NTNU
Det skapende universitet

Kriminalteknisk analyse av rutingsinformasjon

Njaal Brøvig Andresen

Master i datateknikk
Oppgaven levert: Juni 2007
Hovedveileder: Svein Johan Knapskog, ITEM

Norges teknisk-naturvitenskapelige universitet
Institutt for telematikk

Oppgavetekst

I denne oppgaven vil en vurdere metoder og usikkerheter relatert til bruk av kriminalteknisk analyse av rutingsinformasjon for etterforskningsformål. Teoretiske aspekter, inkludert BGP, internett topologi og systemmessige sårbarheter vil bli studert, og en prototype av en metode for å hente inn og analysere rutingsinformasjon på en kriminalteknisk holdbar måte vil bli utviklet. Det primære målet med analysen er å indikere den logiske eller fysiske lokaliteten til en IP adresse, i tillegg til om mulig å detektere uregelmessigheter som kan indikere kapring av IP adresser

Oppgaven gitt: 20. januar 2007
Hovedveileder: Svein Johan Knapskog, ITEM

Sammendrag

I denne oppgaven skulle jeg vurdere metoder og usikkerheter relatert til bruk av kriminalteknisk analyse av rutingsinformasjon for etterforskningsformål. Jeg har gjort dette ved å se nærmere på feltet IP-hijacking. Jeg presenterer her først litt bakgrunnsstoff rundt ruting og IP-hijacking, for så å ta en dypere analyse av de mest kjente deteksjonsteknikkene for IP-hijacking, med tanke på bruk i etterforskning. Ut fra dette arbeidet presenterer jeg et forslag til å forbedre en eksiterende teknikk ”Reflect Scan”. Og en ny deteksjonsmetode som baserer seg på Idlescan teknikken. Jeg har også med et forslag til en implementasjon av et ikke-distribuert system for deteksjon av IP-hijacking. Alt arbeid er utført i samarbeid med KRIPOS, som er oppdragsgiveren for oppgaven.

Forord

Denne oppgaven kommer på bakgrunn av at KRIPOS stadig trenger nye verktøy til støtte under etterforskning av forbrytelser som tar sted på internett. Med tanke på dette har jeg utført en analyse av dagens teknikker for deteksjon av IP-hijacking. Og kommet med forslag til nye deteksjonsteknikker og forbedringer på en eksiterende. Jeg vil også benytte anledningen til å takke alle som har gjort dette arbeidet mulig:

- Professor Svein Johan Knapskog ved NTNU for støtte, faglig hjelp og veiledning under hele perioden.
- Dr Andrè Årnes ved KRIPOS for faglig hjelp og veiledning.
- ITEA og UNINETT for informasjon om hvilke filter og begrensinger som er lagt på nettverkene jeg har hatt tilgang til.
- Dr Z. Morley Mao fra University of Michigan for opplysinger rundt deteksjonsteknikken Reflect Scan.
- Min samboer Henriette Kampesveen for retting og støtte gjennom hele perioden.

Innholdsfortegnelse

Liste med figurer

Liste med forkortelser

IP	Internet Protocol
AS	Autonomous System
BGP	Border Gateway Protocol
EGP	Exterior Gateway Protocol
IGP	Interior Gateway Protocol
CIDR	Classless Interdomain Domain Routing
MOAS	Multiple Origin Autonomous System
HTTP	Hypertext Transfer Protocol
MAC	Medium Access Control
MD5	Message-Digest Algorithm 5
NTNU	Norges Teknisk-Naturvitenskaplige Universitet
TCP	Transmission Control Protocol
TTL	Time To Live
IDS	Inntrenger Deteksjons-system (Intrusion Detection System)
PC	Personal Computer
ISP	Internet Service Provider
DNS	Domain Name Server
DFRWS	Digital Forensics Research Workshop

1 Introduksjon

Siden introduksjonen av internett i 1990, har det etablert seg som et nett for utveksling av informasjon for kommersielle, ikke-kommersielle bedrifter og privatpersoner. Dette globale nettverket av rutere, switcher, huber, servere og klienter, vokser nesten eksponentielt i størrelse siden det ble introdusert. Med denne økningen i størrelse og bruk, har også den kriminelle aktiviteten på dette nettverket økt. For at vi skal kunne bekjempe dette nasjonalt og internasjonalt, kreves det nye verktøy, metoder og kvalifisert personell. Politiet i Norge har overlatt denne oppgaven til KRIPOS. I denne konteksten ønsker KRIPOS å utvikle metoder og verktøy som kan hjelpe dem. Jeg har i samarbeid med Andrè Årnes og Svein Knapskog valgt å evaluere metodene som er tilgjengelige i dag til å detektere IP-Hijacking, samt foreslå hvordan et deteksjonssystem for IP-hijacking kan implementeres. Jeg kommer også med forslag til hvordan enkelte metoder kan forbedres, og forslag til nye.

Jeg hadde liten bakgrunnskunnskap på området, da jeg tok på meg denne oppgaven. NTNU eller KRIPOS, har heller ikke gjennomført noe forskning på området tidligere. Så jeg måtte starte på bar bakke.

Dette dokumentet inneholder først en del bakgrunnsinformasjon på området ruting, før det går mer inn på IP-hijacking som tema. Dokumentet er skrevet med tanke på at leseren har god kjennskap til internett fra før.

1.1 Metode

For å løse denne oppgaven har jeg først sammenlignet dagens deteksjonsteknikker for IP-hijacking. Dette med tanke på disse teknikkene kan brukes i etterforskning for KRIPOS senere. Her kommer det også inn hvordan både resultater og rådata bør lagres, slik at integriteten til bevisene beholdes.

I dette fagfeltet er det mange engelske faguttrykk som det ikke finnes i det norske språk. Jeg vil i denne oppgaven bruke flere engelske faguttrykk, som IP-hijacking, Reflect Scan, Multiple Origin Autonomous System og flere, fordi jeg føler at en leser kan lettere misforstå meningen om jeg selv finner på fornorskede utrykk.
Jeg vil bruke utrykket falske negativer når jeg snakker om tester som ikke detekterer hijacking når det faktisk er hijacket. Og bruker falske positiver om tester som detekterer hijacking, når det ikke er hijacket.

I dette dokumentet er det flere figurer, de som er basert på andres tegninger har jeg referert til dette i teksten. Er det derimot en ren kopi er det referert i bildeteksten under bildet.

1.2 Motivasjon

Motivasjonen min for å ta denne oppgaven var i utgangskpunktet at det var gjort lite arbeid fra NTNU og KRIPOS sin side på dette temaet. Jeg liker å jobbe med nye emner som det er gjort lite arbeid på fra før. Etter hvert som jeg har jobbet med oppgaven har jeg sett hvor sårbar infrastrukturen til internett egentlig er, der muligens det best kjente eksempelet er en ruter i AS7007 som var feilkonfigurert og tok ned større deler av internett, April 1997 [20]. Jeg har også sett hva IP-hijacking brukes til, der i mine øyne de styggeste tilfellene der det har blitt brukt til å spre blant annet dokumentasjon av overgrep mot barn [12]. Det er derfor viktig

for Politiet både nasjonalt og internasjonalt, og kunne overvåke internett på en bedre måte. Og kunne detektere/spore kriminalitet på internett.

1.3 Begrensninger/målsetting

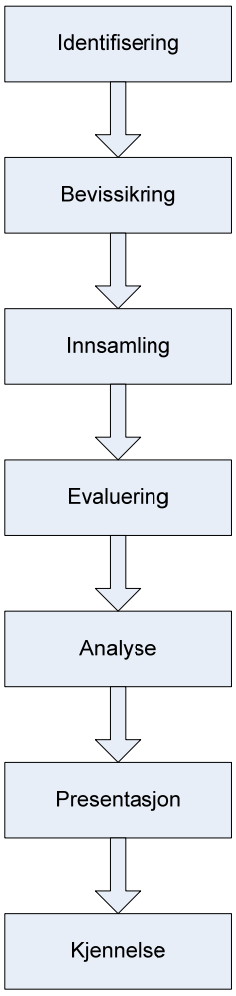
Kriminalteknisk analyse av rutingsinformasjon, er en ganske bred oppgavetekst. Jeg har begrenset den til å omhandle deteksjon av IP-hijacking. Her er det ikke lagt vekt på hvordan en Hijacker har oppnådd sin IP-hijacking, heller ikke forslag til mulige forandringer til eksisterende rutingsprotokoller for å begrense muligheten IP-hijacking. Men å analysere dagens teknikker for å detektere IP-hijacking, og komme med forslag til forbedringer av disse.

2 Etterforskning i vår digitale verden

Mye av dette kapittelet baserer seg på hva jeg skrev i min prosjektoppgave ”Internet Investigations” [17]. Etter at internett kom til verden, muliggjorde dette å gjøre nye forbrytelser. Dette gjorde at det ble krav om nye metoder å bevise slike forbrytelser, det var ikke lenger nok med et fingeravtrykk. Denne nye vitenskapen fikk det engelske navnet ”Digital Forensic Science”. Digital forensic science, er et relativt nytt begrep som først ble introdusert i 2001 på den aller første Digital Forensics Research Workshop (DFRWS) [14]. Jeg skal videre i dette kapittelet fortelle litt mer om hvordan etterforskningen foregår i vår nye digitale verden.

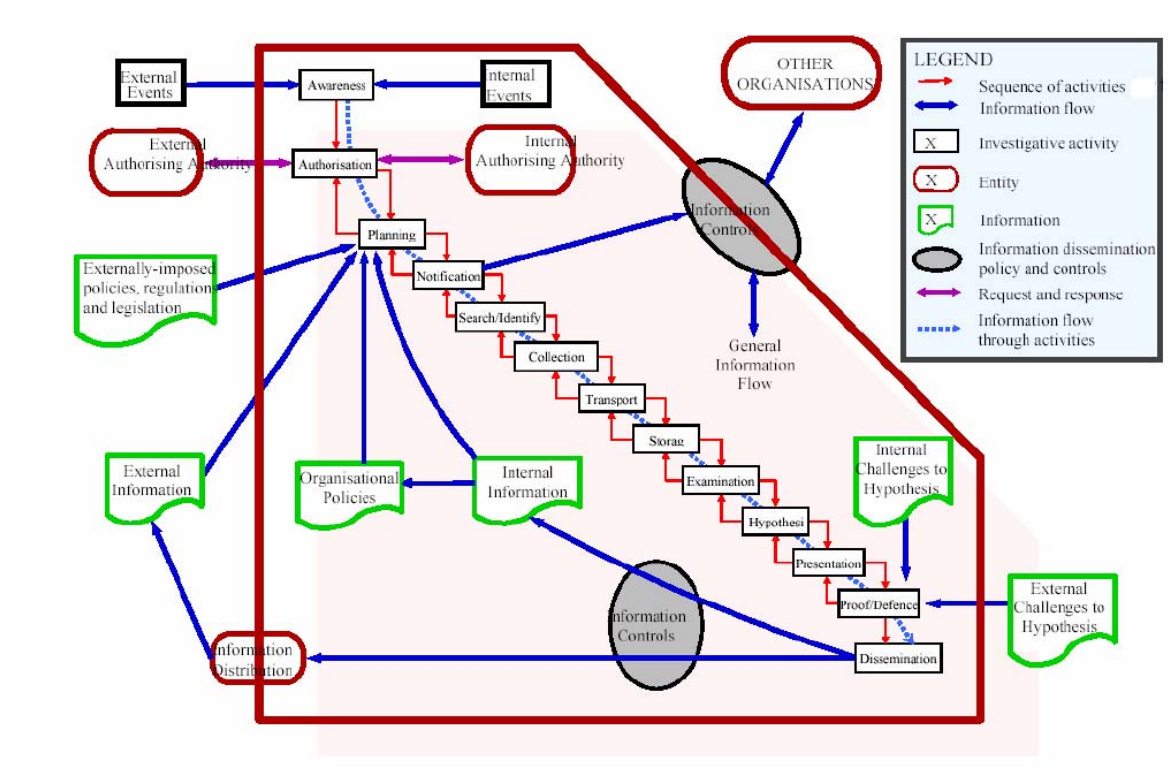
2.1 Etterforskningsprosessen

Som i en vanlig etterforskning er det etterforskerne som skal bevise hvem som har utført forbrytelsen. Det er da viktig at prosessen som blir brukt til å finne og sikre bevis er pålitelig. Så på en måte er prosessen like viktig som bevisene i seg selv. For kan ikke bevisene brukes i retten, er de jo nesten ubrukelige. Som tidligere nevnt er etterforskningen i vår nye digitale verden, en ny vitenskap innen det kriminaltekniske området. Og det var derfor viktig å utvikle nye metoder/prosesser som passet til dette nye feltet. Figur 1 viser en forenklet modell av en etterforskningsprosess for dette feltet.



Figur 1 Internett etterforskningsprosses (DFRWS) [14]

En mer dekkende modell ble laget av Séamus Ó Ciardhuáin [13] på Figur 2, som er mer kompleks, men forklarer prosessen bedre. Det er en toveismodell som mer nøyaktig viser at de noen ganger er ønskelig å gå et eller flere steg tilbake for å forbedre bevisene eller samle inn flere bevis. Denne modellen forklarer også hvordan informasjonen bør flyte i en slik etterforskning. Denne flyten av informasjon skal også sørge for at beviskjeden (”chain of custody”) holder på bevisets integritet.
Det må også nevnes at denne typen etterforskning er et nytt felt og man må regne med at det vil komme mange flere nye modeller og forbedringer på det jeg har forklart nå, i årene som kommer.



Figur 2 Internett etterforskningsprosses (Séamus Ó Ciardhuáin) [13]

2.2 Deteksjon

En forbrytelse detekteres ofte etter at den har tatt sted. Slik er det også med etterforskninger på internett. Vi kan detektere at en side er endret eller at vi ikke lenger kommer til riktig server når vi skal nå tjenester på internett. I noen tilfeller kan vi detektere at noen prøver å utføre en kriminell aktivitet før eller under et eventuelt angrep, tilsvarende en husalarm i et fysisk tilfelle. I min oppgave skal jeg ikke ha fokus på deteksjon av kriminalitet på internett, men i stedet se på måter å detekter IP-hijacking på, som kan hjelpe KRIPOS i å etterforske kriminalitet på internett.

2.3 Bevissikring

Som at et fysisk bevis må sikres, slik at det kan brukes i en rettssak senere, må vi på samme måte sikre bevisene vi samler inn når vi har en etterforskning med digitale bevis på internett.

Vi ønsker at bevisets integritet beholdes, gjennom hele etterforskningen. Vi bruker ofte da det engelske uttrykket ”Chain of custody”, som er alt som har skjedd med beviset siden det ble beslaglagt/innhentet, kontroll, analysert og lignende. Slik at det kan bevises i retten at ikke beviset kan ha blitt tuklet med, slik at det mister sin notoritet. For da kan ikke beviset lengre brukes i en rettssak. Espen Andre Fossen har i sin prosjektoppgave ”Automatic tracing of Internet adresses” [16] kommet med et forslag på hvordan man kan sikre elektroniske bevis bedre. Der han bruker MD5 [31] hash av beviset sammen med et tidstempel, for alle lagringer av bevis i sin database. Dette forbedrer integriteten til bevisene, slik at de står sterkere i en rettssak.

3 Ruting og BGP

I denne delen skal jeg forklare litt bakgrunnsstoff om internett, Border Gateway Protocol(BGP) som rutingsprotokoll og svakheter ved denne. Dette for å skape en base for IP-hijacking kapittelet som kommer senere. Vi starter med tildeling av IP-adresser og AS-nummer på internett.

3.1 Tildeling av AS-nummer og IP-adresser på internett

Internet Assigned Numbers Authority(IANA)[07] tildeler IP- adresser via Regional Internet Registry (RIR). Hver samling av nettverk som blir administrert uavhengig av resten av internett vil få tildelt et AS(Autonomous System) nummer av deres respektive RIR. Hvert AS er selv ansvarlig for rutingen og tildeling av IP-adresser innen sitt nettverk. Alle AS deler informasjon om sitt nettverk til alle nettverk den er tilknyttet (Peering).

3.1.1 IP-adresser til intern bruk

Ikke alle IP-adresser er ment brukt på internett. RFC 1918 [10] definerer følgende adresser som IP-adresser til privat bruk:
10.0.0.0 – 10.255.255.255
172.16.0.0 – 172.31.255.255
192.168.0.0 – 192.168.255.255
Disse adressene skal aldri bli annonsert på internett, og må ikke blandes med uallokerte IP-adresser som ennå ikke er tildelt et AS. Altså adresser som ikke er i bruk i dag.

3.1.2 Private AS-nummer

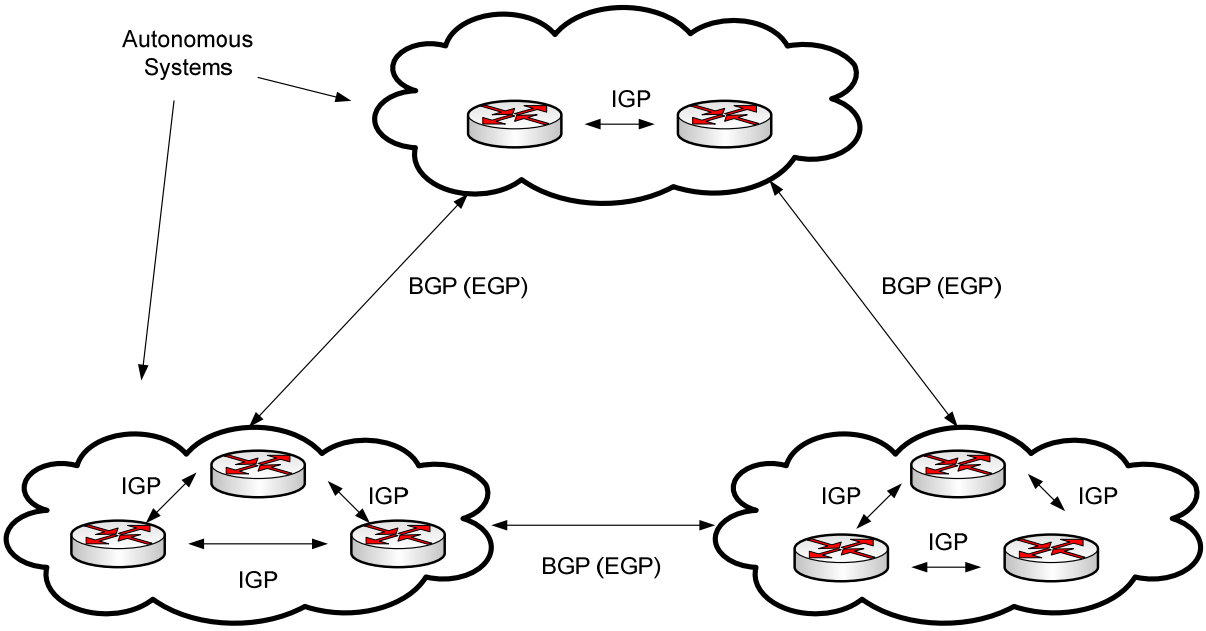
I RFC 1930 [09] har The Internet Assigned Numbers Authority (IANA) reservert AS numrene 64512 til 65535, til privat bruk. Det vil si at de kan brukes internt i et AS om de ønsker å dele opp rutingen sin internt innen sitt AS. Disse numrene skal aldri bli annonsert på internett.

3.2 EGP og IGP

Interior Gateway Protocol(IGP) definerer hvordan nettverk innen et AS utveksler ruting informasjon. Man kan også her bruke EGP-protokoller, som BGP internt sammen med private AS-nummre. De meste kjente IGP-protokollene er:

- RIP (Routing Information Protocol versjon 2, RFC 2453) [05]
- OSPF (Open Shortest Path First, RFC 1247) [06]

Exterior Gateway Protocol (EGP), den første protokollen ble definert av BBN (Bolt Beranek and Newman Inc) i 1982 i RFC 827, som måten ASer skulle utveksle ruting informasjon. Dagens EGP protokoll for internett er BGP (Border Gateway Protocol) versjon 4 definert i RFC 1771 [04] i 1995.



Figur 3 IGP og EGP

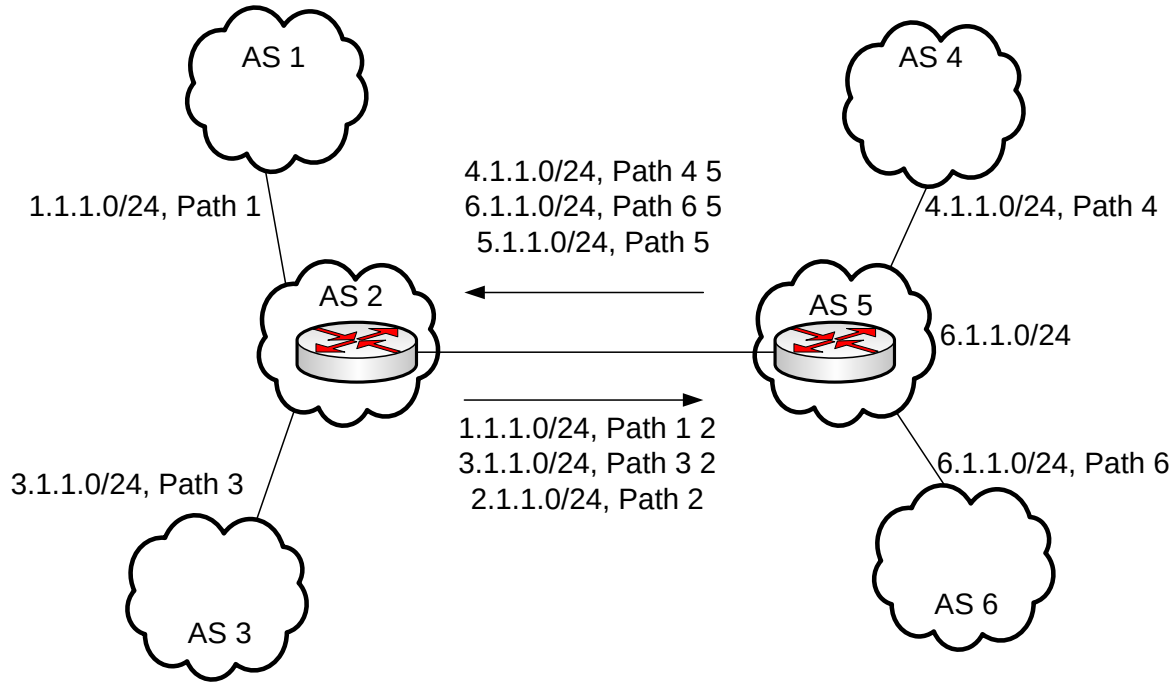
3.3 BGP-versjon 4

Border Gateway Protocol (BGP) har gjennom tidene gått igjennom flere oppgraderinger, fra det første versjonen BGP-1 i 1989. Til BGP-4 ble prøvd for første gang i 1993[01] med tilhørende RFC 1771 [04]. Denne tilbød som første EGP ruting protokoll, klasseløs ruting (CIDR).

3.3.1 Hvordan opererer BGP-4

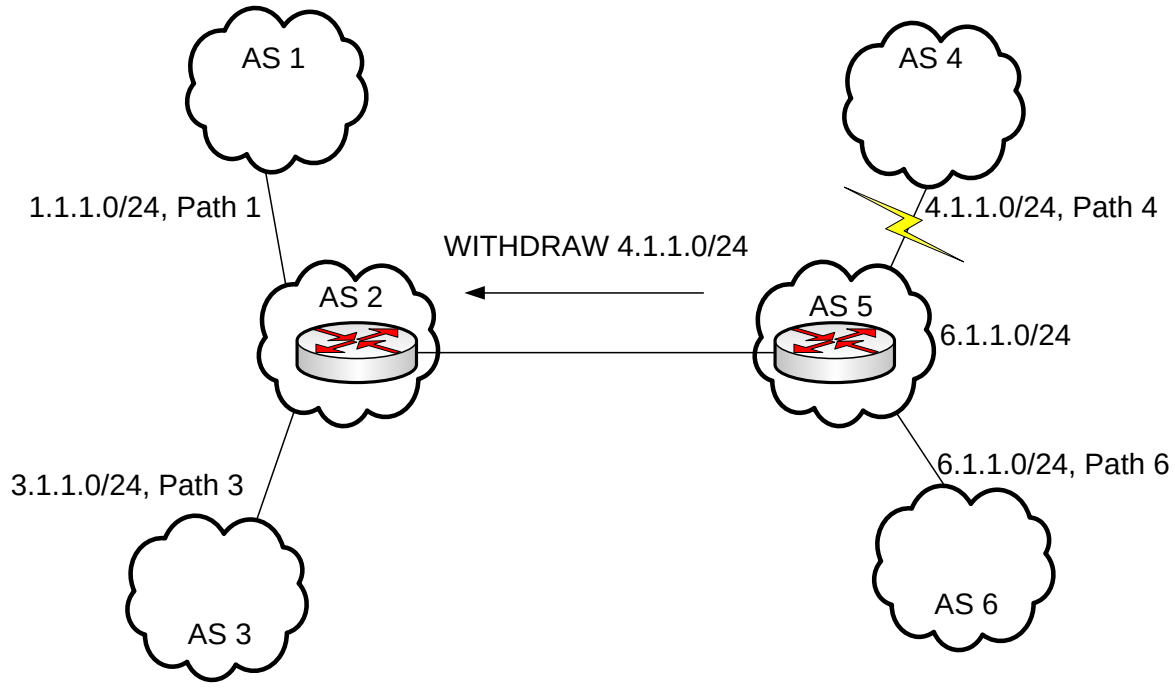
Helt grunnleggende kan man si at BGP-4 er en vei vektor protokoll, som brukes til å utveksle ruting informasjon mellom autonome systemer.
BGP-4 bruker TCP (port 179) som sin transport protokoll, for at man skulle slippe å tenke på pålitelighet som for eksempel retransmisjon, i BGP. BGP-4 brukes kun mellom nabo rutere eksternt i internett. En slik BGP kobling kan bare settes opp manuelt.

