Säker grannupptäck i IPv6
(Secure Neighbor Discovery in IPv6)

PHILIP HUSS

KTH Information and
Communication Technology

Bachelor of Science Thesis
Stockholm, Sweden 2011

TRITA-ICT-EX-2011:22

Säker grannupptäck i IPv6 (Secure Neighbor Discovery in IPv6)

Philip Huss

2011-01-20

Bachelor of Science Thesis

Examiner and Academic Supervisor
Prof. Gerald Q. Maguire Jr.

Department of Communication Systems
School of Information and Communication Technology
Royal Institute of Technology
Stockholm, Sweden

Abstract

The IPv6 protocol offers with some new functions, one of them is auto configuration. With auto configuration it is possible for nodes, i.e. hosts and routers, for automatically associated with IPv6 addresses without manual configuration. Auto configuration it is another protocol as it uses Neighbor Discovery protocol (ND) messages (ND is mandatory in the IPv6 stack). The main purpose of ND is that nodes can discover other nodes on the local link, perform address resolution, check that addresses are unique, and check the reachability with active nodes.

There are exactly the same vulnerabilities of IPv6 as IPv4 and is now exception, ND if not properly secured. IPsec is a standard security mechanism for IPv6 but it does not solve the problem of secure auto configuration due the bootstrapping problem. Therefore the Internet Engineering Task Force (IETF) introduced Secure Neighbor Discovery (SEND). SEND is a mechanism for authentication, message protection, and router authentication. One important element of SEND is the use of Cryptographically Generated Address (CGA) an important mechanism to prove that the sender of the ND message is the actual owner of the address it claims

NDprotector is an open-source implementation of SEND served as the basis for the analysis presented in this report. This implementation was evaluated in a small lab environment against some attacks in order to establish if it can defend itself from these attacks.

Abstrakt

IPv6 protokollet kom det ett par nya funktioner där en av dem är autokonfiguration. Autokonfiguration gör det möjligt för noder, d.v.s. hostar och routrar för att automatiskt bli tilldelade IPv6 adresser manuell konfigurering. För att autokonfiguration ska fungera så används Neighbor Discovery (ND) meddelanden som är ett obligatoriskt protokoll i IPv6-stacken. ND har till huvudsaklig uppgift att noder kan upptäcka andra noder på den lokala länken, utföra adressöversättningar, kolltrollera så att adresser är unika samt kontrollera tillgängligheten hos aktiva noder.

Precis som IPv4 så har IPv6 en hel del sårbarheter och med ND så är det inget undantag då det inte är säkrat. IPsec som är en den standard säkerhets mekanism till IPv6 löser inte problemet på grund av bootstrapping problemet. Det var därför Internet Engineering Task Force (IETF) introducerade Secure Neighbor Discovery (SEND). SEND är en

mekanism för autentisering, meddelande skydd och router autentisering. En viktig del av SEND är Cryptographilcally Generated Address (CGA), en teknik som används för att försäkra sig så att det är den sändaren av ND meddelandet som är den riktiga ägaren av den hävdade adressen.

NDprotector är en öppen källkods implementation av SEND som jag har valt att ha som grund för denna rapport. Jag kommer att sätta upp NDprotector i en liten labbmiljö där jag kommer att utföra olika attacker samt se efter om det klarar att försvara sig emot attackerna.

Keywords: IPV6 security, Neighbor Discovery, Secure Neighbor Discovery, Cryptographically Generated Addresses

Förord

Vill bara passa på att tacka Professor Gerald Q. "Chip" Maguire Jr som har varit min examinator samt handlare. Jag är verkligen glad för att jag fått göra mitt examensarbete för honom, han är så himla entusiastisk till allt han gör för sina elever. Vill även tacka Viking som har varit min opponent samt Tony som hjälpt till med NDprotector.

Innehållsförteckning

1. Inledning

2. Teoretisk bakgrund

3. Implementation

4. Slutsatser

Figurer

Tabeller

1. Inledning

1.1 Problemdefinition

IPv6 protokollet kom det ett par nya funktioner där en av dem är autokonfiguration.
Autokonfiguration gör det möjligt för noder, d.v.s. hostar och routrar att automatiskt bli tilldelade IPv6 adresser utan någon som helst konfigurering. För att autokonfiguration ska fungera så används granneupptäckt(English: Neighbor Discovery (ND)) meddelanden som är ett obligatoriskt protokoll i IPv6-stacken. ND har till huvudsaklig uppgift att noder kan upptäcka andra noder på den lokala länken, adressöversättningar, kolltrollera så att adresser är unika samt att kontrollera tillgängligheten hos aktiva noder.

ND uppfyller en mycket bra funktion men som sagt det uppstår en del sårbarheter när det inte är säkrat. En hacker kan enkelt utföra olika attacker så som mannen i mitten attack(English: Man-in-the-Middle (MitM)) eller förnekad av tjänst(English: Denial-of-Service (DoS)).

När ND definierades så antogs det att på den lokala länken skulle det enbart bestå av förtroende noder. Men med dagens teknik utveckling som går i en radikal takt där det inte enbart fysiskt säkrade lokalt nätverk(English: Local Area Network (LAN)) utan numera är det trådlösa nätverk(English: Wireless Area Network (WLAN)) som gäller. WLAN är en mycket vanligt förekommande teknik på t.ex. hotell, flygplatser samt caféer. Noder på lokala länkar kan inte lita på varandra, det blir inte ens helt säkert då noder skyddas på lager-2 nivå med antingen autentiserad 802.1X authentication eller kryptering med ”Wi-Fi Protected Access2” (WPA2).

Målet med autokonfiguration är ju att hela processen ska vara automatiskt utan någon som helst manuell konfigurering. Internet protokol säkerhet(English: Internet Protocol Security (IPsec)) som är standard säkerhets mekanism för IPv6 löser inte problemet för att säkra autokonfiguration mekanismen. IPsec fungerar inte för detta ändamål därför att IPsec behöver internet nyckelutbyte(English: Internet Key Exchange (IKE)) för att sätta upp säkerhetsförbindelse(English: Security Association (SA)). I sin tur så behöver IKE en giltig IPv6 adress för att möjliggöra kommunikation mellan två IPsec noder. Det blir lite utav hönan och ägget(English: chicken-and-egg) problemet.

De var därför Internet Engineering Task Force (IETF) introducerade säker grann pptäckt(English: Secure Neighbor Discovery (SEND)). SEND använder sig utav kryptografiskt genererade adresser(English: Cryptographilcally Generated Address (CGA)), en teknik som används för att försäkra sig så att det är den sändaren av ND meddelandet som är den riktiga ägaren av den hävdade adressen.

NDprotector som är en öppen källkod implementation av SEND som jag har valt att ha som grund för denna rapport. Jag kommer att titta närmare på dessa sårbarheter till ND samt sedan se hur NDprotector klarar av att skydda sig emot dem. Mer om NDprotector och andra SEND implementationer hittar du i sektion ”1.5”.

1.2 Syfte

Under rapportens gång vill vi få svar på ett antal frågor som vi anser är viktigast inom ämnet just nu:

- Varför är SEND implementeringar så ovanligt förekommande i dagens nätverk?
- Skyddar SEND verkligen emot diverse olika attacker?
- Finns det andra lösningar till SEND för att säkra ND?
- Är DHCPv6 en utmanare?
- Varför finns det inte som standard i nuvarande Microsoft Windows (7/2008 server) och olika Linux distributioner?
- Är NDprotector en bra implementation?

Det bara måste komma en effektiv lösning på hur man kan skydda sig ute på publika nätverk.
Inte vill man att någon ska kunna läsa ens epost eller bankkonto (med hjälp av ”SSLstripping”) uppgifter t.ex. när man sitter och surfar på ett café. Det är ju numera löjligt enkelt för vem som helst utan några som helst avancerade datorkunskaper att ta reda på ens facebook profil med hjälp av Firefox pluginen: firesheep.

SEND implementationer existerar inte i det flesta nätverk idag och kanske skulle SEND kunna vara en framtida säkerhets mekanism i lokala nätverk. Det är därför jag har valt att fördjupa mig inom det. Men givetvis även för att skaffa mig en mer kunskap kring säkerheten till IPv6.

1.3 Målgrupp

Denna rapport vänder sig främst till studenter från min utbildning, Datateknik med inriktning Datornätverk, Högskoleingenjör 180hp. Givetvis även för handledaren och examinatorn Gerald Maguire samt författaren, jag själv. Sen självklart andra intressenter som är intresserade av säkerheten kring IPv6.

1.4 Avgränsningar

Eftersom denna rapport är skriven i slutet av min utbildning så tänker jag inte lägga ner onödig tid på att skriva sådant som man bör ha kunskaper inom. I och med det så hänvisar jag till mina referenser där man kan läsa djupare om något som man kanske saknar kunskaper inom.

Rapporten är som sagt därför skriven för den med relativt goda kunskaper inom datorkommunikation då främst inom IPv6 protokollet. Det kan vara bra att ha en grundläggande förståelse inom IT-säkerhet eftersom rapporten just handlar om bristerna till ND protokollet och vissa termer kommer att tas upp frekvent. Kommer inte mer än att ta upp IPsec problemet, alltså inga praktiska implementationer. Kommer inte heller att gå in något hur man kan logga ND meddelanden i form av nätverksövervakning t.ex. med programmet NDPmon. Eftersom implementationen kommer vara i en Linux baserad miljö så kan det vara bra att ha grundläggande Linux kunskaper inom det.

1.5 Vad har andra gjort?

Det finns ett väldigt fåtal olika SEND implementationer ute på internet idag.

Docomo är väl den mest kända öppen källkods projekt och är utvecklad av DoCoMo USA labs. Den ska fungera i ”user-space” på operativsystemen Linux med kernel 2.6 samt Freebsd 5.4. Det finns en känd nackdel med denna applikation och det är att den inte hanterar kollisioner med Duplicate Address Detection (DAD) processen, på grund av integrationen till kärnan.

JSend som finns på SourceForge är en implementation av SEND protokollet i Java. Projektet är nerlagt och ingen information eller källkod finns tillgänglig att ladda ner.

Sen finns det Easy-SEND som också är en öppen källkods källkodprojekt som är utvecklad av ett universitet *"Central"* i Venezuela. Denna applikation ska ha en bra logghanterare för att hantera felsökning. Easy-send har en nackdel och det är att den saknar support för Router Discovery (RD).

IPv6-SEND-CGA är en Linux kernel implementation skriven i programmerings språket C. Utvecklad av företaget Huawei Technologies i sammarbete med Beijing Universitet Post och telekommunikation.

NDprotector är en annan relativt ny öppen källkods källkodimplementation skriven i python och är baserad på scapy6. Det finns inte så mycket information och analyser som visar hur NDprotector skyddar emot diverse olika attacker samt hur det är prestandamässigt. Det är därför jag har valt att studera närmare kring just det. Tony Cheneau som utvecklat denna implementation har som plan att fortsätta utveckla den.

1.6 Metod

Rapporten kommer vara baserad på både litteraturstudie men även hur NDprotector fungerar i praktiken. Genom att studera och implementera NDprotector i en liten labbmiljö så förväntar jag mig att få en bra uppfattning till det hela. Till det kommer ett par olika attacker att genomföras för att sedan titta närmare på hur man kan skydda sig med hjälp av NDprotector.

Vi kommer givetvis för det mesta att använda oss av olika dokument som kallas för Request for Comments (RFC). RFC är tekniska beskrivningar med ledande och föreslagna internetstandarder som publicerat av IETF som arbetar med nya frågor om tekniker och standarder som ska användas på internet.

1.7 Disposition

Här ser vi hur rapportens uppbyggnad och struktur på ett överskådligt sätt och det gör att vi på ett enklare sätt kan följa rapportens 5 olika delar.

Inledning (Kapitel 1)
Inledning kommer att omfatta en förklaring till problemet kring ND, syftet med denna rapport, målgruppen, metod, vad andra har gjort samt avgränsningar.

Teoretisk bakgrund (Kapitel 2)
Här kommer den teoretiska bakgrunden till ND och dess sårbarheter samt hur man kan undvika sig emot dessa med hjälp utav SEND. Det är alltså denna del som ligger till grund för att man ska kunna förstå hur det hela fungerar, vilket är viktigt inför nästa kommande kapitel om implementationen.

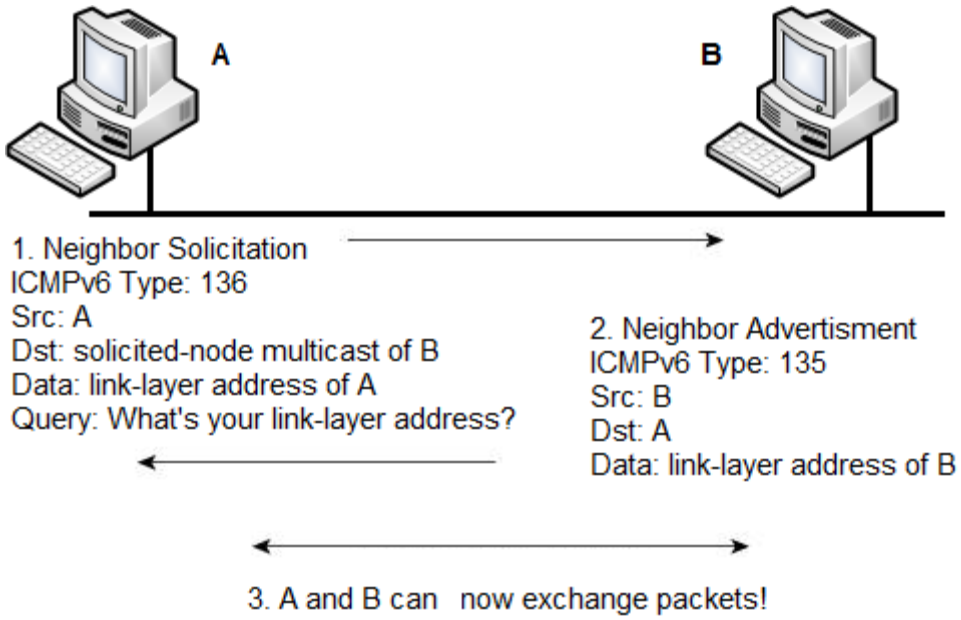
Implementation (Kapitel 3)
Implementation av NDprotector kommer att konfigureras i en liten labbmiljö. Efter kommer det att utföras diverse attacker och slutligen kommer vi studeras närmare hur NDprotector förhindrar dem.

