IPV6 BEVEILIGING

Versie 1.1 - 14 maart 2014
Geschreven voor: SURFnet
Door: Iljitsch van Beijnum

SURF NET

IPV6 BEVEILIGING

Versie 1.1, 14 mrt 2014

Geschreven voor

SURFnet

door Iljitsch van Beijnum

INHOUD

1. INLEIDING

IPv6 is nieuw voor velen onder ons. Maar **zo** nieuw is het niet. IPv4 vierde zijn dertigste verjaardag een paar jaar geleden, en IPv6 bestaat al half die tijd: de eerste IPv6-specificatie werd gepubliceerd in 1995. Dit was voordat veel dingen die we nu voor vanzelf sprekend aannemen aan IPv4 toegevoegd werden, zoals DHCP en NAT. Dus in sommige opzichten is (of lijkt) IPv6 **ouder** dan IPv4. Uiteraard heeft IPv6 nieuwe eigenschappen en details die IPv4 niet heeft. Andere aspecten zijn gewoon anders. Uit het oogpunt van beveiliging is het belangrijk om te weten wat al die verschillen zijn en hoe ze relateren aan beveiliging. Vaak is er de verleiding om IPv6 in te passen in de huidige manier waarop een netwerk werkt, maar zulk "IPv4-denken" kan het realiseren van sommige IPv6-voordelen in de weg staan of tot problemen leiden op de langere duur. Dit document behandelt de meest prominente eigenschappen van IPv6 gezien vanuit het perspectief van netwerkbeveiliging en legt uit hoe die eigenschappen te managen, evenals de relatie en samenwerking tussen IPv6 en IPv4. Er zijn uiteraard veel andere beveiligingsaspecten die komen kijken bij het verbinden van PC's/werkstations/hosts aan het (IPv6- of IPv4-) internet die buiten het kader van dit document vallen; gebruik a.u.b. de geëigende "best practices".

De doelgroep voor dit document bestaat uit systeem- en netwerkbeheerders van organisaties die met een campusnetwerk aan het SURFnet verbonden zijn en gelijksoortige organisaties.

ⓘ Omdat de meeste wijdgebruikte besturingssystemen IPv6 in de standaardconfiguratie aan hebben staan zijn sommige IPv6-beveiligingszaken ook relevant in netwerken die niet met het IPv6 internet verbonden zijn. We bevelen daarom aan dat ook beheerders van netwerken die geen IPv6 draaien zich op de hoogte stellen van de zaken die in dit document besproken worden.

2. KENMERKEN VAN IPV6

Naast de langere adressen en de voor de hand liggende wijzigingen die hiervoor nodig zijn, zoals de nieuwe IPv6-header, het nieuwe AAAA DNS record en nieuwe of gewijzigde routing-protocollen, heeft de IETF ook gebruik gemaakt van het feit dat veranderingen doorgevoerd werden om een aantal nieuwe dingen toe te voegen.

Het kenmerk dat radicaal anders is in IPv6 ten opzichte van IPv4 is de lengte van de adressen. IPv4 heeft 32-bit adressen, wat een maximum van 4,3 miljard adressen mogelijk maakt. (Hoewel veel niet in praktijk bruikbaar zijn om verschillende redenen.) IPv6 vervierdubbelt het aantal adresbits. De 128-bit IPv6-adresruimte maakt een maximum van 340 miljard miljard miljard miljard adressen mogelijk.

De onmiddellijke consequentie van die grote adresruimte is dat het scannen van IPv6-adressen zeer veel moeilijker is dan het scannen van IPv4-adressen. Dit wordt besproken in sectie 4.1. De extra bits maken het mogelijk voor hosts om op een aantal nieuwe manieren aan een IPv6-adres te komen: stateless autoconfig met en zonder privacy extensies, beschreven in sectie 2.2.

2.1. Adresnotatie

IPv4-adressen worden opgeschreven als vier decimale nummers van 0 tot 255 gescheiden door punten: 192.0.2.31. Sommige systemen accepteren voorloopnullen (192.000.002.031) of oude, klassegebaseerde notaties (16.1 in plaats van 16.0.0.1), maar deze notaties zijn ongebruikelijk en er is maar één standaardmanier om IPv4-adressen te schrijven.

Bij IPv6 worden adressen in tekst weergegeven als acht viercijferige hexadecimale waarden gescheiden door dubbele punten: 2001:db8:31:1af:20a:95ff:fef5:246e. Net als bij IPv4 mogen voorloopnullen gebruikt worden, maar die worden normaal weggelaten; in IPv6-adressen worden gewoonlijk kleine letters gebruikt, maar hoofdletters zijn ook toegestaan. Wanneer er een reeks van nullen gescheiden door dubbele punten in een IPv6-adres voorkomt mag deze weggelaten worden: 2001:db8:31:0:0:0:0:53 wordt 2001:db8:31::53.

Wanneer er een reeks van twee of meer nullen gescheiden door dubbele punten voorkomt mag die weggelaten worden. Dus 2001:db8:0:0:1:0:0:1 kan opgeschreven worden als 2001:db8::1:0:0:1 of 2001:db8:0:0:1::1. (Maar **niet** 2001:db8::1::1!) Tenslotte mogen de laatste 32 bits van een IPv6-adres opgeschreven worden als een IPv4-adres. Dus 2001:db8:31::192.0.2.1 is een geldige manier om 2001:db8:31::c000:201 weer te geven. In veel gevallen zal een adres dat door een gebruiker wordt ingegeven in de eerste variant door het systeem in de tweede, meer gebruikelijke notatie weergegeven worden. Bijvoorbeeld:

```
$ ping6 2001:db8:31::192.0.2.1
PING6(56=40+8+8 bytes) 2001:db8:3100:a006:1:: --> 2001:db8:31::c000:201
```

Het bovenstaande heeft twee consequenties. Allereerst is het erg lastig om invoervalidatie te doen die alle legale notatievarianten die IPv6 kent toestaat, maar geen ongeldige IPv6-adressen accepteert. Ten tweede kan bij het zoeken naar een specifiek IPv6-adres in tekstnotatie in bijvoorbeeld tekstdocumenten, databases en spreadsheets makkelijk voorkomen dat het betreffende adres niet gevonden wordt terwijl het wel aanwezig is, maar in een net iets andere notatie.

Om deze problemen te voorkomen adviseert RFC 5952 de volgende standaard-tekstnotatie voor IPv6-adressesn:

- Voorlopnullen dienen weggelaten te worden
- :: dient gebruikt te worden om zoveel mogelijk nullen weg te laten
- Maar een enkele :0: moet niet vervangen worden door ::
- Wanneer twee even lange reeksen van nullen vervangen kunnen worden, vervang dan de eerste
- De letters in IPv6-adressen dienen kleine letters te zijn
- De laatste 32 bits in een IPv6-adres mogen alleen in IPv4-vorm opgeschreven worden wanneer het adres binnen een "well-known" prefix valt, zoals ::ffff:0:0/96

Het is verstandig bestaande functies of bibliotheken te gebruiken om IPv6-adressen te valideren en converteren in plaats van hier eigen code voor te schrijven, het is te makkelijk om fouten te maken.

Een andere complicatie bij de tekstrepresentatie van IPv6-adressen is het meenemen van een poortnummer. De meest gebruikte en aanbevolen manier om dit te doen is [2001:db8:31::c000:201]:80, maar dit hangt van de applicatie af.

2.2. Adresconfiguratie

Toen IPv6 gemaakt werd was DHCP een nieuw protocol dat nog niet in algemeen gebruik was. Maar de mogelijkheid om automatisch aan een adres te kunnen komen zonder handmatige configuratie was een voor de hand liggende toevoeging, waardoor IPv6 "stateless autoconfiguration" kreeg. (Soms wordt dit stateless address autoconfiguration of SLAAC genoemd.) Stateless autoconfig lijkt veel op de manier waarop het oude IPX-protocol werkt: een host (werkstation, computer) combineert een 64-bit prefix die door routers uitgezonden wordt in periodieke Router Advertisements met een 64-bit "interface identifier" om een 128-bit IPv6-adres te creëren.

De interface identifier is min of meer een 64-bit MAC-adres. Een normaal 48-bit MAC-adres wordt verlengd tot 64 bits door de bits **fffe** in het midden toe te voegen, en het unique/local bit wordt omgezet. Dus het MAC-adres 40:6c:8a:32:4b:c3 resulteert in de interface identifier 426c:8aff:fe32:4bc3. Het resultaat van stateless autoconfiguration is dat een host altijd hetzelfde adres genereert, ook al is er geen database die de adressen bijhoudt.

Hosts die stateless autoconfig gebruiken kunnen gevolgd worden wanneer ze zich tussen verschillende netwerken bewegen aan de hand van het MAC-adres dat zich in het IPv6-adres bevindt. Bijvoorbeeld, als een gebruiker thuis adres thuis adres 2001:db8:31:1af:426c:8aff:fe32:4bc3 en op het werk 3ffe:a00:6:993:426c:8aff:fe32:4bc3, dan kunnen diensten die met het betreffende systeem gebruikt worden concluderen dat het in beide gevallen hetzelfde systeem is. Om dit te voorkomen specificeert RFC 4941 privacy extensies voor stateless autoconfig: in plaats van een stabiel MAC-adres worden de laagste bits van het IPv6-adres gevuld met een willekeurig getal. Gewoonlijk wordt een nieuw getal (en daarmee een nieuw adres) gegenereerd iedere keer als het systeem (opnieuw) aan het netwerk gekoppeld wordt, of iedere 24 uur. Windows en Mac OS X vanaf versie 10.8 Mountain Lion genereren privacy-adressen en ook stabiele adressen; het privacy-adres wordt gebruikt voor uitgaande verbindingen, het stabiele adres kan gebruikt worden voor inkomende verbindingen. In het geval van Windows, is het stabiele adres niet (direct) gebaseerd op het MAC-adres dat het systeem heeft. Linux en de BSD-familie besturingssystemen ondersteunen privacy-adressen, maar of deze ook gebruikt worden hangt af van de standaardinstellingen in de distributie en de configuratie die gebruikt wordt.