Første gang man kobler en ny ruter via BGP-4 vil ruterne dele full ruting informasjon(ruting tabeller). Figur 4 viser et eksempel på dette, her blir ruten mellom AS2 og AS5 koblet opp. Vi ser at AS5 sender all ruting informasjon til AS2 via BGP.



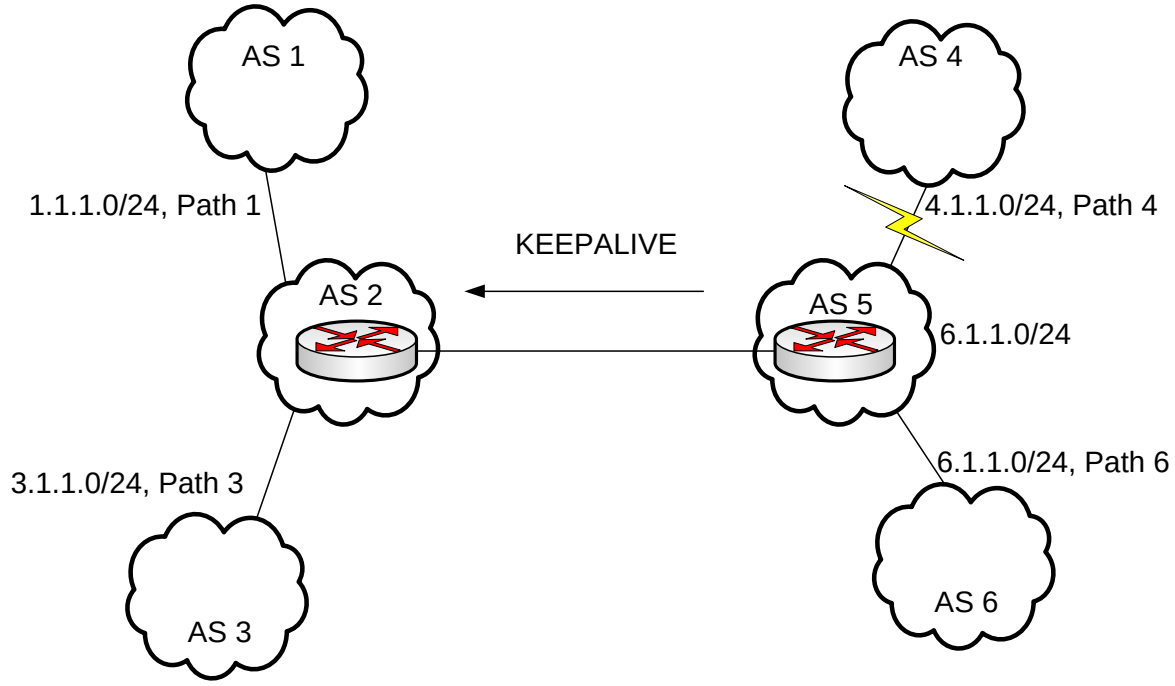
Figur 4 BGP første tilkobling.

Senere når det blir endringer i nettverket/rutingen, vil kun selve oppdateringen bli sendt, ikke hele ruting tabellen slik som på Figur 4. I eksempelet på Figur 5 ser vi at AS5 har mistet ruten til AS4, og sender derfor denne oppdateringen via en UPDATE(withdraw) melding til AS2 sin ruter. I det AS2 mottar denne meldingen vil den slette ruten til AS4 via AS5 fra sin rutingstabell, og videresende withdraw meldingen til sine tilknyttede rutere i AS1 og AS3.



Figur 5 BGP Update

Om det ikke er noen oppdateringer ønsker BGP likevel å holde oppe TCP-koblingen og sender derfor KEEPALIVE pakker med jevne mellomrom, typisk 60sekunder. Figur 6 viser et eksempel på dette, der AS 4 er fortsatt nede.

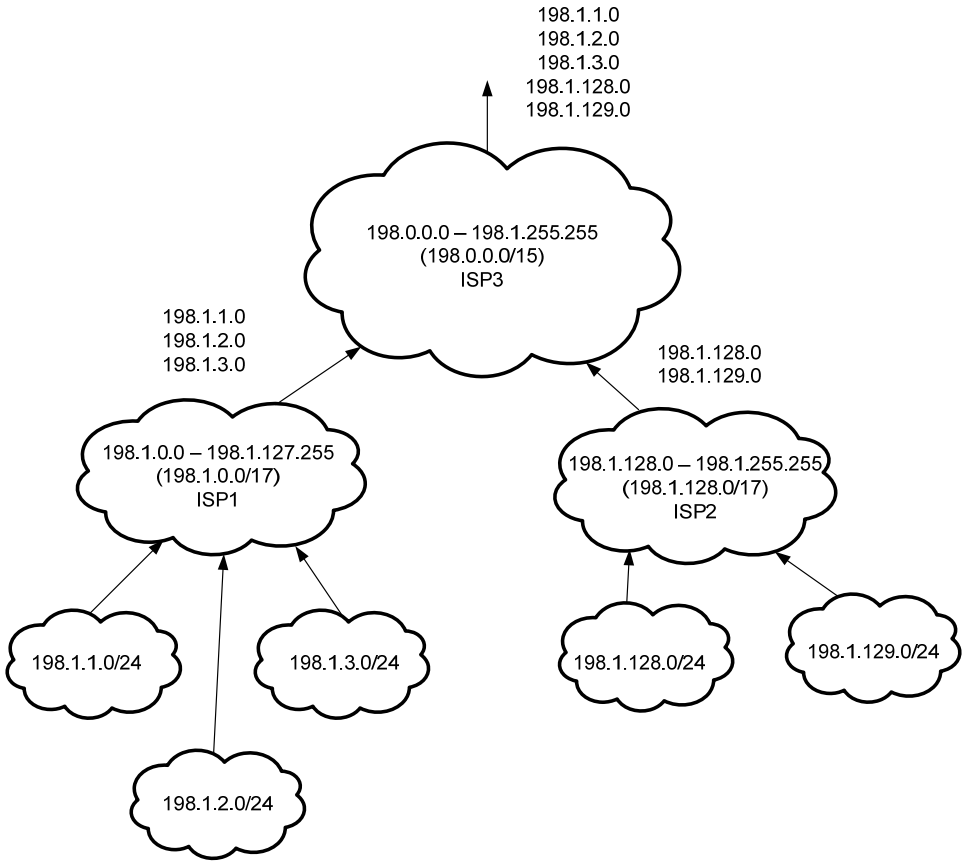


Figur 6 Keepalive pakker

3.3.2 Klasseløsruting

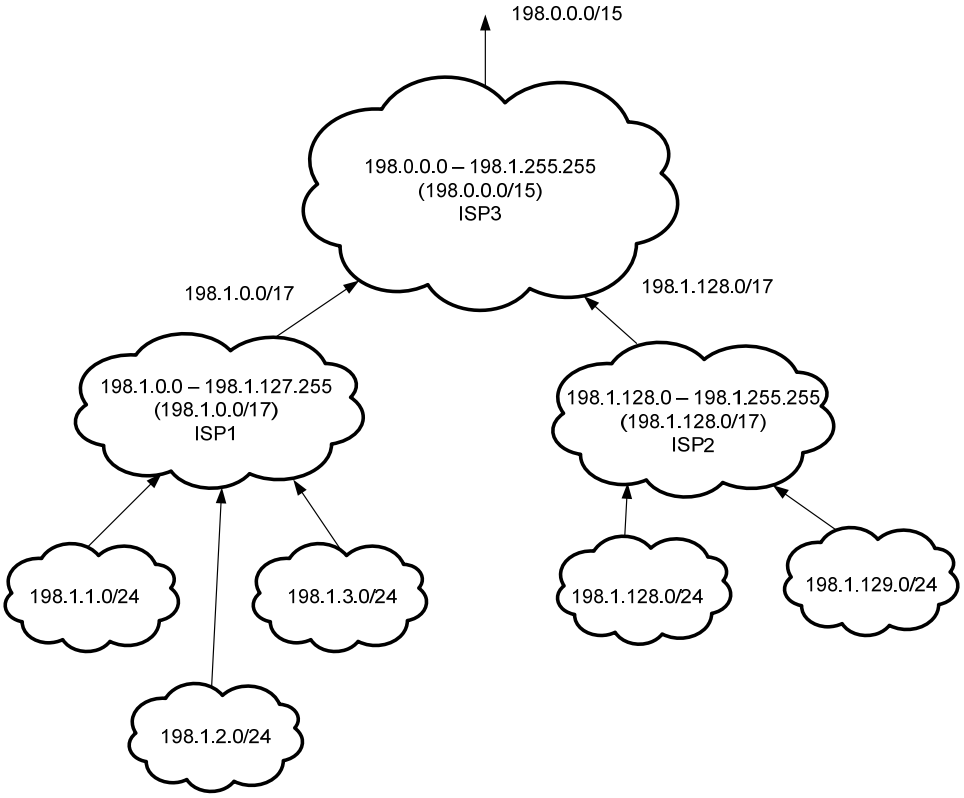
Ved innføringen av BGP versjon 4 ble det åpnet for klasseløs ruting, det vil si at man ikke lengre trengte å forholde seg til klassene A, B og C i ruting tabellene. For eksempel har et

gammelt klasse C nettverk 230.25.1.0 med nettmaske 255.255.255.0, sin tilhørende CIDR [08] adresse 230.25.1.0/24 der /24 betyr at de 24 første bittene fra venstre er nettmasken. Men den store forskjellen ligger i at det ikke lengre trenger å være en hel klasse for at det skal kunne annonseres som en rute, for eksempel kan to klasse C nettverk 221.40.0.0/24 og 221.40.1.0/24 nå annonseres som 221.40.0.0/23. Vedlegg viser forskjellen mellom klasse og klasseløs ruting. Og det er av denne grunnen at klasseløs ruting ble innført på grunn av at ruting tabellene i de enkelte ruterne ble for store, og tiden/prosessorkraften det tok å søke igjennom ruting tabellen for hver enkelt IP-pakke ble for stor. Og ved å innføre CIDR ble ruting tabellene betydelig redusert. Men dette ga også muligheter for nye typer IP-hijacking, noe jeg skal komme tilbake til.



Figur 7 Klasseruting [01]

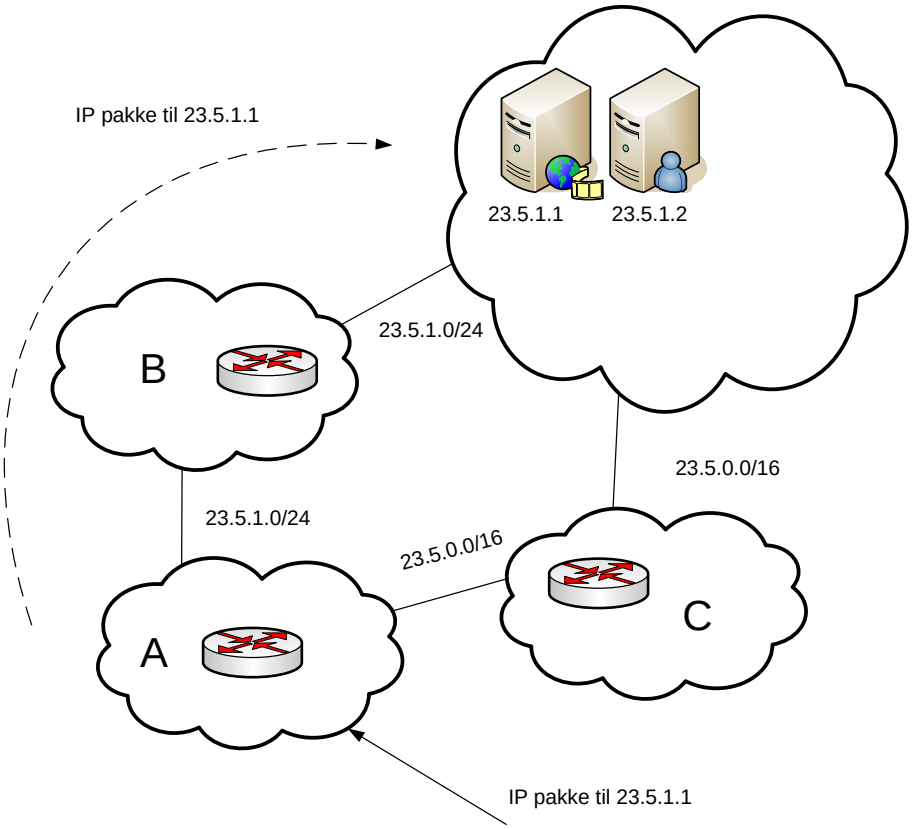
Om vi ser på Figur 7, viser denne hvilke ruter som vil ut av dette nettverket bli annonsert videre til andre nettverk via BGP. Som vi ser ut av denne figuren så blir det da annonsert fem forskjellige ruter. Med klasseløs ruting, som Figur 8 viser, ser vi at det i samme tilfelle får kun en rute 198.0.0.0/15 i stedet for fem på den eldre klasse rutingen.



Figur 8 Klasseløs ruting [01]

Treff i flere linjer i rutingtabell:

I enkelte tilfeller der et AS er koblet til flere ISPer og ønsker at et subnett av sitt større nettverk skal rutes spesielt, for eksempel på grunn av levering sanntidsdata eller andre data med krav til spesiell linjekvalitet. Prioriteringen i ruterne kalles på engelsk ”The Longest Match Routing Rule” [03]. Og om vi ser på Figur 9, viser den hvordan en pakke som er adressert til 23.5.1.1, alltid vil bli rutet via B fordi 23.5.1.0/24 er den mest spesifikke ruten som passer adressen. Figur 9 er basert på en figur fra ”Internet Routing Architectures” [01].



Figur 9 Longest Match Routing Rule

3.4 BGP-4 svakheter

Det er finnes svakheter i de fleste systemer, og det er selvfølgelig også i BGP-4. På grunn av at det i dag er så mange bedrifter og enkeltpersoner som er avhengige av at internett fungerer, så har det store konsekvenser når internettet blir angrepet. Og på grunn av selve infrastrukturen til internett, så vil ”små” feil kunne føre til meget store konsekvenser. Et eksempel på dette er da AS7007 25. April 1997 [20] en feilkonfigurert ruter startet å annonsere, ”at via meg kommer du raskest til mer eller mindre hele internett”. Dette endte med at mange nettverk var umulig å kontakte via internett i flere timer. Et nyere tilfelle er da Google ble tatt av nett 7.mai 2005 i en periode fra 15 til 60 minutter [21], der det i ettertid viste seg å være en MOAS konflikt i flere rutere, selv om Google utad sa at det var en DNS-server feil.

3.4.1 Blackholing

Her ønsker man rett og slett å få deler eller hele internett til ikke å kunne nå et prefiks, ved å bare droppe pakkene man mottar til dette prefikset. Dette finnes det flere legitime tilfeller av, for eksempel er det brukt til å sørge for at ikke private eller uallokerte IP-adresser kan nås på internett. En hacker kan bruke det til å få trafikken rutet mot en spesiell ruter der han bare dropper pakkene. Eller bare droppe alle pakkene den mottar.

3.4.2 Hijacking (Omdirigering av trafikken)

Her tar hijackeren og omdirigerer slik at trafikken til et spesielt nettverk blir omdirigert til annet nettverk som hijackeren har kontroll over. På denne måten kan hijackeren lettere utgi seg for å være noen andre og så få tak i sensitive opplysinger. Han kan også bruke denne typen hijack til å omdirigere trafikken, slik at en annen adresse blir jammet ned av trafikk. Mer om denne typen angrep kommer jeg tilbake til i kapittel 4.

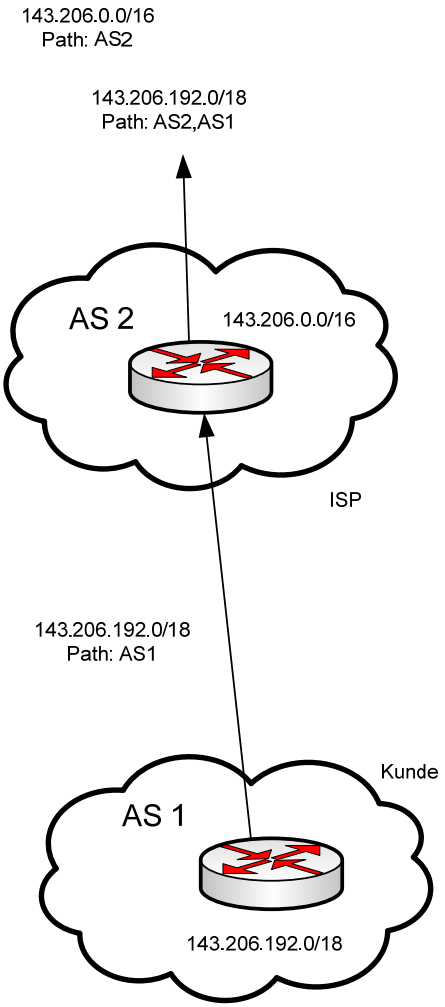
3.4.3 Ustabilitiet

Dette kan man gjøre ved å annonsere og trekke tilbake ruten flere ganger og i et raskt tempo. På denne måten vil man kunne generere stor BGP-trafikk i hele eller deler av internett, eller få ”route dampening” til å slå inn. Om route dampening slår inn kan dette resultere i at trafikken vil ta en annen rute på internett, eller at et nettverk blir utilgjengelig [19] om det ikke finnes andre ruter.

3.5 Multiple Origin AS

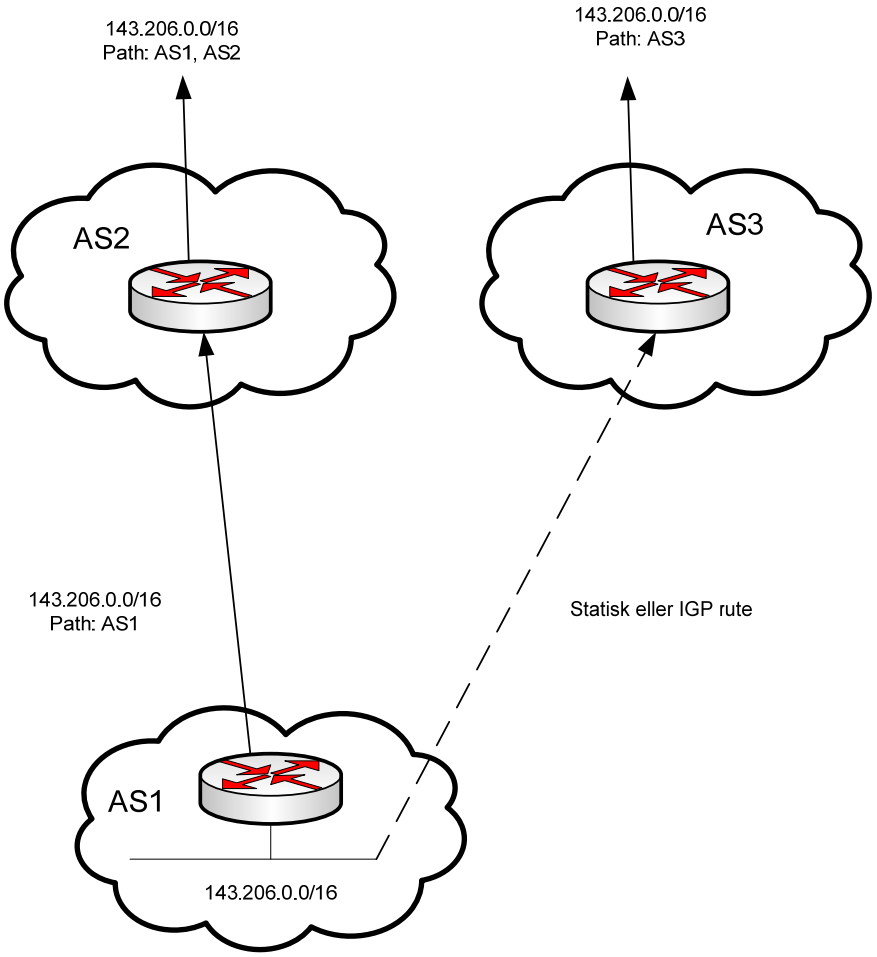
I henhold til RFC1930 [09], skal et i IP-prefiks kun ha et AS som annonserer dette til resten av nettet. Om ikke dette gjøres får vi noe vi kaller Multiple Origin AS(MOAS). MOAS er noe vi ofte knytter til IP-hijacking. Men kan i flere tilfeller også oppstå i legitime situasjoner.

Hvis vi ser på Figur 10 kan vi se et eksempel på dette, der en kunde har leid et sub-prefiks av ISP’en sitt prefiks. Om dette sub-prefikset har fått sitt eget AS nummer oppstår det sub-MOAS, fordi AS 2 i dette tilfellet gir to ruter til adressen 143.206.192.0/18, med to forskjellige AS-nummer.



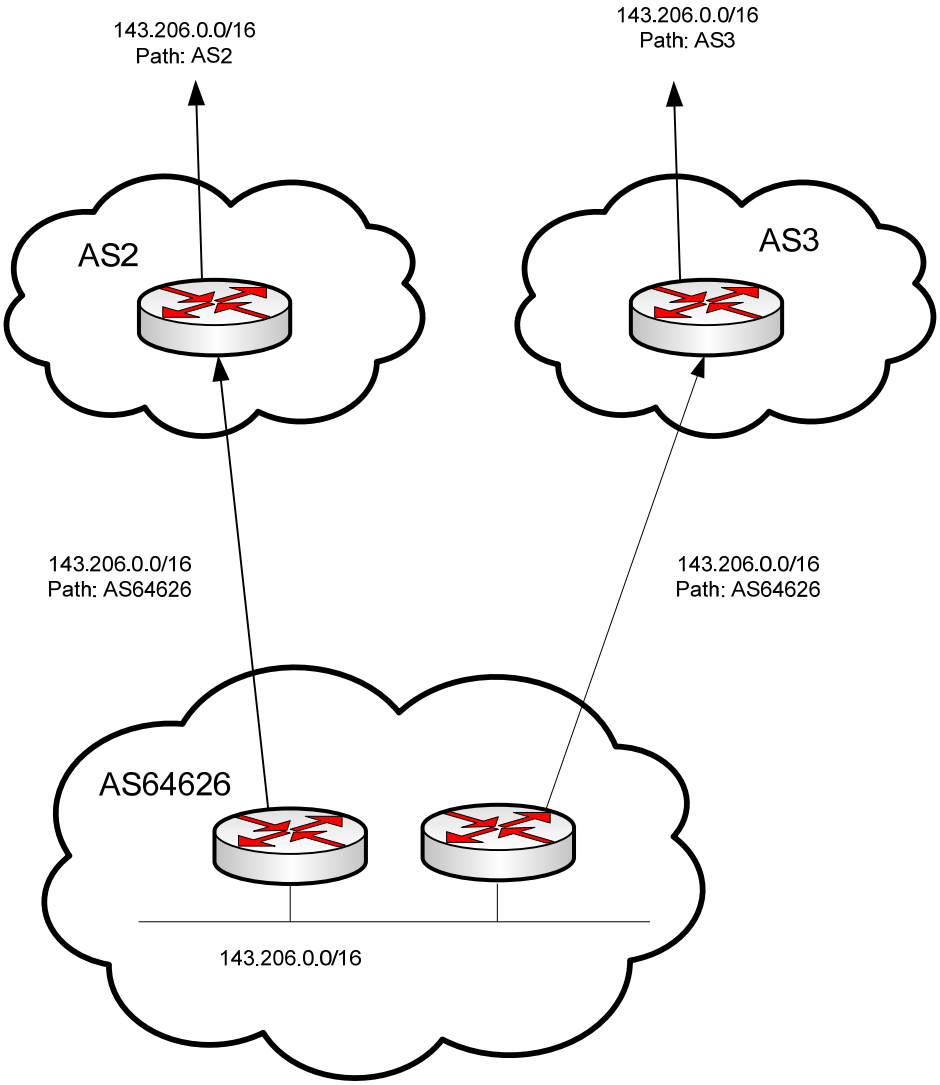
Figur 10 Sub-MOAS

Figur 11 viser oss hvordan det oppstår MOAS ved bruk av en statisk-link i kombinasjon med en rutet-link. Her ser vi at både AS 2 og AS 3 annonserer ruten til prefikset 143.206.0.0/16. Ruterne som mottar begge disse annonseringene vil detektere dette som MOAS.



Figur 11 MOAS med statisklink

Også ved bruk av private AS nummer kan MOAS oppstå i legitime tilfeller, slik Figur 12 viser. Fordi de neste ruterne ikke vil distribuere et privat AS nummer videre til andre rutere. Dette vil føre til at flere rutere annonserer at de har prefikset 143.206.0.0/16 med sitt AS-nummer.

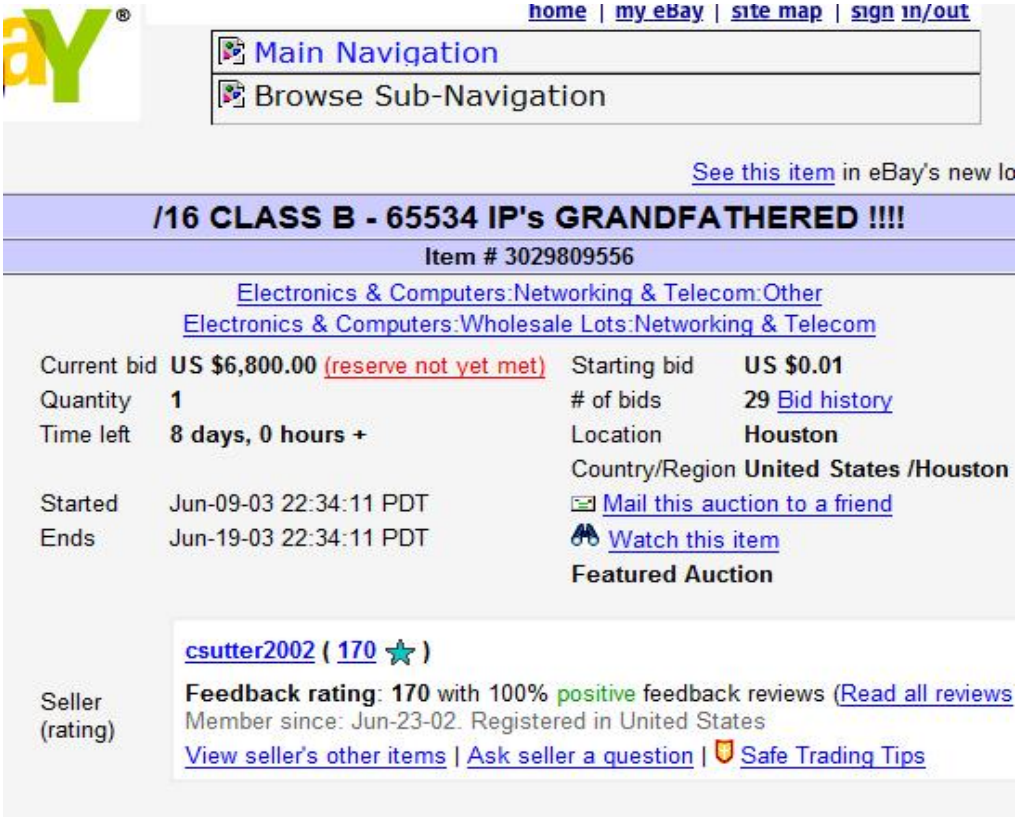


Figur 12 MOAS med bruk av privat AS nummer.