Slutsats (Kapitel 4)
Här kommer det att diskuteras slutsatser kring implementationen NDprotector samt den generella SEND lösningen.

2. Teoretisk bakgrund

Teoretiska bakgrunden ligger till grund för att förstå vad problemet innebär samt inför implementationen.

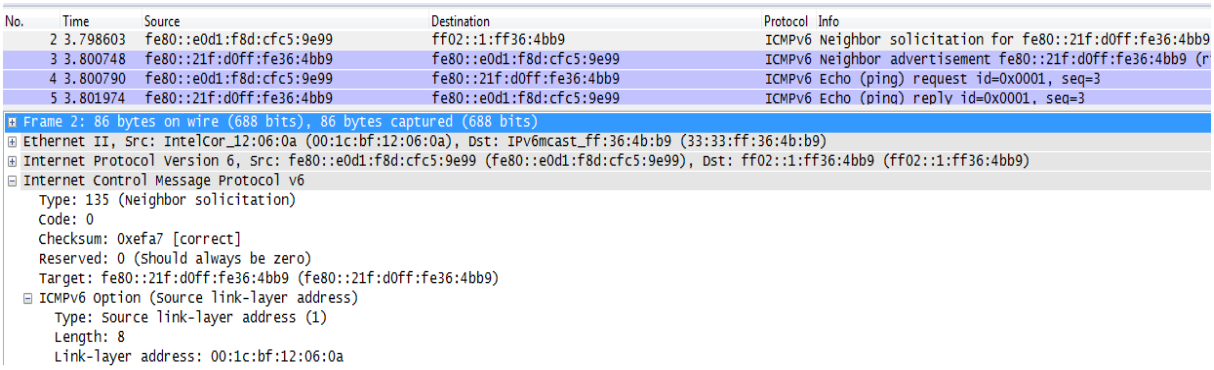
2.1 Neighbor Discovery (ND)



Figur 1: Neighbor Discovery (ND)

ND är till för att samla nätverks information på den lokala länken(English: link-local) som behövs för att kunna göra adressöversättningar mellan IPv6 adresser och nätverkskortets hårdvaruadress(Media Access Control (MAC)). När man pratar i IPv6 termer så pratar man inte om MAC adresser längre utan då är det länklager(English: link-layer) adresser som gäller.

ND tillåter därmed noder att upptäcka andras länklageradresser, precis som i IPv4.
ND protokollet är en mycket viktig del utav internet protokoll meddelanden version 6(English: Internet Control Massage Protocol Version 6 (ICMPv6)). ND är ett komplext protokoll och kommer med en mänga olika meddelanden och processer som används i IPv6 noder. För att bara nämna några av dem adressöversättningar, autokonfiguration, samt för att utforska tillgängligheten hos noder samt integrera med varandra med hjälp av ICMPv6 meddelanden. Vi kommer senare att titta närmare på varje funktion var för sig.



Figur 2: Wireshark adress översättning

2.1.2 Adress översättning

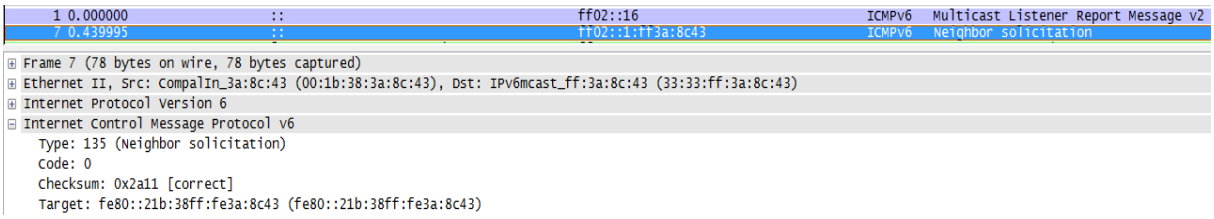
Adress översättning (English: Address Resolution) fyller precis samma funktion som adress översättning protokollet(English: Address Resolution Protocol (ARP)) i IPv4. Det är alltså en mekanism som gör det möjligt att översätta en IPv6 adress till en länklager adress.

När en node vill skicka unicast paket till andra noder på den lokala länken så kollar den först i efter länklager adressen i Neighbor cache tabellen (densamma som i IPv4's ARP cache). Om inte länklager adressen existerar där så skickas ett multicast grannevärvning(English: Neighbor Solicitation (NS)) meddelande med sorce länklager adress. Mål noden kommer nu att svara tillbaka med ett unicast grannannonsering(English: Neighbor Adertisment(NA)) med tillhörande länklager adressen. Det fungerar alltså precis så som i *Figur 1*.

2.1.3 Stateless Autokonfiguration

Till IPv4 så används det väldigt ofta en så kallad dynamisk konfiguration av host protokoll(English: Dynamic Host Configuration Protocol (DHCP)) server för att tilldela adresser, antingen från en router eller DHCP server. På ett rent generellt sätt så kanske det fungerar bra, men för att vara lite kritisk så krävs det ju faktiskt en DHCP server med lite konfigurering. Eller om man nu använder en router så kan man inte garantera att man just får den adress man vill ha vid en senare tidpunkt. Notera att en DHCP server inte garanterar identiteten av noder efter att adresser tilldelats utan den enbart reserverar en adress som är knuten till länklager adressen.

Stateless autokonfiguration är en ny funktion i IPv6 för att automatiskt tilldela adresser utan att behöva en DHCP server. Det är en väldigt praktiskt funktion och det fungerar faktiskt som plug-and-play d.v.s. principiellt bara dra i gång datorn och du får en IPv6 adress, väldigt användbart för WLAN.



Figur 3: Autokonfiguration

Viktigt att notera är att en IPv6 adress består utav två delar, de första 64-bitarna är till för ett nätverks prefix och de resterande 64-bitarna är till för nätverksgränssnitt identifierare(English: Interface identifier (IID)). Nätverks prefix kan antigen vara till för länklokala adressen med prefixet *"fe80::0/64"* eller en global unicast adress . IID är rent generellt baserat på länklager adressen med hjälp av en utökad unik identifierare(English: Extended Unique Identifiter (EUI-64)) adressering där 64 står för antal bitar. Senare kommer vi att titta närmare på hur det fungerar med CGA. Det sammanlänkade nätverksprefixet och IID skapar tillsammans en adress som kallas för tentative adress.

Hostar och routrar som har IPv6 aktiverat har alltid en konfigurerad länklokal adress på alla gränssnitten. Länklokala adressen är erhållen från gränssnittens länklager adress men för att vara garanterad om att vara unik så används en funktion som kallas för duplikat adress upptäck(Duplicate Address Detection (DAD)) som kommer att beskrivas här näst.

2.1.4 Duplikat Adress Upptäckt

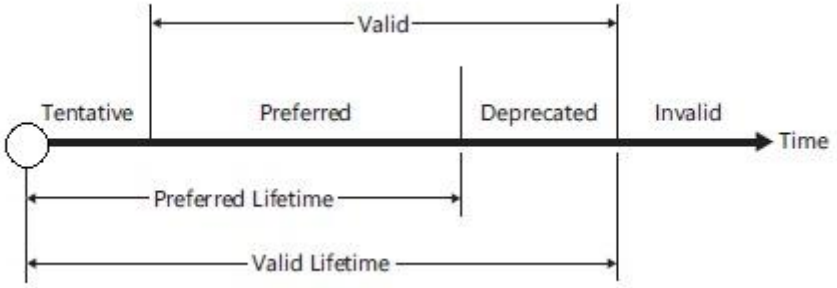
Duplikat Adress Upptäckt (English: Duplicate Address Detection (DAD)) är efter att en node har erhållit en unicast IPv6 adress så måste den kontrollera så att den är unik innan den kan tilldelas till gränssnittet. Det är alltså DAD mekanismen som kontrollerar proceduren och det fungerar så att den använder sig utav ett NS meddelande som skickas med

källadressen den ospecificerade adressen(English: unspecific address)) ”*::*” samt destinationesadressen den begärda nätverksnodens(English: solicited-node)) multicast adressen. Den begärda nätverksnodens adress skaps med hjälp av prefixet ”ff02:0:0:0:0:1:ff00::/104” samt dom 24 sista bitarna av IPv6 adressen. Säg att vi har fe80::21f:d0ff:fe36:4bb9 så kommer det att bli ff02::1:ff36:4bb9.

Om någon annan node redan använder sig utav den befintliga adressen som efterfrågas så måste den svara med ett ND meddelande till destinations multicast adressen alla noder(English: all-nodes) länklokala ”ff02::1”. Annars om det inte kommer något ND i gensvar så kommer den anta att den är unik och där med tillgänglig att använda för unicast trafik.

2.1.5 Adress tillstånd

IPv6 adresser har många olika adresstillstånd(English: address states), notera att olika operativsystem har olika livslängder på tillstånden. Vid eventuella problem med adresser så kan det vara mycket lämpligt att felsöka i vilket tillstånd en IPv6 adress befinner sig i.



Figur 4: IPv6 olika tillstånd

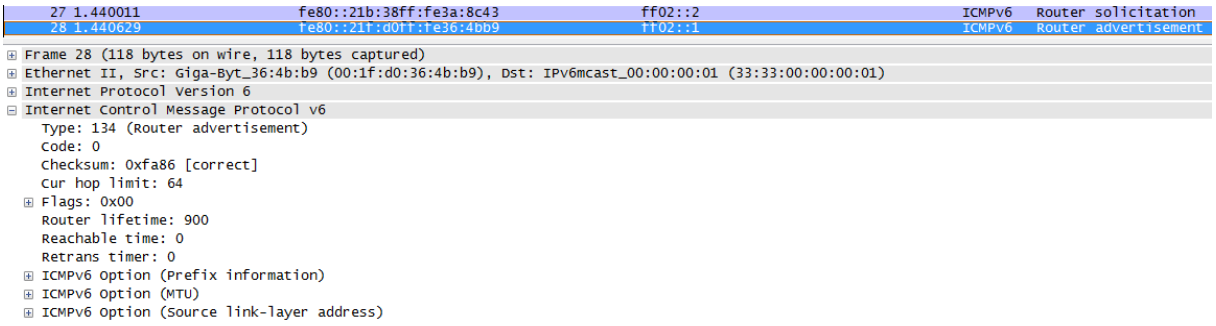
- Preliminiär
  Preliminär(English: tentative) är när ett interface försöker generera en ny adress så kommer den vara preliminär tills det att DAD har försäkrat sig så ingen annan på länken använder den valda adressen.
- Dubblett
  Dubblett (English: duplicate) är när DAD misslyckas, d.v.s. att den

visar att ett annat interface redan använder den adressen så kommer adressen bli visad som dubblett.

- Giltig
  Giltig (English: vaild) är när DAD processen är avklarad utan några problem så kommer adressen till tillståndet ”vaild”. Med autokonfiguration så finns det två olika ”vaild” tillstånd, ”preferred” eller ”deprecated”.
- Prefererad
  Preferera(English: preferred) är när adressen har verifierats och är därmed unik. En node kan då skicka och ta emot unicast trafik till och från en prefererad adress. Det kan vara så också att noden får ett RA meddelande med ett värde på den tid som den kan vara i tillståndet, ”prefererad”.
- Föråldrad
  Föråldrad(English: deprecated) då befinner sig adressen i tillståndet föråldrad och då kan den fortfarande etablera kommunikation under den existerade sessionen. Men kan inte användas för nya sessioner.
- Ogiltig
  Ogiltig(English: invaild) är när livslängs tiden har gått ut så kan adressen inte längre användas för att skicka och ta emot unicast trafik och därmed ogiltig.

2.1.6 Router upptäckt

Router upptäckt(English: Router Discovery (RD)) fungerar så att efter att en host skaffat sig en unik länklokal adress så är det dags att utforska tillgängliga routrar och prefix på den lokala länken. Det fungerar så att en host skickar ut ett RS meddelande till multicast adressen alla routrar(English: all-routers) ”FF02::2”. Det är en adress som alla aktiva routrar lyssnar på. Alla routrar kommer nu att svara tillbaka till hosten med ett RA meddelande med ett antal olika länklokala parametrar t.ex. prefix.



Figur 5: RD processen

2.1.7 Granne onåbarhets upptäckt

Upptäckt av nåbara grannar(English: Neighbor Unreachability Detection (NUD)) fungerar så att noderna brukar förlita sig på övre lagets information för att avgöra om grannarna fortfarande är tillgängliga. Men om det finns en tillräckligt lång fördröjning på övrelagers trafik eller om noden slutar att få gensvar så kommer NUD funktionen att bli involverad i processen.