In halverwege de jaren 2000 werd DHCPv6 toegevoegd, wat grotendeels hetzelfde werkt als IPv4 DHCP. Een verschil is dat IPv4 DHCP, naarst een IP-adres en informatie zoals DNS server-adressen, ook een netmask en het adres van een default router / gateway uitdeelt. Bij IPv6 wordt in plaats van een netmask een prefixlengte gebruikt. Maar DHCPv6 geeft geen prefixlengte of default gateway-adres mee. IPv6 worden geacht een /64 prefixlengte te hebben en meestal werkt alles ook wel als er geen prefixlengte gegeven is, maar uiteraard is een default gateway noodzakelijk om extern te kunnen communiceren. Daarom moeten routers Router Advertisements uitzenden om deze informatie beschikbaar te maken. En de Router Advertisements vertellen hosts of ze "stateful" DHCPv6 (om adressen te configureren), "stateless" DHCPv6 (alleen voor extra informatie zoals DNS server-adressen) of helemaal geen DHCPv6 moeten gebruiken.

Een ander verschil tussen DHCP voor IPv4 en DHCPv6 is dat DHCPv6 een DHCPv6 Unique Identifier (DUID) gebruikt om DHCPv6-cliënten te identificeren in plaats van het MAC-adres van het betreffende systeem of een client ID. Dit maakt het lastiger om cliënt-specifieke instellingen aan een DHCPv6-serverconfiguratie toe te voegen. RFC 6936 heeft als doel dit probleem op te lossen.

Het is mogelijk om zowel stateless autoconfig als DHCPv6 voor adresconfiguratie te gebruiken; hosts die beide ondersteunen zullen dan drie adressen configureren: een stabiel autoconfig-gebaseerd adres, een privacy-adres en een DHCPv6-gebaseerd adres. Alle systemen die IPv6 aankunnen ondersteunen stateless autoconfig. Windows ondersteunt DHCPv6 vanaf Vista, Mac OS X ondersteunt DHCPv6 sinds 10.7 Lion. De meeste open source besturingssystemen / distributies ondersteunen DHCPv6, maar soms moet het expliciet geactiveerd of geïnstalleerd worden.

Een Router Advertisement kan meerdere prefixen voor stateless autoconfig bevatten. Meerdere routers kunnen dezelfde of verschillende subnetprefixen en/of verschillende DHCPv6-configuraties uitsturen. Over het algemeen zullen hosts een combinatie van deze informatie gebruiken. Dus als één router twee prefixen en geen DHCPv6 uitstuurt en een andere een derde prefix en stateful DHCPv6, dan zullen hosts zes adressen configureren door middel van stateless autoconfig (drie stabiele en drie privacy-adressen) en contact zoeken met een DHCPv6-server voor een zevende adres.

2.3. Geen NAT

De reden dat Network Address Translation (NAT) bestaat is zodat meerdere IPv4-hosts een publiek IPv4-adres kunnen delen. Om dit te faciliteren krijgen deze hosts een privé- (RFC 1918) adres en een NAT-apparaat vertaalt tussen het publieke adres van het NAT-apparaat en de interne adressen van de betreffende hosts. Dit belemmert het functioneren van protocollen die adressen doorgeven als onderdeel van het protocol, zoals FTP en SIP. Afgezien daarvan werkt dit voor sessies die geïnitieerd worden door de hosts achter de NAT: inkomende pakketten worden doorgestuurd naar de host die eerder gerelateerde uitgaande pakketten zond. Maar voor communicatie die vanaf buiten geïnitieerd wordt heeft het NAT-apparaat geen manier om te bepalen naar welke host de betreffende pakketten doorgestuurd moeten worden.

Dus NATs filteren inkomende TCP-sessies en ongevraagde UDP-pakketten als bijeffect. Maar er zijn verschillende typen NAT, en sommige laten inkomende sessies/pakketten door in sommige situaties. Bijvoorbeeld, wanneer een uitgaande sessie naar bestemming A poort X gebruikt dan is het mogelijk dat de NAT inkomende pakketten naar poort X toestaat van externe systemen anders dan A.

Omdat IPv6 zo veel adressen heeft is er geen reden om meerdere hosts een enkel adres te laten delen. Om die reden wordt NAT zelden gebruikt in combinatie met IPv6, en producten die IPv4 NAT ondersteunen bevatten over het algemeen geen ondersteuning voor IPv6 NAT. Omdat het gebruik van NAT niet verwacht wordt in combinatie met IPv6 zijn de maatregelen om door NAT heen te kunnen werken vaak niet aanwezig voor IPv6. Hierdoor zal het gebruik van NAT met IPv6 meer problemen opleveren dan met IPv4.

RFC 4864 geeft een overzicht van de beveiligingsmechanismen die gebruikt kunnen worden met IPv6 om zonder adresvertaling een zelfde mate van beveiliging te krijgen als NAT geeft met IPv4.

Er is een experimentele specificatie voor IPv6 prefixvertaling (RFC 6296), maar hierbij wordt een volledige prefix vertaald, waardoor iedere interne host zijn eigen externe IPv6-adres heeft. Zodoende biedt deze vorm van NAT niet dezelfde filtering die een doorsnee IPv4 NAT biedt.

Bij het aanbieden van IPv6-connectiviteit aan hosts die zich achter een IPv4 NAT bevinden kan het nodig zijn een stateful firewall te installeren. Zie sectie 5.7. Let ook op dat zowel "native" als getunnelde IPv6-connectiviteit extern zichtbare, publieke IPv6-adressen gebruikt en daarmee informatie over het netwerk kan vrijgeven die hiervoor verborgen werd door de IPv4 NAT.

2.4. Gebruikers identificeren aan de hand van hun IPv6-adres

Veel netwerkbeheerders houden er niet van dat hosts hun eigen adressen genereren door middel van stateless autoconfig, en geven zodoende de voorkeur aan DHCPv6. Daarmee is het mogelijk om de relatie tussen IPv6-adres en MAC-adres te loggen (of vooraf te configureren).

Wel geldt de beperking dat het loggen van welke host welk IPv6-adres heeft gekregen met DHCPv6 alleen onder normale omstandigheden effectief is: een kwaadwillende gebruiker kan nog altijd een niet-DHCP-adres genereren of een DHCP-adres stelen van een andere host. Het loggen van de relatie MAC-adres - IPv6-adres helpt hiertegen, behalve dat het MAC-adres ook vervalst kan worden. De oplossing is het gebruik van 802.1x op draadgebonden netwerken en WPA2 enterprise voor Wi-Fi gecombineerd met het loggen van welke gebruiker welk IPv6-adres krijgt.

Let op dat zelfs wanneer alleen stateless autoconfig gebruikt wordt om adressen te configureren, hiernaast vaak DHCPv6 ingezet wordt om de IPv6-adressen van DNS resolvers te configureren. Er is ook een RA-optie om dit te doen, maar niet alle routers en operating

systems (met name Windows) implementeren deze. (En in dual-stack netwerken kan IPv4 gebruikt worden voor het raadplegen van de DNS.)

Er zijn verschillende situaties waar aanbieders van diensten er behoefte aan hebben om gebruikers aan de hand van IP-adres te identificeren. Bijvoorbeeld, om een bepaalde gebruiker te blokkeren of om te limiteren hoe vaak een bepaalde gebruiker een bepaalde actie mag uitvoeren. Dit wordt vaak gedaan op basis van een gebruiker's IP adres. In de IPv4-wereld kan dit problematisch zijn als meerdere gebruikers hetzelfde publieke IPv4-adres delen, maar bij IPv6 kan het omgekeerde makkelijk voorkomen: een gebruiker krijgt met enige regelmaat een ander IPv6-adres, zelfs als de gebruiker verbonden blijft met hetzelfde netwerk. Een mogelijkheid is om te filteren op de eerste 64 bits van het IPv6-adres, wat dezelfde "resolutie" geeft als filteren op een individueel IPv4-adres. Echter, veel ISPs geven hun gebruikers meer dan een enkele /64, dus gemotiveerde gebruikers zouden in staat zijn zulke filters te omzeilen. Ottow et al. (2012) suggereren het opschalen naar grotere and grotere blokken als ongewenste activiteiten continueren.

Uiteindelijk is er niet een enkele aanpak die het probleem oplost van het filteren van gebruikers die niet gefilterd willen worden; het gebruiken van IP-adressen voor dit doel is laster bij IPv6 dan bij IPv4.

2.5. (Niet langer) verplichte IPsec

RFC 4294, de IPv6 Node Requirements, schrijft voor dat IPv6-nodes (hosts en routers) IPsec **moeten** ondersteunen. Maar RFC 6434 laat die eis vieren en stelt dat IPv6-nodes IPsec **zouden moeten** ondersteunen. (**Should** in plaats van **must**, oftewel doen er goed aan, maar het is mogelijk hiervan af te wijken.) IPsec is een mechanisme om communicatie te authenticeren en/of versleutelen op het niveau van een individueel pakket. Dit maakt het meer algemeen toepasbaar en effectiever dan TLS/SSL.