4 IP-Hijacking

4.1 Hva kan IP-hijacking brukes til?

IP-hijacking har mange formål for en hijacker. Som oftest er det for å tjene penger, da i salg eller å leie IP-adressene ut. Det er faktisk tilfeller der en hijacker har prøvd å selge IP-adresser på ebay.com se Figur 13. Men i de fleste tilfeller brukes det til Spam/Porno [12], og de verste tilfeller brukes det til å spre dokumentasjon av overgrep mot barn [12].
Men det også viktig å tenke på er at internett i dag brukes av bedrifter som om det var leide linjer. Dette resulterer i store tap for mange bedrifter når deres nettverk påvirkes av IP-hijacking og andre typer ustabilitet ved internett.



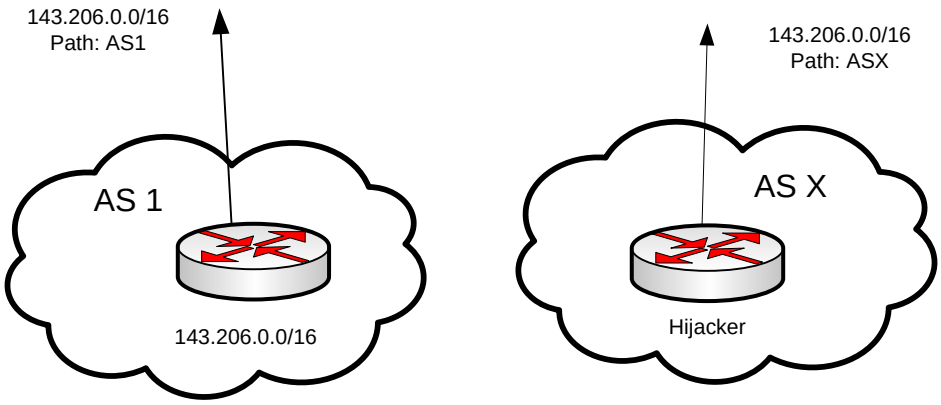
Figur 13 Salg av IP-adresser [11]

4.2 Forskjellige typer IP-hijacking

IP-hijacking kalles ofte BGP-hijacking, noe som ligger naturlig i at det er Border Gateway Protocol (BGP) som brukes til å utveksle ruting informasjon på internett i dag [01]. For at noen skal kunne gjennomføre IP-hijacking må denne personen først få kontroll over en internettruter. Etter at dette er gjennomført kan angriperen sette opp denne ruteren til å annonsere falsk ruting informasjon og på den måten få hele eller deler av internett til å rute trafikk til denne ruteren. Vi har flere typer av IP-hijacking, men vi kan rangere disse inn i fem hovedtyper som jeg nå skal ta for meg.

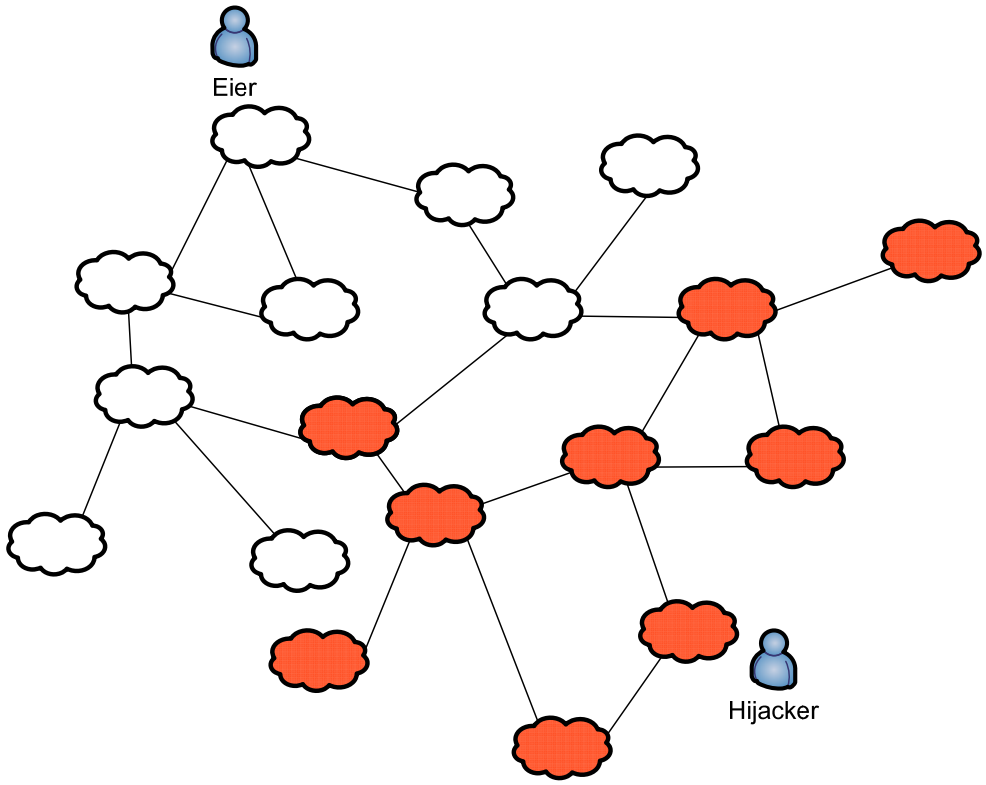
4.2.1 Hijack et helt prefiks:

Figur 14 viser denne typen for angrep. I dette eksemplet har en hacker tatt over kontrollen av en ruter i AS X, og bruker denne kontrollen til å annonsere til resten av internett at prefikset 143.206.0.0/16 skal rutes mot AS X. Dette vil ende med Multiple Origin AS, det vil si at det vil bli konflikter i rutingtabellene til ruterne fordi to forskjellige ASer har annonsert samme IP prefiks. Men vi kan ikke bare ut fra dette konkludere med at det har blitt foretatt en IP-hijacking, fordi som vi så på Figur 11 og Figur 12 kan dette også oppstå i legitime tilfeller.



Figur 14 Hijack av et prefiks

Ved hijacking av et helt prefiks vil dette ende med at internett blir delt i to, der ene delen ruter trafikken til selve eieren av prefikset og andre halvdel ruter trafikken til hijackeren. Dette er altså snakk om hvilken av de to rutene som er mest attraktive å ta for hvert enkelt nettverk. Figur 15 viser et eksempel på et miniinternett der de røde nettverkene vil rute trafikken mot hijackeren, og de hvite vil rutes til eieren av prefikset.



Figur 15 Miniinternett

4.2.2 Deteksjon av denne typen hijack

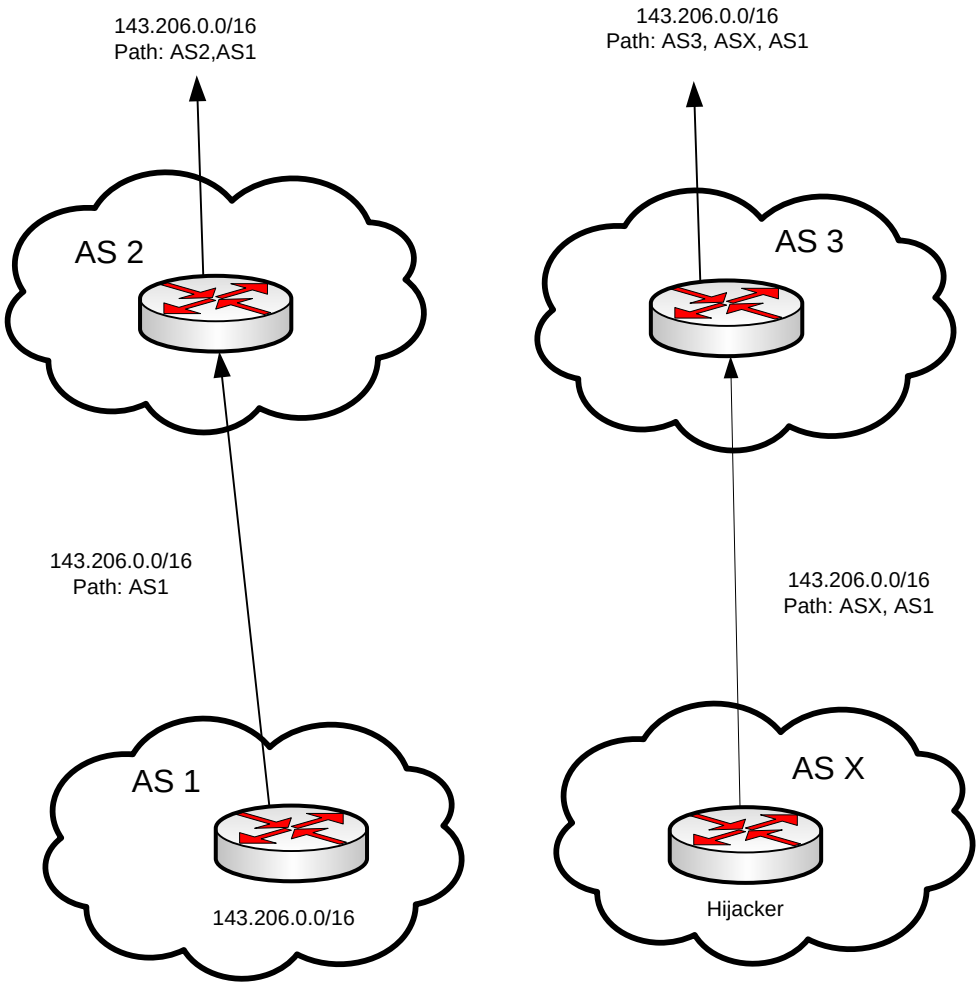
Mer utfyllende informasjon om deteksjonsteknikkene er beskrevet i kapittel 4.3.

- MOAS konflikt
- Elektronisk fingeravtrykk
- Geografisk lokasjon
- Logisk lokasjon

4.2.3 Hijack et helt prefiks og sitt tilhørende AS

Som vi ser på Figur 16 så har her hackeren tatt over en ruter i AS X, og bruker denne til å annonsere at den har en kobling til AS 1 og sitt tilhørende prefiks. På denne måten vil deler av internett som mener at ruten via AS X til AS 1 er kortest, prøve å koble seg opp via denne ruten (fra de adressene som mener den ruten er mest attraktiv). Denne typen hijacking vil ikke føre til en MOAS konflikt, så denne typen er litt vanskeligere å oppdage at har blitt utført.

Denne typen IP-hijacking vil føre til at internett blir splittet som i kun hijacking av et prefiks, se Figur 15.



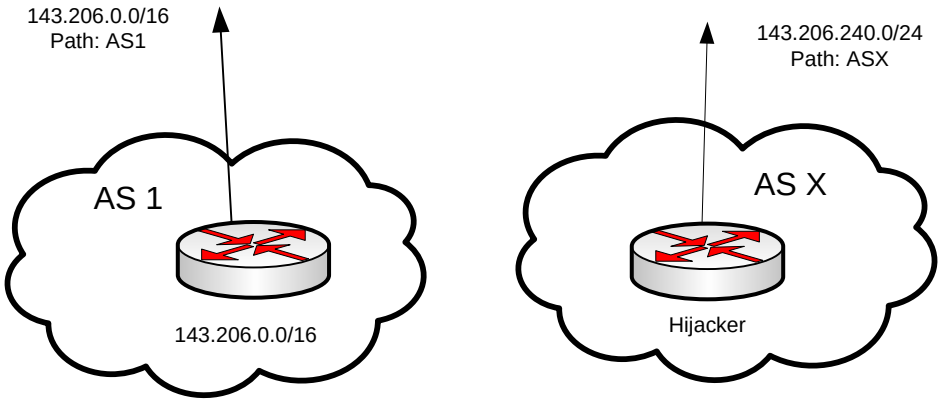
Figur 16 Hijack av prefiks og tilhørende AS

4.2.4 Deteksjon av denne typen hijack

- Elektronisk fingeravtrykk
- Geografisk lokasjon
- Logisk lokasjon

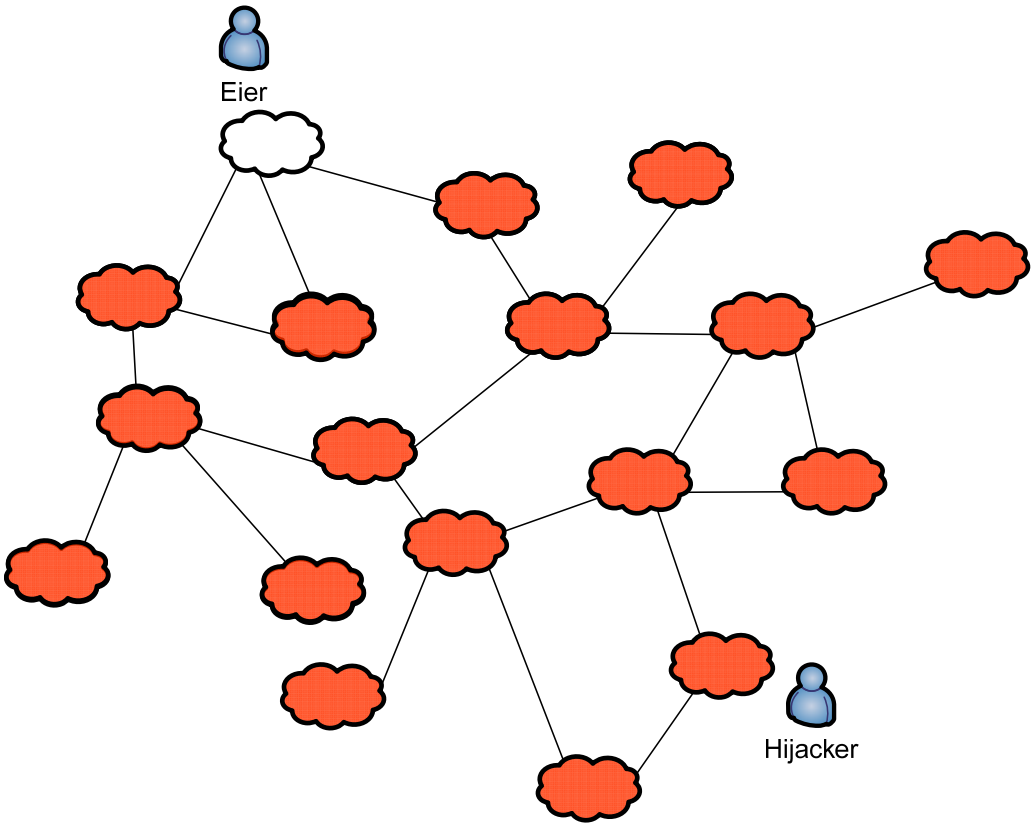
4.2.5 Hijacking av et subnett

I dette tilfellet (Figur 17) velger hackeren å bruke sin kontroll over ruteren i AS X til å annonsere at trafikk til 143.206.240.0/24 som er et subnett av AS 1 sitt IP-prefiks, skal rutes til AS-X. På denne måten vil hackeren oppnå at trafikk til disse adressene rutes fra hele internett til AS X, men det vil oppstå en sub-MOAS konflikt i ruterne. På grunn av at det mest spesifikke prefiks blir distribuert videre (Engelsk: Longest Prefix Based Forwarding [03]). Så på denne måten å hijacke på, slipper hackeren å bekymre seg over å gjøre ruten til hans AS mest attraktiv.



Figur 17 Hijacking av et subnett av et større prefiks

Denne typen hijacking vil føre til at alle adresser i subnettet hijackeren annonserer, vil bli rutet til han, slik Figur 18 viser. Her ser vi at hele internett bortsett fra nettet til den virkelige eieren av prefikset som blir rutet via IGP, blir rutet til hijackeren.



Figur 18 Miniinternett

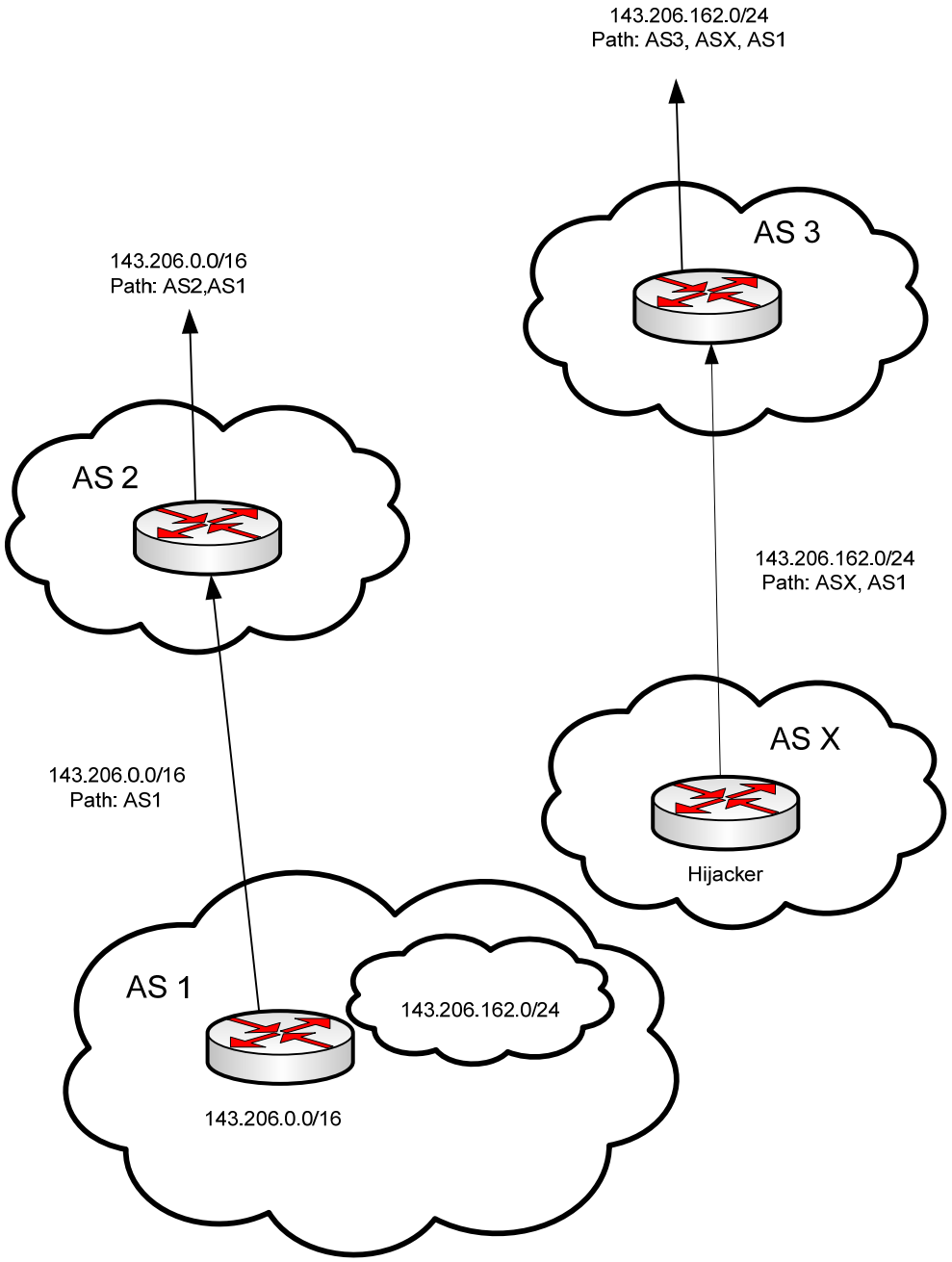
4.2.6 Deteksjon av denne typen hijack

- Sub-MOAS konflikt
- Geografisk lokasjon
- Logisk lokasjon
- Reflect Scan

4.2.7 Hijack et subnett og tilhørende AS

Her blir et subnett av et større prefiks IP-hijacket, med sitt tilhørende AS. Måten hijackeren gjør dette på er å annonsere at han har en rute til subnettet 143.206.162.0/24 gjennom sitt AS, se Figur 19. På denne måten unngår hijackeren at det blir MOAS-konflikter, derfor er denne måten å hijacke på noe vanskeligere å detektere.

Man vil med denne typen IP-hijacking oppnå at alle adresser som skal til det hijackede subnettet vil bli rutet til hijackeren, som Figur 18 viser et eksempel på.



Figur 19 Hijack subnett og tilhørende AS.

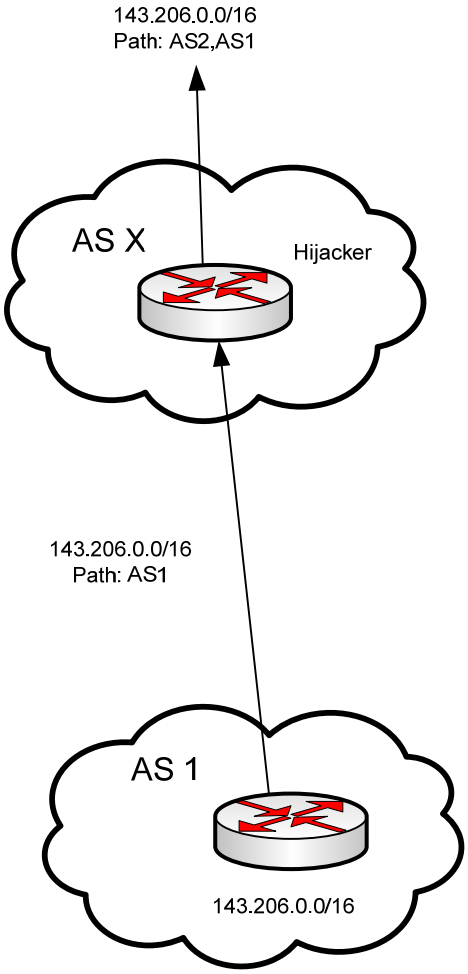
4.2.8 Deteksjon av denne typen hijack:

- Geografisk lokasjon av subnett versus hele prefiks

- Logisk lokasjon
- Reflect Scan

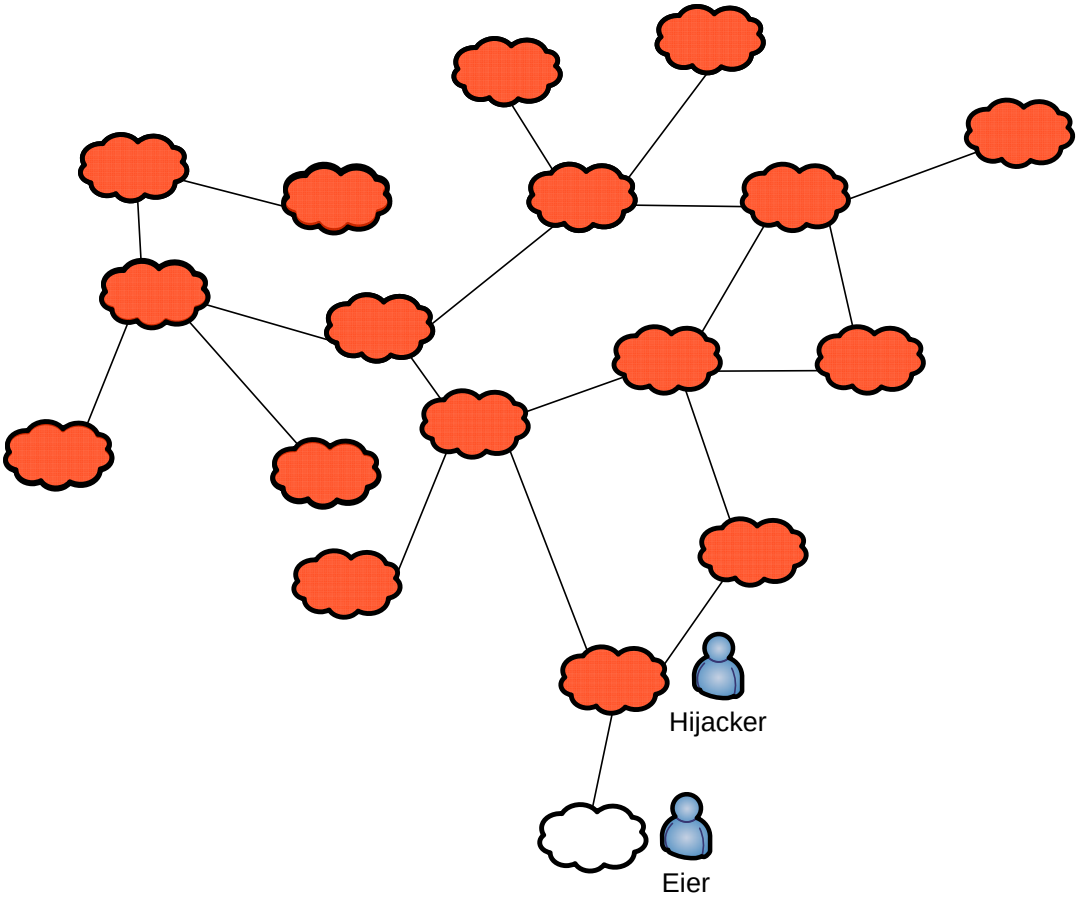
4.2.9 Hijack et AS i vanlig rutingsvei

Hvis vi ser på Figur 20 ser vi at om en hacker har tatt over ruteren i AS X, kan han mer eller mindre styre trafikken som går fra, og som skal til, AS 1 som han selv vil. Dette kan oppdages om han stopper all trafikk til AS 1 i AS X, men om han bare ønsker å stoppe enkelte adresser kan det være meget vanskelig å detektere. Denne typen hijacking vil heller ikke føre til noen form for MOAS.