Hostar och routrar sänder aktivt ut NS meddelanden för att hålla kolla så att grannarna är tillgänglig. Om grannarna inte skulle svara med ett NA inom den angivna tidsperioden så är det troligtvis något som är fel, systemet kommer då att ta bort grannens länklager adress ur granncache tabellen. Det är NUD som är själva proceduren för att hålla kolla på dom olika tillstånden av grannarna som är lagrade i granncache tabellen.

Denna teknik kan vara användbart då det finns flera routrar i nätverket då en router går ner så kommer hosten att automatiskt byta standard portgång(English: default gateway).

2.1.8 Granneupptäckt cache

Granneupptäckt(English: neighbor discovery cache) är en möjlig

konceptuell datastruktur som hostar och i viss mån routrar kommer att upprätthålla samverkan med närliggande noder.

Granne cache
Granne cache tabellen är ekvivalent med ARP cachen tabellen i IPv4. Den lagrar alltså grannens IPv6 adress med tillhörande länklager adress samt en indikation av tillgänglighetens tillstånd.

Destinations cache
Lagrar information om destinations next-hop IPv6 adresser för senast skickade trafik. Destination cachen innehåller direktanslutna samt icke direktanslutna destinationer och den är uppdaterad med information som den lärt sig från omdirigerade(English: redirect) meddelanden.

Prefixlist
Prefix listan innehåller poster för alla prefix på länken. Varje post har IPv6 adresser för destinationer som är direkt tillgängliga. Listan är uppdaterad med prefix som den lärt sig utav RA meddelanden.

Standard router list
Listan visar vilka routrar som är villiga att agera som standard router.

2.1.9 Granne cache tillstånd

Granne cache tabellen innehåller ett som sagt ett register över nåbarheten för sina grannar. När en granne lagras så kommer den att befinna sig i olika tillstånd.

- **Ofullständig**
  Ofullständig(English: incomplete) Länklager adressen är inte ännu fastställd på grund av adress översättning pågår.

- **Tillgänglig**
  Tillgänglig(English: reachable) bekräftelse av tillgänglighet har nyligen gjorts.

- **Förlegad**
  Förlegad(English: stale) tiden är slut och ingen bekräftelse om tillgänglighet.

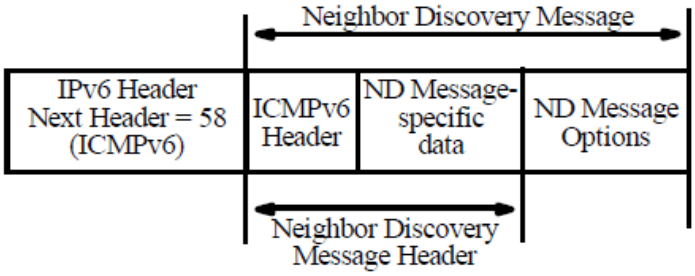
- **Fördröjning**
  Fördröjning(English: delayed) Ingen bekräftelse om tillgänglighet

under den senaste tiden men ett unicast NS sond kommer att skickas till grannen för att verifiera tillgängligheten.

- **Sond**
  Sond(English: probe) ett unicast NS har precis skickats men har inte ännu fått något svar.

2.1.10 ND meddelanden

Ett ND meddelande består utav ett ND meddelande huvud, bestående av ett ICMPv6 huvud, ND meddelande specifik data, och ett valfritt alternativ.



Figur 6: ND meddelande format

De olika alternativen för ND meddelanden förser ytterligare med information så som länklager adresser, prefix information, omdirigeringss huvud samt maximal överförings storlek (Maximum Transmission Unit (MTU)).

Typ	Alternativ
1	Source Link-Layer Address
2	Target Link-Layer Address
3	Prefix Information
4	Redirected Header
5	MTU

Tabell 1: ND alternativ

- **Käll länklager adressen**
  Käll länklager adressen(English: source link-layer address) är inkluderad i NS, RS och RA meddelanden för att indikera länklager adressen hos den sändande noden.

- Mål länklager adressen
  Mål länklager adressen(English: target link-layer address) är inkluderad i NA och omdirigerade meddelanden för att indikera länklager adressen av målet.

- Prefix Information finns I RA meddelanden för att visa hostar direktansluten prefix och prefix som behövs för autokonfiguration.

- Omdirigeringshuvud
  Omdirigeringshuvud(English: redirected header) används i omdirigerade meddelanden och innehåller alla eller delar av paket som ska omdirigeras.

- Maximal överförings storlek används i RA meddelanden för att försäkra så att alla noder på länken använder samma MTU.

ICMPv6 används som stöd för att underlätta tjänster som ND ger. ND protokollet har fem olika meddelanden eller protokoll data enhet(English: Protocol Data Unit (PDU)) med olika funktionalitet.

Router värvning
Router värvning(English: Router Solicitation (RS)) fungerar så att efter att ett interface har aktiverats så kommer hostar att skicka ut RS meddelanden för att se efter om det finns några tillgängliga routrar på länklokal. RS skickas till destinations multicast adressen hos alla routrar

Router annonsering
Router annonsering(English: Router Adevertisment (RA)) meddelanden skickas periodvis ut till alla-noders mutlticast adress för att informera om deras tillgänglighet men även som gensvar till RS meddelanden. RA meddelanden innehåller olika länk relaterade parameter värden så som länk prefix, MTU och hop-limit.

Granne värvning
Granne värvning(English: Neighbor Solicitation (NS)) har tre olika huvudfunktioner och det är dels utforska länklager adresser på länken som är en del av adressöversättning. För det andra så är det tillgängligheten till andra noder. Sen används det även till DAD funktionen för att avgöra om samma unicast adress har tilldelats på mer än en node.

Granne annonsering
Granne annonsering(English: Neighbor Advertisement (NA)) är till för

både svara på NS meddelanden eller så skickas det för att informera om sin länklager adress. Vid ett mottagande av ett NA meddelande så kommer noden att uppdatera sin granne cache med en IPv6 adress och tillhörande länklager adressen.

Omdirigering

Används av routrar för att informera hostar om en bättre destinations adress för sitt first-hop.

Meddelande	ICMPv6 type	Sändare	Mottagare	Alternativ
Router Solicitation	133	Noder	Routrar (multicast)	Source Link-Layer adresser
Router Advertisment	134	Routrar	1.När Solicited: Skickas av RS (unicast) 2.När unsolicited All noder (multicast)	Source Link-Layer adresser, autokonfiguration parameters. MTU, Prefix, Router,
Neighbor Solicitation	135	Noder	Solicited Node eller Målets node	Source Link-Layer Adress
Neighbor Advertisment	136	Noder	Svarar på NS eller all noder (multicast)	Target Link-Layer addresser

Tabell 2: ND meddelande typer

2.2 Threat modell

Enligt RFC 3766 så beskrivs det tre olika huvudsakliga nätverks scenarion samt en hel del olika potentiella sårbarheter till dem. Nedan kommer ett litet sammandrag av de olika attackerna man kan utföra. Dom är uppdelade i tre olika kategorier: ND attacker, routing attacker samt replay och remote attackerna.

- Alla autentiserade noder på ett stängt nätverk (företags nätverk)
- Pålitliga routrar med opålitliga hostar (Publikt trådlöst nätverk)
- Nätverk där alla noder är opålitliga (Ad hoc)

2.2.1 Neighbor Discovery attacker

Här nedan kommer det olika attackerna kring ND protokollet att beskrivas lite närmare.
Det är framförallt DAD och NUD till autokonfigurationen.

2.2.2 Neighbor Solicitation/Advertisement spoofing

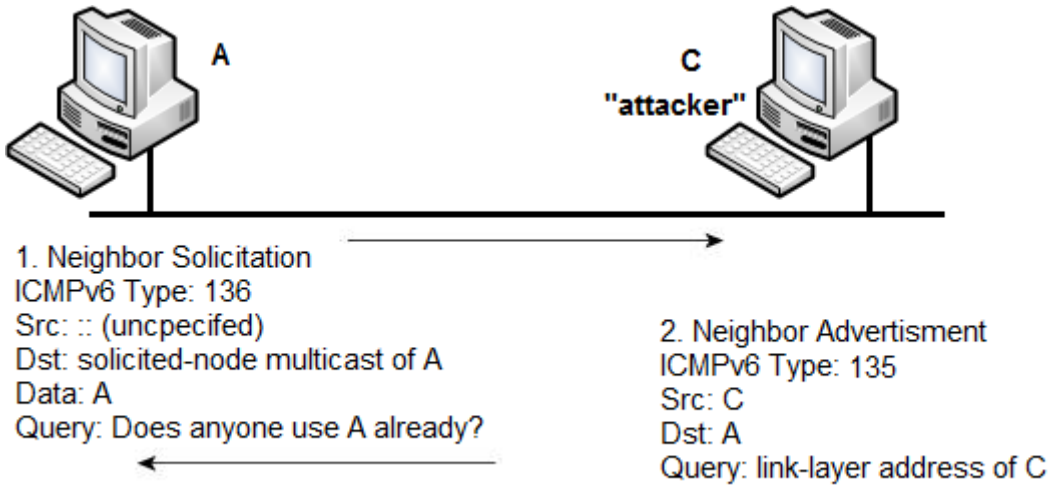
En hacker kan skicka ut falska NS meddelanden med felaktig käll länklager adress som alternativ, eller varför inte ett NA meddelande med fel mål, länklager adress.
I och med detta så skulle den noden få felaktig information i sin granne Cache. Vilket leder till att det blir en MITM attack där lösenord och annan känslig information kan sniffas upp.

2.2.3 NUD misslyckande

NUD mekanismen används ju för att kolla länklokala tillgängligheten hos andra noder. Normalt sätt så bruka noderna förlita sig på högre övre lager för att avgöra om peer noder fortfarande är tillgängliga. Men om det går en tillräckligt lång tid för trafiken övre lagers skikt, eller om noden slutar att få svar, så kommer NUD mekanismen att bli involverad. Efter ett antal försök så kommer peer noden att försvinna ur granncachen. Hackern kan då svara NS med falska NA meddelanden med en falsk länklager adress som gör att ingen kommunikation kommer att uppstå. Detta leder till en DoS attack.

2.2.4 Duplicate Address Detection DoS

I ett nätverk där en host försöker att erhålla en adress med stateless adress autokonfiguration så kan en hacker starta en DoS attack med att svara med NA till alla DAD försök. Om hackern hävdar denna adress så kommer hosten aldrig att bli erhållen någon giltig adress. Denna attack är faktiskt rätt farlig då den kan används till multicast som gör att den når ut på hela local-link. Vilket i sin tur då leder till att ett stort antal noder kan få kommunikations problem.



Figur 7: DAD attack

2.3.1 Routing attacker

Här förklaras det olika teknikern för att attackera RD och andra relevanta funktioner kring just routern.

2.3.2 Skadlig sista Hopp Router

Skadlig sista hopp router(English: Malicious Last Hop Router) är en teknik där en hacker kan omdirigera all trafik som skickas genom den. Hackern maskeras som sista-hopp router genom att skicka ut legitima RA meddelanden antingen som unicast eller multicast. Om nu en host skulle välja hackern som standard router så kan hackern leda bort trafiken och kan användas som proxy men även för att avlyssna trafiken.

2.3.3 Standard routern är död

Standard routern är död(English: default router is ”killed”) är när en hacker kan dödar standard routern för andra noder. Om default routern inte existerar hos en node så kommer den tro att destinations noden är direktansluten. Noden kommer då att försöka skicka paket direkt alltså inte via någon router. Efter det så kan hackern använda NS/NA förfalskning även med icke direktanslutna destinationer.

Det finns två tillvägagångs sätt för att döda en standard router. Den ena är att utföra en DoS attack så routern inte kommer att svara något mer. Den andra är att skicka förfalskade RA precis som i Malicious Last Hop Router.

2.3.4 Säker router kan äventyras

Säker router kan äventyras(English: **Good Router Goes Bad**) är om en hacker skulle komma över en router, fysiskt så skulle alla paket att

äventyras. Kanske skulle man kunna införa router rättigheter, detta är som sagt en frågeställning och det finns i dags läget ingen generell lösning till detta fall.

2.3.5 Falska omdirigerade meddelanden

Falska omdirigerade meddelanden(English: Spoofed redirected messages) innebär att vem som helst kan skicka ut förfalskade omdirigerade meddelanden på den lokala länken. Detta är ett stort problem för det finns inga rättigheter för att säkra upp det hela.

2.3.6 Falska direktansluta prefix

Falska direktansluta prefix(English: Bogus On-Link Prefix) En hacker kan skicka ut felaktiga RA meddelanden specificerade med icke existerade prefix. Hostar som accepterar dessa RA meddelanden kommer då tro att prefixen är direktanslutna. Då kommer hosten att försöka kontakta dem direkt utan att gå via en router när den ska utföra NS/NA utbyte. Om inte hackern förfalskar i gensvar med ett NS så vill ursprungs hosten vara kvar på länken men har ingen att kommunicera med, DoS attack. Om hackern nu utnyttjar ett falskt prefix med mer falska NA meddelanden så kan detta leda till en potentiell MitM attack.

2.3.7 Falska adress konfigurations prefix

Falska adress konfigurations prefix(English: Bogus Adress Configuration Prefix) är snarlikt föregående attack, ett förfalskat och ogiltigt nätverks prefix kan skickas ut med RA meddelanden till en host som försöker med autokonfiguration. Hosten kommer utifrån ogiltigt prefix skapa en adress som inte tillhör nätverket och därmed tappa kommunikation.

2.3.8 Parameter Spoofing

Hostar kan skicka ut RA meddelanden med extra parametrar t.ex. säg till om de ska använda statefull adressconfgiration. En hacker kan då skicka ut RA meddelanden som ser ut att komma från en default router med felaktiga parametervärden. Säg att hackern skickar ut massvis med sådana som säger att använda en DHCPv6 server som inte existerar, vilket då leder till att hosten aldrig kommer att få en användbar IPv6 adress.