De meeste systemen die IPv6 aankunnen ondersteunen inderdaad IPsec. Wel is het zo dat IPsec alleen helpt als je het daadwerkelijk gebruikt. Wat moeilijk te doen is. In de praktijk wordt IPsec alleen op significante schaal gebruikt voor VPN-tunnels. IPsec is ook niet langer onderscheidend tussen IPv4 en IPv6 omdat ondanks het feit dat IPsec origineel ontwikkeld is voor IPv6, het even goed functioneert met IPv4.

2.6. Link-local adressen

Waarschijnlijk de belangrijkste nieuwe eigenschap van IPv6 afgezien van de langer adressen is het feit dat IPv6 link-local adressen heeft. Elk interface waarop IPv6 actief is **heeft altijd een link-local adres**. Deze adressen bevinden zich in de fe80::/64 prefix en de onderste 64 bits are een stabiele interface identifier die ook gebruikt wordt voor stateless autoconfig-adressen. De fe80::/64 prefix wordt hergebruikt op elk interface. Dit is mogelijk omdat deze adressen alleen geldig zijn binnen een "link" of subnet; ze kunnen niet door een router passeren. Dit betekent wel dat bij het gebruik van deze adressen noodzakelijk is om het uitgaande interface aan te geven. De meeste systemen voegen een procentteken gevolgd door de interfacenaam toe aan link-local adressen wanneer nodig. Bijvoorbeeld, dit ping-commando gebruikt het en0-interface:

```
$ ping6 fe80::8a1f:a1ff:fe29:923c%en0
PING6(56=40+8+8 bytes) fe80::bae8:86ff:fe3a:69f2%en0 -->
fe80::8a1f:a1ff:fe29:923c%en0
```

En dit het en4-interface:

```
$ ping6 -c 3 fe80::8a1f:a1ff:fe29:923c%en4
PING6(56=40+8+8 bytes) fe80::426c:8fff:fe3b:4bc3%en4 -->
fe80::8a1f:a1ff:fe29:923c%en4
```

Link-local adressen maken het mogelijk om via IPv6 te communiceren wanneer systemen geen "echt" IPv6-adres hebben (die global unicast adressen genoemd worden), of wanneer systemen adressen hebben in verschillende subnetprefixen. Om deze reden worden link-local adressen gebruikt voor de interne huishouding van IPv6, zoals het versturen van Router Advertisements, ICMPv6 redirects en dergelijke.

Een bijeffect van link-local adressen is dat wanneer systemen een service beschikbaar maken over IPv6, de betreffende service bereikbaar is voor iedereen op het lokale subnet. Het is dus erg belangrijk om altijd de juiste toegangsrestricties toe te passen voor IPv6, ook als een systeem geen externe IPv6-connectiviteit zal krijgen.

2.7. Neighbor Discovery en SEND

IPv4 gebruikt ARP over Ethernet en andere laag 2 netwerken die zich als Ethernet gedragen, zoals Wi-Fi. IPv6 gebruikt Neighbor Discovery om te bepalen welk MAC-adres bij welk IP-adres hoort. Neighbor Discovery gebruikt een aantal verschillende soorten ICMPv6-pakketten:

- Neighbor Solicitation, om op te vragen welk MAC-adres bij welk IPv6-adres hoort
- Neighbor Advertisement, in antwoord op een Neighbor Solicitation
- Router Solicitation, om om een Router Advertisement te vragen
- Router Advertisement, om de aanwezigheid van routers aan te geven en voor adresconfiguratie

IPv6 maakt geen gebruik van broadcasts (pakketten die naar alle aangesloten systemen "omgeroepen" worden), dus wanneer deze pakketten niet naar een enkel systeem gestuurd worden dan worden ze geadresseerd aan een multicast-adres; zie RFC 4861 voor details.

Aanvallers kunnen uiteraard nep-Neighbor Discovery-pakketten versturen om IPv6-verkeer in de war te brengen, of om pakketten te herrouteren om ze zo te kunnen inspecteren of modificeren. (Vergelijkbaar met ARP-spoofing bij IPv4.) Of de Duplicate Address Detection-fase saboteren waardoor een systeem denkt dat zijn adres al in gebruik is, waardoor het achterblijft zonder werkend IPv6-adres. Met name in het geval van het link-local adres van een router is dit problematisch, omdat als deze aanval slaagt de router geen link-local adres heeft en dan geen pakketten over het betreffende interface kan routeren.

SEcure Neighbor Discovery (SEND, RFC 3971) -adressen geven systemen de mogelijkheid valse Neighbor Discovery-pakketten te detecteren en verder negeren die door onbetrouwbare systemen op het lokale subnet verstuurd worden. Bij SEND hebben hosts en routers een certificaat dat vertrouwd kan worden via een pad van vertrouwd systeem naar vertrouwd systeem (trust path). Adressen worden vervolgens gebaseerd op een certificaat via Cryptographically Generated Addresses (CGAs). Een CGA is een IPv6-adres waar het interface identifier deel bestaat uit een hash over de public key van een certificaat. Signatures (digitale handtekeningen) worden via opties in de Neighbor Discovery pakketten gecommuniceerd om dit te kunnen controleren.

SEND maakt het mogelijk voor hosts en routers om Neighbor Discovery functies correct uit te voeren ook al bevinden zich systemen op het lokale subnet die dit door het versturen van foutieve ND-pakketten proberen te voorkomen. Met name worden vervalste Router Advertisements genegeerd.

SEND is echter niet wijd geïmplementeerd.

2.8. Hop Limit = 255 voor lokale pakketten

Neighbor Discovery en ICMPv6 redirects hebben alleen zin tussen hosts en routers die verbonden zijn aan hetzelfde subnet. Dus wanneer deze pakketten binnenkomen vanaf het internet dan is er sprake van een kwaadwillende actie. Om aanvallen gebaseerd op deze pakketten te voorkomen controleert IPv6 of deze pakketten gegenereerd werden door een lokaal systeem, of dat ze vanaf elders verstuurd werden (via het internet).

Wanneer een systeem een ND- of redirect-pakket verstuurt dan zet het het Hop Limit-veld op 255, de hoogste waarde die binnen dat veld van de IPv6-header past. De ontvanger controleert vervolgens of de Hop Limit inderdaad 255 is. Zo ja, dan wordt het pakket geaccepteerd als zijnde lokaal en verder verwerkt.

Als het pakket niet lokaal gegenereerd werd en dus één of meer routers gepasseerd heeft, dan zouden die routers de Hop Limit verlaagd hebben. Dus niet-lokale pakketten kunnen nooit een Hop Limit van 255 hebben, en pakketten met een Hop Limit van lager dan 255 worden dan ook genegeerd. Deze controlestap maakt het mogelijk voor systemen om vervalste ND- en redirect-pakketten te negeren zonder filters of andere complexe beveiligingssystemen. Deze controle is onderdeel van de basisspecificatie van IPv6 en kan niet uitgezet worden.

3. MIGRATIE VAN EN COEXISTENTIE MET IPV4

Het is simpeler een puur IPv6-netwerk te runnen dan een IPv4-netwerk. Subnetten zijn (bijna) altijd een /64, er is genoeg adresruimte om subnetten apart te houden voor speciale doeleinden en er is geen NAT. Maar het runnen van een "dual stack" IPv4 + IPv6 netwerk combineert alle complexiteit van IPv4, alle complexiteit van IPv6 en dan nog extra complexiteit voor het naast elkaar bestaan van de twee.

3.1. Dual stack

Dual stack wil zeggen het naast elkaar runnen van IPv4 en IPv6. Aangezien het (nog) zelden realistisch is om IPv4 uit te zetten op dit moment is dual stack de meestgebruikte manier om IPv6 te implementeren. Hoewel applicaties normaliter dezelfde APIs gebruiken om over IPv4 en IPv6 te communiceren opereren beide protocollen binnen de netwerkinfrastructuur geheel onafhankelijk. Als een DHCP-server een IPv4-adres en een IPv4 default gateway uitgeeft aan een host, dan draait die host IPv4. Als Router Advertisements een host van een IPv6 default gateway en RAs of DHCPv6 een IPv6-adres voorzien, dan draait de host IPv6. Wanneer beide gebeuren draait de host dual stack.

Let op dat dezelfde router zowel de IPv4-router als de IPv6-router op een subnet kan zijn, of één router kan de IPv4 afhandelen en een andere de IPv6. Wanneer meerdere IPv6-routers aanwezig zijn dan kunnen IPv6-hosts automatisch van de één naar de ander omschakelen.

Dus in de meeste opzichten komt dual stack neer op IPv4 runnen zoals te doen gebruikelijk en IPv6 runnen zoals te doen gebruikelijk. Bij het opzetten van uitgaande sessies vragen hosts met IPv4-connectiviteit A records op in de DNS en communiceren over IPv4. Hosts met IPv6-connectiviteit vragen AAAA records op en communiceren over IPv6. Dual stack hosts vragen zowel A als AAAA records op, en als beide aanwezig zijn moeten ze beslissen of ze communiceren over IPv4 of IPv6.

Tot niet zo lang geleden gaven dual stack hosts de voorkeur aan IPv6 wanneer mogelijk. Dit heeft wel als nadeel dat als de IPv6-connectiviteit niet werkt of langzaam is de gebruiker last heeft van een niet-werkende of trage dienstverlening. Om dit te voorkomen implementeren Mac OS X en Windows in de huidige versies "happy eyeballs"-technieken die de kwaliteit van de

IPv4- en IPv6-connectiviteit proberen te bepalen en dan de betere van de twee gebruiken. Dit heeft als bijeffect dat het moeilijk is om vantevoren te weten of een uitgaande verbinding over IPv4 of IPv6 zal gaan. (Maar Windows en de meeste Unix(-achtige) besturingssystemen maken het mogelijk dit te beïnvloeden met behulp van de RFC 6724 policy table.)