Figur 20 Hijack gjennom vanlig rutingsvei.

Denne typen hijacking vil som tidligere nevnt gjøre at hijackeren kan velge hvilke adresser som skal gå igjennom til eieren av adressene, og stoppe/manipulere de han selv kan tenke seg. Figur 21 viser et mer helhetlig bilde av hvordan dette påvirker internett.



Figur 21 Miniinternett

4.2.10 Deteksjon av denne typen hijack

- Traceroute
- Elektronisk fingeravtrykk (om hijackeren filtrere adressene)

4.3 Deteksjonsmekanismer for IP-hijacking

Det er nevnt tidligere flere mekanismer for å kunne detektere IP-hijacking. Jeg skal forklare de mer i detalj, og til slutt oppsummere hva som er bra og dårlig med de enkelte metodene.

4.3.1 Traceroute

Traceroute (RFC 1393) [24] bruker ICMP (Internet Control Message Protocol) til å finne ruten en pakke bruker igjennom internett for å nå en bestemt internettadresse. Dette kan brukes til å prøve å finne ut om pakkene faktisk kommer fram til riktig mottaker, mot eventuelle hijack i vanlig rutingsvei. Her må man være oppmerksom på at ICMP-pakker kan bli stoppet av brannmurer. På denne måten kan det da i enkelte tilfeller se ut som om ikke pakken kommer fram til destinasjonen sin. Dette vil da gi en del falske positiver, om man ikke tar utgangspunkt i at man tar med at det er en ugyldig probing om man ikke får svar.
Det er også gjort arbeid for å utvikle et AS-nivå traceroute verktøy [33]. Denne baserer seg på vanlige ICMP-pakker, men løfter traceroute som verktøy opp på AS-nivå. Det samme arbeidet viste også at tilnærmet 50 % av alle tracerout testene ga et tvetydig svar.

4.3.2 Reflect Scan

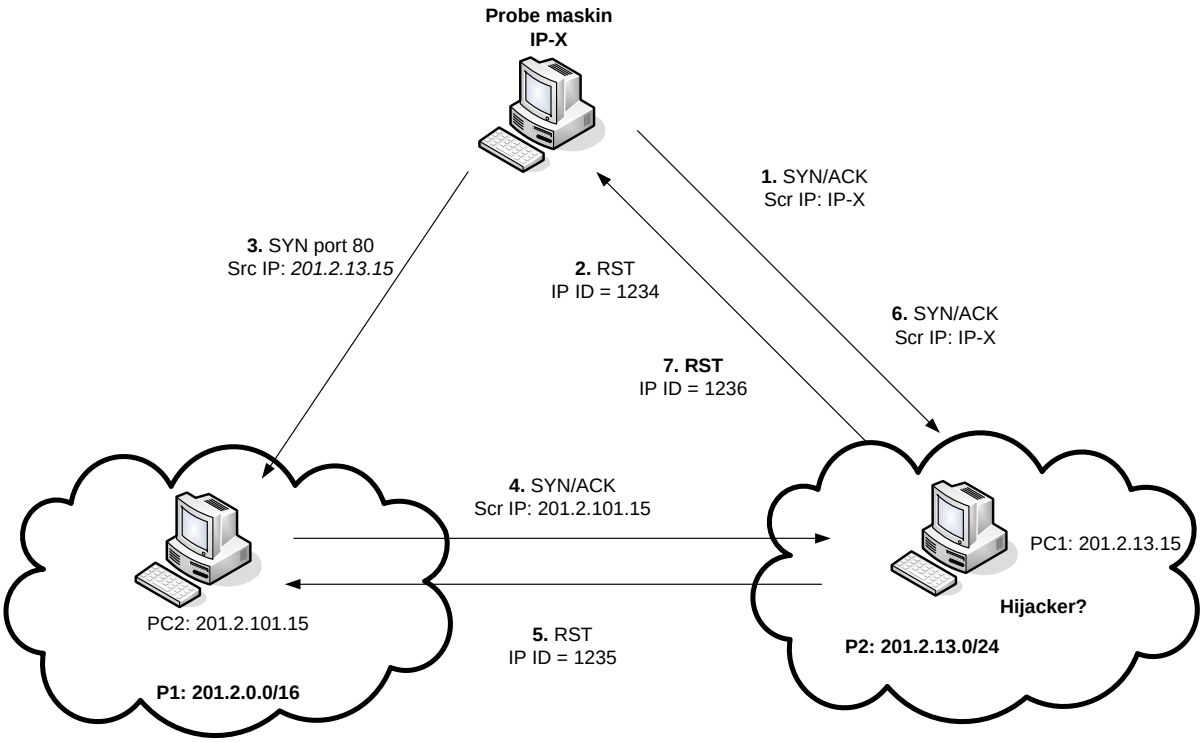
Reflect-scan er en måte å kunne detektere subnett IP-hijacking, og er brukt som en metode av Xin Hu og Z. Morley Mao [03]. Denne teknikken baserer seg på Idlescan [25] som er en del av NMAP [26] nå. Den tar utgangspunkt i å bruke en annen PC til å skanne porter på en IP du ønsker å vite mer om. I en reflect scan er vi ikke interessert i å finne hvilke porter som er åpne, men vi vil heller bruke denne teknikken til å finne ut om vi har kontakt med samme PC via 2 forkjellige ruter på internett, ved hjelp av å sende ut pakker med falsk avsender adresse (spoofet pakke). For å kjøre denne testen kreves det at du finner en ”idle” vert, som IP-Identification feltet øker med 1 for hver pakke den sender ut, samt at det er bare vi som genererer nettverkstrafikk på denne verten mens testen gjennomføres.

Reflect Scan uten IP-Hijacking:
Her tar vi som utgangspunkt i at det kan være en sub-MOAS konflikt, og at et subnett av et større nettverk er annonsert med egen rute på internett. Teknikken går så ut på å finne en PC som ikke er i bruk (”idle), i det subnettet som vi mistenker for å være hijacket (PC1), og en PC (PC2) i det større nettverket som ikke er en del av det subnettet som vi mistenker å være hijacket.
Om vi ser videre på Figur 22 viser denne hvordan man går frem for å gjennomføre en Reflect Scan.

1. Vi sender en SYN/ACK pakke fra vår egen probe maskin til en ”idle” vert, her PC1.
2. Vi får da et svar fra PC1 med en RST pakke, vi leser IP ID 1234 ut fra denne pakken.
3. Vi sender så en SYN til port 80 på 201.2.101.15, men med falsk avsender IP (Src IP) nemlig IP: 201.2.13.15, IPen til PC1.
4. PC2 vil da svare på denne SYN pakken via IGP, med en SYN/ACK til PC1.
5. PC1 svarer på denne med en RST pakke med IP ID: 1235, den har altså lagt til 1 på sin IP ID.
6. Vi vil så sende ut ny SYN/ACK til PC1
7. Får et svar med en RST pakke fra PC1 med IP ID: 1236.

Vi kan så sammenligne IP IDen fra de to RST pakkene vi har mottatt (2 og 7), og om verdien på denne har økt med 2 så har vår SYN pakke fra punkt 3 kommet til samme PC (PC1) som

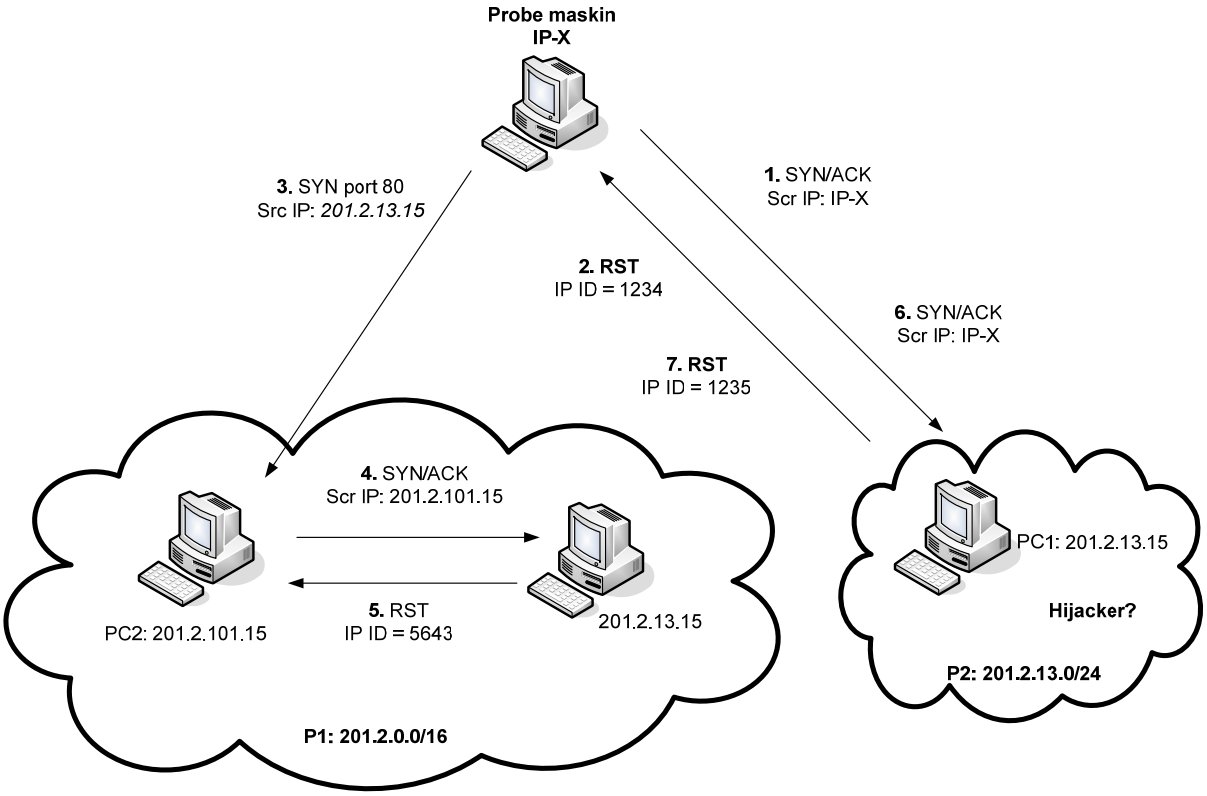
våre SYN/ACK pakker sendt i punkt 1 og 6. Når vi vet dette kan vi konkludere med at P2 ikke er IP-hijacket. Vi skal nå se på hva som skjer hvis P2 faktisk er IP-hijacket.



Figur 22 Reflect Scan uten IP-Hijacking [03]

Reflect Scan med IP-hijacking

Nå skal vi bruke samme metode på et eksempel der det faktisk er hijacket et subnett, se Figur 23. Vi gjør her akkurat det samme fra vår probe PC som i Figur 22. Men vi kan se på IP ID verdiene vi mottar fra PC1, at den ikke har mottatt SYN/ACK pakken fra PC2 i punkt (4). Derfor kan vi ut fra dette konkludere med at det faktiske subnettet P2 er IP-Hijacket i dette tilfellet. Men det skal vise seg å være en drøy konklusjon, om vi ser på hva som vil skje om det er en brannmur på nettet P1.

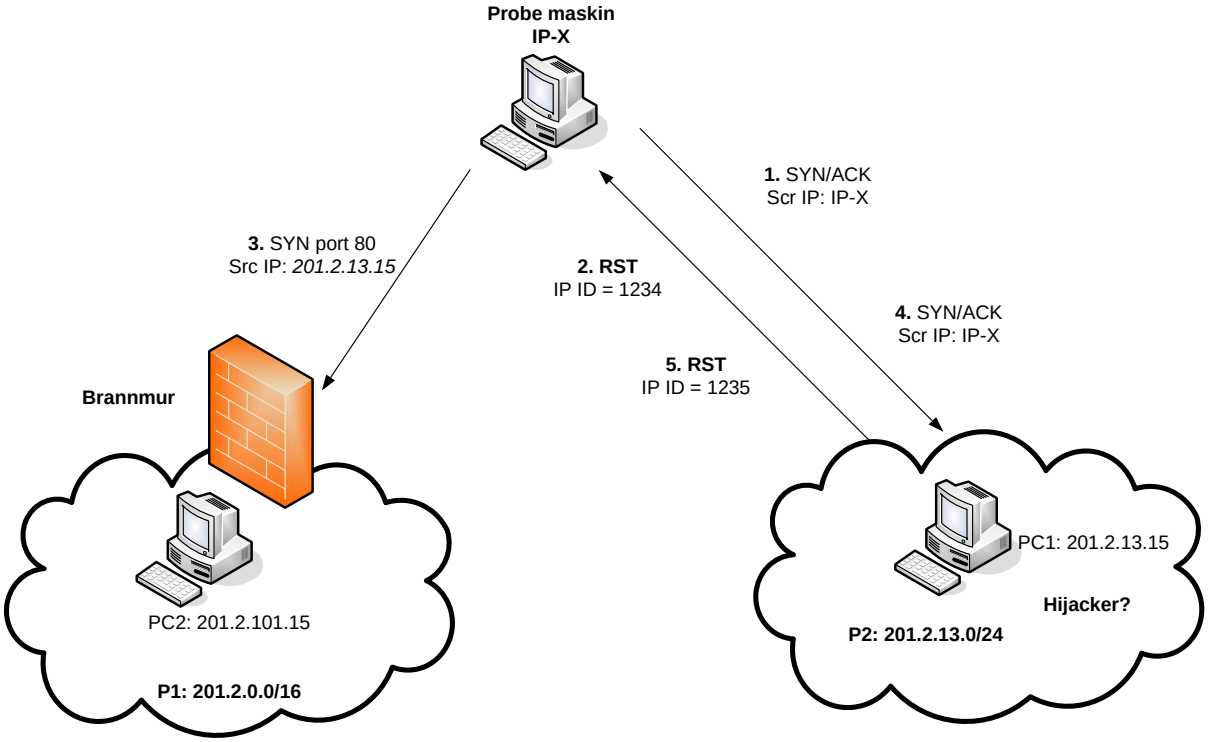


Figur 23 Reflect Scan med IP-hijacking [03]

Reflect Scan uten IP-Hijacking med brannmur

Om vi ser på Figur 24 hvordan denne typen test vil gå på et system med en brannmur, ser vi at IP IDene som vi mottar ikke har økt med mer enn 1, ergo vil vi i utgangspunktet konkludere med at P2 er hijacket selv om dette ikke er tilfelle.

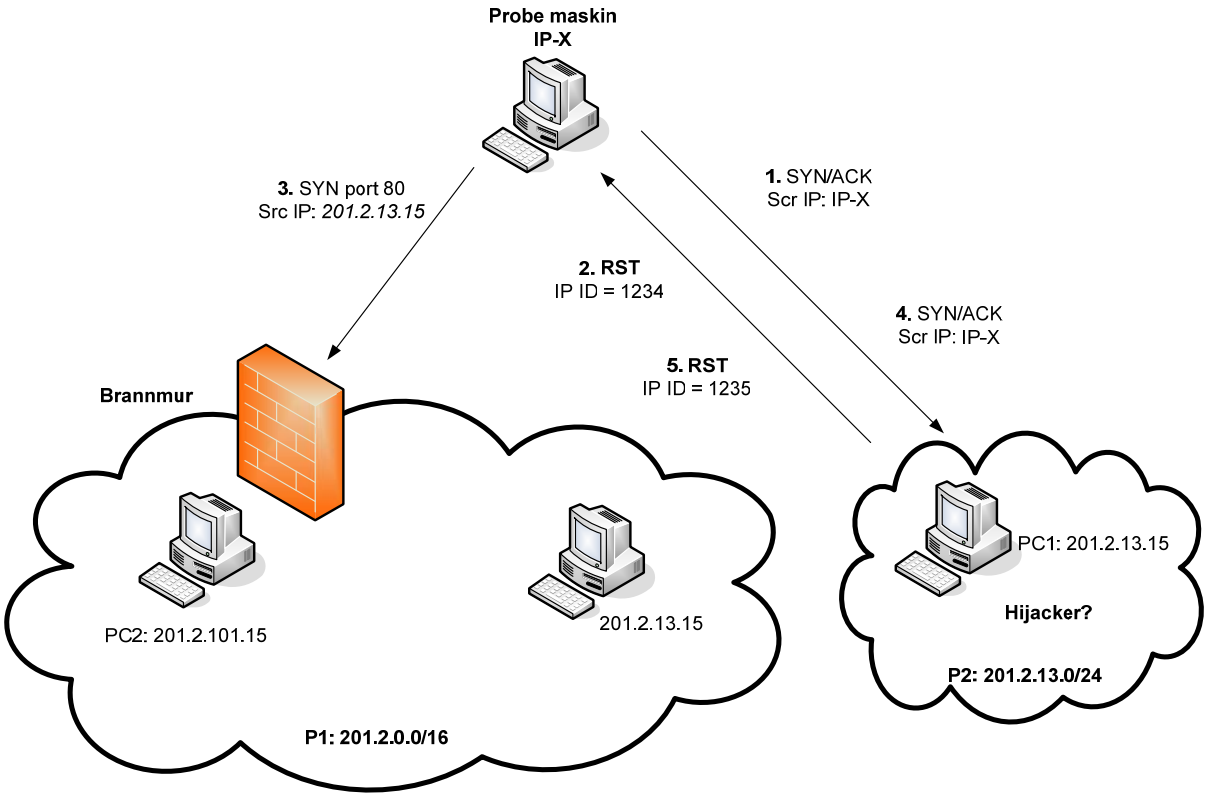
Så om det større nettverket P1 skulle ha en brannmur på nettverket sitt, vil denne testen konkludere med at P2 er hijacket, på grunn av at pakken som burde gå fra PC2 til PC1 ikke kommer fram.



Figur 24 Reflect Scan uten hijack, med brannmur

Reflect Scan med IP-Hijacking og brannmur

Et eksempel på dette kan vi se på Figur 25. For oss som sitter på probe maskinen vil det se ut akkurat likedan som på Figur 24, selv om situasjonen er en helt annen. Her vil vi da konkludere ”riktig”, at det P2 som er hijacket. Men ut fra dette kan vi si at Reflect Scan som metode gir falske positiver, når en brannmur stopper vår spoofede pakke.



Figur 25 Reflect Scan med hijack og brannmur

Bruk av Reflect Scan til detektering av IP-Hijacking

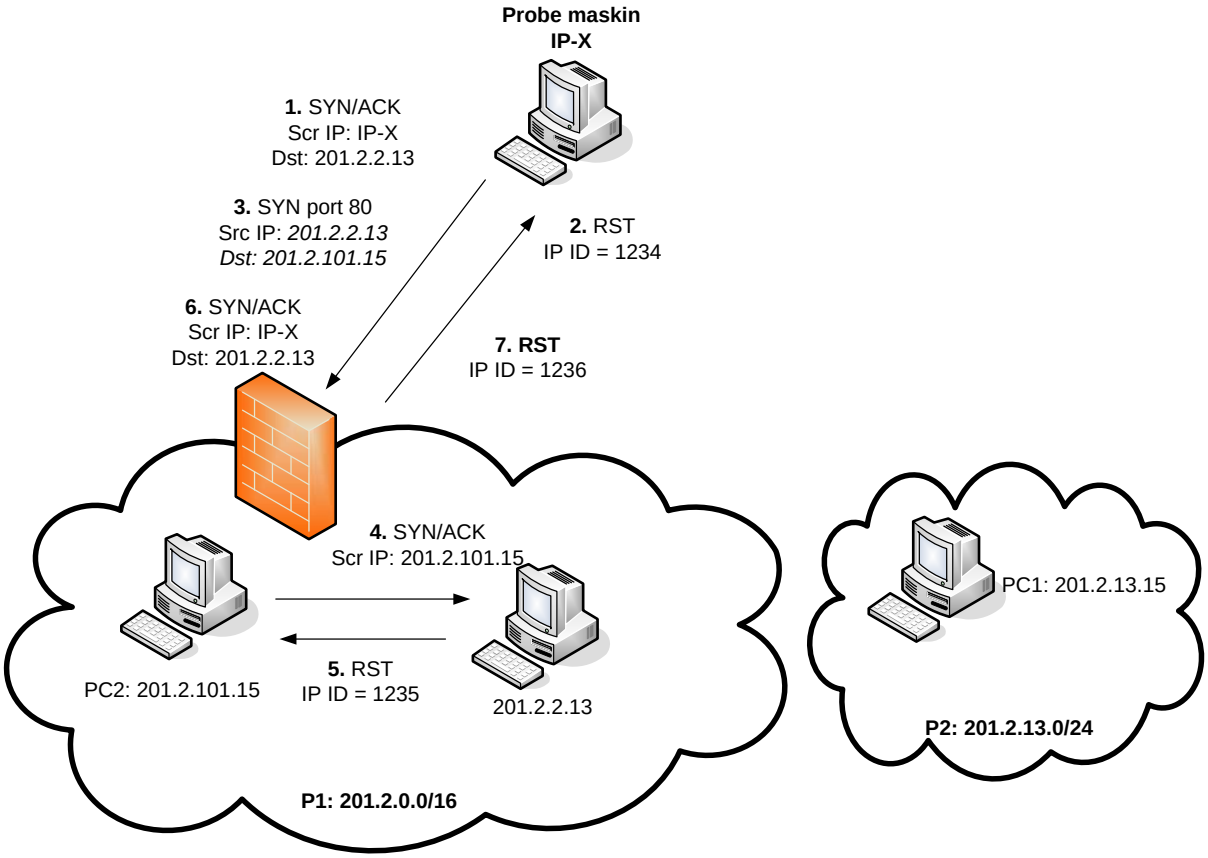
I dag er brannmurer vanlig å ha på store og små nettverk. De aller fleste brannmurer vil stoppe IP-pakker med falsk avsender adresse, og i hvert fall når den falske avsender IP-adressen tilhører det interne nettverket til brannmuren. Og med tanke på at vi vil i de tilfellene der brannmuren stopper vår spoofede pakke i punkt (3), vil dette ende med at vi konkluderer med at en IP-hijacking har blitt foretatt, selv om dette ikke er tilfelle. Slike falske positiver er ikke ønskelig i forbindelse med etterforskning, så derfor foreslår jeg en forbedring av Reflect Scan som metode, se lengre ned.

Det eneste Reflect Scan metoden kan gjøre er å bekrefte at et subnett ikke er hijacket. Men denne testen er også litt svak, for det kan hende at PC1 svarer på andre pakker enn de vi sender ut. Da vil vi konkludere feil med at P2 ikke er hijacket, selv om den er det. Disse falske negativene kan forbedres ved å kjøre testen flere ganger etter hverandre. Dette kan ikke fjerne muligheten for falske negativer, men gjøre sannsynligheten for at de oppstår mindre.

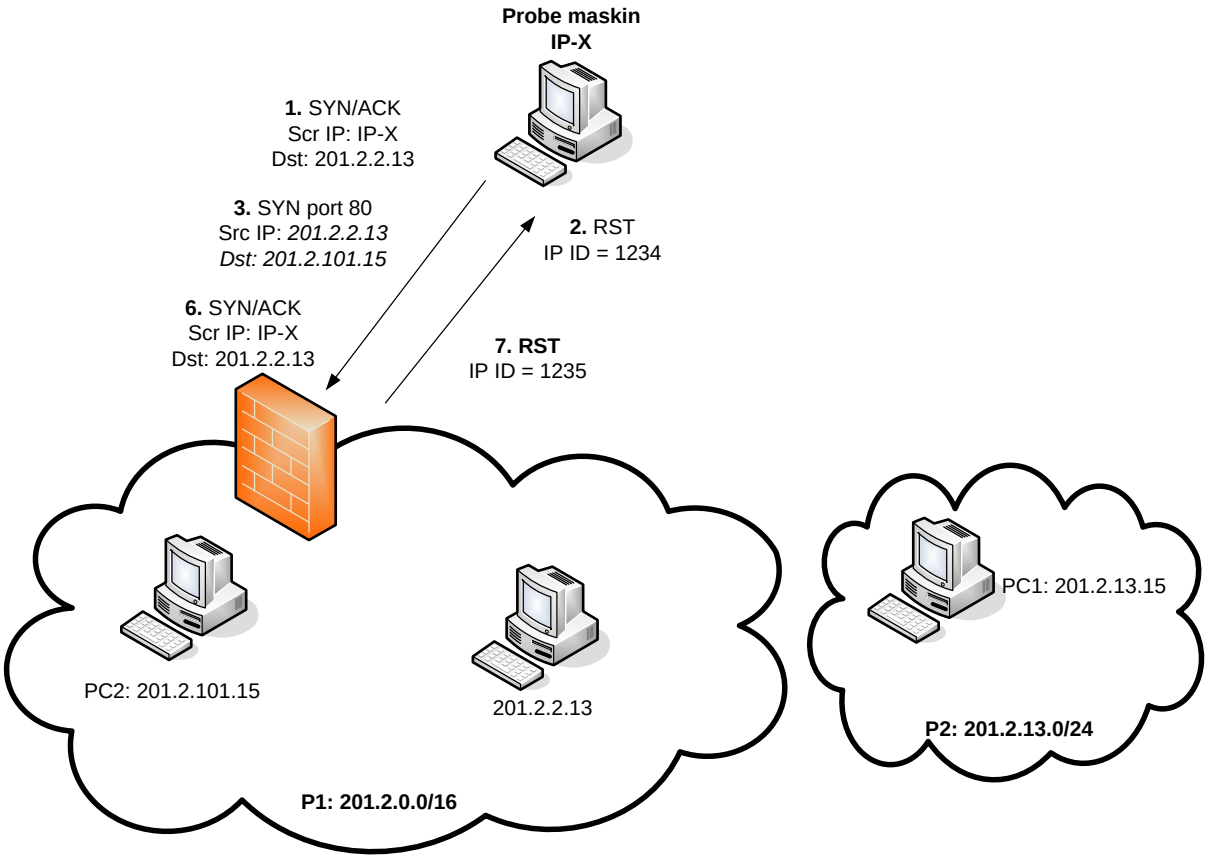
En forbedring av Reflect Scan som metode

Om vi kunne detektert om en eventuell brannmur i prefikset P1 stopper mine spoofede pakker, så ville vi kunne bruke denne testen til mer enn å bare kunne frikjenne en eventuell IP-hijacking. Da ville det være mulig å bruke reflect scan også til å bekrefte at en IP-hijacking har funnet sted. Jeg velger å løse dette ved å legge inn en test som skal detektere om det er en brannmur som stopper spoofede pakker. Figur 26 viser et eksempel der brannmuren ikke stoppet den spoofede pakken og Figur 27 for hvordan testen går når brannmuren stopper den spoofede pakken. Tar her og finner 1 ”idle” PC i nettverket P1, som ikke er del av subnettet P2. Bruker så denne som testpunkt mot å teste om brannmuren stopper mine spoofede pakker. Og vi ser på ID-ID feltet på de to mottatte pakkene for å konkludere om en brannmur har

stoppet den spoofede pakken. Denne teknikken er helt lik Reflect Scan, men her kan altså ikke en IP-hijacking påvirke resultatet og derfor kan vi bruke testen til å se om en brannmur stopper pakken min.