2.4 Replay och remote attacker

2.4.1 Replay attacks

Efter att en host går offline så kan en hacker ta över en gammal hostar adress och där med att initiera nya förbindelser med routrar och andra noder som hosten kommunicerat med tidigare. Med hjälp av SEND så kan

det lösas med nonces och timestamps, förutsatt att noderna är synkroniserade.

2.4.2 ND DoS Attack

Men denna attack så kan en hacker skicka ut massivt med ND förfrågningar till den lokala routern. Routern kommer då vara upptagen med att svara på dessa, och andra hostar förfrågningar kommer därmed bli försenade och kanske till och med ignorerade helt.

2.4.3 Summering över det olika attckerna

Attack	N/R	R/D	Msgs	1	2	3
NS/NA spoofing	ND	Redir	NA NS	+	+	+
NUD failure	ND	DoS	NA NS	-	+	+
DAD DoS	ND	DoS	NA NS	-	+	+
Malicious router	RD	Redir	RA RS	+	+	R
Default router killed	RD	Redir	RA	+/R	+/R	R
Good router goes bad	RD	Redir	RA RS	R	R	R
Spoofed redirect	RD	Redir	Redir	+	+	R
Bogus onlink prefix	RD	DoS	RA	-	+	R
Bogus address config	RD	DoS	RA	-	+	R
Paramter spoofing	RD	DoS	RA	-	+	R
Replay attack	All	Redir	All	+	+	+
Remote ND DoS	ND	DoS	NS	+	+	+

Tabell 3: ND Trust Models and Threats

(1) Företags nätverk
(2) Publikt nätverk
(3) Ad hoc nätverk
(-) Hotet är inte närvarande eller inte ett problem.
(+) Hotet är närvarande och minst en lösning är känd.
(R) Hotet är närvarande men att lösa det är ett forskningsproblem.

2.5 Secure Neighbor Discovery (SEND)

Förhoppningsvis efter att ha läst föregående sektioner så har ni nu lärt er hur ND protokollet fungerar samt fått en inblick hur känsligt det är för en mängd olika sårbarheter.
ND protokollet har ju ingen mekanism för autentisering utan den har enbart två väldigt måttliga säkerhets mekanismer. Den första är att ND är ju ett länklokalt protokoll och käll adressen måste vara antigen den ospecificerade adressen eller en länklokal adress. Den andra är att antal hop måste vara satt till 255 och det skyddar ju emot att paket kan bli injekterad. Dessa mekanismer är absolut inte tillräckligt säkra för att skydda hela ND protokollet.

Det var tänkt från en början att IPsec skulle lösa problemet, men så var inte fallet. Precis som vi beskriv tidigare så är IPsec väldigt olämpligt för syftet att säkra ND protokollet. Syftet med ND är ju att automatiskt konfigurera adresser utan någon som helst manuell konfiguration. IPsec kan inte lösa problemet fullt ut automatiskt eftersom IPsec behöver IKE för sätta upp SA och IKE behöver en giltig IPv6 adress för att möjliggöra kommunikationen mellan IPsec parterna. Det var därför IETF introducerade SEND, som är en säkerhets funktion för ND protokollet och till det introduceras det två nya meddelanden samt olika alternativ.

Målet med SEND är att använda en asymmetrisk kryptering alltså privat och publika nycklar för att skydda emot autentisering och identitet utan att behöva konfigurera. SEND är baserad på CGA och är en IPv6 adress som är kryptografiskt generad med hjälp av en one-way hash funktion från en noder publika nyckel och ett antal parametrar. CGA är alltså en funktion som knyter samman en IPv6 adress och den publika nyckeln som gör att det går att autentisera noder.

2.5.1 SEND alternativ

Till SEND så kommer det ett par olika alternativ som man kan välja att ha med i sitt SEND meddelande, där vissa är obligatoriska. De olika alternativen är CGA, Rivest Shamir Adlema (RSA) signatur, timestamp, nonce, och trust anchor. Tillsammans med dessa komponenter så ska man försvara sig emot attacker för identitet och integritet. Nedan beskrivs det olika alternativen lite mer utförligt.

- CGA används för att kontrollera så att avsändarens ND meddelande är den som den hävdar. En publik-privat nyckel par är genererad av alla noder innan de kan hävda en adress.

- RSA signatur försäkrar integriteten hos meddelanden. Innehåller information om en publik nyckel som är till grund för signaturen. Det används för att säkerhetsställa alla meddelanden till Neighbor och Router Discovery.
- Timestamp säkerhetsställer så att en hacker inte kan fånga upp advertisments för att sedan spela upp dem när informationen inte längre är giltig. På så vis krävs det att noden är tidssynkroniserade med t.ex. Network Time Protocol (NTP). Det ska noteras att det kan vara bra att skydda NTP meddelanden kryptografiskt antingen med asymmetrisk kryptering eller med publik nyckel infrastruktur(Public Key Infrastructor (PKI)). RSA signaturen garanterar så att timestamp inte kan manipuleras.

- Nonce är till för att försäkra så att advertisment inte är manipulerad på något sätt. Solicitation meddelandet innehåller ett nonce som är ett random värde som ska vara minst 6bytes långt. Eftersom detta värde är rätt stort är det ganska osannolikt att två meddelanden under en kort tidsperiod innehåller samma nonce.

- Trust anchor innehåller information som en host behöver när den ska identifiera en trust anchor, alltså autentisera en router. Detta alternativ används enbart i certifierings vägs värvning(English: Certification Path Solicitation (CPS)) och certifierings vägs annonsering(English: Certification Path Advertisment (CPA)) meddelanden.

- Certificate används enbart i CPA meddelanden och är till för routrar att skicka ut en förteckning med certificates

2.5.2 SEND meddelanden

Authorization Delegation Discovery (ADD) är en mycket viktigt mekanism till SEND. Vem som helst kan ju sätta upp en router och skapa egna nycklar. Det var därför ADD mekanismen blev introducerad till för att på ett säkert sätt kunna autentisera en router. ADD mekanismen fungerar så att en host kan autentisera en router och se efter att den är tillåten att annonsera specifika prefix. Hur det fungerar kan ni se efter i *figur 6*.

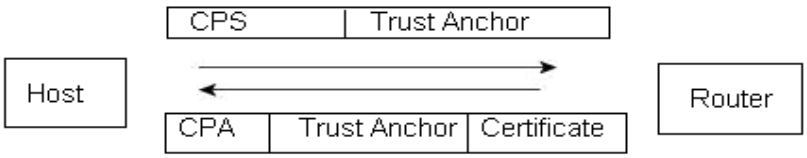
Till ADD så kommer det två olika ICMPv6 meddelanden:

- Certification Path Solicitation
  Certification Path Solicitation (CPS) meddelandet skickas från hostar till routrar för att efterfråga certification paths.

- Certification Path Advertisment
  Certification Path Advertisement (CPA) medelandet skickas ut av routern som gensvar av CPS.

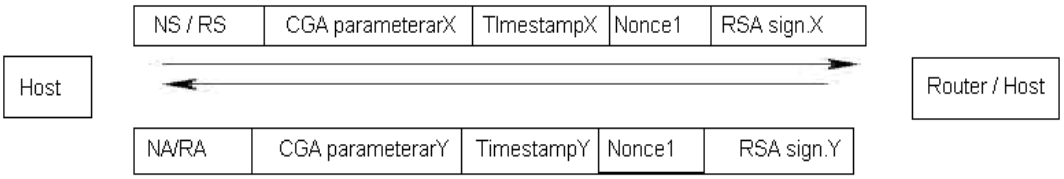
När en SEND node ansluter till ett nätverk där SEND används så kommer den skicka ut ett RS meddelanden för att få information om eventuella nätverks prefix.

Om noden nu får ett RA meddelanden tillbaka så kommer noden att verifiera routern och validera prefixen. Om den nu accepterar autentiseringen så skickas ett CPS meddelanden där trust anchor alternativet är inkluderat. Sen är det routern som svarar med ett CPA meddelanden med tillhörande certifikat vägar som noden behöver för att verifiera routern. Efter att noden verifierat certifikat vägarna så anses routern som autentiserad. Noden kommer nu att skapa CGA adress med hjälp av prefixet och ett par andra parametrar. Noden kan nu kommunicera med sina grannar med SEND meddelanden.



Figur 8: CPA och CPS meddelanden

Nu när noden skickar ett NS eller RS meddelande till en router eller node så kommer den behöva skicka med följande alternativ CGA parametrar, RSA signatur, timestamp, nonce. Svaret kommer antingen från ett NA eller RA meddelande och kommer med samma nonce värde för att matcha. Notera att NA eller RA medlanden inte behöver nonce t.ex. inte när en node byter adresser (IPv6, länklager, CGA).



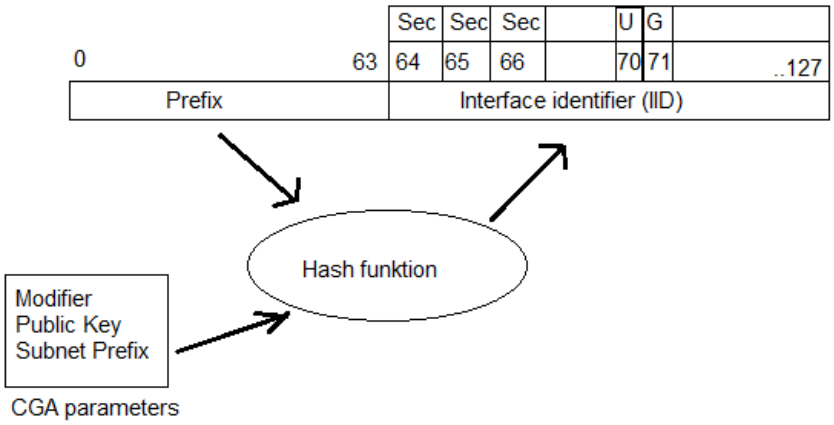
Figur 9: NS/RS och NA/RA meddelande

2.5.3 CGA parameter strukturen

En CGA adress ser ut som en helt vanlig IPv6 adress, den är uppdelad i två delar om 64bitar vardera. De första 64bitarna är nätverkets prefix, själva subnet värdet och de resterande 64bitarna är IID. Då det är relativt enkelt med dagens datorer att utföra brute-force attacker, 64bitars IID är inte direkt tillräckligt. Därför överkom CGA med 3bitar baserad på collisions count, det är att försvårar emot brute-force och de bitarna kallas för Security (Sec) bits.

I IPv6 adresser så är bitarna 7 respektive 8 av IID speciella flagor. Bit 7 menas med universal eller local och kallas vanligen för "U" biten. Bit 8 kallas för "G" och är satt till 0 om det är unicast eller 1 om det är multicast. Det finns ju inga globala multicast adresser, som just uppstår när både U och G bitarna är satta till 1 och som [3] beskriver så ska CGA adresser använda dessa bitar.

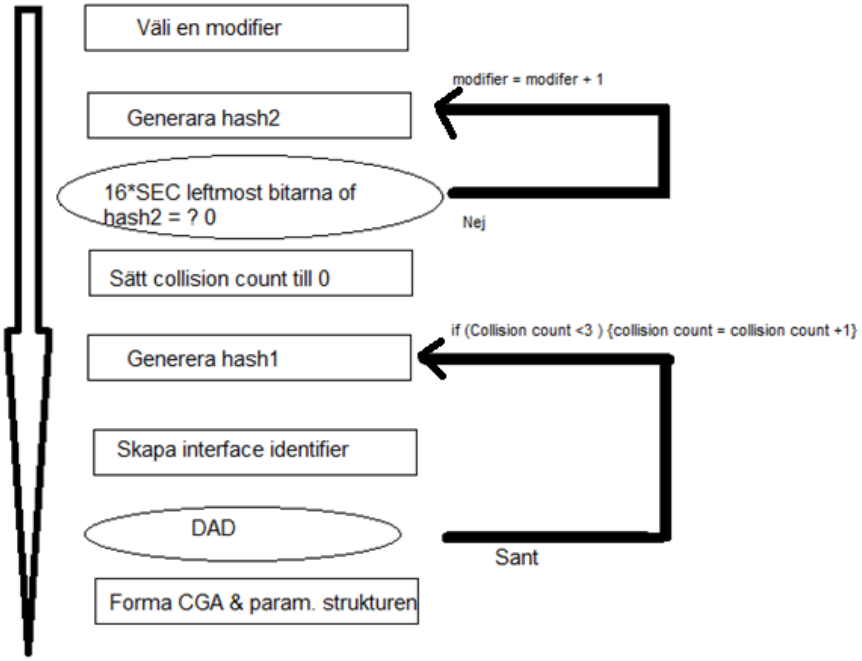
- **Modifer** innehåller 128bitars random värde som väljs vid CGA genereringen.
- **Subnet prefix** innhåller ett subnet värde
- **Publik nyckel** innehåller den publika nyckeln av hosten och är Distinguished Encoding Rules (DER) kodad och genererad av RSA med minst 384bitar.