3.2. Tunnels

IPv6-tunnels zijn een techniek om IPv6-pakketten door te geven over een deel van het netwerk dat alleen IPv4 ondersteunt. Er zijn verassend veel verschillende tunnelmechanismen; zie RFC 7059 voor een overzicht.

Een gewone PC of ander systeem dat aan het netwerk gekoppeld is kan geconfigureerd worden om IPv6-pakketten te routeren tussen een IPv6-tunnel en zijn Ethernet of Wi-Fi interface, waarbij het Router Advertisements stuurt om verkeer aan te trekken van andere hosts. Als die RAs niet gefilterd worden dan zullen hosts op hetzelfde subnet hun IPv6-pakketten naar de "schurkenrouter". (Rogue router in het Engels, vergelijkbaar met rogue state: schurkenstaat.)

Het beste middel hiertegen is RA Guard, wat foutieve RAs wegfiltert. Dit werkt beter dan proberen tunnelpakketten te filteren omdat het relatief makkelijk is foute RAs te identificeren, maar tunnels kunnen versleuteld zijn of op een andere wijze aan het zicht onttrokken worden.

Wanneer een tunnel gebruikt wordt voor externe IPv6-connectiviteit dan is het belangrijk de tunnel te behandelen als iedere andere externe verbinding en de juiste filtering toe te passen. Dit is lastiger bij automatische tunnelmechanismen zoals 6to4, waar er een relatie is tussen bits in het IPv6-adres in de IPv6-header en het IPv4-adres in de IPv4-header die voor het IPv6-pakket geplaatst wordt.

Kwaadwillende gebruikers kunnen van zulke tunnelmechanismen gebruik maken om filters die naar het bronadres kijken te omzeilen. Bijvoorbeeld, iemand op het internet netwerk met adresreeksen 192.0.2.0/24 en 2001:db8:31::/48 is normaal niet in staat om pakketten te versturen met bronadres 145.0.2.10 of 2001:610:188:301:145::2:10. Maar iemand zou kunnen proberen om een 6to4-pakket te genereren waarbij het IPv4-bronadres 192.0.2.15 is maar het IPv6-pakket wat zich binnen het IPv4-pakket bevindt heeft als bronadres 2001:610:188:301:145::2:10. Door op deze manier een kleine DNS-aanvraag te versturen waarop een veel groter antwoord komt kan een kwaadwillende gebruiker een DNS amplification attack uitvoeren die heel lastig terug te voeren is naar de aanvaller.

6to4-tunneling staat standaard aan op Windows-systemen wanneer deze een publiek IPv4-aldres hebben maar geen global unicast IPv6-adres. Teredo-tunneling, wat in tegenstelling tot 6to4 wel door IPv4 NAT werkt, staat standaard aan op een Windows-systeem dat een privé-IPv4-adres heeft en geen global unicast IPv6-adres. Dit betekent dat 6to4 en Teredo gebruikt kunnen worden om firewallrestricties te omzeilen, zowel vanaf het interne netwerk om geblokkeerde externe bestemmingen te bereiken als bij externe systemen om systemen binnen het netwerk te kunnen bereiken.

Zodoende is het niet ongebruikelijk voor organisaties die publieke IPv4-adressen gebruiken voor PCs/werkstations om protocol nummer 41 uit te filteren, wat gebruikt wordt door 6to4 en verschillende andere tunnelmechanismen. In het verleden was dit problematisch omdat Windows-systemen dachten dat ze IPv6-connectiviteit hadden, maar dat die connectiviteit niet werkte. Het resultaat was dat bestemmingen met een IPv6-adres in de DNS moeilijk bereikbaar waren. Vanwege dit probleem wordt niet langer de voorkeur gegeven aan 6to4-connectiviteit boven IPv4, waardoor het filteren van protocol 41 aanzienlijk minder problematisch is. Teredo-connectiviteit wordt alleen gebruikt als er niks anders beschikbaar is, waardoor het filteren van UDP-poort 3544 over het algemeen niet tot problemen leidt.

Maar simpelweg "native" (niet-getunnelde) IPv6-connectiviteit beschikbaar stellen levert hetzelfde resultaat op. Een andere goede oplossing zou zijn een firewall en/of IDS die ook binnen getunnelde pakketten kan kijken en de zich in de tunnel bevindende IPv6-pakketten kan filteren.

Zie RFC 3964 voor meer informatie over de beveiliging van 6to4.

3.3. Vertalers (NAT64)

NAT64 vertaalt tussen de IPv6- en IPv4-pakketindelingen. Met stateless NAT64 (RFC 6145) wordt er een één-op-één relatie gelegd tussen een individueel IPv6-adres en een individueel IPv4-adres. Dat betekent dat stateless NAT64 niet helpt tegen het tekort aan IPv4-adressen. Daarnaast is er stateful NAT64 (RFC 6146) waarbij meerdere IPv6 clients (hosts die de verbinding initiëren) kunnen communiceren met IPv4 servers. Stateful NAT64 werkt in combinatie met DNS64. Een normale DNS-server geeft een leeg antwoord terug wanneer een IPv6-host het IPv6-adres opvraagt van een server die alleen een IPv4-adres heeft. DNS64 daarentegen genereert een synthetisch IPv6-adres door een IPv6-prefix die naar de NAT64-vertaler gerouteerd wordt te combineren met het IPv4-adres van de bestemming. Wanneer de IPv6-host dan een verbinding tracht op te zetten naar dat synthetische adres komen de pakketten terecht bij de NAT64, die ze vertaalt van IPv6 naar IPv4 en dan IPv4 NAT toepast. De pakketten die terugkomen van de server worden terugvertaald naar IPv6.

Het belangrijkste beveiligingsaspect aan NAT64 bestaat eruit dat als de NAT64-vertaler pakketten van buiten accepteert en vertaalt, kwaadwillende externe gebruikers de vertaler als een open proxy kunnen gebruiken. Het is daarom belangrijk de toegang tot de NAT64 te beperken tot legitieme gebruikers, gewoonlijk door pakketten van buitenaf naar de NAT64-prefix uit te filteren.

3.4. IPv4-adressen in IPv6-adressen

Zoals gezegd in sectie 3.2 (en zie ook het SURFnet IPv6 nummerplandocument) is het mogelijk een IPv4-adres op te nemen in een IPv6-adres. Maar de x:x:x:x:x:x:y.y.y.y notatie is alleen cosmetisch; zulke adressen blijven gewone IPv6-adressen. Er zijn echter verschillende manieren waarop IPv4-adressen in IPv6-adressen kunnen terugkomen op een manier die wel betekenis heeft. De meest zichtbare hiervan is bij het gebruik van tunnels, die later aan de orde komen.

IPv4-adressen worden ook in IPv6-adressen opgenomen bij het gebruik van NAT64. Er is een "well-known" (algemeen bekende) prefix gereserveerd voor stateful NAT64: 64:ff9b::/96. Netwerkbeheerders kunnen echter ook een eigen prefix inzetten voor dit doel. Het IPv4-adres wordt meestal in de onderste 32 bits geplaatst. Bijvoorbeeld, als de NAT64 de well-known prefix gebruikt en de IPv6-host wil een verbinding opzetten naar IPv4-adres 192.0.2.31, dan zal de DNS64 het synthetisch adres 64:ff9b::192.0.2.31, ofwel 64:ff9b::c000:201 genereren. Organisaties die een stateful NAT64-vertaler runnen moeten zorgen dat er filters aanwezig zijn zodat alleen legitieme gebruikers hun pakketten door de vertaler kunnen sturen.

Als laatste is er de prefix ::ffff:0:0:/96 die de IPv4-adresruimte representeert zodat applicaties kunnen volstaan met het gebruiken van de IPv6 Socket API om communiceren met de IP netwerkstack en dus niet de originele Socket API hoeven te gebruiken voor IPv4 en de IPv6 Socket API voor IPv6. Dit betekent dat als een applicatie een verbinding opzet naar bijvoorbeeld ::ffff:192.0.2.1, het systeem IPv4-pakketten naar 192.0.2.1 zal versturen. Zodoende kunnen er nooit geldige IPv6-pakketten zijn met adressen uit deze prefix.

Algemeen gebruikte tunnelmechanismen zoals 6to4 en Teredo genereren IPv6-adressen gebaseerd op een vaste prefix (2002::/16 voor 6to4, 2001::/32 voor Teredo) en het IPv4-adres van de gebruiker. Ze verpakken dan automatisch IPv6-pakketten voor deze bestemmingen in een IPv4-pakket en kopiëren de IPv4-bestemmingsbits uit de IPv4-bits in het 6to4 of Teredo

IPv6-adres. Teredo voegt ook de UDP-poort die gebruikt wordt om de host die Teredo runt te bereiken toe in het IPv6-adres.

4. IPV6-RISICO'S

Voordat we ingaan op specifieke nieuwe eigenschappen van IPv6 is het belangrijk om te onderkennen dat IPv6-implementaties fouten kunnen bevatten simpelweg omdat ze nieuwer zijn. Het geheel aan IPv6-specificaties bestaat al geruime tijd, dus de meeste bugs **zouden** ondertussen gevonden en opgelost moeten zijn. Implementaties worden ook volwassener, maar ze hebben zeker nog niet het aantal uren in productie meegemaakt als hun IPv4-tegenvoeters. Het is daarom zeker mogelijk dat fouten die jaren geleden al verholpen zijn in IPv4-implementaties nog aanwezig zijn in de corresponderende IPv6-implementatie.