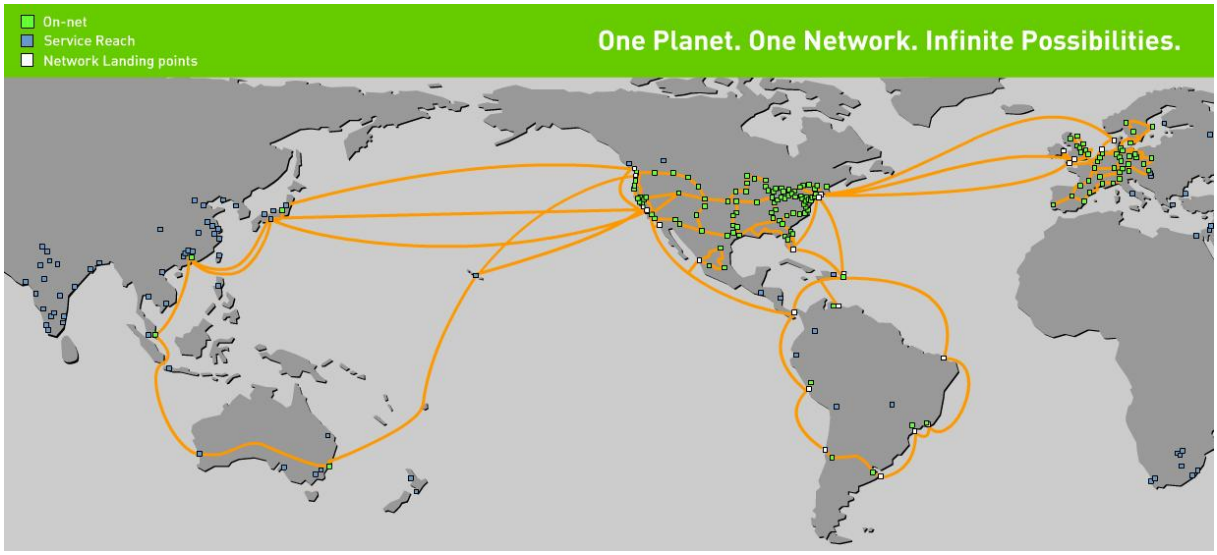
Figur 26 Deteksjon av brannmur, brannmuren slipper pakkene igjennom.



Figur 27 Deteksjon av brannmur, der brannmur stopper testen.

4.3.3 Geografisk posisjonering

Man kan bruke en geografisk posisjonerings test, spesielt på subnett hijacking, for der rutes alle pakker til samme nett. Dette muliggjør bruk at metoden Espen A. Fossen og André Årnes forsket på i ” Forensic Geolocation of Internet Addresses using Network Measurements” [15]. På ren prefiks hijacking er ikke denne metoden egnet, med tanke på at pakker fra ulike lokasjoner kan bli rutet til forkjellige verter. Men til subnett hijacking kan denne metoden brukes for å se om subnettet vi mistenker er hijacket, ligger geografisk nære resten av prefikset. Man må være oppmerksom på at flere AS er spred over hele verden, som for eksempel AS3549, se også Figur 28. Om vi skulle mistenkte AS3549 å være subnett hijacket, og kjørt en geografisk posisjonerings test, så ville vi sikkert konkludert med at det var hijacket, selv om det ikke var tilfelle.



Figur 28 AS3549 Global Crossing koblet til 922 andre AS [34].

4.3.4 Logisk posisjonering i nettverket

Det er flere måter man kan detektere en IP-hijacking ved å se på de forkjellige rutene IP-pakkene vil ta for å nå målet sitt. For eksempel er det meget mistenkelig om du sitter i Trondheim, og skal nå www.vg.no i Oslo og pakkene dine går via en ruter i USA. Jeg skal nå ta for meg flere kjente metoder for å detektere unormal ruting ved å se på hvor pakken går logisk i internett.

Lixin Gao

Lixin Gao er ikke en metode, men en person som har gjort viktig arbeid rundt analysering av relasjonene mellom autonome systemer på internett, og har skrevet ”On Inferring Autonomous System Relationships in the Internet” publisert i 2000 [27]. Han presenterer også i dette arbeidet en algoritme for å detektere unormal ruting (Vedlegg).

Customer-Provider Check

Denne sjekken ble introdusert av Xin Hu og Z. Moerly Mao i ”Accurate Identification of IP Hijacking” [03], som bygger videre på arbeidet til Lixin Gao [27]. Den tar utgangspunkt i to antagelser, den første er at en nettverksleverandør ikke vil IP-hijacke en kunde sitt nettverk. Den andre er at en kunde ikke kan hijacke IPer til andre kunder av deres egen nettverksleverandør. Dette sammen med at vi vet at internett rutingen er såkalt ”walley free” [27], det vil si at rutingen ikke først kan gå fra leverandør -> kunde, for så å gå kunde -> leverandør igjen.

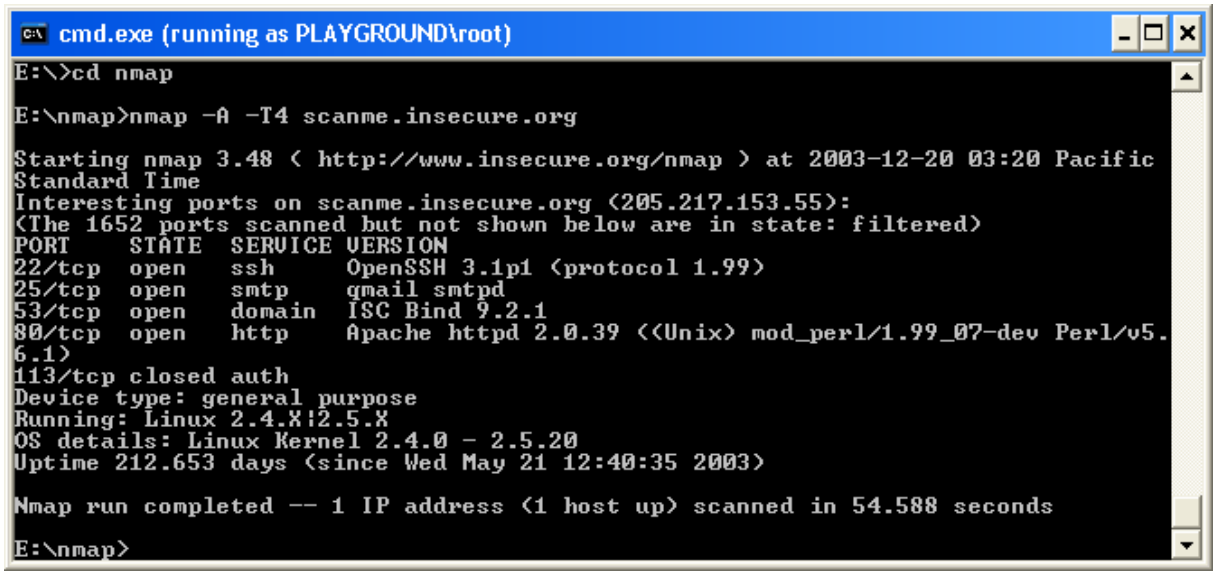
Denne sjekken er rimelig god, men tar ikke høyde for legitime MOAS ved bruk av flere leverandører som på Figur 11 og Figur 12.

4.3.5 Elektronisk fingeravtrykk

Et elektronisk fingeravtrykk er å kunne skille en PC fra en annen ved å se på informasjonen vi kan få ved å kontakte den via nettverket, og analysere svarene. Det er her mulig å bruke flere programmer med tilnærmet lik funksjonalitet, men jeg tar har valgt å beskrive det mest kjente programmet NMAP.

NMAP

Network Mapper (NMAP) [26]. Dette er et program utviklet av Gordon Lyon (Fydor) [23], som blant annet kan skanne porter på en spesiell IP, eller et prefiks. Her kan man få informasjon om hvilke porter som er åpne og hvilke servicer som kjøres på disse, oppetid, systemtid og lignende (Figur 29). Slik at man kan i ettertid sammenligne NMAP skanningen fra flere lokasjoner og se om "fingeravtrykket" / portskanningen har samme resultat fra alle disse lokasjonene. Dette krever et distribuert system i dette tilfelle fordi man må ha fingeravtrykk av samme IP-adresse fra flere lokasjoner i internett.



Figur 29 NMAP portskanning [25]

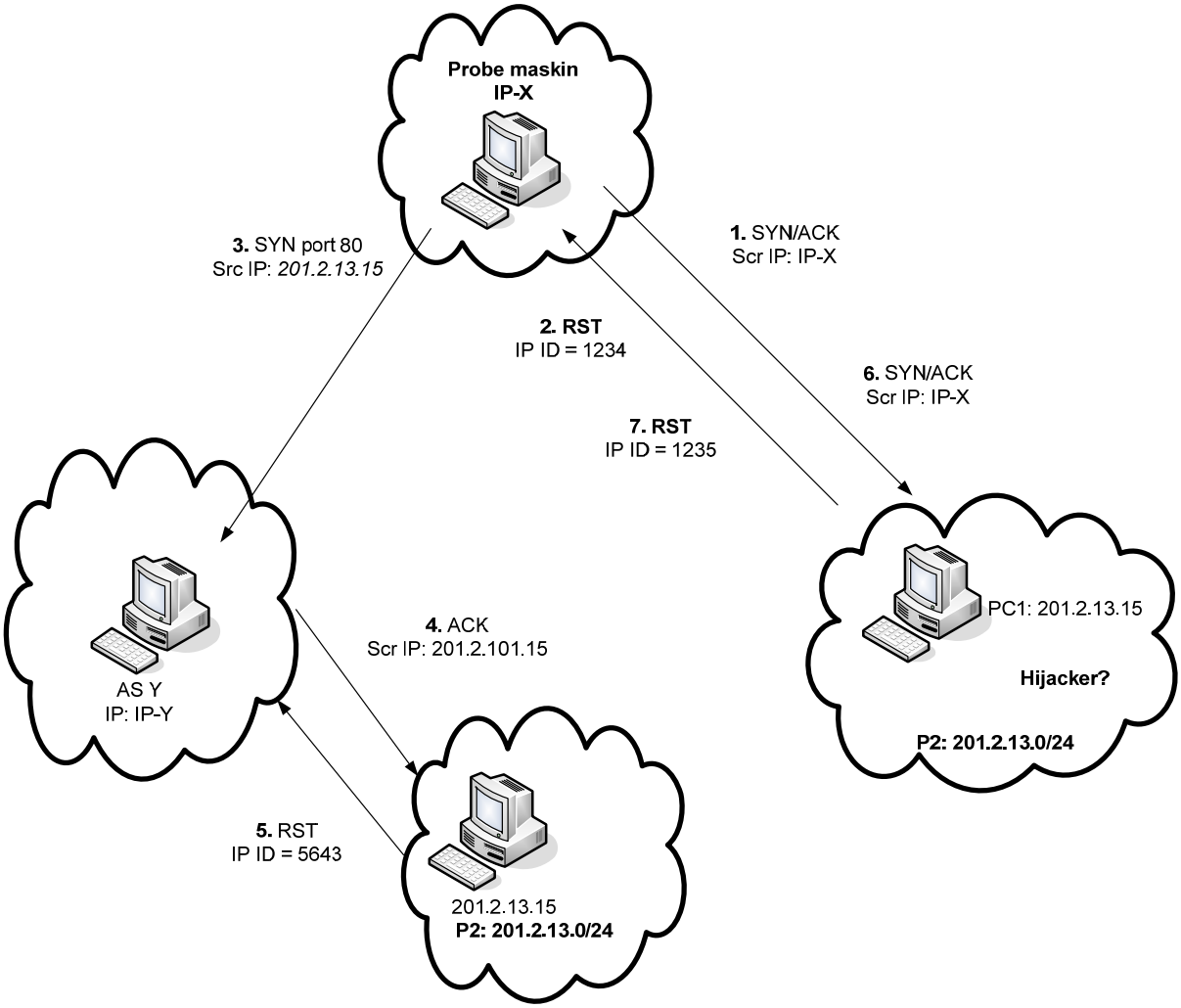
Sertifikater

Sertifikater er mulig det beste elektroniske fingeravtrykket vi kan finne. Om en server bruker sertifikater, kan vi kontrollere dette mot rotsertifikatet til leverandøren av sertifikatet. Og dermed kunne bekrefte om det virkelig er riktig server vi har kontakt med. Dette krever også en form for distribuert system, slik at vi kan se om sertifikatet er ekte fra alle lokasjoner på internett. Et eksempel på slike sertifikater er X.509 [29].

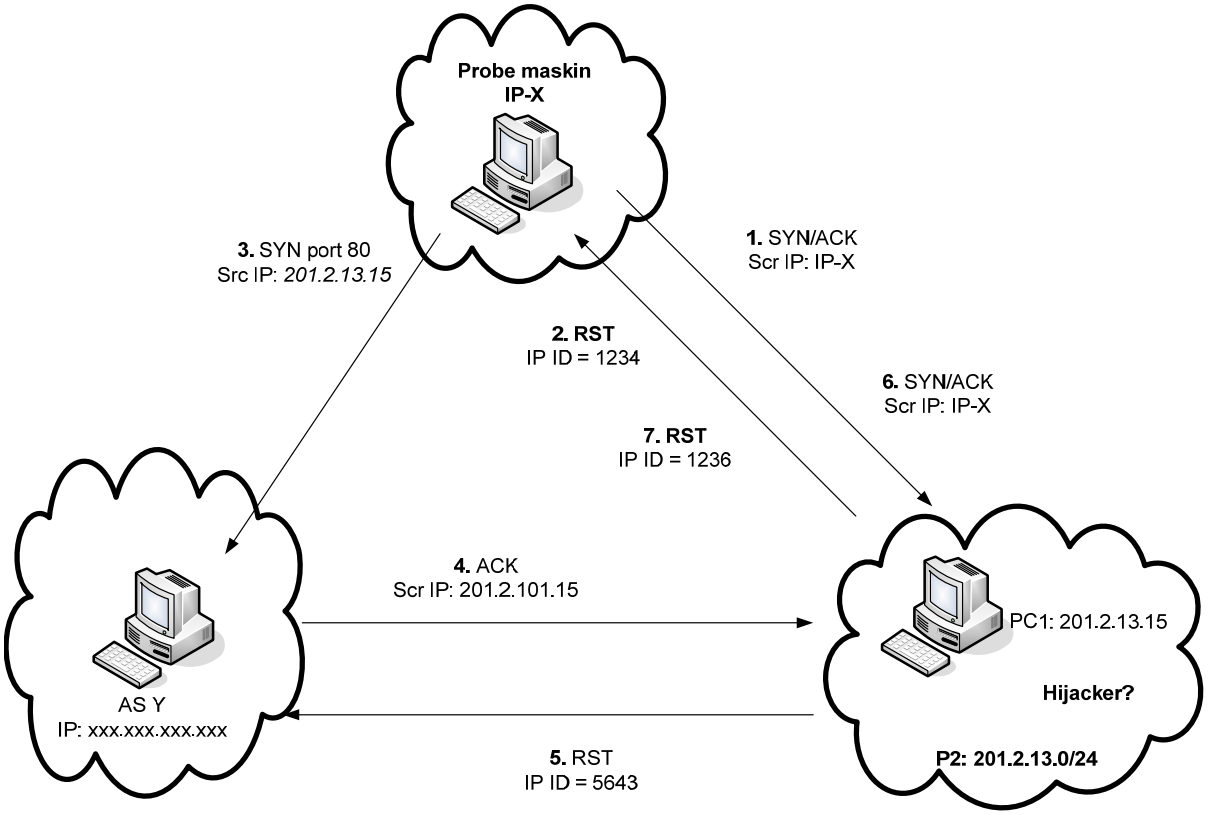
4.3.6 Reflect Scan for prefiks hijacking

Her tenker jeg da ikke på subnett hijacking, for der vil vanlig Reflect Scan ta sin del av dette. Men på hijacking av hele prefiks foreslår jeg at vi kan bruke tilsvarende metode, med en liten vri. Vi tar da utgangspunkt i at vi i vår probelokasjon alltid blir rutet enten til hijacket nett eller til den ekte eieren av nettverket. Vi vet også at ved prefiks hijacking vil nettet bli splittet i to, som Figur 15 viser. Det vi nå skal bruke testen til er å se om andre nettverk på internett ruter trafikk til samme nettverk som vi gjør fra vår lokasjon. Vi trenger derfor en åpen port på flere andre nettverk for å klare å kjøre denne testen. I eksempelet mitt nå vil jeg bruke en webserver i dette andre nettverket (AS Y) for å teste om AS Y ruter pakker til samme nettverk som det jeg gjør fra min probe lokasjon, Figur 30 viser hvordan testen går når P2 er hijacket og Figur 31 viser et eksempel der prefikset P2 ikke er hijacket.
Vi får i denne testen kun testet om et annet AS ruter trafikken til samme vert som jeg når fra min probelokasjon. Så om vi skal være sikker på at et prefiks ikke er hijacket teste dette via alle AS og det kan ta godt over 24 timer å kjøre. Man kan dog begrense tiden det tar å kjøre testen hvis man velger strategisk hvilke AS man prober via.

Vi slipper i denne testen å tenkte på om en brannmur stopper pakken vår, som i den vanlige reflect scan, på grunn av at den vil se ut som en helt normal pakke for alle brannmurer.



Figur 30 Reflect Scan på prefiks, med hijacking



Figur 31 Reflect Scan prefiks, uten hijacking

4.3.7 Oppsummering/diskusjon

Jeg skal nå se litt mer spesifikt hva som er svakheter og styrker ved de enkelte deteksjonsmetodene for IP-hijacking. Og hva som kreves for å kjøre de enkelte testene.

MOAS og Sub-MOAS konflikter

MOAS og Sub-MOAS er enkelt å detektere ved å se på rutingen fra forskjellige lokasjoner på internett. Vi vet også at MOAS og Sub-MOAS kan oppstå i legitime tilfeller. Jeg har dog ikke funnet noen eksakt statistikk på prosentandelen av MOAS og Sub-MOAS tilfeller som faktisk er IP-hijacking. Derfor er det vanskelig å si noe om hvor sannsynlig det er at en IP-hijacking som er foretatt i når vi har oppdaget en MOAS konflikt. På denne måten kan man si at MOAS og Sub-MOAS gir falske positiver med tanke på at det likevel ikke er sikkert at en IP-hijacking har blitt utført. Det er også som vi har sett mulig å foreta en IP-hijacking uten å skape en MOAS eller Sub-MOAS konflikt, og derfor vil en ren test på MOAS også gi oss falske negativer. Så man kan si at denne testen bare er med og klassifiserer type IP-hijacking, den er altså ikke tilstrekkelig til å bekrefte eller avkrefte en IP-hijacking.

Elektroniske fingeravtrykk

Jeg har tidligere nevnt programmet NMAP [26] til bruk for å lage et elektronisk fingeravtrykk av en IP-adresse fra flere lokasjoner på internett. Denne typen test kan bare brukes på

hijacking av hele prefiks, for ved subnett vil hijacking bare bli rutet til hijackeren fra hele internett som Figur 21 viser. Dette vil kreve et distribuert system. Det er her viktig at vi velger en IP-adresse ut fra det hijackede nettet som vi kan få mest mulig info ut fra, desto mer informasjon vi får, desto vanskeligere er det for hijackeren å forfalske en annen som har likt elektronisk fingeravtrykk. Et økende antall elektroniske fingeravtrykk vi tar fra forskjellige lokasjoner på internett vil redusere muligheten for falske negativer. En slik test har veldig liten sannsynlighet for falske positiver, men man skal være oppmerksom på at det kan forekomme ved IP-filter, lastbalansering, inntrengingsdeteksjons-systemer (IDS) og noen ganger ved overbelastning av IP-adressen du skanner.

Om nettet vi har en IDS, vil nok detektere en slik skanning som en portskanning. Og ut fra dette om mulig stoppe skanningen.

Jeg nevnte også bruk av sertifikater som et elektronisk fingeravtrykk. Om det er tjenester som benytter sertifikater på IP-adressen vi etterforsker, kan vi som nevnt laste disse ned og kontrollere de mot rotsertifikater. Dette er en kontroll metode som gir veldig få falske negativer, om du faktisk når det hijackede nettverket via noen av lokasjonene du laster ned sertifikatet fra. Dette krever også et distribuert system som ved NMAP, som jeg nevnte ovenfor. Denne testen vil også gi nesten ingen falske positiver, på grunn av at det er tilnærmet umulig å forfalske sertifikater.
Sertifikater som test er meget bra om man har mange lokasjoner på internett man kan kontrollere sertifikatet på, men det er nok mange tilfeller av IP-hijacking der det ikke finnes en server/IP-adresse som har sertifikat i det aktuelle nettet. Det som også er positivt med denne testen er at den også kan brukes på hijacking av subnett, fordi vi kontrollerer sertifikatet mot rotsertifikatet.

Geografisk posisjonering
Geografisk posisjonering som en deteksjonsmetode for IP-hijacking, blir mest aktuelt opp mot subnett hijacking der man kan se om det subnettet man mistenker for å være hijacket ligger geografisk i nærheten av resten av prefikset den tilhører, her kan man benytte metoden som Espen A. Fossen og André Årnes i ” Forensic Geolocation of Internet Addresses using Network Measurements” [15]. Ved bruk av en slik metode skal man være oppmerksom på at den vil gi falske negativer, om faktisk subnett-hijackingen rutes til et nett som er i nærheten av eieren av nettverket. Og falske positiver kan også oppstå i tilfeller der det er varierende mengde nettverkstrafikk.

Om man skal bruke samme geografisklokasjonstest for å avkrefte eller bekrefte en IP-hijacking av et helt prefiks, må man allerede vite fra hvilke lokasjoner man blir rutet til hijacker eller virkelig eier av nettet, ellers vil resultatet bli upålitelig. Og når man må vite fra hvilke posisjoner man kan kontrollere fra, så må man allerede bekrefte at det er en IP-hijacking. Men kan brukes som ekstra bevis i en sak om påtalemakten ønsker dette. Det vil da gi litt flere falske positiver og negativer enn ved subnett hijacking, på grunn av at man får begrensing på hvilke målepunkter man kan bruke når man skal finne den geografiske lokasjonen til de 2 nettverkene.

Logisk Posisjonering
På dette området er det allerede forsket mye [27], [28], [03]. Det å utføre slike tester kan detektere de fleste typer hijacking, men de vil i enkelte tilfeller ikke kunne skille mellom legitime MOAS og Sub-MOAS tilfeller og IP-hijacking. Slik at testen vil ha både falske positiver og negativer i slike tilfeller. Jeg mener derfor dette er metoder som må brukes i

samband med andre deteksjonsteknikker, for at det skal kunne brukes som bevis. Men også dataene fra slike tester vil også være verdifull i en etterforskning for å få en bedre forståelse av hva som har hendt. For å gjennomføre en slik test må man ha tilgang til fulle rutingstabeller fra flere rutere som er lokalisert på forskjellige lokasjoner på internett.

Reflect Scan (Subnett)
Reflect Scan som test kan i noen tilfeller gi falske positiver. Men om man legger inn min forbedring av metoden ved å legge inn en brannmursjekk, skal det i teorien begrense antall falske negativer til et minimum. Denne testen kan også gi falske negativer, ved at det er annen trafikk som påvirker vår ”idle” vert. Det kan reduseres ved å kjøre testen flere ganger etter hverandre. Begrensingen i denne metoden er jo det å finne en ”idle” vert i nettverket, som også er av typen som øker IPID for hver pakke den sender ut (Printere, Windows, eldre Linux maskiner, FreeBSD, og Mac OS [25]). Og sannsynligheten for at ingen andre kontakter denne verten samtidig som du kjører testen blir mindre og mindre desto lengre du velger å kjøre den. Brannmurtesten vil også kreve at du finner en ”idle” vert til i prefikset du mistenker å ha blitt utsatt for subnett hijacking. Jeg mener at dette er en av de bedre testene å kjøre mot et subnett hijack om du ikke har tilgang til sertifikater, men likevel finner ”idle” verter.

Inntrengingsdeteksjons systemer (IDS) kan også detektere en reflect scan som en ”zombie” skanning [25]. Og ut fra dette stoppe våre probepakker. Men ettersom vi kjører først denne brannmurtesten vil sannsynligheten for at en IDS stopper vår test senere være mindre.

Reflect Scan (prefiks)
Her er det som tidligere nevnt ikke nødvendig med en brannmurtest først, fordi alle pakker som blir sendt til målet vi ønsker å probe ikke har avsender adresser som er innenfor nettverket. Det betyr ikke at en IDS ikke kan detektere dette som en zombie skanning, men sannsynligheten er mindre enn ved reflect scan for subnett hijacking. Her bruker vi da andre maskiner til å probe vår vert, og sjekker på denne måten om vår vert svarer disse. Denne testen vil dog ta mye lengre tid enn Reflect Scan for subnett, fordi den prober via kun en annen vert, mens vi her bruker tilsvarende metode via mange andre verter, derfor vil tiden øke med antallet verter vi prober via. Vi får selvfølgelig her det samme problemet som ved subnett, at vi må finne en ”idle” vert som bruker økende IP ID.