Figur 10: CGA process

2.5.4 CGA genererings processen

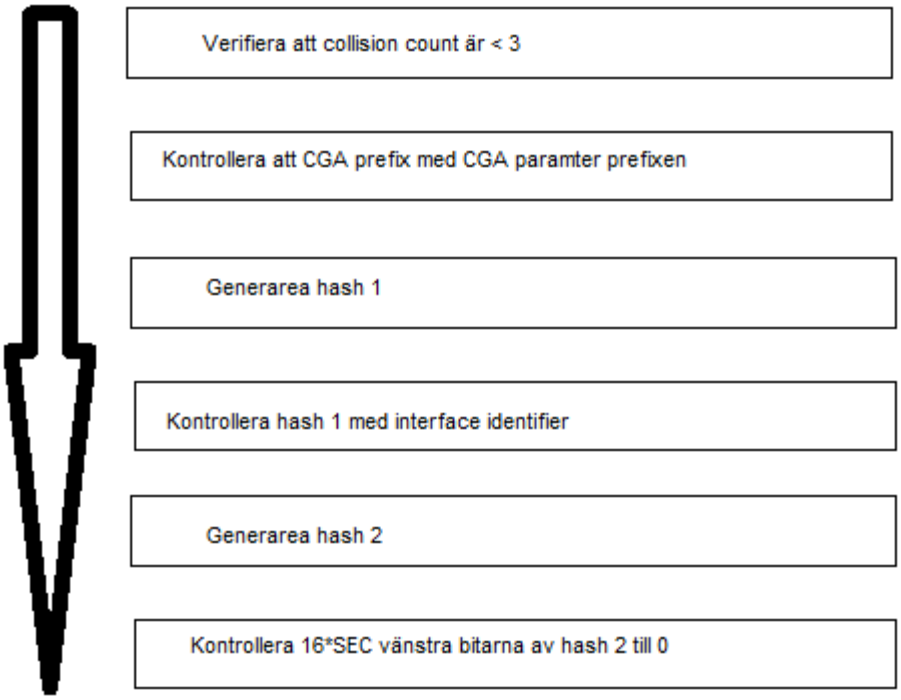
CGA genererings processen är beroende av tre olika parametrar, en publik nyckel, subnet prefix samt SEC bitarna. Själva processen går ut på att skapa två specifika hashar. Det inleds med att generera en 128bitars modifier. Sen skapas en $hash_2$ med SHA-1 och ser till så att collision count värdet är satt till 0 men även subnet prefixet till 0. Sen ska 16*Sec bitarna (längst från vänster) av $hash_2$ bli lika med 0 för att gå till nästa steg i processen annars minska med 1 och börja om. Efter att det är gjort så ska $hash_1$ skapas utifrån en modifier, subnet prefix, collision count samt den publika nyckeln. $Hash_1$ kommer nu att bli IID och bitarna Sec, U och G kommer få värdet 0. Sen formar man subnet prefixet och IID till en komplett IPv6 adress. Till sist så görs en DAD för att avgöra om adressen är unik.



Figur 11: Gernerera CGA algoritm

2.4.5 CGA verifierings processen

När en node får ett paket från en annan SEND node så kommer noden att inleda en verifierings process. Paket ska innehålla ett CGA alternativ samt tillhörande CGA parametrar. Verifierings processen börjar med att den kollar så att collision count värdet är mindre än 3. Efter det så kontrolleras det att subnet prefixet från käll IPv6 adressen är den samma som finns i CGA parametrarna. Sen ska $hash_1$ räknas ut och jämföras med IID med undantag Sec bitarna och U och G bitarna. När det är klart så extraheras Sec bitarna ut från IID samt $hash_2$ räknas ut. Till sist kontrolleras det att de 16*Sec bitarna är lika med 0. Om något i denna CGA verifierings process går fel så kommer processen genast att avslutas annars om så är den verifierad och paketet kommer att accepteras.



Figur 12: Verifiera CGA algoritm

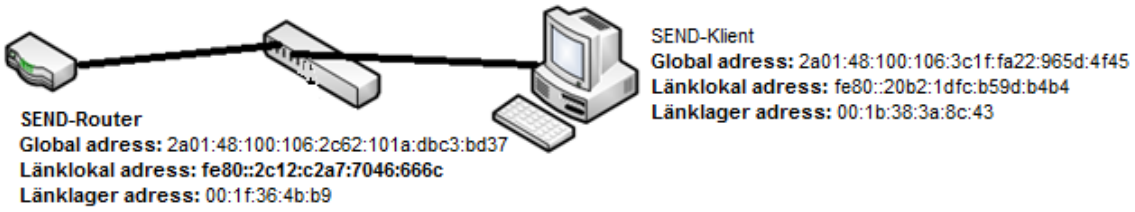
3. Implementation

Labbmiljön kommer att bestå av en router, switch samt två klienter. Routern och de två klienterna kör operativsystemet Linux med distributionen *Ubuntu* version *10.10* samt en kernel *2.6.35-22-generic-pae*. I första sektionen kommer enbart installation och konfigurationen av NDprotector att genomföras. Sedan kommer en DAD DoS attack att genomföras och till sist se efter hur NDprotector tar hand om attacken.

3.1 Installation och konfigurering av NDprotector

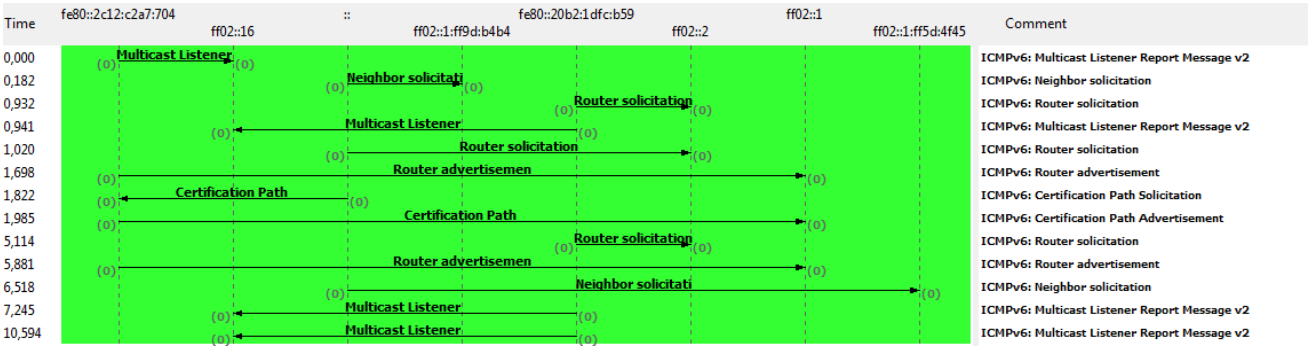
Nu är det god tid för att sätta upp NDprotector som är den SEND implementation jag har valt att använda i min labbmiljö. I appendixet hittar du steg för steg hur man gör för att sätta upp NDprotector. Som router demon har jag använt mig utav *radvd* och konfigurationsfilen hittar du i appendixet. I denna sektion kommer vi enbart att sätta upp NDprotector mellan routern och en klient för att forska vidare hur SEND fungerar med diverse olika meddelanden. Notera att routern kör en IPv6-in-IPv4 tunnel från tunnelförmedlaren *IPv6tf* och konfigurationen till den hittar du i appendixet.

Nu när allt är konfigurerat och klart så kommer vi att se att de länklokala adresserna inte är lik länklager adresserna utan den är numera en CGA adress.



Figur 13: SEND labbmiljö

Nedan i figuren ser du en bra bild där klienten frågar routern med ett CPS meddelanden och får i gensvar ett CPA meddelande. Hur ett CPA meddelande ser ut ser du i figuren 14.



Figur 14: SEND meddelanden

Nedan i tabellen så ser vi att det tar en viss tid för CGA adressen att generas.

```
root@router:/etc# ndprotector.py -v
Verbose output enabled
Reading configuration file /etc/NDprotector/sendd.conf
ECC support is available
Generating CGA
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
Sent 1 packets.
flushing all addresses on interface eth1
assigning 2a01:48:100:106:2c62:101a:dbc3:bd37 to interface eth1
CGA address 2a01:48:100:106:2c62:101a:dbc3:bd37 computed and
assigned in 5.261644 seconds
Generating CGA
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
Sent 1 packets.
assigning fe80::2c12:c2a7:7046:666c to interface eth1
CGA address fe80::2c12:c2a7:7046:666c computed and assigned in
7.083882 seconds
available plugins: []
setting filtering rules
Neighbor Cache initialized
flushing kernel neighbor cache
running In/Out/CPSCPA NFQueues
signing an outgoing RA message
No matching Signature Algorithm to sign the message, using RSA/SHA-
1…
dropping an unsecured RS message
signing an outgoing RA message
No matching Signature Algorithm to sign the message, using RSA/SHA-1
sending a CPA message
.
Sent 1 packets.
```

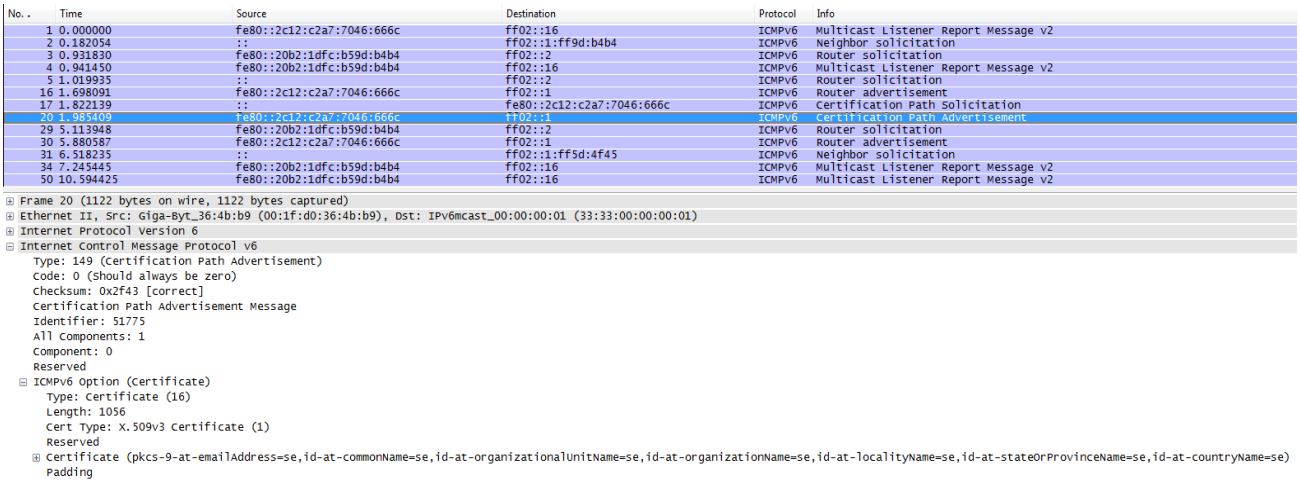
```
NC updating an entry with an secured RS message from the address:
fe80::20b2:1dfc:b59d:b4b4
signing an outgoing RA message
No matching Signature Algorithm to sign the message, using RSA/SHA-1
```

Tabell 4: SEND-router

```
root@klient:/etc/NDprotector# ndprotector.py -v
Verbose output enabled
Reading configuration file /etc/NDprotector/sendd.conf
ECC support is available
Generating CGA
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
Sent 1 packets.
flushing all addresses on interface eth0
assigning fe80::20b2:1dfc:b59d:b4b4 to interface eth0
CGA address fe80::20b2:1dfc:b59d:b4b4 computed and assigned in
2.651621 seconds
available plugins: []
setting filtering rules
Neighbor Cache initialized
flushing kernel neighbor cache
Certification Cache starting
.
Sent 1 packets.
Sending an RS on interface eth0
running In/Out/CPSCPA NFQueues
RA contains following prefix(es): ['2a01:48:100:106::']
Sending a CPS message
storing ID 51775 for a new CPS
.
Sent 1 packets.
Receiving a CPA message
checking ID 51775 against a previously sent CPS
storing on cert for Certificate Path #51775
Verifying certification path for #51775
We received a complete and valid Certification Path
signing an outgoing RS message
No matching Signature Algorithm to sign the message, using RSA/SHA-1
RA contains following prefix(es): ['2a01:48:100:106::']
Public Key associated to the signature corresponds to a certificate
NC updating an entry with an secured RA message from the address:
fe80::2c12:c2a7:7046:666c
Generating CGA
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
```

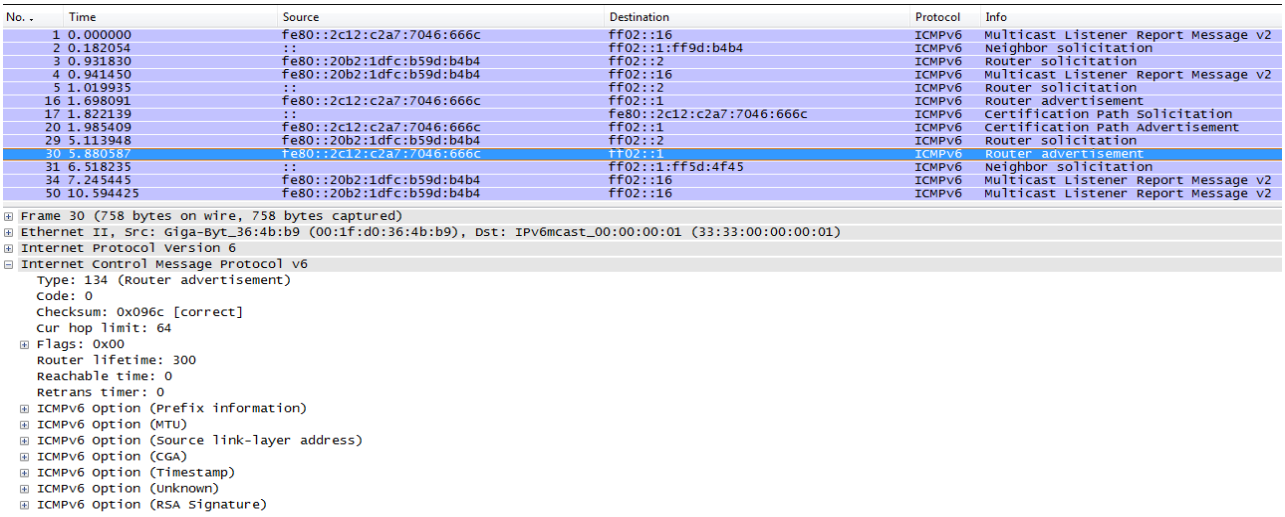
```
Sent 1 packets.
assigning 2a01:48:100:106:3c1f:fa22:965d:4f45 to interface eth0
CGA address 2a01:48:100:106:3c1f:fa22:965d:4f45 computed and
assigned in 1.308976 seconds
created a new address (2a01:48:100:106:3c1f:fa22:965d:4f45) for
prefix 2a01:48:100:106::
```

Tabell 5: SEND-klient



Figur 15: CPA meddelande

Här kan vi se hur klienten skickar ett CPS meddelanden till routern. Routern kommer nu att snabbt svara tillbaka med ett CPA med tillhörande certifikat. Efter att certifikatet anlänt så kommer klienten att verifiera CPA med det CPS meddelande som skickades tidigare. Nu när det stämmer så kommer klienten skicka ett RS meddelande med tillhörande SEND parametrarna. Routern kommer sedan att svara med ett RA meddelande precis som ni kan se i figuren nedan. Om det nu hade varit ett felaktigt certifikat så kommer den att ignorera meddelandet.