Een voorbeeld van zo'n fout is de Routing Header type 0 (RH0), die gebruikt werd voor source routing bij IPv4. Deze header kon gebruikt worden om (sommige) firewalls te passeren en ook om vermenigvuldings-aanvallen (amplification attacks) te lanceren. RH0 was "deprecated" door de IETF in RFC 5095 in 2007.

Let erop dat er procedures zijn voor het up-to-date brengen van de software of firmware van IPv6-routers en -hosts (inclusief embedded systemen) wanneer beveiligingslekken verholpen worden.

4.1. Scannen

In 2003 verspreidde de SQL Slammer worm zich door het versturen van kopiëen van de worm naar willekeurige adressen. Omdat de wormcode maar een paar honderd bytes lang was en de dienst die gevoelig was voor de fout over UDP bereikbaar was kon de worm zichzelf verspreiden in één enkel pakket. Deze pakketten werden met hoge snelheid naar willekeurige IPv4-adressen gestuurd: sommige systemen verstuurden 15000 exemplaren van de worm per seconde. Het resultaat was dat 80% van de gevoelige systemen die aan het internet verbonden waren binnen tien minuten geïnfecteerd werden.

Pakketten naar willekeurige adressen versturen is niet de beste manier om de volledige adresruimte te scannen, maar de gedistribueerde aanpak van de Slammer worm maakte dat meer dan goed. Als we uitgaan van tien minuten om elk IPv4-adres te bereiken dan zou de volledige IPv6-adresruimte scannen $1{,}5 \times 10^{24}$ jaar duren. Dus als iemand was begonnen met het scannen van IPv6-adressen ten tijde van de oerknal en vervolgens de volledige levensduur van het heelal IPv6-adressen gescand had in dit tempo, dan zou deze persoon nu rond adres 0:0:2:9422:415f:8800:0:0 aangeland zijn.

Met andere woorden, **elk** IPv6-adres scannen is volledig onwerkbaar. Maar vaak zijn veel adresbits al vantevoeren bekend, en dan is scannen wellicht wel mogelijk. Bijvoorbeeld, het scannen van één /64 subnet op systemen die stateless autoconfiguration zonder privacy-adressen gebruiken kost nog altijd twee jaar met een snelheid van een miljoen pakketten per seconde (1 Gbps met pakketten van 125 bytes). Maar MAC-adressen worden uitgegeven in blokken van 16.8 miljoen aan fabrikanten. Eén zo'n blok scannen kost maar 17 seconden met een snelheid van 1 MPPS, dus een subnet scannen op (bijvoorbeeld) Apple-apparatuur die stateles autoconfig gebruikt kost tijd, maar is zeker te doen.

Een beveiligingsonderzoeker (Heuse, 2012) vond dat 79% van de netwerken reageert op scans naar de ::0, ::1 of ::2 adressen in tenminste één subnet. Dit maakt het voor een kwaadwillend persoon mogelijk om snel te bepalen welke subnetten in gebruik zijn en vervolgens deze subnetten verder te onderzoeken. Het kan nuttig zijn om te voorkomen dat routers en andere systemen zulke voor de hand liggende adressen krijgen en/of pakketten van buiten naar deze

adressen te blokkeren. Maar wees wel voorzichtig met het filteren van uitgaande pakketten, zie sectie 5.8.

Bij het expliciet toekennen van individuele adressen aan individuele hosts (via handmatige configuratie of DHCPv6) is het gebruikelijk om hosts opeenvolgende te nummeren, een IPv4-adres in de onderste bits van het IPv6-adres op te nemen op enige manier of om woorden te gebruiken in IPv6-adressen. (Bijvoorbeeld, het IPv6-adres van www.facebook.com is 2a03:2880:f006:101:**face:b00c**::1.) In al deze gevallen kunnen de betreffende adressen geraden worden. Adressen kunnen ook ontdekt worden via de DNS. In het algemeen zullen aanvallers proberen het gebruikte nummerplan te raden.

Automatische tunnelmechanismen zoals 6to4 en Teredo (besproken in sectie 3.2) genereren IPv6-adressen van een systeem zijn IPv4-adres. Deze adressen zijn relatief makkelijk te raden/scannen.

4.2. Het netwerk in kaart brengen via multicast

IPv6 gebruikt geen broadcasts, maar het all-hosts multicast adres ff02::1 dient hetzelfde doel als het IPv4 255.255.255.255 broadcastadres. Veel niet-Windows systemen reageren op pings naar dit adres:

```
$ ping6 -c 2 -I en0 ff02::1
PING6(56=40+8+8 bytes) fe80::bae8:86ff:fe3a:69f2%en0 --> ff02::1
16 bytes from fe80::bae8:86ff:fe3a:69f2%en0, icmp_seq=0 hlim=64 time=0.130 ms
16 bytes from fe80::8a1f:a1ff:fec9:323c%en0, icmp_seq=0 hlim=64 time=1.334 ms
16 bytes from fe80::60c:ceff:fe93:1860%en0, icmp_seq=0 hlim=64 time=112.837 ms
16 bytes from fe80::10e9:bd67:b1ad:a78c%en0, icmp_seq=0 hlim=64 time=119.508 ms
16 bytes from fe80::bae8:86ff:fe3a:69f2%en0, icmp_seq=1 hlim=64 time=0.175 ms
```

Nog behulpzamer voor mensen die pogen het netwerk te debuggen (al dan niet geautoriseerd) zijn node information query's (RFC 4620). Dit zijn ICMPv6-pakketten die IPv6-nodes vragen hun naam of adresinformatie terug te sturen. Sommige implementaties van het ping6-commando kunnen deze query's versturen en de antwoorden afbeelden:

```
$ ping6 -c 2 -I en0 -w ff02::1
PING6(72=40+8+24 bytes) fe80::bae8:86ff:fe3a:69f2%en0 --> ff02::1
36 bytes from fe80::bae8:86ff:fe3a:69f2%en0:macbookpro.local
28 bytes from fe80::8a1f:a1ff:fec9:323c%en0: basestation
40 bytes from fe80::10e9:bd67:b1ad:a78c%en0: Apple-TV
36 bytes from fe80::bae8:86ff:fe3a:69f2%en0: macbookpro.local
```

Apple-producten geven nog steeds antwoord op lokale node information query's, maar de meeste andere systemen doen dit niet meer. Sommige oudere BSD-systemen reageerden zelfs op niet-lokale NIQs met uitgebreide informatie:

```
$ ping6 -c 1 -a acgslA 2001:db8:31:1af::1
PING6(72=40+8+24 bytes) 2001:470:1f0b:1289:a5ad:a91f:9f44:d0e9 -->
2001:db8:31:1af::1
116 bytes from 2001:db8:31:1af::1:
  fe80::212:3fff:feed:d76c(TTL=infty)
  2001:db8:31:1af::1(TTL=infty)
  ::1(TTL=infty)
  fe80::1(TTL=infty)
  2002:c000:201::1(TTL=infty)
```

4.3. Neighbor Discovery cache-uitputting

Stel de volgende situatie: iemand scant IPv6 subnet 2001:db8:dead:beef::/64, beginnend bij het 2001:db8:dead:beef:0:0:0:0 (2001:db8:dead:beef::) adres. De router moet dan een Neighbor Solicitation pakket (vergelijkbaar met een IPv4 ARP request, zoals later beschreven) sturen voor adres 2001:db8:dead:beef::. En dan voor 2001:db8:dead:beef::1, 2001:db8:dead:beef::2 enzovoort. Routers hebben een beperkte Neighbor Discovery cache, die na verloop van tijd gevuld raakt met incomplete informatie voor de gescande adressen. Afhankelijk van de

implementatie kan dit problemen opleveren voor hosts en routers die adressen gebruiken die daadwerkelijk gebruikt worden in het betreffende subnet.

Hetzelfde probleem kan in principe voorkomen bij IPv4, maar daar is het zeldzaam. In de meeste gevallen zal NAT gebruikt worden, zodat het onmogelijk is om adressen binnen een subnet te scannen vanaf buiten de organisatie. Maar zelfs zonder NAT zijn IPv4 subnets over het algemeen klein genoeg dat de ARP cache niet uitgeput raakt. Wel is het zo dat sommige routers geheugen delen tussen de IPv6 ND cache en de IPv4 ARP cache, zodat een aanval tegen het één problemen voor het ander op kan leveren.

Filters voor de soorten pakketten die veel voor scannen gebruikt worden, zoals ICMPv6 echo request (ping) stoppen ook cache-uitputting als bij-effect van "normale" scans. Maar het is lastig het probleem volledig op te lossen met filters, aangezien alle typen pakketten die niet uitgefilterd worden gebruikt kunnen worden voor een opzettelijke Neighbor Discovery cache-uitputtingsaanval. Alleen een stateful firewall die alleen inkomende pakketten toestaat die horen bij eerdere uitgaande pakketten lost het probleem volledig op. Aan de andere kant, zulke restrictieve filters kunnen problemen opleveren voor peer-to-peercommunicatie.

Idealiter implementeren routers Neighbor Discovery op zo'n manier dat cache-uitputting niet tot problemen leidt. Het kan verstandig zijn om leveranciers van routers te vragen hoe zij dit probleem ondervangen. Op subnetten met alleen routers en/of servers is het mogelijk om een kleiner blok IPv6-adressen dan een /64 te gebruiken om dit probleem te omzeilen. (Zie sectie 4.12 van de SURFnet IPv6 nummerplanhandleiding.) Het kan ook nuttig zijn om wel een /64 te reserveren maar alleen een /120 te gebruiken, op die manier is het mogelijk later over te gaan naar de gebruikelijke /64 subnetgrootte zonder te hoeven hernummeren.