Tracert
Dette er en test for å detektere en IP-hijacking langs vanlig rutingsvei. Denne type test skal i teorien være ganske god til å detektere slike typer hijacking, men mange rutere har blitt konfigurert slik at de ikke svarer på ICMP meldinger der TTL er lik 0. Og om en hijacker er ute etter et spesielt nett han ønsker å hijacke / lytte på trafikken til, så kan vi regne med at han også velger å skru av svar på ICMP pakker for å forhindre at han blir oppdaget. Dette i tillegg til at det allerede er en del rutere som har skrudd av denne funksjonaliteten, så kan vi heller ikke konkludere med at det er noe mistenkelig ved at vi ikke får svar. I følge AS-nivå traceroute [33], var det tilnærmet 50 % av alle traceroutene som endte uten å få svar fra siste vert.

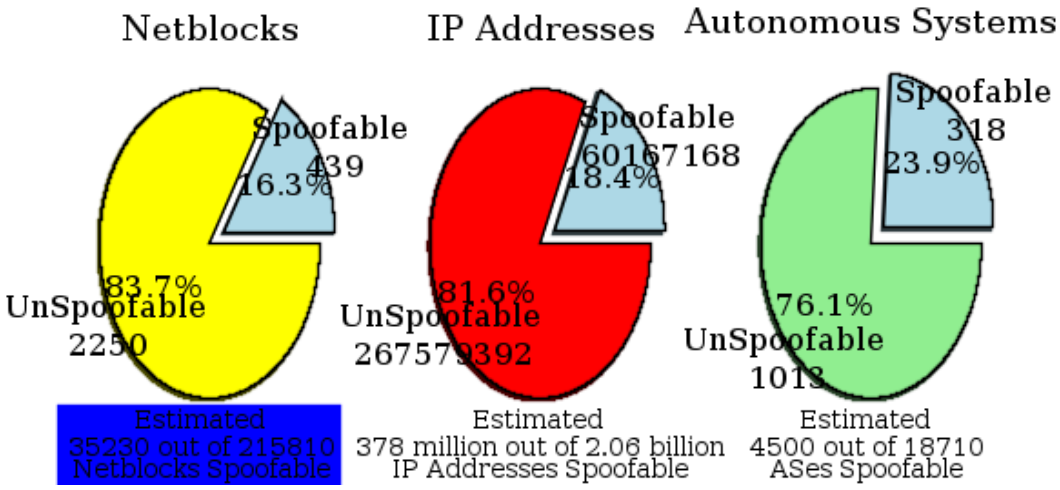
Inntrengingsdeteksjons-systemer (IDS)
Jeg har nevnt at flere av deteksjonsmekanismene jeg har tatt for meg kan detekteres av IDSer. Man skal være på vakt når man bruker svakheter i TCP for å kjøre en skanning av et nettverk, fordi dette kan sees på som misbruk / spionasje på andres nettverk. Om dette er direkte ulovlig er en annen sak, for vi tar opp svært lite ressurser på andre sine nettverk ved å kjøre slike tester, men det kan likevel ses på som et angrep på andre sine nettverk.

Type Hijack	Deteksjonsmekanismer
Hijack av et prefiks uten AS	MOAS konflikt El. Fingeravtrykk Geografisk posisjon Logisk posisjon Reflect Scan for prefiks hijacking
Hijack av et prefiks med AS	El. Fingeravtrykk Geografisk posisjon Logisk posisjon Reflect Scan for prefiks hijacking
Hijack av sub-prefiks uten AS	Sub-MOAS Geografisk posisjon Logisk posisjon Reflect Scan
Hijack av sub-prefiks med AS	Geografisk posisjon Logisk Posisjon Reflect Scan
Hijack langs vanlig rutingsvei	Tracert El. Fingeravtrykk Reflect Scan for prefiks hijacking

Tabell 1 Hijack og deteksjonsmekanismer

5 Prototyp og implementasjon

Mine forslag til forbedringer av eksisterende teknikker, og forslag til ny teknikk baserer seg på at man kan sende ut spoofede pakker (IP-pakker med falsk avsenderadresse). Og det viser seg at verken NTNU eller UNINETT i dag har nett, der dette er mulig. Noe som er forståelig med tanke på at denne svakheten ved TCP/IP har vært kjent i snart 20år [39]. Det må også nevnes at i henhold til et prosjekt ved Massachusetts Institute of Technology(MIT) er fortsatt 23 % av nettverkene på internett åpent for å sende spoofede pakker [32], se også Figur 32. Jeg derfor valgt i samarbeid med faglærer Svein Johan Knapskog å forklare hvordan jeg ville implementere mitt ikke-distribuerte system for å detektere IP-hijacking. Det vil ikke bli implementert som en prototyp fordi jeg ikke har tilgang til nettverk som lar meg utføre alle deteksjons testene. Og det ville være for tidkrevende å sette opp et nettverk som testene kan kjøres på.



Figur 32 Spoofer Project [32]

6 Kravspesifikasjon:

I dette kapittelet presenteres funksjonelle krav og kvalitetskrav, samt rammer og begrensninger for deteksjonssystemet som skal utvikles. Dette er et system for å detektere IP-hijacking. Systemet skal kjøres i Java [37], dette slik at den kan kjøres uavhengig av operativsystem. Det taes også utgangspunkt i at deteksjonssystemet skal kjøre på et nettverk som ikke har noen form for utgående filtrering, som ville stoppet spoofede pakker. Og med spoofede pakker mener jeg, IP-pakker med falsk avsender adresse.

Kravspesifikasjonen består av to hoveddeler.

- Funksjonelle krav – her en beskrivelse av de funksjonelle kravene som stilles til systemet, dette er kravene som går direkte på at systemet skal kunne detektere de forkjellige typer IP-hijacking.
- Rammer og begrensinger – her beskrives omgivelsene systemet skal kjøres i.

6.1 Funksjonelle krav

I denne delen av dokumentet presenteres de funksjonelle kravene for systemet. Systemets oppgave er å kontrollere om en spesifikk IP-adresse er kapret, og gjøre dette med lave falske negativer og lave falske positiver.

6.1.1 Overordnede krav

Jeg har valgt å definere to overordnede funksjonelle krav, som senere blir dekomponert til mer praktiske testbare krav. De to overordnede kravene er:

F1 Systemet skal detektere IP-hijacking.
Dette innebærer at systemet skal kunne detektere både prefiks hijacking og subnett hijacking. Hijacking via vanlig rutingsvei skal ikke systemet detektere.

F2 Systemet skal lagre all rådata på en slik måte at bevisets integritet beholdes.
Herunder kommer både utgående og innkommende nettverks trafikk, samt data fra rutingservere.

6.1.2 Dekomponerte krav

Med utgangspunkt i de overordnede kravene, har jeg utledet de dekomponerte kravene i Tabell 2 og Tabell 3. Disse kravene er definert slik at de skal være logisk testbare.

F1: Systemet skal detektere IP-hijacking	
F1.1	Systemet skal kunne detektere Sub-MOAS og MOAS konflikter.
F1.2	Systemet skal detektere Subnett Hijacking.
F1.3	Systemet skal detektere Prefiks Hijacking.

Tabell 2 F1 Funksjonelle krav

F2: Systemet skal lagre alle rådata på en slik måte at bevisets integritet beholdes.	
F2.1	Systemet skal lagre alle rådata.
F2.2	Systemet skal tidsstemple alle rådata.
F2.3	Systemet skal lage en hash til alle rådata som lagres. Hashen skal lagres sammen med rådataene.

Tabell 3 F2 Funksjonelle krav.

6.2 Rammer og begrensinger

Hensikten med dette kapittelet er å gi en oversikt over hvilke krav og begrensinger det er til omgivelsene til systemet. Dette vil være momenter som er viktig å ta hensyn til når jeg senere skal lage arkitekturen, og også for en fremtidig implementasjon av dette systemet.
Jeg vil dele dette kapittelet opp i underdelene nettverk, system og IP-hijack.

6.2.1 Nettverk

Det må stilles spesielle krav til nettverket systemet skal kjøres fra, det vil si hvordan vi er tilknyttet internett. Dette systemet baserer seg på at det er mulig å sende ut IP-pakker med falsk avsender adresse, for å kunne generere trafikk til og fra lokasjoner på internett som vi ikke har kontroll over.
Det er derfor et krav til nettverket om at det ikke finnes utgående filtrering av avsenderadresse på IP-pakkene mine. Da gjelder det også at selve maskinen systemet skal kjøre på har en ekte IP-adresse, og ikke ligger bak en NAT [35] ruter.

6.2.2 System

Her tenker jeg da spesielt på programmeringsspråk og operativsystem, altså generelle omgivelser til maskinen systemet skal kjøre på. Jeg velger her Java [37] som programmeringsspråk, både fordi det kan kjøres på de aller fleste operativsystemer og fordi det er det programmeringsspråket jeg har best kjennskap til.
Og når valget av Java som programmeringsspråk er det også klart at Java ikke støtter utsending av pakker med falsk avsenderadresse (spoofede pakker). Så vi må utvide Java med et bibliotek som støtter dette. Valget mitt falt på Jpcap ”Java package for packet capture” [36].

Og for Windows systemer krever Jpcap at det installeres Winpcap [38] for full funksjonalitet, dette kreves dog ikke på andre Linux baserte systemer.

6.2.3 IP-hijack

Denne deteksjonsmetoden skal bare fungere i de tilfeller der det er to eller flere parter som annonserer samme IP-adresser av internett. Det finnes jo også i enkelte tilfeller som for eksempel AS30040 og AS10741 [12], der noen andre har hijacket adressene til et selskap som er nedlagt. Da er det kun en ruter som annonserer disse adressene, og min deteksjonsmetode vil ikke se på dette som en IP-hijacking, selv om det ikke er eieren som annonserer de.

7 System for deteksjon av IP-hijacking

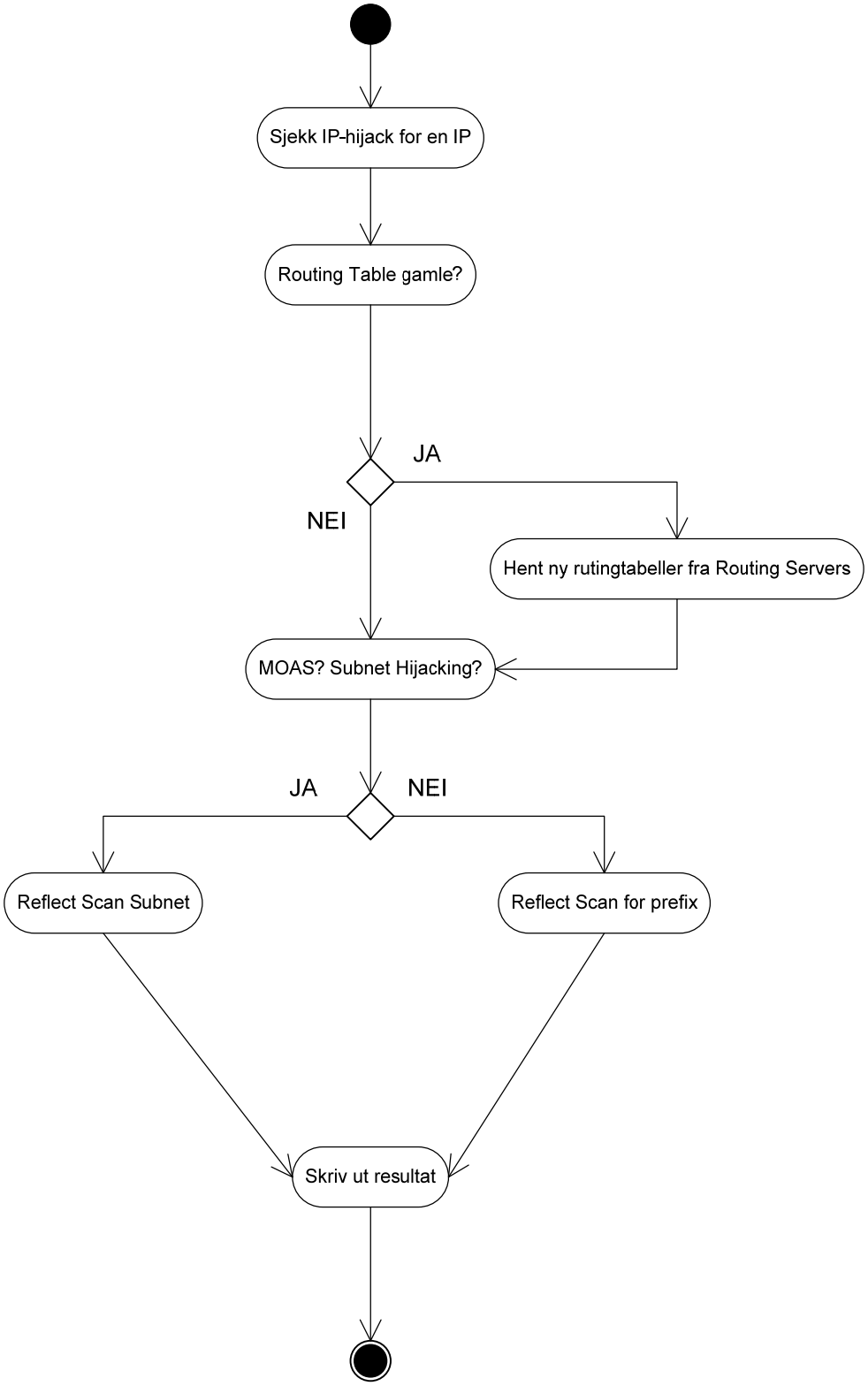
I denne delen skal jeg forklare hvordan jeg vil bygge opp et system for deteksjon av IP-hijacking med utgangspunkt i kravspesifikasjonen. Og jeg skal komme med detaljerte løsninger på oppgavene systemet skal utføre.

7.1 Introduksjon

Det helt klart beste er et distribuert system for å detektere IP-hijacking. Men på grunn av at dette også krever at man har avtaler med mange ulike ISP'er i flere ulike lokasjoner og land, er dette noe som jeg velger bort. Dette fordi en del av testene man kjører med et distribuert system som for eksempel port-skanning/elektronisk fingeravtrykk kan sees på som et "angrep". Og derfor er det ønskelig å få avtaler på plass med alle ISP'ene man må leie nett og maskiner av.
Dette er noe jeg bevisst har valgt og ikke bruke tid på i min masteroppgave. Jeg vil heller legge vekt på å få en bedre oversikt over hvilke deteksjonsmekanismer som er tilgjengelige i dag og hvordan disse fungerer. Jeg har også et ønske om å vise hvordan metodene jeg har foreslått burde implementeres.

7.2 Overordnet

Systemet skal ta inn en IP-adresse som vist på Figur 33, som den senere skal analysere om er IP-hijacket eller ikke. Systemet vil så sjekke om hvilke treff denne IP-adressen har i runtingstabellene systemet har lastet ned fra ruting serverne.
Her kan vi sjekke om det er MOAS konflikter, og om IP-adressen har treff i flere linjer i rutingstabellene, noe som kan tyde på et mulig subnett hijack.
Ut fra dette vil systemet bestemme om det skal kjøres en Reflect Scan for subnett, eller for prefiks. Disse testene vil så til slutt konkludere at IP-adressen er del av et nett som har blitt hijacket, eller ikke.
Jeg vil nå ta for meg de enkelte delene av dette overordende systemet.

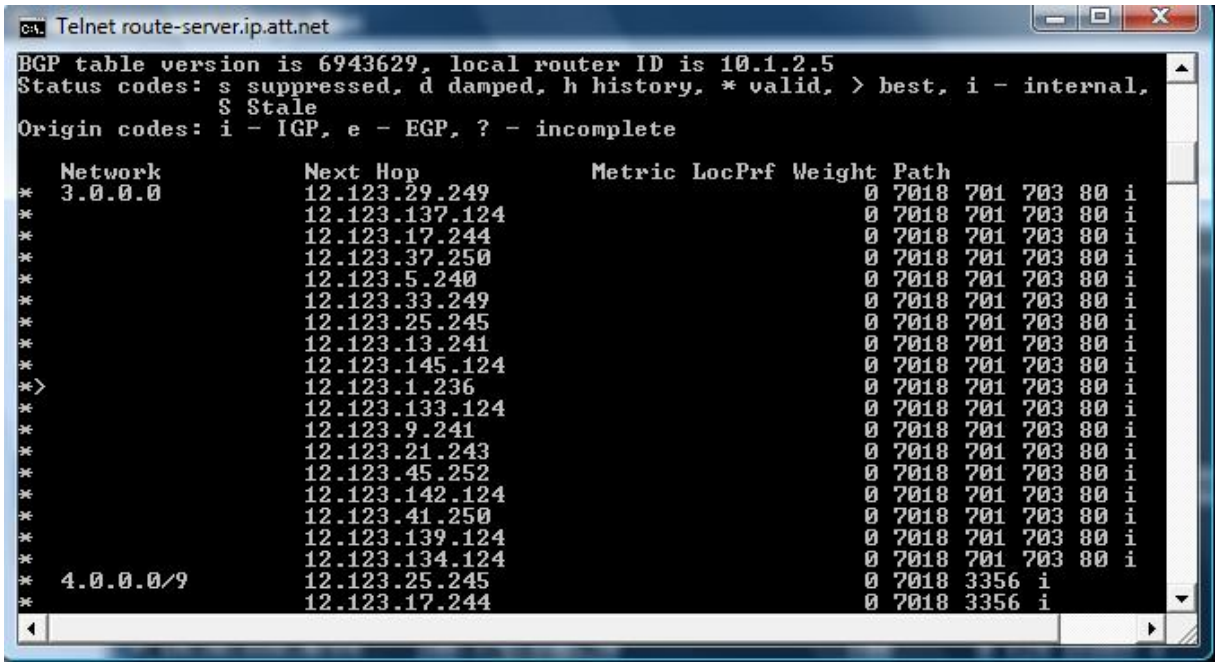


Figur 33 Overordnet UML aktivites diagram

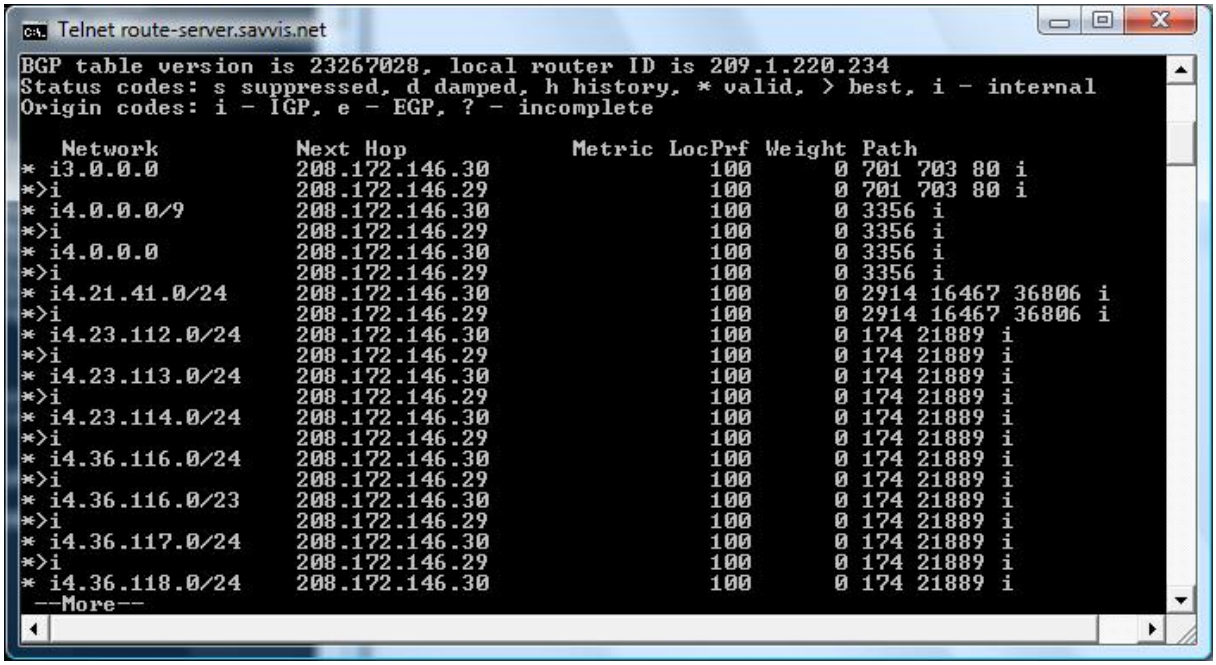
7.2.1 Ruteservere

Ruteservere (Routing Servers) er systemets måte å få tak i routingtabeller fra forskjellige lokasjoner på internett. Ruteservere er nok en av de enkleste måtene å få tak i denne informasjonen. Man kan også samle mer informasjon via lookingglass servere, men disse er som regel kun tilgjengelige via HTTP protokollen, og gir svar i html som kan ha svært forskjellig koding. Det er derfor ønskelig å bruke ruteservere som har mer like svar. Man skal

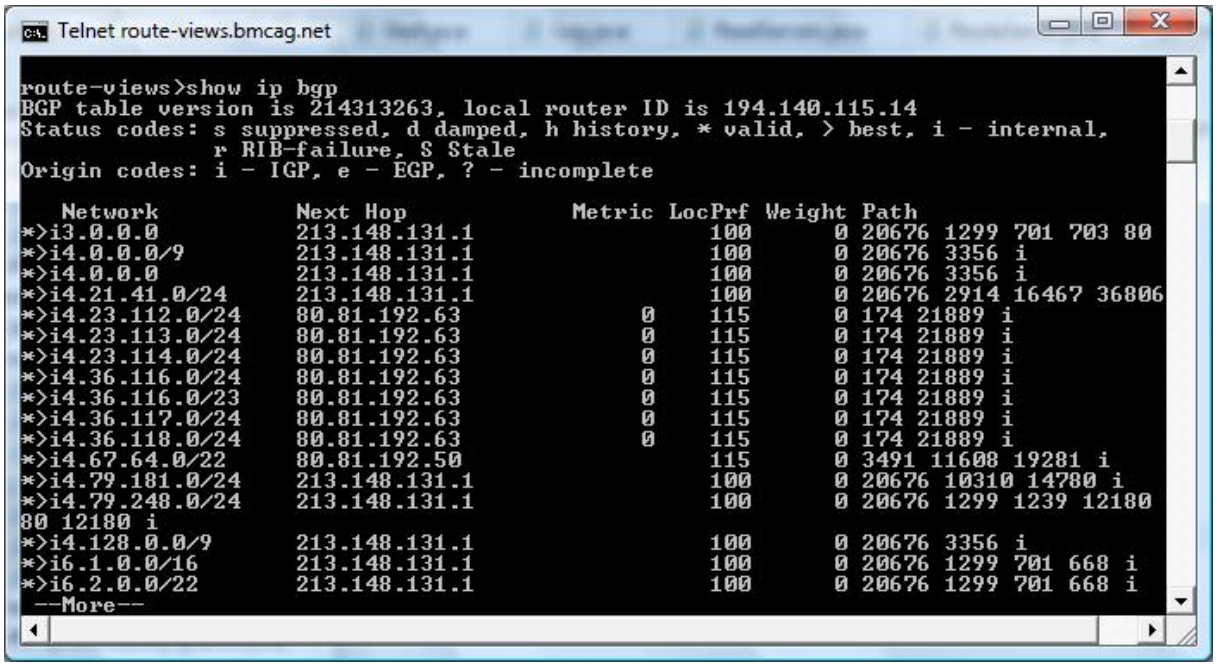
være klar over at ruteservere kan også ha ulike svar på samme kommando, dette bestemmes av både konfigurasjon og softwareversjon, ruteserveren kjører. Figur 34, Figur 35 og Figur 36 viser tre forskjellige ruteservere som svarer på samme kommando ”*show ip bgp*”.



Figur 34 route-server.ip.att.net



Figur 35 ruote-server.savvis.net



Figur 36 route-views.bmcag.net

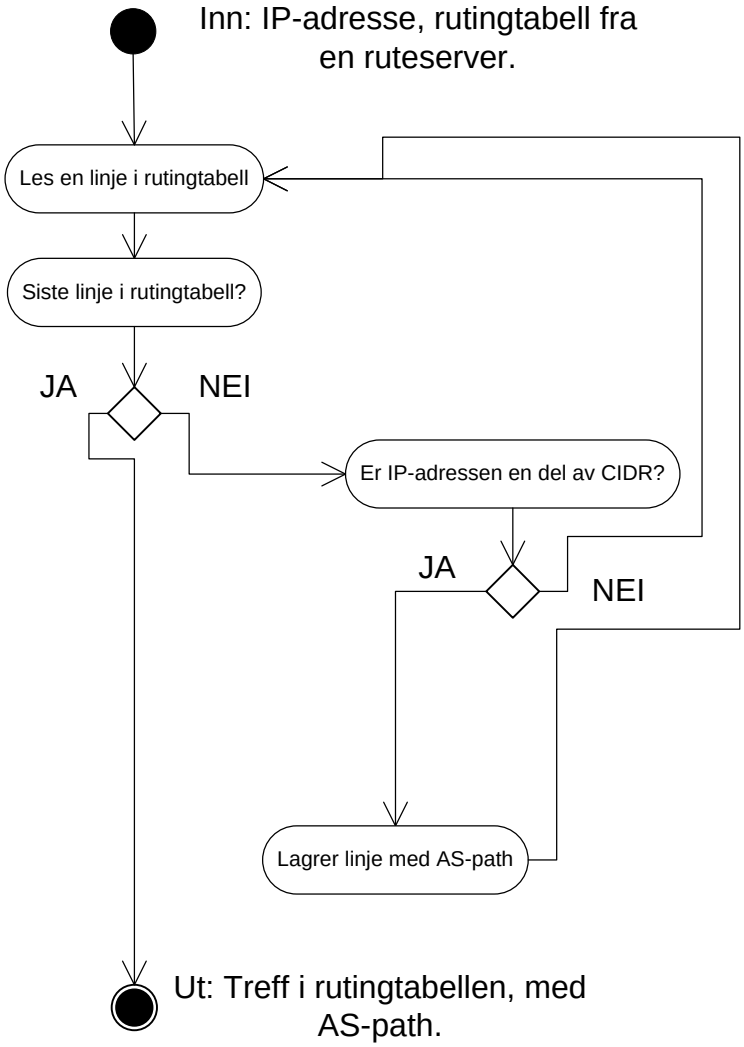
Når man skal koble seg til disse ruteserverne via telnet, må man også være klar over at enkelte servere krever brukernavn og passord, men som regel står dette i velkomstteksten på ruteserveren. Derfor må implementasjonen ha mulighet for dette. Jeg vil derfor ha en egen fil med de ruteserverne vi ønsker å bruke, på formen:

#servernavn(dns) brukernavn passord
tpr-route-server.saix.net saix saix

Det er mye data man skal motta via denne telnet tilkoblingen, i gjennomsnitt rundt 20Mbyte, men noen servere gir opp mot 80MByte i svar på kommandoen ”show ip bgp”. Dette kan derfor ta en del tid, og det er ønskelig å kjøre nedlastingen av rutingtabellene parallelt. Jeg kjørte derfor alle telnet tilkoblingene i tråder som kjørte samtidig.
Det er også en del av våre rådata som må lagres med hash og tidsstempling.