Figur 16: RA med SEND relaterade parametrar

Om vi testar att skicka iväg några ICMPv6 echo requests med hjälp av *ping* kommandot från en host med icke-SEND aktiverat så kommer SEND-klienten att ignorera meddelandet och till sist slänga det.

```
params argument is not a CGAParams structure
an ingoing packet has failed CGA verification test
NC not updating an entry with an unsecured NA message from the
address: fe80::a00:27ff:fe54:7a5e
```

Tabell 6: NA meddelande från en icke SEND klient.

Om vi kör kommandot *ip -6 neigh* snabbt efter varandra så ser vi hur tillstånden i grannetabellen hoppar mellan otillgänglig och felaktig.

```
root@klient:/# ip -6 neigh
fe80::247e:a98f:ce39:a44e dev eth0 lladdr 00:1f:d0:36:4b:b9 router
STALE
fe80::a00:27ff:fe54:7a5e dev eth0  INCOMPLETE
root@laptop:/home/#
```

Tabell 7: Granne cache med tillståndet otillgänglig

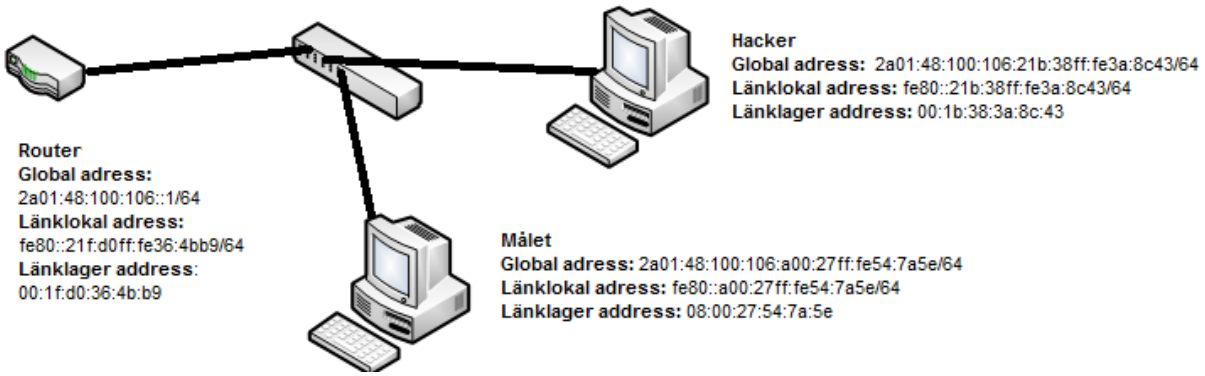
```
root@klient:/# ip -6 neigh
fe80::247e:a98f:ce39:a44e dev eth0 lladdr 00:1f:d0:36:4b:b9 router
STALE
fe80::a00:27ff:fe54:7a5e dev eth0  FAIL
root@laptop:/home/#
```

Tabell 8: Granne cache med tillståndet felaktigt

3.1.2 Slutsats

NDprotector fungerade precis som teorin säger vilket man kunde se under de olika testerna. Det fungerar precis som den teorin med alla meddelanden i sektion 2.4.2. Om du tittar i tabellen 8 så ser vi ett tillstånd i granntabllen ”FAIL” det har jag inte nämnt i teoridelen. Det är Linuxkärnan som valt att ha med det, jag tycker det är användbart att den påpekas att den misslyckades med tanke på att annars hade det varit lite förvirrande om den nu hade blivit borttagen direkt.

3.2 DAD DoS Attack utan SEND



Figur 17: DAD DoS labbmiljö

I länklokala nätverk där stateless adress autokonfiguration används kan en hacker enkelt utföra en DoS attack förutsatt att ingen autentisering används. Det fungerar så att ett program kommer att svara på alla DAD försök, om hackern hela tiden begär den hävdade adressen så kommer målet aldrig att få en giltig adress. Dessa attacker kan leda till riktigt stora problem för att dessa utförs med multicast och det leder ju till alla noder på det lokala nätverk samtidigt. Programmet som kommer att används för att utföra denna attack är utvecklat av THC-attack och finns i deras IPV6-toolkit, programmet heter dos-new-ip6.

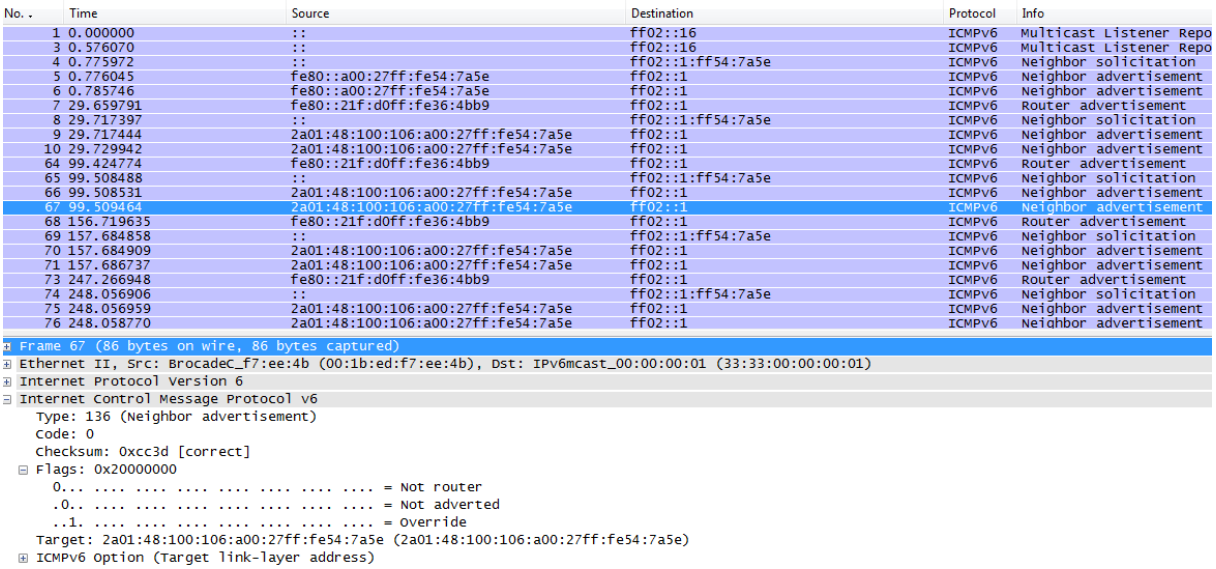
Nu startar vi programmet dos-new-ip6. Här ser vilka IPv6 adresser som mål maskinen försöker associera till sitt nätverkskort. Den börjar med länklokala adressen därefter globala adressen som den svarar RS med ett RA meddelande.

```
root@hacker:~/thc-ipv6-1.2# ./dos-new-ip6 eth0
Started ICMP6 DAD Denial-of-Service (Press Control-C to end) ...
Spoofed packet for existing ip6 as fe80::a00:27ff:fe54:7a5e
Spoofed packet for existing ip6 as 2a01:48:100:106:a00:27ff:fe54:7a5e
Spoofed packet for existing ip6 as 2a01:48:100:106:a00:27ff:fe54:7a5e
```

```
Spoofed packet for existing ip6 as 2a01:48:100:106:a00:27ff:fe54:7a5e
Spoofed packet for existing ip6 as 2a01:48:100:106:a00:27ff:fe54:7a5e
```

Tabell 9: DOS hacker program

Vi börjar med att avaktivera nätverksgränssnittet *eth0* på mål maskinen, sedan aktiverar det igen. Det görs för att vi vill att autokonfigurationen ska starta.



Figur 18 Dos DAD attack

Här ser vi i det falska NA meddelandet att den har en intressant flagga nämligen ”1” som gör att den skiver över alla värden i grannetabellen.

Testade även att lägga till både en global och länklokal adress manuellt.

```
root@victim:~# ip -6 addr add 2a01:48:100:106:a00:27ff:1111:1337/64
dev eth0
root@victim:~# ip -6 addr add fe80::a00:27ff:fe54:1337/64 dev eth0
```

Tabell 10: manuellt tilldelning av IPv6 adresser

Här nedan i tabellen ser vi hur alla adresser är dadfailed.

```
root@victim:~# ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
inet6 2a01:48:100:106:a00:27ff:fe54:7a5e/64 scope global tentative
dadfailed
valid_lft forever preferred_lft forever
inet6 2a01:48:100:106:a00:27ff:1111:1337/64 scope global tentative
```

```
dadfailed
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe54:1337/64 scope link tentative dadfailed
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe54:7a5e/64 scope link tentative dadfailed
valid_lft forever preferred_lft forever
root@victim:~#
```

Tabell 11: IPv6 adress tillstånd

```
root@victim:~# ping6 -I eth0 fe80::21f:d0ff:fe36:4bb9
connect: Cannot assign requested address
root@victim:~#
```

Tabell 10: Misslyckat ping6 försök

Försöker att komma åt routern genom att ping:a men det går ju inte på grund av att målet inte har någon giltig unicast IPv6 adress vilket behövs.

Det ska noteras att den har olika påverkan då målet kan använda olika typer av adresser.

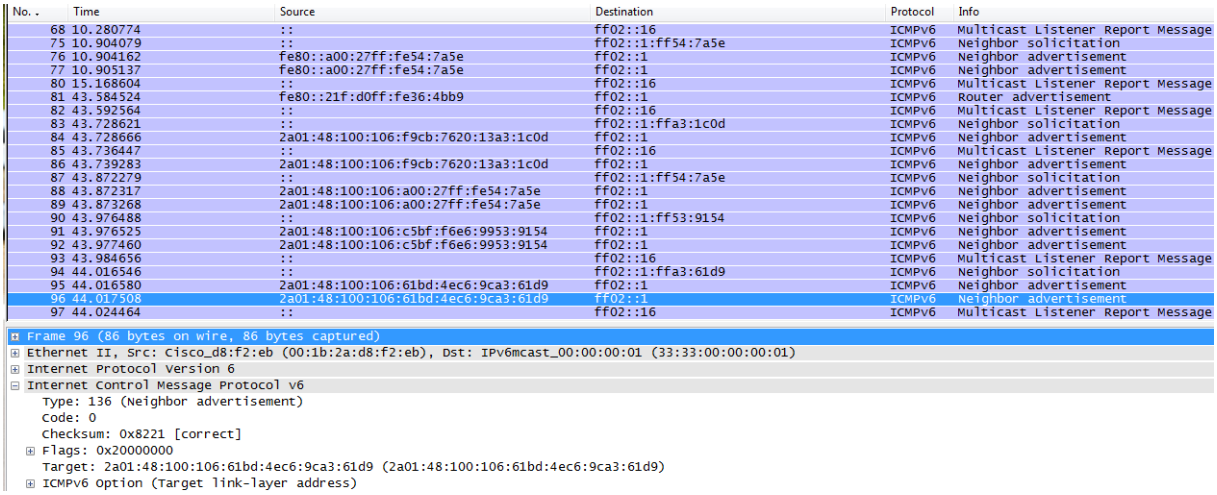
- I stateless adress autokonfiguration som beskrivs i RFC 4861 och 2462 så är det en länklokal adress som blir formad efter IID hårdvaru adress (EUI-64) och den ska i princip vara unik.

- RFC 3041 och 4911 beskriver även så kallade temporära adresser som genererarnas slumpmässigt. Om den efterfrågade adressen redan är upptagen så kommer nya adresser att genereras. RFC 4911 som är den senaste kommer att utföra upp till 5 olika adresser som standard.

