytte mot man-in-the-middle angrep, derfor anbefales det at denne headeren blir satt. Les mer her: https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html. Under er et utklipp fra Owasp Zap vist.



Sårbarhet 23 - X-Content-Type-Options Header mangler

Risiko: Lav

Dette er en sårbarhet som lar angripere utføre content-sniffing angrep. En angriper kan f.eks hvis det går an å laste opp bilder på webserveren laste opp et bilde som inneholder javascript kode og dermed kan en søkemotor som kjører content sniffing bli lurt til å kjøre koden på applikasjonen. Dette kan ligne et XSS angrep, derfor anbefales det å sette X-Content-Type-Options Header. Les mer her: https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)?redirectedfrom=MSDN. Under er det vist et utklipp fra Owasp Zap.



Konklusjon

Denne sikkerhetsrevisjonen har avdekket flere alvorlige sårbarheter i systemet, som må fikses omgående. De mest kritiske sårbarhetene involverer eksponering av de sensitive portene 21, 9999, og 22. Port 21 tillater anonym innlogging, mens port 9999 og port 22 er sårbare for brute-force angrep. Webapplikasjonen er i tillegg sårbar for IDOR, og tillater uautoriserte brukere tilgang til andre brukere. Disse sårbarhetene utgjør en kritisk risiko og kan gi angripere uautorisert tilgang og/eller sensitiv informasjon.

Videre er det påvist alvorlige sårbarheter som SQL Injection, manglende validering av inndata, XSS, svake passord, og Path Traversal, som forbindes med høy risiko. Disse kan føre til kompromittering av sensitiv informasjon.

Sårbarhetene som utgjør middels risiko, slik som manglende CSRF tokens, utlevering av feilmeldinger fra databasen, manglende headere, mm.., viser at selskapet må forbedre sin sikkerhetspraksis.

På lavrisikonivået finner vi svakheter som cookies uten sikkerhetsflagg, servere som lekker versjonsnummer, og manglende sikkerhets headere som Strict-Transport-Security og X-

Content-Type-Options. Selv om disse utgjør en lavere risiko, bør de utredes for å forbedre den totale sikkerheten på webapplikasjonen.

Disse funnene tyder på en nødvendighet for å styrke systemets sikkerhet. Det anbefales å prioritere retting av kritisk og høy risiko sårbarheter så fort som mulig, men også middels og lav risiko sårbarheter trenger oppmerksomhet. Å fikse disse sårbarhetene vil bety mye for selskapets fremtid. Dersom denne rapporten ikke blir tatt på alvor vil det utgjøre en stor sikkerhetsrisiko for selskapet, med tanke på at webapplikasjonen er et lett bytte for ondsinnede aktører.