7.2.2 Analyse av rutingtabellene

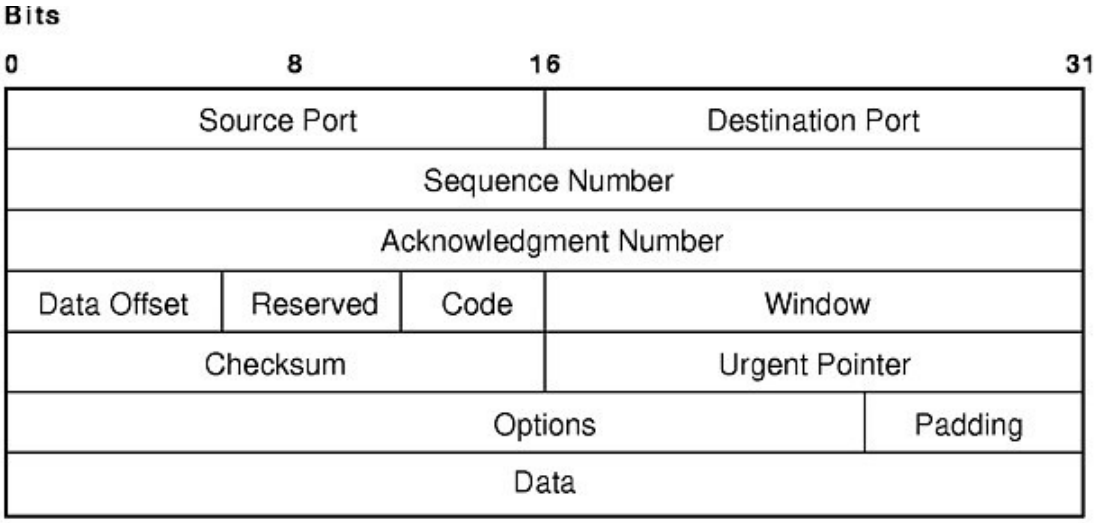
Da er vi over på å hente ut relevant informasjon av rutingtabellene som vi lastet ned fra ruteserverne. Med tanke på at ruteserverne svarer litt forskjellig, vil det lønne seg å implementere Java ”Regular Expressions”. Dette for lettere å kunne luke ut ønskelig informasjon fra hver linje i rutingtabellen.
Vi ønsker følgende informasjon: treff i hvilken CIDR nettverk, og AS path/vei til dette CIDR nettverket. Med denne informasjonen kan vi detektere MOAS og Sub-MOAS, samt se om dette er en mulig prefiks eller subnett hijacking.
Figur 37 viser hvordan analysen av en rutingtabell vil gjennomføres. Etter dette sammenligner vi resultatene fra alle rutingtabellene, for å se om vi har ulike CIDR treff, og om AS nummeret er likt i alle resultatene. Systemet vil så bruke dette til å velge og sette igjen en av to mulige tester, Reflect Scan for subnett eller for prefiks.



Figur 37 UML aktivitetsdiagram for analyse av en rutingtabell.

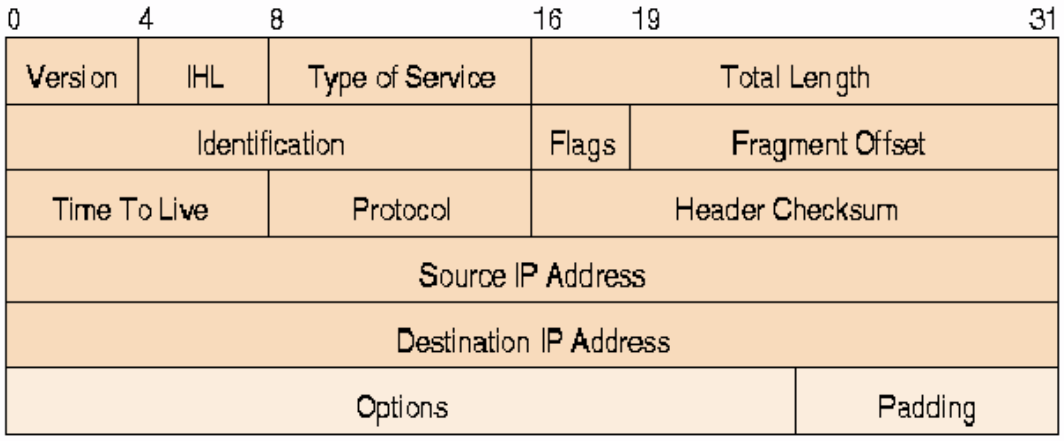
7.2.3 Nettverk

Før vi går inn på selve Reflect Scan og portskanningen må vi sette noen parametere på IP-pakken, TCP-pakken og ethernet pakken.
Vi starter med TCP-pakken, se Figur 38. Her er det viktig å få satt riktig Destination Port. Destination Port må settes til den porten vi ønsker å nå på den verten vi sender pakker til. Ved Source Port er det faktisk ikke så nøye hva vi setter til ved vårt bruk, fordi Jpcap sin JpcapCaptor tar imot alle pakker ethernet-grensesnittet mottar, om porten er åpen eller ikke har ikke noe å si. Men vi kan sette Source Port til for eksempel 2007. Slik kan vi lettere skille pakker som vi mottar fra hverandre som svar på utsendte pakker, og fra andre pakker vi vil motta.
Ettersom vi ikke skal holde en TCP kobling oppe, trenger vi heller ikke tenkte så mye på hva squence number og acknowledgment number skal settes til. Disse kan settes til en tilfeldig verdi. I Code feltet kan vi sette hva slags type TCP pakke det er, dette deteksjonssystemet kommer bare til å sende ut pakker av typen SYN og SYN/ACK, så SYN bittet må settes til 1, og på SYN/ACK pakken må også ACK bittet settes til 1. Checksum settes automatisk av Jpcap. Vindustørrelsen settes til 10, det gjør vi også med Urgent Pointer. Og for at datafeltet ikke skal være tomt kan vi fylle det med byte-sekvensen til en liten streng med tekst, for eksempel ”data”. Da er TCP-pakken klar for å pakkes inn i en IP-pakke.



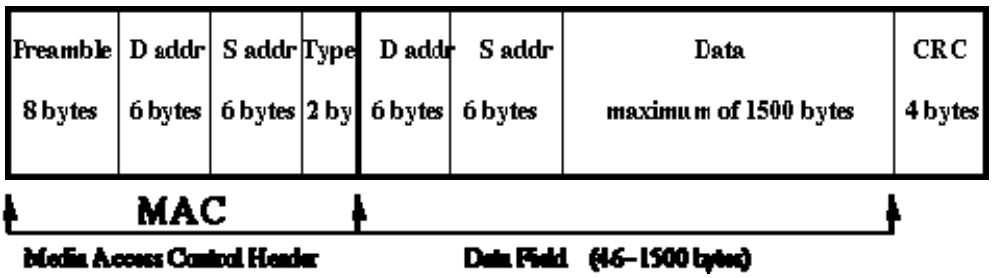
Figur 38 TCP-pakke

I IP-pakken setter vi IP-adressen til avsender og mottaker, se Figur 39. Når vi sender pakker via Jpcap kan vi sette både Source IP og Destination IP til de verdier vi ønsker selv. Dette påvirkes ikke av din lokale IP-adresse. I denne pakken er det viktigste å sette Protocol til TCP. Og TTL kan vi sette til 255 (maks).
Identification kan vi selv velge å sette på pakkene vi sender ut, det har ingen betydning. Men dette feltet skal vi bruke når vi mottar pakker senere i Reflect Scan. Nå har vi satt de mest nødvendige feltene, og det er på tide å pakke IP-pakken inn i en Ethernet pakke.



Figur 39 IP-pakke

Ethernet pakken kan vi se på Figur 40. På vårt eget lokalnett vil pakken rutes via MAC-adresser. Derfor må disse settes for at vi kan nå riktige verter på vårt lokalnett. I dette systemet kommer vi ikke til å sende pakker til eget lokalnett, bortsett fra default gateway. Derfor kan vi sette D addr til MAC-adressen til vår egen default gateway. Og S addr kan settes til MAC-adressen til nettverkskortet på maskinen systemet kjøres på. Når dette er gjort vil pakken være klar til å sendes ut på lokalnettet.



Figur 40 Ethernet pakke.

7.2.4 Portskanning

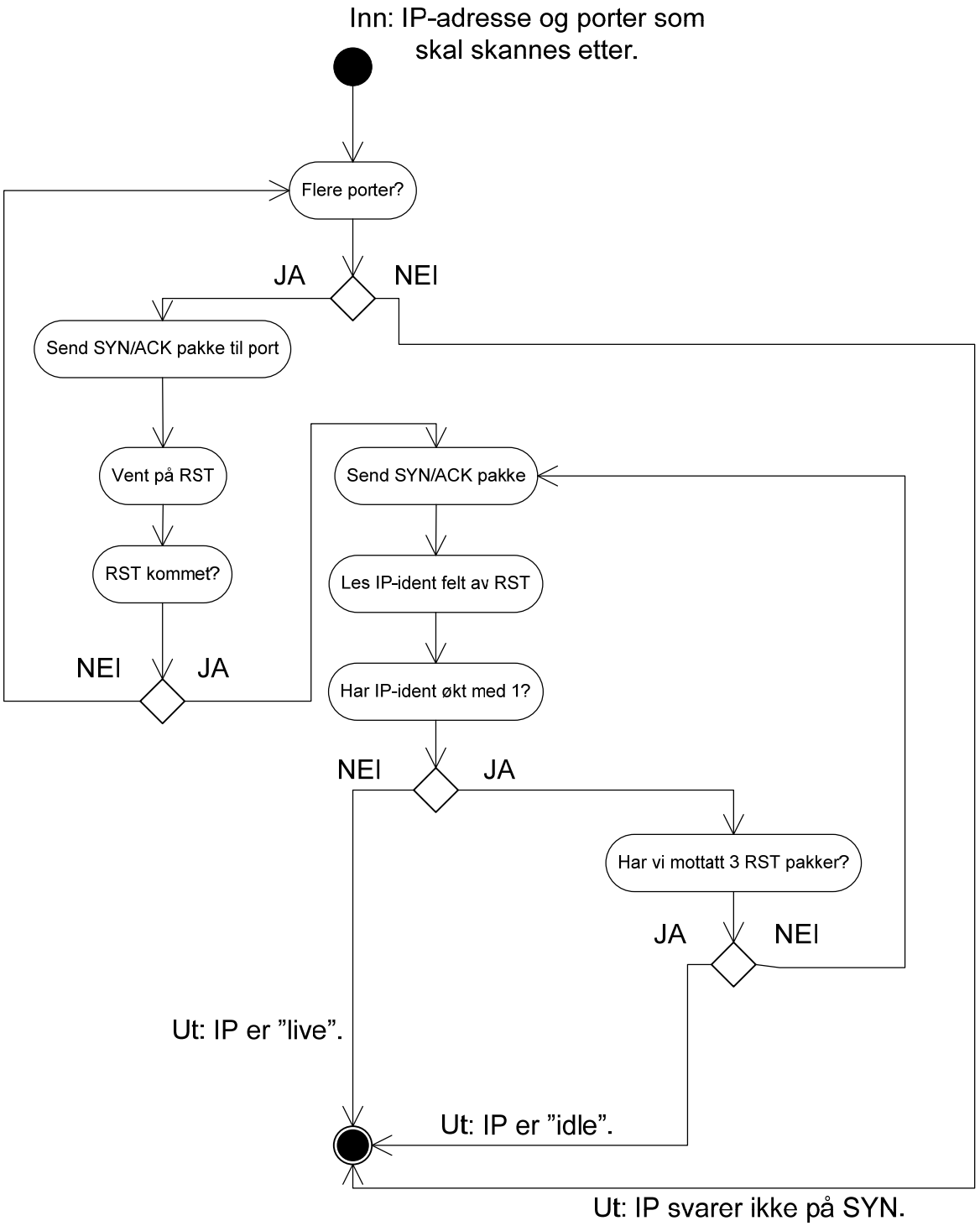
Før vi går inn på de to variantene av Reflect Scan, må vi først skanne nettverket for ”idle” og ”live” verter. En ”idle” vert er en som svarer på utsendte SYN/ACK-pakker med RST-pakker, der IP-Identification feltet øker med ên for hver pakke vi mottar.
En ”live” vert svarer på våre SYN/ACK-pakker , men har enten fast IP-Identification, eller den øker med mer enn ên for hver pakke vi mottar, noe som kan tyde på at andre også bruker nettverket på denne verten.
Figur 41 viser fire forskjellige typer svar fra ulike type servere. Vi kan her se at www.vg.no har fast IP-Identification felt på sine pakker, mens verden.pvv.ntnu.no har økende IP-Identification, men det er mer aktivitet på nettverkskortet til verden.pvv.ntnu.no enn det vi har generert. Disse vil da bli kategorisert som ”live” verter i systemet. Mens kofod.kofod-petersen.org serveren svarer med IP-Identification som øker med ên for hver pakke vi mottar, kategoriseres i systemet som en ”idle” vert.

www.vg.no	kofod.kofod-petersen.org
193.69.165.21 ident: 27061	129.241.210.49 ident: 64877
193.69.165.21 ident: 27061	129.241.210.49 ident: 64878
193.69.165.21 ident: 27061	129.241.210.49 ident: 64879

starbuck.nvg.ntnu.no	verden.pvv.ntnu.no
129.241.210.75 ident: 0	129.241.210.224 ident: 19524
129.241.210.75 ident: 0	129.241.210.224 ident: 19584
129.241.210.75 ident: 0	129.241.210.224 ident: 19606

Figur 41 IP-Identification

Selve portskanningen går ut på å sende ut en SYN/ACK-pakke til en valgt port, deretter vente for å se om man får en RST-pakke i svar. Får man ikke svar, så er ikke porten hos mottaker åpen for TCP tilkoblinger.
Figur 42 viser hvordan portskanningen skal gå fram. I dette systemet er vi ikke interessert i å finne alle åpne porter på en maskin, men har vi funnet bare en port som svarer på våre SYN/ACK-pakker har vi det vi er ute etter. Så vil vi sende flere SYN/ACK-pakker til denne porten for så å analysere IP-Identificationfeltet og klassifisere den som en ”idle” eller ”live” vert. For ikke å sende ut unødvendig mange pakker, kan man begrense portskanningen til kun mest brukte porter.

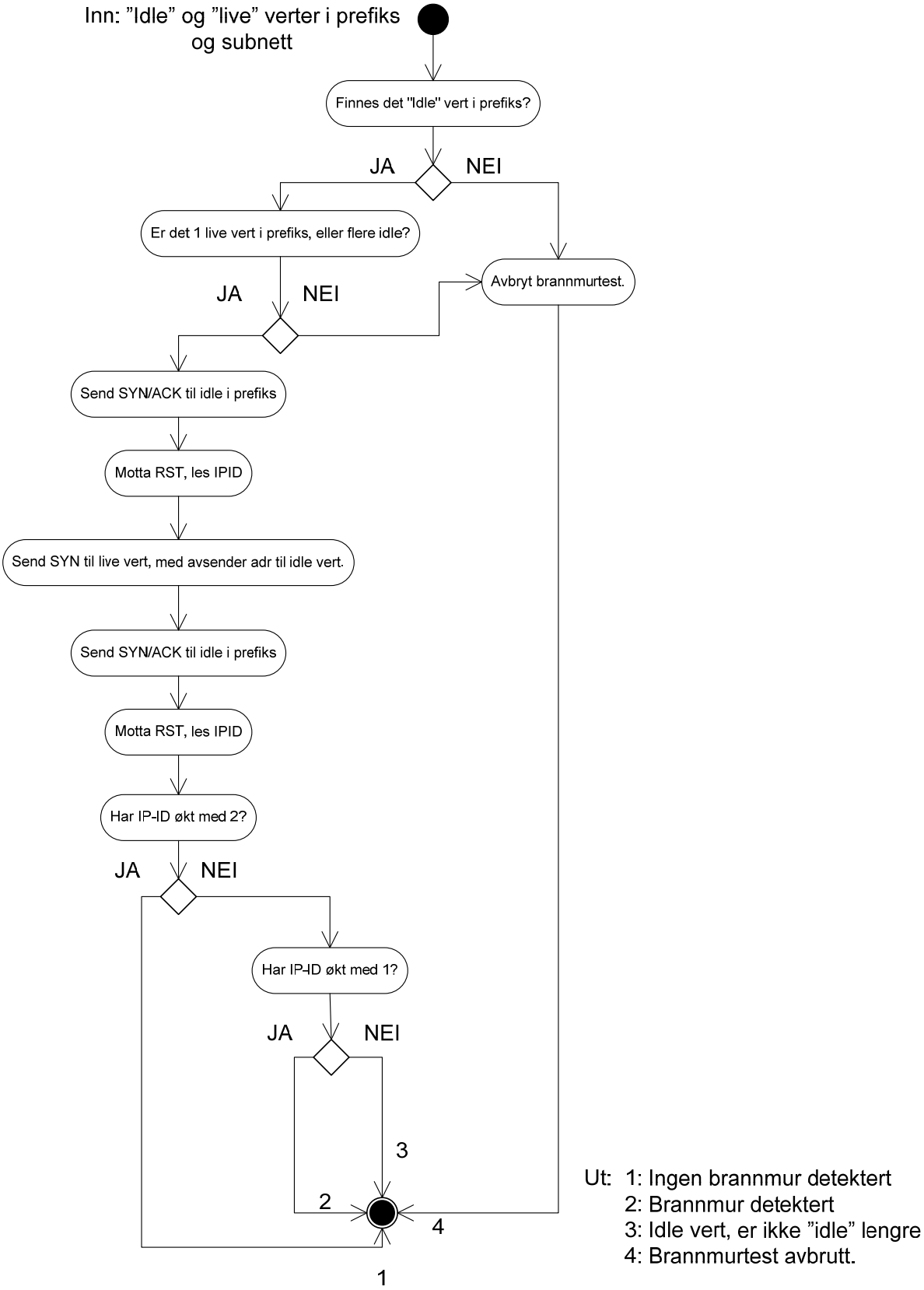


Figur 42 UML aktivitetsdiagram portskanning.

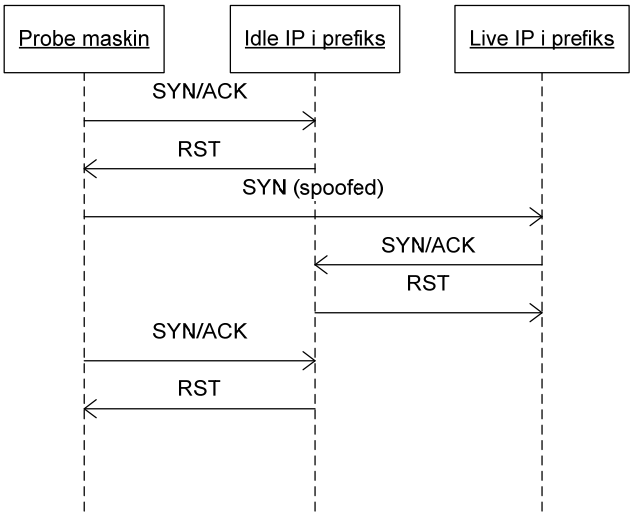
7.2.5 Reflect Scan Subnett

Vi bestemmer å kjøre en Reflect Scan for subnett, etter at vi har sett på hvilke treff IP-adressen hadde i rutingtabellene tidligere. Etter at vi har gjennomført portskanningen av nettet, er vi klare for å kjøre en Reflect Scan.

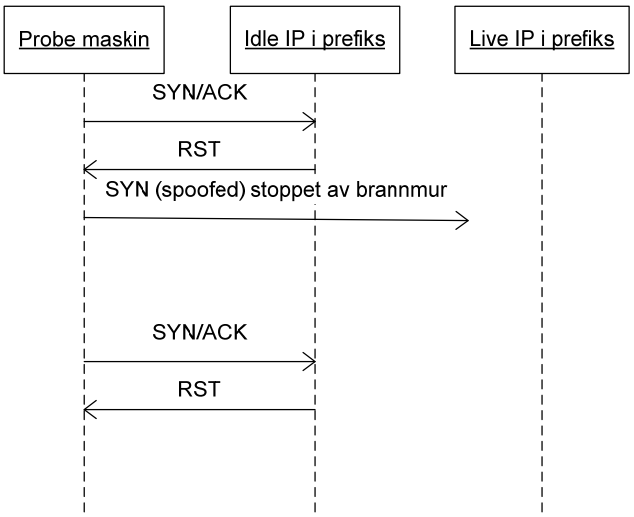
Vi har nå en liste over ”idle” og ”live” verter i nettet vi skal skanne. Vi trenger nå å skille disse ”idle” og ”live” vertene inn i ”idle, kun subnett”, ”idle, kun prefiks”, ”live, kun prefiks” og ”live, kun subnett”.
Det første vi starter med er en brannmurtest, se Figur 43. For å kjøre denne testen er vi avhengige av minimum en ”idle” vert i prefikset, og en vert til, som enten er ”idle” eller ”live”. Vi sender så SYN/ACK-pakke til den ”idle” verten i prefikset, for å lese IP-Identification på RST-pakken vi mottar. Deretter sender vi en spoofed SYN-pakke til den ”live” verten med avsenderadresse til den ”idle”.
Dette vil generere en SYN/ACK og en RST pakke som Figur 44 viser. Om derimot en brannmur stopper den spoofede pakken vil det gå som Figur 45 viser.
Vi sender så en ny SYN/ACK-pakke til den ”idle” verten, og undersøker IP-Identifaication feltet på RST på pakken vi mottar. Har den økt med kun en, har en brannmur stoppet vår spoofede pakke, og resten av Reflect Scan metoden kan avbrytes. Har den økt med 2 kan vi konkludere med at vår spoofede-pakke ikke har blitt stoppet, og vi kan nå kjøre en Reflect Scan. Om IP-Identification feltet har økt med mer enn to kan vi konkludere med at den ikke er ”idle” lengre, og brannmurtesten kan ikke kjøres mot denne IPen.



Figur 43 UML aktivitetsdiagram for brannmurtest

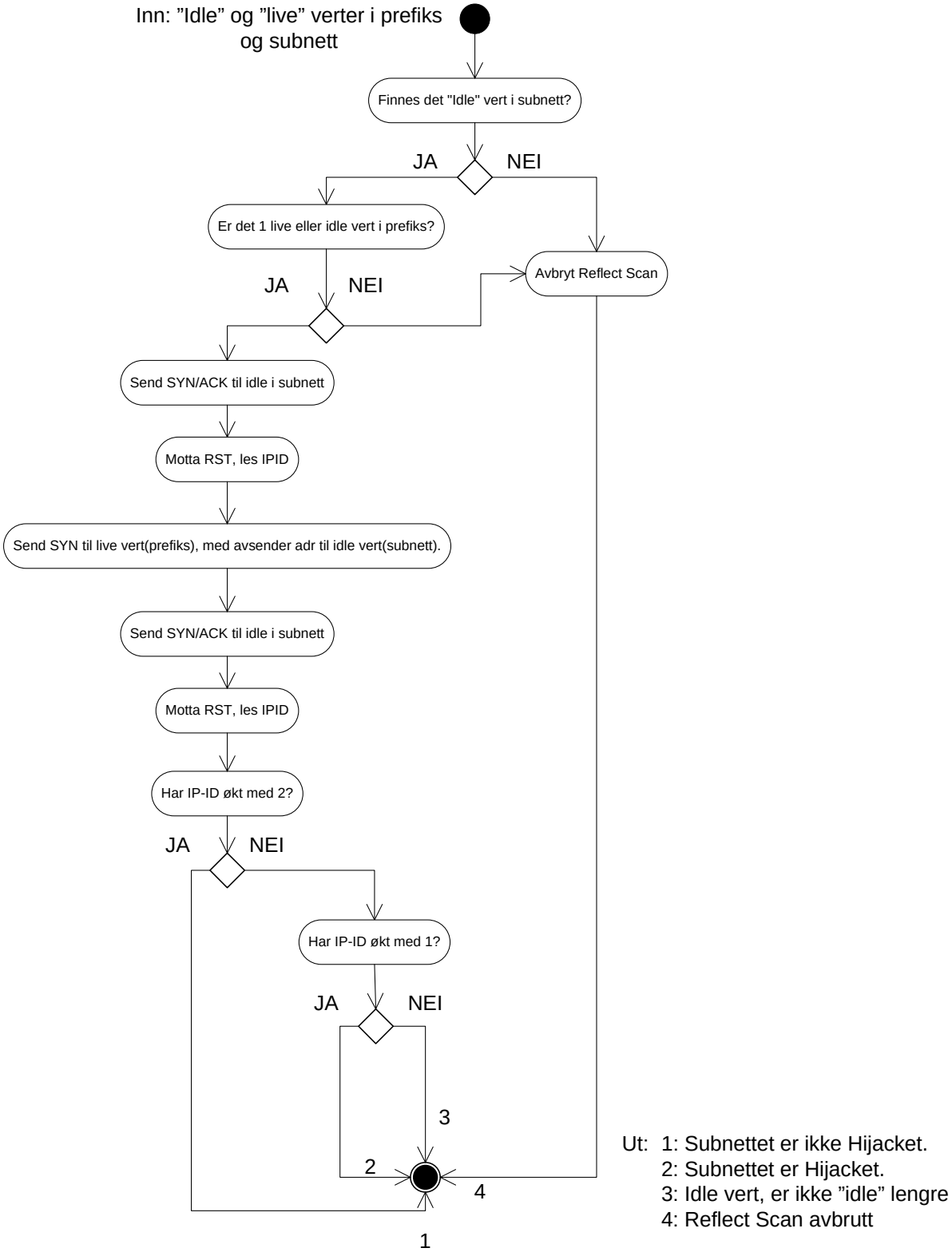


Figur 44 UML sekvensdiagram uten brannmur.