4.4. Ongeautoriseerde Router Advertisements en RA Guard

Bij IPv4 is er de mogelijkheid dat iemand een ongeautoriseerde "rogue" DHCP-server runt en daarmee het netwerk verstoort. Op dezelfde manier kan iemand IPv6 Router Advertisements versturen, zowel per ongeluk als met kwade opzet.

In netwerken die geen IPv6 draaien kunnen ongeautoriseerde Router Advertisements gebruikt worden om hosts zover te krijgen dat ze het verkeer voor IPv6-bereikbare bestemmingen via een aanvaller versturen, die dan het betreffende verkeer kan inspecteren en/of modificeren. In tegenstelling tot bij een ongeautoriseerde IPv4 DHCP-server heeft dit geen impact op het IPv4-netwerk, dus de situatie kan geruime tijd onontdekt blijven. In de praktijk gaat het hier meestal om Windows-systemen die ingesteld zijn om hun internet verbinding te delen (internet connection sharing) zonder dat er sprake is van kwade opzet. In dit geval zijn IPv6-bestemmingen onbereikbaar of veel trager voor hosts die zowel IPv4 als IPv6 aan hebben staan.

RFC 6105 specificeert Router Advertisement Guard (RA Guard), een systeem om ongeautoriseerde RAs weg te filteren in de laag 2 infrastructuur. RA Guard kan op verschillende manieren werken, maar de meest voor de hand liggende is om RAs alleen toe te laten vanaf de switchpoorten waar IPv6-routers aangesloten zijn, en RAs uitfilteren die binnenkomen op alle andere poorten. RA Guard kan ook gebruikt worden op draadloze netwerken, aangezien alle pakketten het Wi-Fi basisstation moeten passeren, zelfs pakketten tussen twee draadloos gekoppelde hosts.

Waar mogelijk dient RA Guard gebruikt te worden naast soortgelijke filtering van IPv4 DHCP en DHCPv6. Omdat dit selectief bepaalde IPv6- (en IPv4-) pakketten wel en andere niet filtert is deze functionaliteit meestal beperkt tot meer geavanceerde switches en Wi-Fi basisstations.

Zie ook de opmerking over fragmentatie en filteren in sectie 5.2.

4.5. Ongeautoriseerde DHCPv6-servers

Net als bij IPv4 DHCP is het mogelijk voor kwaadwillende gebruikers om een ongeautoriseerde DHCPv6-server te draaien. Dit is echter ongebruikelijk. De impact van zo'n ongeautoriseerde DHCPv6-server varieert afhankelijk van de waarde van de M- (managed config) en O- (other stateful config) bits in Router Advertisements, zie tabel 1.

Tabel 1, M en O bits in Router Advertisements en gebruik van DHCPv6

Soort network	M en O bits	Impact ongeautoriseerde DHCPv6-server
DHCPv6 wordt gebruikt voor adresconfigurattie	M = 1	Aanvaller kan adresconfiguratie ontregelen en adressen kwaadaardige DNS-servers injecteren
DHCPv6 wordt gebruikt voor overige congfiguratie (DNS etc.)	M = 0, O = 1	Aanvaller kan adressen kwaadaardige DNS-servers injecteren
DHCPv6 wordt niet gebruikt	M = 0, O = 0	Geen impact, hosts maken geen contact met DHCPv6-server

Dus in netwerken waar DHCPv6 gebruikt wordt is het voornaamste risico dat een aanvaller hosts zover krijgt dat ze een kwaadaardige DNS-server gebruiken. Wanneer DHCPv6 voor adresconfiguratie gebruikt wordt kan een denial-of-service aanval uitgevoerd worden door adresconfiguratie te verstoren, maar omdat DHCPv6 geen informatie over de default gateway/ router communiceert kan een kwaadwillende DHCPv6-server niet verkeer omrouteren.

Er is een groot aantal additionele DHCPv6-opties voor veel verschillende doeleinden. Afhankelijk van de wijzigende implementatiestatus van deze opties is een onge-autoriseerde DHCPv6-server wellicht in staat hier gebruik van te maken. Zodoende is het belangrijk om in netwerken die DHCPv6 gebruiken UDP pakketten met bestemmingspoort 546 weg te filteren tenzij ze afkomstig zijn van een legitieme DHCPv6-server of DHCPv6-relay.

In netwerken waar DHCPv6 niet gebruikt wordt zou een aanvaller ongeautoriseerde RAs met M = 1 or O = 1 moeten versturen naast het draaien van een ongeautoriseerde DHCPv6-server. In deze gevallen **zou** RA Guard dus moeten volstaan als bescherming tegen onge-autoriseerde DHCPv6-servers. Het is echter mogelijk dat hosts toch DHCPv6-verzoeken versturen zelfs als zijn de M en O bits nul, dus waar mogelijk is het filteren van (ongeautoriseerde) DHCPv6-servers aanbevolen.

5. HET ADRESSEREN VAN RISICO'S

De voornaamste manier om IPv6-gerelateerde risico's uit te sluiten is door het uitfilteren van pakketten die gebruikt kunnen worden om van deze risico's gebruik te maken.

5.1. Beveilingsafwegingen adresconfiguratie

Over het algemeen staan gemakkelijke autoconfiguratie en beveiliging tegenover elkaar. De enige manier om alle mogelijke beveiligingsproblemen met toegangscontrole en adrescon-figuratie te voorkomen is door authenticatie te vereisen voordat adresconfiguratie plaatsvindt (door middel van IEEE 802.1x of WPA2 Enterprise) en dan IPv6-adressen en Neighbor Discovery MAC-adres tabellen statisch te configureren en alleen vantevoren vastgelegde MAC-adressen toe te staan in Wi-Fi basisstations en switches. Op deze manier kunnen alleen geautoriseerde gebruikers een verbinding met het netwerk maken en daarna kunnen ze niet

hun MAC- of IPv6-adres vervalsen. Uiteraard vereist dit een zekere hoeveelheid werk vooraf om een nieuwe gebruiker en/of nieuw apparaat aan het netwerk toe te voegen.

Een beter uitvoerbare aanpak is het scheiden van gebruikers en/of werkstations en andere apparaten gebaseerd op beveiligingsvereisten en verschillende groepen gebruikers of apparaten aan verschillende subnetten toe te wijzen. Immers, servers met gevoelige data hebben andere vereisten op beveiligingsgebied dan mobiele telefoons en tables van gasten, en door deze te scheiden is het mogelijk voor beide groepen de juiste afweging tussen veiligheid en gebruikersgemak te maken: op een serversubnet kan handmatige configuratie op zijn plaats zijn om ongelukken te voorkomen, maar het loggen van adressen is dan verder niet nodig. Voor gasten kan toegangscontrole onhaalbaar en/of ongewenst zijn, maar het loggen van adressen kan nodig zijn om zich misdragende gebruikers te identificeren.

SEND zou erg nuttig zijn in netwerken die zich hier tussenin bevinden: de gebruikers en apparaten zijn te talrijk en divers voor handmatige configuratie, maar de gebruikers zijn bekend en de communicatie is potentieel gevoelig. In dit geval zou een vereiste om een beveiligingscertificaat te installeren en/of configureren niet onredelijk zijn. Maar SEND is niet beschikbaar in algemeen gebruikte besturingssystemen. Een mogelijk alternatief is het gebruik van een VPN.

Tabel 2, adresconfiguratiewijzen met en zonder SEND en statische ND/MAC-tabellen

	Op zichzelf	Met SEND	Met statische ND/MAC-tabellen
Handmatige configuratie	Geen autoconfig, kapen IP/MAC en DoS mogelijk	Geen autoconfig, kapen IP onmogelijk, MAC DoS wel	Geen autoconfig, kapen IP/MAC en DoS onmogelijk
Stateless autoconfig	Moeilijk om IP-adressen te loggen, kapen IP/MAC en DoS mogelijk	Loggen makkelijker, kapen IP onmogelijk, MAC DoS wel	Statisch ND onmogelijk, statisch MAC maakt loggen makkelijker, DoS moeilijker
DHCPv6	Makkelijk om IP-adressen te loggen, kapen IP/MAC en DoS mogelijk	Makkelijk om IP-adressen te loggen, kapen IP onmogelijk, MAC DoS wel	Per-host DHCP-config vereist, kapen IP/MAC en DoS onmogelijk

In deze context betekent denial-of-service (DoS) dat host A niet in staat is host B's IPv6- of MAC-adres over te nemen, maar host B kan **wel** host A's communication verstoren door Neighbor Discovery of DHCPv6 te verstoren met vervalste pakketten. Bijvoorbeeld, SEND zorgt ervoor dat alleen host A het IPv6-adres van host A kan gebruiken, maar host B kan nog wel voorgeven de eigenaar van host A's MAC-adres te zijn.