Nu när målet använder sig utav temporära adresser så ser det ut så här när hackern försöker köra en DAD DoS attack.

```
root@hacker:~/thc-ipv6-1.2# ./dos-new-ip6 eth0
Started ICMP6 DAD Denial-of-Service (Press Control-C to end) ...
Spoofed packet for existing ip6 as fe80::a00:27ff:fe54:7a5e
Spoofed packet for existing ip6 as
2a01:48:100:106:f9cb:7620:13a3:1c0d
Spoofed packet for existing ip6 as 2a01:48:100:106:a00:27ff:fe54:75ae
Spoofed packet for existing ip6 as
2a01:48:100:106:c5bf:f6e6:9953:9154
Spoofed packet for existing ip6 as
2a01:48:100:106:61bd:4ec6:9ca3:61d9
```

Tabell 13: dos-new-ip6 hacker program



Figur 19 DoS attack med temporära adresser

3.2.2 Slutsats

Dos-new-ip6 tar enbart ett argument och det är nätverkskortet. Det är det som ska lyssna efter ND meddelanden och svara tillbaka med falska ND meddelanden. DAD förlitar sig på ND protokollet med ingen som helst autentisering och därmed kan en hacker enkelt utföra en DoS attack.

Vi kan se efter att målet har aktiverat nätverkskortet så kommer autokonfigurations proceduren att starta igång. Den skickar ut NS meddelande till multicast adressen som alla IPv6 noder lyssnar på. Sen när ett RA kommer så kommer den snabbt att svara två NA meddelanden. När privacy extensions användes så ser vi att den generar globala IPv6 adresser men även där så spoofas adresserna. För övrigt kan vi notera att privacy extensions används i Windows 7 som standard. Dos attacken är väldigt enkel att utföra och är verkligen en effektfull attack. Tänkt dig i öppna nätverk där vem som helst kan ansluta sig till med minimal autentisering eller ingen länklager autentisering. NA meddelanden kommer att skickas med flaggan ”override” vilket betyder att alla kommer att skriva över deras nuvarande värde i deras grannetabeller. Det finns två sätt: att lyssna efter NS meddelanden med den ospecificerade adressen som mål adress och genast svara med ett spoofat NA meddelande. Tabell 3 säger att attacken inte är ”concern” i nätverk där nodern är pålitliga, men om en node äventyras så kommer den att bli oskyddad. I både publika nätverk samt ad hoc nätverk så är detta ett stort problem och en lösning finns vilket är SEND.

3.3 DAD DoS Attack med SEND

Vi utgår från samma labbmiljö som i figur 15. Det är bara att NDprotector är konfigurerat på samtliga enheter.

Vi börjar med att aktivera NDprotector på routern, samt köra igång dos-new-ip6 på hacker klienten se tabell 16. När det är klart så kör vi igång NDprotector på mål klienten, här nedan i tabellen ser man vad det är som händer.

```
root@victim:~# ndprotector.py -v
WARNING: No route found for IPv6 destination :: (no default route?)
Verbose output enabled
Reading configuration file /etc/NDprotector/sendd.conf
ECC support is available
Generating CGA
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
Sent 1 packets.
collision on fe80::38c5:ba6d:11e2:13df detected
trying DAD on new address fe80::2ca0:9312:b1cd:6238
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
Sent 1 packets.
flushing all addresses on interface eth0
assigning fe80::2ca0:9312:b1cd:6238 to interface eth0
CGA address fe80::2ca0:9312:b1cd:6238 computed and assigned in
4.793120 seconds
available plugins: []
setting filtering rules
Neighbor Cache initialized
flushing kernel neighbor cache
Certification Cache starting
.
Sending an RS on interface eth0
running In/Out/CPSCPA NFQueues
RA contains following prefix(es): ['2a01:48:100:106::']
Sending a CPS message
storing ID 24483 for a new CPS
.
Sent 1 packets.
Receiving a CPA message
checking ID 24483 against a previously sent CPS
storing on cert for Certificate Path #24483
Verifying certification path for #24483
We received a complete and valid Certification Path
RA contains following prefix(es): ['2a01:48:100:106::']
Public Key associated to the signature corresponds to a certificate
NC updating an entry with an secured RA message from the address:
fe80::34d3:6d33:af87:5cd4
Generating CGA
```

```
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
Sent 1 packets.
collision on 2a01:48:100:106:3400:92a1:cf10:a7f3 detected
trying DAD on new address 2a01:48:100:106:340f:34aa:b088:5971
No matching Signature Algorithm to sign the message, using RSA/SHA-1
.
Sent 1 packets.
params argument is not a CGAParams structure
one packet with an invalid CGA address was received during the DAD
procedure
params argument is not a CGAParams structure
one packet with an invalid CGA address was received during the DAD
procedure
assigning 2a01:48:100:106:340f:34aa:b088:5971 to interface eth0
CGA address 2a01:48:100:106:340f:34aa:b088:5971 computed and
assigned in 4.935388 seconds
created a new address (2a01:48:100:106:340f:34aa:b088:5971) for
prefix 2a01:48:100:106::
signing an outgoing RS message
No matching Signature Algorithm to sign the message, using RSA/SHA-1
params argument is not a CGAParams structure
an ingoing packet has failed CGA verification test
NC not updating an entry with an unsecured NA message from the
address: 2a01:48:100:106:3400:92a1:cf10:a7f3
signing an outgoing RS message
No matching Signature Algorithm to sign the message, using RSA/SHA-1
params argument is not a CGAParams structure
an ingoing packet has failed CGA verification test
NC not updating an entry with an unsecured NA message from the
address: 2a01:48:100:106:3400:92a1:cf10:a7f3
params argument is not a CGAParams structure
an ingoing packet has failed CGA verification test
NC not updating an entry with an unsecured NA message from the
address: 2a01:48:100:106:340f:34aa:b088:5971
params argument is not a CGAParams structure
an ingoing packet has failed CGA verification test
NC not updating an entry with an unsecured NA message from the
address: 2a01:48:100:106:340f:34aa:b088:5971
RA contains following prefix(es): ['2a01:48:100:106::']
Public Key associated to the signature corresponds to a certificate
NC updating an entry with an secured RA message from the address:
fe80::34d3:6d33:af87:5cd4
changing valid and prefered of 2a01:48:100:106:340f:34aa:b088:5971
to 86400 and 14400
unsetting filtering rules
root@victim:~#
```

Tabell 11: Ndprotector under DAD DoS attack

```
root@laptop:~/thc-ipv6-1.2# ./dos-new-ip6 eth0
Started ICMP6 DAD Denial-of-Service (Press Control-C to end) ...
Spoofed packet for existing ip6 as fe80::38c5:ba6d:11e2:13df
Spoofed packet for existing ip6 as fe80::2ca0:9312:b1cd:6238
Spoofed packet for existing ip6 as
2a01:48:100:106:3400:92a1:cf10:a7f3
Spoofed packet for existing ip6 as
2a01:48:100:106:340f:34aa:b088:5971
```

Tabell 125: Hacker DAD DoS klienten

```
root@victim:~# ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2a01:48:100:106:340f:34aa:b088:5971/64 scope global
dynamic
       valid_lft 86389sec preferred_lft 14389sec
    inet6 fe80::2ca0:9312:b1cd:6238/64 scope link nodad
       valid_lft forever preferred_lft forever
root@victim:~#
```

Tabell 136: Mål klienten adress status

Nedan i tabellen ser vi hur mål maskinen kommer åt internet vilket betyder att SEND skyddar sig emot denna attack.

```
root@victim:~# ping6 -I eth0 ipv6.google.com -c1
PING ipv6.google.com(2a00:1450:8007::69) from
2a01:48:100:106:2812:a968:ad3a:2cb5 eth0: 56 data bytes
64 bytes from 2a00:1450:8007::69: icmp_seq=1 ttl=54 time=133 ms

--- ipv6.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 133.836/133.836/133.836/0.000 ms

root@victim:~# ip -6 neigh
fe80::34d3:6d33:af87:5cd4 dev eth0 lladdr 00:1f:d0:36:4b:b9 router
REACHABLE
root@victim:~#
```

Tabell 14: Mål klienten kan nå internet

3.3.2 Slutsats

Precis som RFC 3971 beskriver så skyddar sig SEND emot denna attack. Vi ser att den upptäcker kollision både länklager och globala adresserna, men det gör den enbart engång.

4. Slutsats

I denna rapport har vi beskrivit hur ND fungerar samt hur SEND skyddar sig emot olika attacker. Slutsatser kommer att tas upp i två delar dels själva implementationen samt generellt om SEND som lösning.

4.1 NDprotector som implementation

NDprotector var det ingen som hade skrivit om eller testat. Det var därför jag valde att testa den och det var lovande till en början. Sen när det väl skulle konfigureras och sättas upp i min lilla labbmiljö så blev det en hel del tid som gick. Så fort det var syntax fel i konfigurationsfilen så vart det python error när man körde igång NDprotector. På grund av att den strikta syntaxen i python som även återfanns i konfigurationsfilen. Sen för det andra så saknar NDprotector dålig kontroll eller felhantering, och undantdags spår.

Efter att så gott som avklarat min utbildning inom datateknik så vad jag har lärt mig när man programmerar applikationer så är felhantering något extremt viktigt. Även för dig själv och slutanvändaren av applikationen. Skaparen av NDprotector vet om detta. Det första felet tog mig ett antal timmar att lista ut. Det är att skaparen har skrivit ett fel variabelnamn i konfigurationsfilen. Det står att det ska vara publika nyckeln men det skall vara den privata nyckeln. Hur det såg ut finner ni i appendixet, som ni ser så är det inte direkt enkelt och se att det ska vara den privata nyckeln. Skaparen kommer givetvis att ändra till den nästa version som planeras att komma ut under våren. Sen var det extremt pilligt med de olika certifikat vägarna som skulle stämma överens. Det skulle vara trevligt att få lite loggar om vad som händer, SEND implementationen Easy-SEND är något skaparen borde studera närmare och ta efter. Efter att man väl fått igång NDprotector utan några större felmeddelanden så gjorde NDprotector sitt jobb. Precis enligt vad som beskrevs i teoridelen samt diverse olika RFC's. Skaparen är mycket medveten att man inte kan använda denna implementation i skarpt läge vilket tydligt står på hemsidan. Det är främst för forsknings och utbildnings syfte.

4.2 SEND som lösning

Generellt sätt så är det ingen skillnad mellan IPv4 och IPv6 och dess säkerhet. Det finns dock ingen känd lösning som skyddar IPv4 som SEND gör till IPv6. IPv4 adresserna håller nu snart på att ta slut och personligen tror jag att detta år är det stora året för IPv6 och dess framfart kommer att växa markant då främst i företagsnätverk.

SEND löser problemet med autentisering det har vi sett i under mina tester med NDprotector. Vi ser också att det klarar att skydda sig emot emot DAD DoS attacken.

SEND använder CGA som är baserat med RSA och SHA-1. I dags läget rekommenderas det att köra 1024bitars för standard säkerhetsnivå. Med längre nycklar än så är det givetvis mer resurskrävande för framförallt wireless operationer samt batteri relaterade enheter.

Nu testas det andra asymmeteriska kryptogratfiska mekanismer såsom Elliptisk kurva kryptografi (English: Elliptic Curve Cryptography (ECC)). Det är bevisat att det är mindre resurskrävande och snabbare än nuvarande CGA med RSA. Något som inte har testats än är CGA med SHA-2. Man kanske kan tycka att CGA adresser tar lite lång tid att generera men det testas med nya olika nyckel standarder som ECC. Jag tror inte det är just det som är problemet dagens smarta mobiltelefoner har ju redan 1GHz processorer samt kraftfulla GPU:er.

SEND är ett viktigt komplement som kan användas i nätverk. Viktig del som tillsammans med andra kända tekniker kan skydda lager 2 attacker. Nya protokoll som IEEE 802.1ae för att försäkra sig om sekretess och integritet. En nackdel med SEND är ju när nya hostar anländer till nätverk så behöver dem vara tidsynkade. Därför krävs det en säker tid synkroniserings teknik.

IPv6 kommer snart att slå och vi kommer alltmer bli beroende av IPv6. Varför inte SEND finns i dags läget är kanske inte så konstigt. Men jag tvivlar inte på att det kommer även till Windows inom en snar framtid. Programmeringsspråk så som *python* och *perl* saknar tyvärr bra fungerande IPv6 stöd för att kunna utveckla nya program. Även är det inte alltför många personer som har tillräcklig kunskap om IPv6. När väl IPv6 har etablerat sig så tror jag att det kommer komma betydligt fler SEND implementationer. SEND skyddar inte emot allt och det gör inte heller en brandvägg utan den hjälper till att försvara sig emot attacker.

Vi har inte pratat om DHCPv6, det är ingen utmanare då de även kan användas tillsammans. DHCPv6 kommer med en mer komplex konfigurations lösning. Till exempel så kan DHCPv6 ge mer komplexa

konfigurationsalternativ. Vid två alternativ överlappande, anser jag att detta är genomförandet specifikt och du kanske inte att kunna avgöra nu vilket som är den vinnande tekniken.

Referenser

[1] J. Arkko, J. Kempf, B. Zill, and P. Nikander. "SEcure Neighbor Discovery (SEND)". RFC 3971, Internet Engineering Task Force, March 2005.

[2] T. Narten, E. Nordmark, W. Simpson, and H. Soliman."Neighbor Discovery for IP version 6 (IPv6)". RFC 4861, Internet Engineering Task Force, September 2007.

[3] T. Aura. "Cryptographically Generated Addresses (CGA)". Internet Engineering Task Force. RFC 3972, March 2005.

[4] S. Thomson, T. Narten, and T. Jinmei. "IPv6 Stateless Address Auto-con_guration". RFC 4862, Internet Engineering Task Force, Septem-ber 2007.

[5] R. Hinden and S. Deering. "IP Version 6 Addressing Architecture". RFC 4291, Internet Engineering Task Force, February 2006.

[6] P. Nikander, J. Kempf, and E. Nordmark. "IPv6 Neighbor Discovery (ND) Trust Models and Threats". RFC 3756, Internet Engineering Task Force, May 2004.