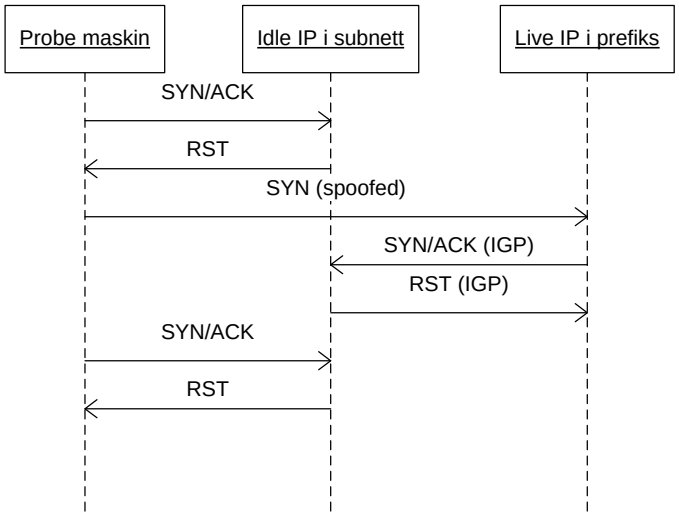


Figur 45 UML sekvensdiagram med brannmur.

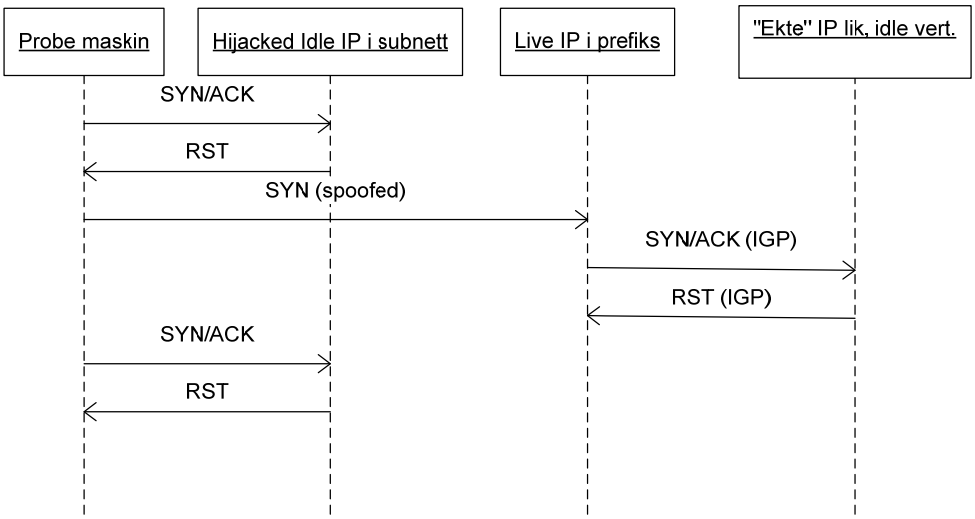
Vi går nå videre med Reflect Scan for subnett hijacking, som den er beskrevet i [03]. Her er det nesten samme prosess som ved brannmurtesten, se Figur 46.
Vi er i denne testen avhengig av at vi har funnet en ”idle” vert i subnettet vi frykter er hijacket, og en ”idle” eller ”live” vert i prefikset. Men her ser vi om svaret på vår spoofede SYN-pakke, som blir rutet via IGP, når samme ”idle” vert som det vi gjør når vi sender pakker på internett. Figur 47 viser sekvensdiagram for en slik skanning når subnettet ikke er hijacket. Og Figur 48 viser sekvensdiagram for når subnettet er hijacket. På denne måten kan vi da se på IP-Identification feltet til de to mottatte RST-pakkene og konkludere om det er en IP-hijacking eller ikke.



Figur 46 UML aktivitetsdiagram Reflect Scan subnett



Figur 47 UML sekvensdiagram for Reflect Scan subnett uten Hijack.

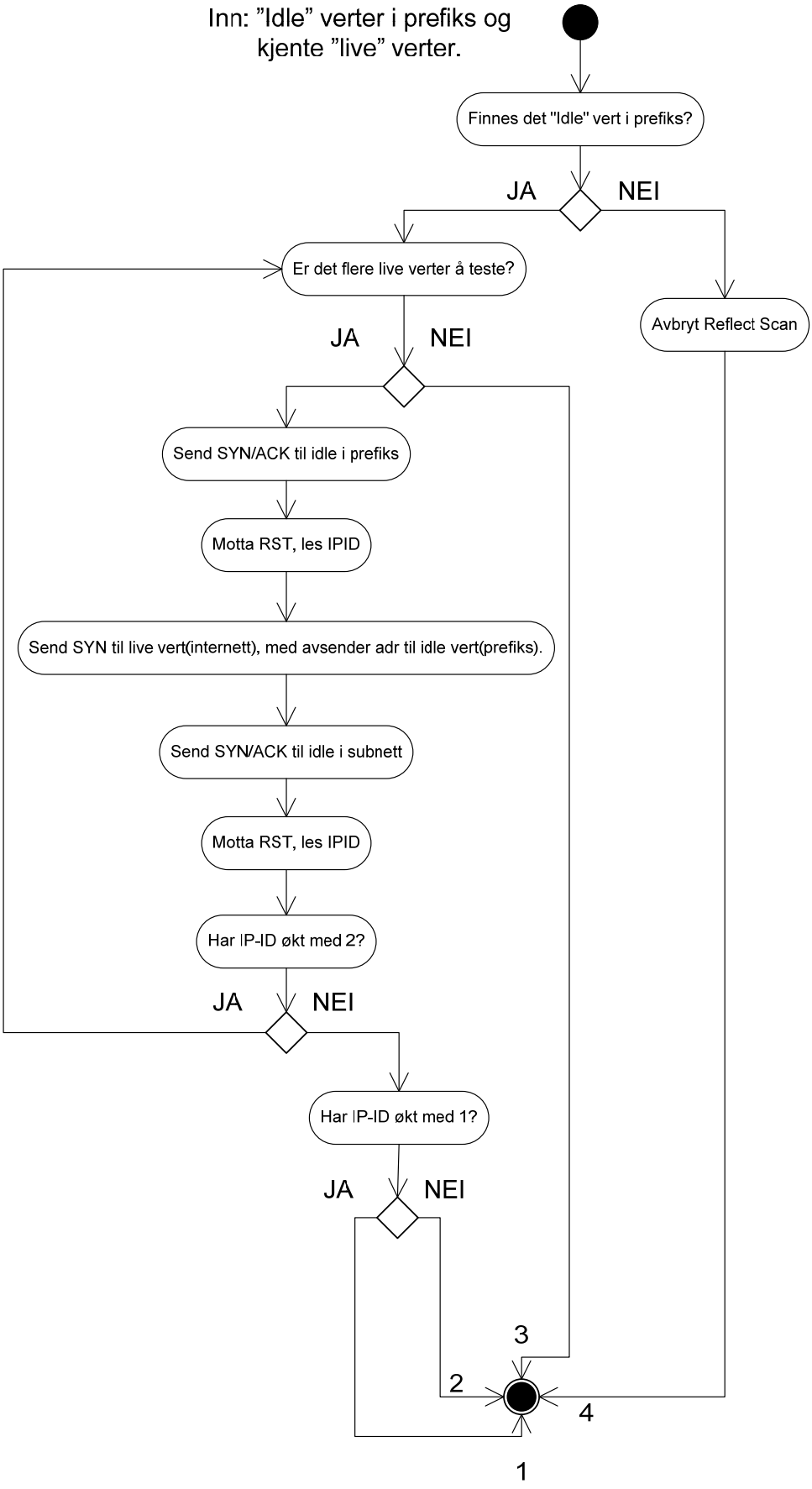


Figur 48 UML sekvensdiagram for Reflect Scan subnett med Hijack.

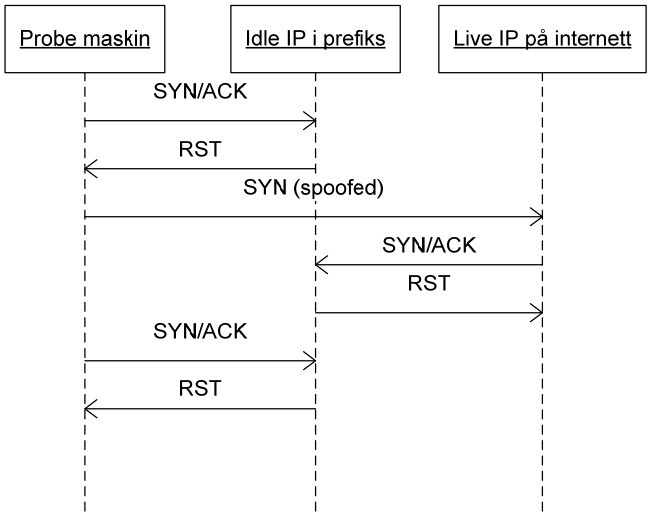
7.2.6 Reflect Scan for prefiks hijacking

Ved Reflect Scan for prefiks hijacking bygger også på Idlescan teknikken[25]. Men som vi vet på prefiks hijacking, så deler internett seg i to som Figur 18 viser.
Denne deteksjonstesten baserer seg på å se om alle nettverk på internett når samme ”idle” vert som vi når fra vårt nettverk. Alt vi trenger er en åpen port på et annet nettverk som SYN-prober via. Her kan man for eksempel bruke en ruteserver eller port 80 på www.vg.no.
Vi ønsker her at vi har ”live” IPer fra så mange AS som mulig, dette begrenser sannsynligheten for falske negativer.
Figur 49 viser aktivitetsdiagram for denne delen. Her kan vi se at vi sender en spoofet SYN-pakke til hver av ”live” vertene vi kjenner fra andre nett. Som da genererer en RST pakke fra

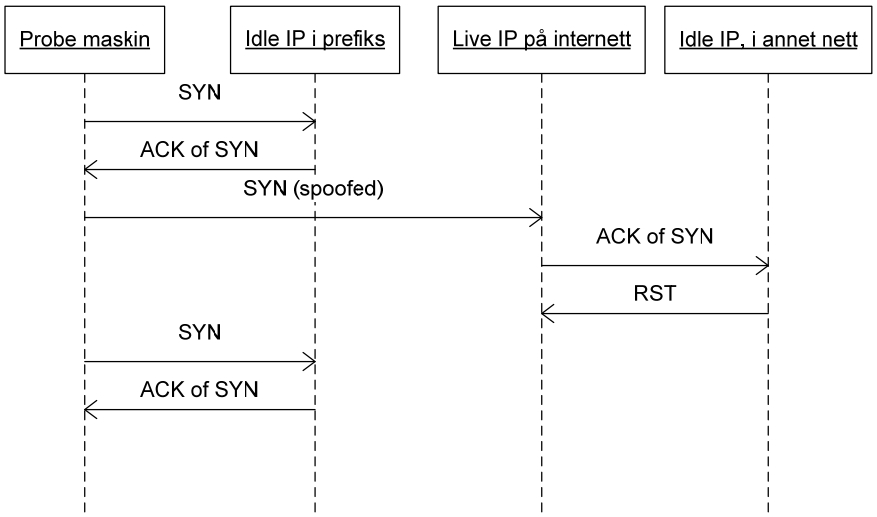
den ”idle” verten i prefikset vi skal kontrollere. Vi kan på denne måten se om andre nettverk på internett har kontakt med samme ”idle” vert som oss.
Er alle det så er ikke prefikset hijacket. Om noen ikke når samme prefiks, kan vi konkludere med at prefikset er hijacket.
Figur 51 viser sekvensdiagrammet for når ”live” verten sender svar til et annet nett. Og Figur 51 viser sekvensdiagrammet for når vi når samme nett som det nettverket vi prober via.



Figur 49 UML aktivitetsdiagram for Reflect Scan for prefiks



Figur 50 UML sekvensdiagram Reflect Scan prefiks, der live IP når samme nett.



Figur 51 UML sekvensdiagram Reflect Scan prefiks, der live IP svarer annet nett.

7.3 Diskusjon av systemet

Jeg har nå vist hvordan jeg ville designe et system for deteksjon av IP-hijacking. Dette systemet kan detektere både subnett hijacking og prefiks hijacking. Men systemet mangler fortsatt en måte å detektere hjiacking via vanlig rutingsvei. Systemet oppfyller alle de funksjonelle kravene jeg har satt.
Brannmurtesten som jeg har innført som et tillegg til Reflect Scan Subnett vil med stor sannsynlighet redusere antall falske positiver, selv om IDS systemer likevel kan føre til falske positiver. Jeg har også vist hvordan jeg ville brukt Idlescanteknikken til å detektere prefiks hijacking. Denne testen vil ta lengre tid å gjennomføre enn Reflect Scan for subnett hijacking. Men om man velger strategisk hvilke ”live” verter man bruker å probe via, kan man likevel begrense antall ”live” verter man må bruke. Om man ikke er strategisk og heller velger å

probe via alle ASene som er i bruk på internett i dag, kan testen ta over 24timer å gjennomføre.
Den store svakheten til dette deteksjonssystemet er at vi må finne ”idle” verter i nettet vi undersøke om er hijacket eller ikke. Finnes det ingen ”idle” verter så får man ikke gjennomført testene. Vi må også være klar over at ”live” vertene som vi prober via, kan være overarbeidet og velger å ikke svare på våre SYN-pakker, dette vil resultere i falske positiver.

8 Konklusjon

I denne oppgaven har jeg analysert ulike kjente deteksjonsteknikkene for IP-hijacking. Jeg har sett på har sett hvor gode de enkelte teknikkene er og kommet til følgene konklusjoner:

- MOAS og sub-MOAS konflikter ikke er godt nok alene som en test for å detektere IP-hijacking. Fordi slike konflikter oppstår noen ganger også i legitime tilfeller.
- Elektroniske fingeravtrykk som bruk for å detektere prefiks hijacking har jeg kommet frem til at er en god deteksjonsteknikk med både lavt antall falske positiver og lavt antall falske negativer, men det krever et distribuert system. Jeg har også sett på sertifikater og hvordan disse kan brukes som meget gode elektroniske fingeravtrykk.
- Geografisk posisjonering som en metode har jeg også analysert, denne er helt klart best for subnett hijacking, men den kan også i enkelte tilfeller ved subnett hijacking ha klare falske positiver.
- Logisk posisjonering har jeg også sett på, og kommet frem til at denne kan brukes i samband med andre deteksjonsteknikker, fordi den alene kan i enkelte tilfeller gi falske positiver ved legitime tilfeller av MOAS og sub-MOAS.
- Reflect Scan for subnett som test har jeg også analysert, og funnet ut at den kan ha gi falske positiver når en brannmur stopper deler av testen. Jeg har derfor kommet med et forslag til forbedring av denne metoden, ved å innføre en brannmur test. Denne ekstra testen vil med stor sannsynlighet redusere antall falske positiver ved bruk av Reflect Scan for subnett.
- I denne oppgaven har jeg også introdusert en ny måte å bruke idlescan teknikken til å detektere prefiks hijacking. Jeg har valgt å kalle denne Reflect Scan for prefiks hijacking, fordi denne har mange likheter med Reflect Scan for subnett som ble introdusert i ”Accurate Real-time Identification of IP Hijacking” [03]. Reflect Scan for prefiks hijacking skal ha falske negativer og falske positiver, men det er vanskelig å si noe spesifikt om antall på grunn av at jeg ikke har fått testet denne metoden i praksis.
- Traceroute som en deteksjonsteknikk for å detektere subnett hijacking og hijacking i vanlig rutingsvei, har jeg også undersøkt. Denne teknikken har vist seg å være veldig lett å unngå for en eventuell hijacker og gir i 50 % av testene et tvetydig svar [33].

Jeg har også foreslått et ikke-distribuert system for å detektere IP-hijacking, som baserer seg på bruk av ruteservere og begge Reflect Scan teknikkene. Dette systemet oppfyller alle de funksjonelle kravene jeg hadde satt. I dette systemet viste jeg også hvordan jeg ville implementert min foreslåtte brannmurtest og den nye deteksjonsteknikken Reflect Scan for prefiks. Fordi jeg ikke fikk testet dette systemet i praksis er det vanskelig å si noe om prosentandel falske negativer og positiver, men min foreslåtte brannmur test vil redusere antall falske positive ved Reflect Scan for subnett hijacking.

I et videre arbeid på denne oppgaven, ser jeg gjerne forsøk i et testnettverk av mitt ikkedistribuerte system for deteksjon av IP-hijacking. Dette krever at man har et høyt antall rutere som støtter BGP-4 protokollen. Man vil her også trenge flere datamaskiner som kjører ulike typer operativsystem og tjenester, for at man skal kunne kategorisere disse ”live” og ”idle”. Men det er likevel bare på internett vi kan virkelig se hvordan IDSer og brannmurer med ulike konfigurasjoner vil påvirke våre tester. Så et forsøk på internett ville være det beste for å virkelig kunne bekrefte hvor godt min foreslåtte brannmur test og Reflect Scan for prefiks fungerer.

9 Referanser

[01] Sam Halabi og Danny McPherson (2001). Internet Routing Architectures, Cisco Press.

[02] W. Richard Stevens (1996). TCP/IP Illustrated Volume 3, Corporate & Professional Publishing Group.

[03] Xin Hu og Z. Morley Mao (2003). Accurate Real-time Identification of IP Hijacking. Proceedings of IEEE Security and Privacy (Oakland) 2007.

[04] RFC 1771, "A Border Gateway Protocol 4 (BGP-4)", http://www.ietf.org/rfc/rfc1771.txt?number=1771

[05] RFC 2453, " RIP Version 2", http://www.ietf.org/rfc/rfc2453.txt?number=2453

[06] RFC 1583, "OSPF version 2" http://www.ietf.org/rfc/rfc1583.txt?number=1583

[07] Internet Assigned Numbers Authority. www.iana.org

[08] RFC 4632, "Classless Inter-domain Routing (CIDR)", http://www.ietf.org/rfc/rfc4632.txt?number=4632

[09] RFC 1930, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", http://www.ietf.org/rfc/rfc1930.txt?number=1930

[10] RFC 1918, "Address Allocation for Private Internets" http://www.ietf.org/rfc/rfc1918.txt?number=1918

[11] The Spamhause Project, "Salg av IP prefiks på Ebay, bevis" http://www.spamhaus.org/rokso/evidence.lasso?rokso_id=ROK2594

[12] Completewhois, http://www.completewhois.com/

[13] Séamus Ó Ciardhuáin (2004). An extended model of cybercrime investigations. http://www.utica.edu/academic/institutes/ecii/ijde/articles.cfm (19 oktober 2006)

[14] Digital Forensic Research Workshop (DFRWS). http://www.dfrws.org (20 oktober 2006)

[15] Espen A. Fossen, André Årnes (2005). Forensic Geolocation of Internet Addresses using Network Measurements. NORDSEC 2005.

[16] Espen A Fossen (2004) Automatic tracing of Internet addresses

[17] Njaal B. Andresen (2006) "Internet Investigations"

[18] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu,Lixia Zhang(2002), An Analysis of BGP Multiple Origin AS (MOAS) Conflicts.

[19] Ola Nordström og Constantintos Dovrolis,(2004) "Beware of BGP Attacks", ACM SIGCOMM Computer Communications Review April 2004.

[20] V. J. Bono,(1997) "7007 Explanition and Apology" http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html (2007 mars 26)

[21] Tao Wan Paul C. van Oorschot ,"Analysis of BGP Prefix Origins During Google's May 2005 Outage" 2nd International Workshop on Security in Systems and Networks (SSN2006), Rhode Island, Greece, Apr.25 2006 (in conjunction with IEEE IPDPS).

[22] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu,Lixia Zhang, "Detection of Invalid Routing Announcement in the Internet" (2002) Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)

[23] Gordon Lyon, Fydor, http://insecure.org/fyodor/

[24] RFC 1393, "Traceroute Using an IP Option", http://www.ietf.org/rfc/rfc1393.txt?number=1393

[25] Gordon Lyon, Fydor,Idlescan NMAP, http://insecure.org/nmap/idlescan.html

[26] Network Mapper (NMAP), http://insecure.org/nmap/

[27] Lixin Gao. "On Inferring Autonomous System relationship in the Internet", IEEE Global Internet Symposium, 2000.

[28] G. Battista, M. Patrignani, and M. Pizzonia. ”Computing the Types of the Relationships between Autonomous Systems”. IEEE INFOCOM Mars 2003.
[29] RFC 2459, “Internet X.509 Public Key Infrastructure”, http://www.ietf.org/rfc/rfc2459.txt?number=2459
[30] Svein J. Knapskog “Informasjonssikkerhet i internett” (2005), Tapir Akademisk Forlag, Trondheim 2005.
[31] RFC 1321, “The MD5 Message-Digest Algorithm”, http://www.ietf.org/rfc/rfc1321.txt?number=1321
[32] Advanced Network Architecture Group. Spoofer Project, http://spoofer.csail.mit.edu/
[33] Z. Morley Mao, Jennifer Rexford, Jia Wang, og Randy Katz, ”Towards an Accurate AS-Level Traceroute Tool” ACM SIGCOMM 2003.
[34] Global Crossing. http://www.gblx.net/
[35] RFC 1631, ”The IP Network Address Translator (NAT)”. http://www.ietf.org/rfc/rfc1631.txt?number=1631
[36] Jpcap ”Java package for packet capture”, http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html
[37] Java Technology, http://java.sun.com/
[38] WinPcap, ”The Windows Packet Capture Library” http://www.winpcap.org/
[39] S.M. Bellovin ,”Security Problems in the TCP/IP Protocol Suite” Computer Communication Review, Vol. 19, No. 2, pp. 32-48, April 1989.

Vedlegg

Vedlegg A

CIDR	Class	Hosts*	Mask
/32	1/256 C	1	255.255.255.255
/31	1/128 C	2	255.255.255.254
/30	1/64 C	4	255.255.255.252
/29	1/32 C	8	255.255.255.248
/28	1/16 C	16	255.255.255.240
/27	1/8 C	32	255.255.255.224
/26	1/4 C	64	255.255.255.192
/25	1/2 C	128	255.255.255.128
/24	1 C	256	255.255.255.000
/23	2 C	512	255.255.254.000
/22	4 C	1024	255.255.252.000
/21	8 C	2048	255.255.248.000
/20	16 C	4096	255.255.240.000
/19	32 C	8192	255.255.224.000
/18	64 C	16384	255.255.192.000
/17	128 C	32768	255.255.128.000
/16	256 C, 1 B	65536	255.255.000.000
/15	512 C, 2 B	131072	255.254.000.000
/14	1024 C, 4 B	262144	255.252.000.000
/13	2048 C, 8 B	524288	255.248.000.000
/12	4096 C, 16 B	1048576	255.240.000.000
/11	8192 C, 32 B	2097152	255.224.000.000
/10	16384 C, 64 B	4194304	255.192.000.000
/9	32768 C, 128B	8388608	255.128.000.000
/8	65536 C, 256B, 1 A	16777216	255.000.000.000
/7	131072 C, 512B, 2 A	33554432	254.000.000.000
/6	262144 C, 1024 B, 4 A	67108864	252.000.000.000
/5	524288 C, 2048 B, 8 A	134217728	248.000.000.000
/4	1048576 C, 4096 B, 16 A	268435456	240.000.000.000
/3	2097152 C, 8192 B, 32 A	536870912	224.000.000.000
/2	4194304 C, 16384 B, 64 A	1073741824	192.000.000.000
/1	8388608 C, 32768 B, 128 A	2147483648	128.000.000.000
/0	16777216 C, 65536 B, 256 A	4294967296	000.000.000.000

Vedlegg B

Final Algorithm:
Input: BGP routing table RT
Output: Annotated AS graph G
Phase 1: Use either basic or refined algorithm to coarsely classify AS pairs' relationships
Phase 2: Identify AS pairs that can not have a peer-to-peer relationship
1.For each AS path $(u_1, u_2, \ldots, u_n)$ in RT,
2. find the AS u_j such that
degree[u_j]=max$_{1 \le i \le n}$ degree[u_i]
3. for $i = 1, \ldots, j-2$,
4. notpeering[u_i, u_{i+1}]=1
5. for $i = j+1, \ldots, n-1$,
6. notpeering[u_i, u_{i+1}]=1
7. if relationship[u_{j-1}, u_j]$\neq$sibling-to-sibling
and relationship[u_j, u_{j+1}]$\neq$ sibling-to-sibling
9. notpeering[u_j, u_{j+1}]= 1
10. else
11. notpeering[u_{j-1}, u_j]= 1
Phase 3: Assign peer-to-peer relationships to AS pairs
1.For each AS path $(u_1, u_2, \ldots, u_n)$ in RT,
2. for j=1, ..., n-1,
3. if notpeering[u_j, u_{j+1}]$\neq$ 1
and degree[u_j]/degree[u_{j+1}]$< R$
and degree[u_j]/degree[u_{j+1}]$> 1/R$
4. relationship[u_j, u_{j+1}] = peer-to-peer