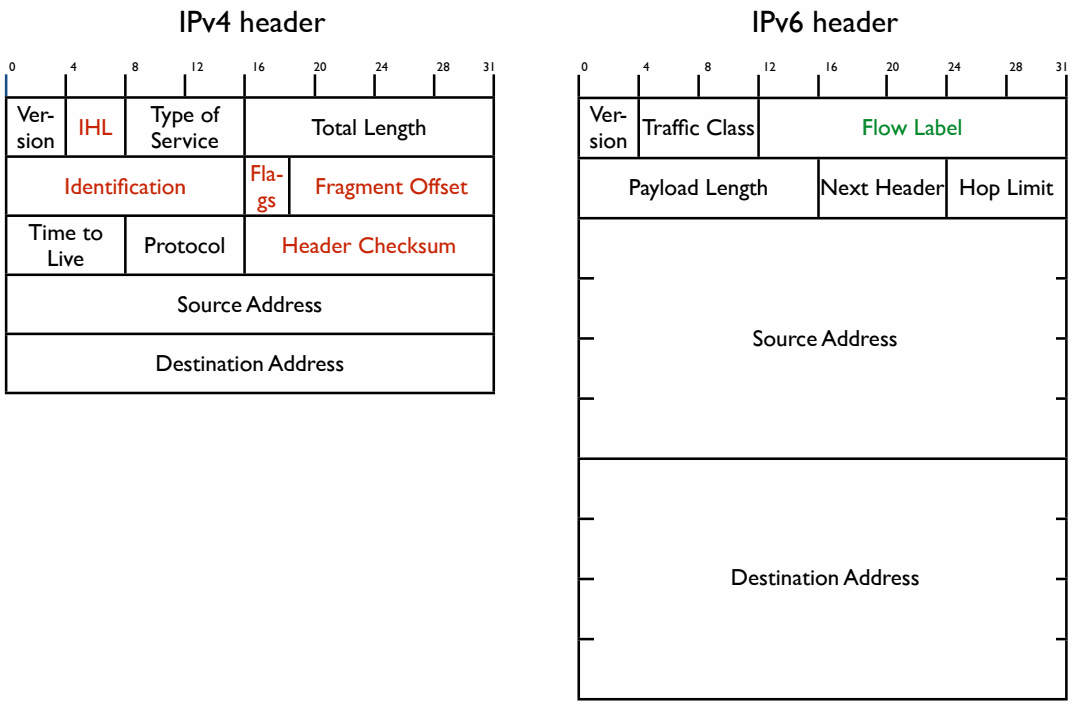
5.2. Het filteren van extra headers en fragmentatie

Bij het firewallen/filteren van IP-pakketten is er geen standaardaanpak die in alle gevallen toepasbaar is. Zodoende doen we geen specifieke aanbevelingen, maar informatie die nuttig is bij het maken van filters.

ⓘ In een dual stack netwerk moet alles dat gefilterd is in IPv4 ook gefilterd worden in IPv6.

De IPv6-header is versimpeld ten opzichte van de IPv4-header. Afgezien van de (veel) langere adressen zijn de belangrijkste verschillen dat de velden die gebruikt worden voor fragmentatie niet meer aanwezig zijn en dat de header altijd 40 bytes lang is. Als additionele functionaliteit vereist is dan worden extra headers (extension headers) geplaatst tussen de IPv6-header en de inhoud van het pakket.

Figuur 1, de IPv4- en IPv6-headers



Bijvoorbeeld, wanneer een IPv6-pakket te groot is om verstuurd te kunnen worden over een interface dan moet het gefragmenteerd worden, net als bij IPv4. Omdat routers niet mogen fragmenteren bij IPv6 moet Path MTU Discovery gebruikt worden wanneer een IPv6-host een pakket verstuurt dat groter is dan 1280 bytes (de minimale maximumpakketgroote (MTU) bij IPv6). Hosts passen dan waar nodig is de pakketgrootte aan voor latere pakketten. Bij TCP kan de pakketgrootte zonder problemen gereduceerd worden, maar bij UDP-gebaseerde protcollen kunnen de pakketgrenzen niet zomaar veranderd worden. UDP moet dus te grote pakketten fragmenteren. Om dit te doen zijn de drie fragmentatie-gerelateerde velden die nu ontbreken in de IPv6-header nodig: Identification, Flags and Fragment Offset. Zodoende wordt een Fragment Header toegevoegd tussen de IPv6-header en de UDP-header.

RFC 6980 bespreekt de problemen die voor kunnen komen wanneer Neighbor Discovery-pakketten gefragmenteerd worden; sommige implementaties negeren zulke ND-pakketten, maar ze kunnen ook gebruikt worden om RA Guard-implementaties in switches te omzeilen omdat de informatie die nodig is om RA Guard uit te voeren over meerdere fragmenten verdeeld is. Om deze reden specificeert RFC 6980 dat ND-pakketten niet gefragmenteerd mogen worden.

5.3. Extra headers

Normaal gesproken werkt het Next Header-veld in de IPv6-header hetzelfde als het Protocol-veld in de IPv4-header en bevat het gewoonlijk de waarden 6 (TCP), 17 (UDP) of 58 (ICMPv6). Maar wanneer er een extra header (extension header) aanwezig is, zoals de Fragment Header, dan bevat het Next Header-veld het nummer van het type extra header, in dit geval 44 voor de Fragment Header. Deze extra headers hebben een eigen Next Header-veld, wat in dit geval de waarde 17 (UDP) bevat.

Dit heeft als gevolg dat firewalls en IP filters de rij opeenvolgende headers langs moeten lopen om te bepalen of een pakket TCP, UDP of ICMP is. In het verleden waren er wel filters die alleen keken naar de waarde in het Next Header-veld in de IPv6-header, en zodoende niet in staat waren gefragmenteerde pakketten naar behoren te filteren. Let hierop bij het kiezen van een firewall of het toepassen van IP filters. Een complicerende factor is dat hoewel de **meeste** extra headers een vaste indeling volgen, dit niet verplicht

is, waardoor een oudere firewall geen extra headers kan verwerken die gedefinieerd zijn nadat de firewallsoftware klaar was.

De Fragment Header is de meest gebruikelijke extra header bij IPv6, maar anderen zijn ook mogelijk, waaronder Shim6, de IPsec Authentication Header, de Routing Header, de Hop-by-Hop Options Header en de Destination Options Header.

De Routing Header wordt gebruikt voor source routing; type 0 (RH0) is "deprecated", wat wil zeggen dat het gebruik ervan zeer sterk afgeraden wordt door de IETF. Type 2 (RH2) blijft wel in gebruik voor Mobile IPv6 (MIPv6). De Hop-by-Hop Options Header wordt gebruikt voor additionele functies waarvoor het nodig is dat iedere router langs het pad dat het pakket volgt een extra actie uitvoert, en de Destination Options Header voor additionele functies waarbij alleen de bestemming extra werk hoeft te doen. Geen van deze wordt veel gebruikt; zie de IANA website voor een overzicht. Zie RFC 4942 voor gedetailleerde informatie over extension headers en fragmentatie.

5.4. Hardware- versus software-filters/firewalls

In het algemeen is het zo dat hardware- en softwarefirewalls op een fundamenteel andere wijze opereren, en het is belangrijk dat verschil te onderkennen. In het algemeen zullen hardware-firewalls die geen IPv6 ondersteunen IPv6-pakketten simpelweg niet "zien" en ze dus ook niet doorlaten. (Controleer dit wel voor het in gebruik nemen van een hardwarefirewall.) Software-firewalls en filters die IPv4 filteren daarentegen zijn simpelweg niet van toepassing op IPv6, en laten dus gewoonlijk **alle** IPv6-pakketten door.

Zorg er dus voor bij het opzetten van een firewall of bij het instellen van filters op routers en servers dat IPv4 en IPv6 **beide** gefilterd worden zoals bedoeld. In veel gevallen zullen aparte filterregels aangemaakt moeten worden voor IPv4 en IPv6, maar sommige firewallsoftware (zoals PF) maken het mogelijk beide protocollen te filteren met één filterregel.

Veel grotere routers, en mogelijk ook firewalls, gebruiken hardware-gebaseerde pakketfilters waarbij de filterhardware alleen naar de eerste 64, 128 of 256 bytes van een pakket kan kijken. Dit betekent dat zulke filters omzeild kunnen worden door het gebruik van grote of veel extra headers. Zie IPv6 ACL bypass.

5.5. (Unique) Site Local adressen

Site-local IPv6-adressen waren bedoeld voor systemen die niet verbonden worden aan het wereldwijde internet. In dat opzicht zijn deze adressen vergelijkbaar met RFC 1918 IPv4-adressen, met het verschil dat RFC 1918-adressen over het algemeen achter NATs gebruikt worden. Maar de IETF kon geen goede manier bedenken waarop routers dienen te functioneren wanneer ze aan twee privénetwerken gekoppeld zijn die beide dezelfde reeks site-localadressen gebruiken.

Zodoende definieerde de IETF een ander type site-localadressen: Unique Site Local Addresses (ULAs). Hierbij is het idee dat elk netwerk z'n eigen reeks site-localadressen gebruikt. Twee soorten zijn gedefinieerd: één waarbij een centraal uitgiftepunt ULA-prefixen uitgeeft en één waarbij gebruikers hun eigen ULA-prefix genereren. Het eerste type is nooit in gebruik genomen, dus organisaties die site-localadressen willen gebruiken dienen één (of meer) /48 ULA-prefixen te genereren op basis van een willekeurig nummer. Verschillende websites kunnen dit doen, of consulteer RFC 4193. 40 bits in de ULA-prefix zijn willekeurig gegenereerd, het is dus zeer onwaarschijnlijk dat twee ULA-gebruikers die elkaar kennen dezelfde ULA-prefix genereren, zolang ze tenminste beide de correcte procedure voor het genereren gebruiken.

ULAs komen uit fc00::/7 en "het wordt niet verwacht dat ze routeerbaar zijn op het publieke internet".

5.6. Global unicast adressen

Op dit moment zijn alleen adressen binnen 2000::/3 gedefinieerd als global unicast adressen. (In andere woorden: gewone publieke adressen.) Dus bijna driekwart van de IPv6-adresruimte is opzij gezet voor toekomstig gebruik. Dit betekent echter niet dat het noodzakelijkerwijs een goed idee is om al die adressen uit te filteren. Veel mensen installeerden filters om ongebruikte IPv4-adresruimte te filteren, maar lieten vervolgens na die filters aan te passen wanneer meer adresruimte in gebruik genomen werd. En het klasse E IPv4-adresblok (240.0.0.0/4) kan op veel systemen niet gebruikt worden omdat het als "gereserveerd voor toekomstig gebruik" te boek staat. Met als resultaat dat nu we die adressen nodig hebben, ze niet bruikbaar zijn.