[7] Scott Hogg and Eric Vyncke, **IPv6 Security**, Cisco Press, ISBN 1587055945, Dec 2008.

[8] A. Conta and S. Deering " **Internet Control Message Protocol (ICMPv6) for the Internet Protocol 6 (IPv6) Specification"** - RFC 2463 , Internet Engineering Task Force, December 1998.

[9] P. Nikander, Ed. And J. Kempf and E. Nordmark," **IPv6 Neighbor Discovery (ND) Trust Models and Threats**" - RFC 3756, Internet Engineering Task Force, May 2004.

[10] J. Arkko, **"Effects of ICMPv6 on IKE",** Work in Progress, March 2003.

[11] J. Arkko, **"Manual Configuration of Security Associations for IPv6 Neighbor Discovery"**, Work in Progress, March 2003.

[12] M. Bagnulo and J. Arkko. **"Cryptographically Generated Addresses (CGA) Extension Field Format" -** RFC 4581. October, 2006.

[13] S. Kent. **"IP Authentication Header."** RFC 4302. December, 2005.

[14] S. Kent. **"IP Encapsulating Security Payload (ESP").** RFC 4303. December, 2005.

[15] Say Chiu and Eric Gamess **"Easy-SEND: A Didactic Implementation of the Secure Neighbor Discovery Protocol for IPv6",** October, 2009

[16] Benedikt Stockebrand **"IPv6 in Practice A Unixer's Guide to the Next Generation Internet"**, ISBN 978-3540245247 November 2006

[17] Daniel Minoli & Jake Kouns **"Security in an IPv6 Environment",** ISBN 9781420092295, 2008

[18] **"Experimentation and Evaluation of IPv6 Secure Neighbor Discovery Protocol",**http://master.apan.net/meetings/xian2007/publication/031_lin.pdf, Sep. 2007

[19] Lin Zhao-Wen, Want Lu-hua & Ma Yan, "**Possible Attacks based on IPv6 Features and Its Detection",** Asia-Pacific Advanced Network (APAN) 24th Meeting in Xi'An, China, 2006

[20] "**Ipv6-send-cga an implementation of SEND protocol in LINUX kernel"** Huawei Technologies Corp. and BUPT (Beijing University of Post and Telecommunications), Dec 2009, http://code.google.com/p/ipv6-send-cga/

[21] Firesheep, **"A Firefox extension that demonstrates HTTP session hijacking attacks".** Oct. 2010, http://codebutler.github.com/firesheep/

[22] LORIA/INRIA, "**NDPMon - IPv6 Neighbor Discovery Protocol Monitor**", Nov 2007, http://ndpmon.sourceforge.net

[23] **"NTT Docomo",** http://www.docomolabs-usa.com/lab_opensource.html

[24] Madhava K. Gaikwad "**JSEND"**,SEND protocol implementation, http://sourceforge.net/projects/jsend/

[25] Tony Cheneau , **"NDprotector":** an implementation of CGA & SEND for GNU/Linux based on Scapy6",Jun. 2010 Paris. http://http://amnesiak.org/NDprotector/

[26] Frederic Beck, Thibault Cholez, Olivier Festor & Isabelle Chrisment **"Monitoring the Neighbor Discovery Protocol",** June 2007

[27] Christian Vogt**,"Security in IPv6 Neighbor Discovery"**, http://doc.tm.uka.de/vogt-2006-security-in-nd6.pdf, 2006

[28] **Van Hauser, "Vh thc ipv6 attack",** The Hackeres Coice (THC) http://freeworld.thc.org/thc-ipv6/ , 2008

[29] Tony Cheneau, Aymen Boudguiga, Maryline Laurent **, "Significantly improved performances of thecryptographically generated addresses thanks to ECC and GPGPU"**, 2010

[30] "**Hurricane Electric IPv4 Exhaustion Counters "**,http://ipv6.he.net/statistics/ , Jan. 2011, *Last access 2011-01-20*

[31] **"The IPv6 Portal",** IPv6 tunnel provider http://www.ipv6tf.org/

Appendix

I detta appendix så hittar du teknisk konfiguration som jag har använt mig utav.

A. IPv6 tunnel

```
ip tunnel add tun mode sit remote 213.172.34.125 local 83.179.39.174 ttl 255
ip link set tun up mtu 1480
ip -6 addr add 2A01:0048:0100:0001:0001::1C2/126 dev tun
ip -6 route add 2000::/3 dev tun
ip -f inet6 addr
```

B. radvd.conf

```
interface eth1

{

        AdvSendAdvert on;
        AdvHomeAgentFlag off;
        MinRtrAdvInterval 30;

        MaxRtrAdvInterval 100;
        AdvLinkMTU 1280;
        prefix 2A01:48:100:106::/64

   {

                AdvOnLink on;
        AdvAutonomous on;
         AdvRouterAddr on;

     };

};
```

C. IPv6 Privacy Extenstion

editera /etc/sysctl.conf

net.ipv6.conf.wlan0.use_tempaddr = 2
net.ipv6.conf.eth0.use_tempaddr = 2
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2

D. NDprotector installations guide

apt-get install openssl libnetfilter-queue-dev nfqueue-bindings-python python-setuptools /python-crypto python-m2crypto python-pyx

wget http://amnesiak.org/NDprotector/files/ndprotector-0.5.tar.gz

tar -zxvf ndprotector-0.5.tar.gz

cd ndprotector-0.5

python setup.py install

apt-get source libssl0.9.8

cd openssl-0.9.8g

pico/nano/vi debian/rules # add "enable-rfc3779" to the relevant line

CONFARGS = --prefix=/usr --openssldir=/usr/lib/ssl no-idea no-mdc2 no-rc5 zlib enable-tlsext no-sslv2 enable-rfc3779

dpkg-buildpackage -rfakeroot

sudo dpkg -i libssl0.9.8_0.9.8g-16ubuntu3.1_i386.deb

konfa scriptet gencert.sh samt kör det

private_key.pem public_key.pem lag dem i exmpel /etc/NDprotector/

kopiera sendd.conf.host eller sendd.conf.router från /ndprotector/examples/ till /etc/NDprotector/

E. Router konfiguration

# /etc/NDprotector/send.conf (ROUTER CONF)
# DO NOT CHANGE theses values unless you know what you're doing
NDprotector.retrans_timer = 1

```
NDprotector.ts_delta = 300
NDprotector.ts_fuzz = 1
NDprotector.ts_drift = 0.01

# indicate which signature algorithm is supported, order matters
NDprotector.SignatureAlgorithms = SigTypeID.keys() # we authorize all
the supported keys

# you can edit the rest of the file :)

# if the node is a router:
#  - it does not interpret RA messages
#  - it does not need the CPS/CPA messages (handled by the modified
radvd daemon)
NDprotector.is_router = True

# mixed mode indicates if the daemon is configured to accept both
secured and
# unsecured NDP messages (an unseccured message will not ovewrite a
secured
# entry though)
NDprotector.mixed_mode = True

# allow NDprotector to flush all the IPv6 addresses on all interfaces
NDprotector.flush_interfaces = True

# path to the plugin directory
NDprotector.pluginpath = "NDprotector/plugins"

# plugins that could be loaded
NDprotector.plugins = []

# available plugin
# NDprotector.plugins = [ 'ephemeraladdress' ]

# default sec value used for address creation
# (value higher than 1, while totally conform to RFC 3971/3972,
# are not recommanded as they require a lot of processing power)
NDprotector.default_sec_value = 1

# If True, the programm automotaticlally assign addresses on the
interfaces
# the program will also clean up and destroy unused addresses
# If set to False, the user must to the address assignement manually
NDprotector.assign_addresses = True
```

```
# Default key size for RSA keys
NDprotector.rsa_key_size = 1024

# Minimum (RSA) key size for  ingoing messages
# when the key is bellow this value, the message is considered insecure
NDprotector.min_RSA_key_size = 384

# Maximum size of a (RSA) key
# key whose length exceed this value are ignored
NDprotector.max_RSA_key_size = 2048

# Path to the default Public Key used for Stateless Address
Autoconfiguration
# if None is provided, generate a new RSA Public Key for new addresses
NDprotector.default_publickey =
"/etc/NDprotector/rsa/level1/private/cakey.pem"

# For host it is [ TA1, TA2 ]
# For router, this is empty ( [] )
NDprotector.trustanchors = []

# Certification path
# Order matters (first certificate is the CA, then level1, etc.
# For routers it is [ [C1,C2,C3], ..., [C1',C2',C3'] ]
# For host, this is empty ( [] )
# Ex: here, there is two paths
NDprotector.certification_path = [ [
"/etc/NDprotector/rsa/level0/cacert.pem",
"/etc/NDprotector/rsa/level1/cacert.pem" ] ]


# list of configured addresses
# (you must at least configure a Link-Local address
# for each interface you want this daemon to listen on)
NDprotector.configured_addresses =  [ Address(interface = "eth1",
prefix = "2a01:0048:0100:0106::"),
Address(interface = "eth1",
prefix = "fe80::") ]
```

F. Host konfiguration

```
# /etc/NDprotector/send.conf (HOST CONF)
# DO NOT CHANGE theses values unless you know what you're doing
```

```
NDprotector.retrans_timer = 1
# timestamp verification algorithm variables
NDprotector.ts_delta = 300
NDprotector.ts_fuzz = 1
NDprotector.ts_drift = 0.01

# indicate which signature algorithm is supported, order matters
NDprotector.SignatureAlgorithms =  SigTypeID.keys() # we authorize all
the suppo
rted keys

# you can edit the rest of the file :)
# path to the plugin directory
NDprotector.pluginpath = "plugins"

# plugins that could be loaded
NDprotector.plugins = []

# available plugin
# NDprotector.plugins = [ 'ephemeraladdress' ]

# if the node is a router:
#  - it does not interpret RA messages
#  - it interperts CPS and answers CPA
NDprotector.is_router = False

# mixed mode indicates if the daemon is configured to accept both
secured and
# unsecured NDP messages (an unseccured message will not ovewrite a
secured
# entry though)
NDprotector.mixed_mode = False

# allow NDprotector to flush all the IPv6 addresses on all interfaces
# (so that only CGA are on the interfaces)
NDprotector.flush_interfaces = True

# /!\ Beware:
# force test on X.509 IP Address extension
# (only used when is_router is set to False)
# by default, the extensions are checked against, however, it needs
# OpenSSL to have been compiled with the -enable-rfc3779 flag
NDprotector.x509_ipextension = True
```

```
# default sec value used for address creation
# (value higher than 1, while totally conform to RFC 3971/3972,
# are not recommanded as they require a lot of processing power)
NDprotector.default_sec_value = 1

# If True, the programm automotaticlally assign addresses on the
interfaces
# the program will also clean up and destroy unused addresses
# If set to False, the user must to the address assignement manually
NDprotector.assign_addresses = True

# Default key size for RSA keys
NDprotector.rsa_key_size = 1024

# Minimum (RSA) key size for  ingoing messages
# when the key is bellow this value, the message is considered insecure
NDprotector.min_RSA_key_size = 384

# Maximum size of a (RSA) key
# key whose length exceed this value are ignored
NDprotector.max_RSA_key_size = 2048

# Path to the default Public Key used for Stateless Address
Autoconfiguration
# if None is provided, generate a new RSA Public Key for new addresses
#NDprotector.default_publickey = None
NDprotector.default_publickey = "/etc/NDprotector/cakey.pem"

# Path to the trust anchor
# For host it is [ TA1, TA2 ]
# For router, this is empty ( [] )

NDprotector.trustanchors = [ "/etc/NDprotector/cacert.pem" ]

# Certification path
# Order matters (first certificate is the CA, then level1, etc.

# For routers it is [ [C1,C2,C3], ..., [C1',C2',C3'] ]
# For host, this is empty ( [] )
# Ex: here, there is two paths
NDprotector.certification_path = []

# list of configured addresses
# (you must at least configure a Link-Local address
# for each interface you want this daemon to listen on)
```

```
NDprotector.configured_addresses = [ Address(interface = "eth0",
prefix = "fe80::") ]
```

G. Ett typiskt NDprotector bug felmeddelande

```
root@itwt:/etc/NDprotector# ndprotector.py –v
Verbose output enabled
Reading configuration file /etc/NDprotector/sendd.conf
ECC support is available
Traceback (most recent call last):
  File "/usr/local/bin/ndprotector.py", line 5, in <module>
    pkg_resources.run_script('ndprotector==0.5', 'ndprotector.py')
  File "/usr/lib/python2.6/dist-packages/pkg_resources.py", line 467, in
run_script
    self.require(requires)[0].run_script(script_name, ns)
  File "/usr/lib/python2.6/dist-packages/pkg_resources.py", line 1200, in
run_script    execfile(script_filename, namespace, namespace)
  File "/usr/local/lib/python2.6/dist-packages/ndprotector-0.5-
py2.6.egg/EGG-INFO/scripts/ndprotector.py", line 24, in <module>
    main()
  File "/usr/local/lib/python2.6/dist-packages/ndprotector-0.5-
py2.6.egg/NDprotector/Core.py", line 42, in main
    readconfig(NDprotector.CONFIG_FILE)
  File "/usr/local/lib/python2.6/dist-packages/ndprotector-0.5-
py2.6.egg/NDprotector/Config.py", line 28, in readconfig
    execfile(config_file)
  File "/etc/NDprotector/sendd.conf", line 67
    "NDprotector.default_publickey =  "/etc/NDprotector/public_key.pem"

                ^
SyntaxError: EOL while scanning string literal
root@itwt:/etc/NDprotector#
```

TRITA-ICT-EX-2011:22

www.kth.se