Dus volgens de IPv6-specificaties moet alle IPv6-adresruimte die gereserveerd is voor toekomstig gebruik hetzelfde behandeld worden als global unicast. Kiest u ervoor om niet-2000::/3 adresruimte te filteren, zorg er dan voor dat er procedures zijn om op de hoogte te blijven van het ingebruiknemen van meer adressen als global unicast en vervolgens filters aan te passen.

5.7. Stateful firewalling als vervanging voor NAT

Hosts die beschermd werden tegen ongevraagde inkomende pakketten door een IPv4 NAT hebben die bescherming niet bij IPv6 bij het gebruik van global unicast adressen. Het is belangrijk om te onderkennen dat de meeste NATs niet waterdicht zijn: er zijn verschillende manieren om er pakketten doorheen te krijgen, zeker als de host achter de NAT dit actief probeert. Daarnaast zijn hosts op hetzelfde lokale netwerk achter dezelfde NAT niet tegen elkaar beschermd. Zodoende dienen hosts met onverwachte inkomende pakketten om te kunnen gaan.

Desondanks is het vaak gewenst om bepaalde pakketten van het lokale netwerk weg te houden. Zonder NAT kan dit gedaan worden met een stateful firewall. Een stateful firewall houdt uitgaande pakketten bij en laat dan inkomende pakketten door die eruit zien als antwoorden op eerdere uitgaande pakketten. Inkomende pakketten die niet overeenkomen met eerder opgezette TCP-sessies of uitgaande UDP-pakketten worden "gedropt" (tegengehouden).

Een stateful firewall heeft dus als doel de functionaliteit die NAT levert als bijeffect. Zodoende is het over het algemeen veel moeilijker een stateful firewall te passeren dan een NAT, met als resultaat dat veel peer-to-peerapplicaties, waaronder SIP voor VoIP en videochat, slechter werken over IPv6 met een stateful firewall dan over IPv4 met NAT. Wel heeft een firewall het voordeel boven NAT dat de firewall selectief uitgeschakeld kan worden.

Bij het implementeren van pakketfilters en/of een stateful firewall is RFC 6092 een goede plek om te beginnen. Deze RFC bevat aanbevelingen over hoe thuisrouters (CPEs), die vaak niet actief beheerd worden, het best de beveiliging van een thuisnetwerk kunnen regelen.

5.8. Filteren van ICMPv6 en adresreeksen

Hier volgt een overzicht van de IPv6-adresruimte en de implicaties van het filteren van verschillende prefixen.

Tabel 3, filterimplicaties van IPv6-prefixen

Prefix	Eerste bits	Doel	Filterimplicaties
::/3	000	Speciale doeleinden	Kan gefilterd worden aan rand netwerk
::1/128		Localhost	In hosts: niet filteren / wees voorzichtig*
::ffff:0:0/96		IPv4-mapped API adressen	In hosts: niet filteren / wees voorzichtig*
64:ff9b::/96	0000 0000 0110 0100	NAT64 well-known prefix	Filter aan rand netwerk
2000::/3	001	Global unicast	Niet filteren
2001::/32		Teredo	Filteren kan in enkel geval tot timeouts leiden
2002::/16	0010 0000 0000 0010	6to4	Filteren leidt soms tot timeouts
4000::/2 - f800::/6	01 - 1111 01	Gereserveerd	Filter alleen als er een updateprocedure is
fc00::/7	1111 110	Unique Site Local	Filter aan rand netwerk
fe80::/10	1111 1110 01	Link-local	Niet filteren / wees voorzichtig**, gaat niet door routers
fec0::/10	1111 1110 11	Oude site-local	Filter
ff00::/8	1111 1111	Multicast	Filter niet, maar doe alleen multicastrouting waar nodig

* Per ongeluk pakketten van/naar ::1/128 of ::ffff:0:0/96 filteren binnen een host zorgt voor grote problemen. Deze pakketten kunnen echter veilig gefilterd worden in routers als onderdeel van een filterregel die ::/3 blokkeert.

** Hoewel er wat te zeggen valt voor het filteren van pakketten van/naar fe80::/10, de link-localprefix, heeft het geen nut dit te doen in routers of firewalls. Routers geven deze pakketten sowieso niet door, en zo'n filter is gevaarlijk omdat gemakkelijk legitieme link-localpakketten geblokkeerd kunnen worden, zoals die voor Neighbor Discovery en routingprotocollen.

Netwerken dienen alleen uitgaande IPv6-pakketten door te laten als die pakketten een bronadres hebben dat toebehoort aan het betreffende netwerk. Zie RFC 2827 / Best Current Practice 38.

Tabel 4 bevat een overzicht van ICMPv6-foutmeldingen (typenummers onder de 128) en de implicaties van het filteren hiervan. Let op dat bij het gebruik van een stateful firewall, het niet nodig zou moeten zijn om inkomende ICMPv6-foutmeldingen te filteren, de firewall hoort deze automatisch te filteren als ze niet het antwoord zijn op eerdere uitgaande pakketten.

Tabel 4, filterimplicaties van ICMPv6-foutmeldingen

Type	Naam	Doel	Filterimplicaties
1	Destination Unreachable	Communiceert onbereikbare bestemmingen	Filteren leidt tot timeouts
2	Packet Too Big	Path MTU Discovery	Filteren maakt communicatie met bepaalde bestemmingen onmogelijk
3	Time Exceeded	Traceroute	Filteren blokkeert traceroute, RFC 4890 beveelt doorlaten Code 0 aan
4	Parameter problem	Communiceert protocolfouten	Moeilijk te zeggen, is ongebruikelijk, RFC 4890 beveelt doorlaten Code 1 + 2 aan

Als laatste in tabel 5 een overzicht van de ICMPv6-informatieberichten (typenummers boven de 127) en de implicaties van het filteren hiervan.

Tabel 5, filterimplicaties van ICMPv6-informatieberichten

Type	Naam	Doel	Filterimplicaties
128	Echo Request	Ping	Ping werkt niet, misschien impact Teredo
129	Echo Reply	Ping	Ping werkt niet, misschien impact Teredo
130 - 132, 143	MLD	Vertelt routers waar multicasts te sturen	Multicast-applicaties zullen waarschijnlijk niet werken
133	Router Solicitation	Triggert Router Advertisement	Default gateway and adresconfiguratie zijn veel langzamer
134	Router Advertisement	Default gateway, adresconfiguratie	Hosts krijgen geen default gateway en geen global unicast-adressen
135	Neighbor Solicitation	Lokale communicatie, adresconfiguratie	Hosts kunnen niet communiceren over IPv6
136	Neighbor Advertisement	Lokale communicatie, adresconfiguratie	Hosts kunnen niet communiceren over IPv6
137	Redirect Message	Optimaliseert hoe pakketten lopen	Pakketten moeten misschien langs een extra router
139	Node Information Query	Debugging	Geen
140	Node Information Response	Debugging	Geen
144 - 147	Mobile IPv6	Mobile IPv6	MIPv6 werkt niet (MIPv6 is niet algemeen geïmplementeerd)
148 - 149	SEND	SEcure Neighbor Discovery	SEND werkt niet (SEND is niet algemeen geïmplementeerd)
152 - 153	Multicast Router Discovery	Optimaliseert multicast forwarding door switches (vgl. IGMP snooping)	Switches kunnen multicasts blokkeren of onnodig doorsturen

Let op het verschil tussen IPv4, waar ARP-pakketten nooit gefilterd worden omdat dit geen IPv4-pakketten zijn, en IPv6, waar Neighbor Discovery-pakketten per ongeluk gefilterd kunnen worden. Cisco routers gebruiken een "implicit deny" waar alle niet expliciet toegestane pakketten uitgefilterd worden, maar die implicit deny is niet van

toepassing op ND-pakketten. Maar een expliciete "deny all" filtert **wel** Neighbor Discovery-pakketten, dus in dit geval moet ND eerst expliciet toegestaan worden.

Het is onnodig om Neighbor Discovery, Redirect en Multicast Listener/Router Discovery and SEcure Neighbor Discovery ICMPv6-pakketten te filteren aan de rand van het netwerk, aangezien hosts deze alleen accepteren als ze lokaal gegenereerd zijn. (De Hop Limit = 255 beveiligingscontrole.) Mocht u ze toch willen filteren om mogelijke implementatiefouten in hosts te omzeilen, let dan op dat ND-pakketten **van en naar de router** toegestaan worden door het filter.

Voor ICMPv6 is een filter dat alle noodzakelijke en gewenste ICMP-types doorlaat (whitelist) en alle overige blokkeert een geëigende keuze.

Zie RFC 4890 voor een volledige discussie over het filteren van ICMPv6-pakketten.

Zoals besproken in sectie 5.2 kan een Fragment Header gebruikt worden door aanvallers om langs filters te komen. Het is echter meestal niet mogelijk om alle pakketten met een Fragment Header uit te filteren, aangezien legitieme pakketten gefragmenteerd kunnen zijn. Het meest gebruikelijke voorbeeld is UDP DNS-pakketten die voor DNSSEC gebruikt worden. Deze zijn vaak groter dan de maximale Ethernet-grootte van 1500 bytes en worden zodoende gefragmenteerd. Neighbor Discovery-pakketten met een Fragment Header kunnen weggefilterd worden.

Er zijn twee manieren om te filteren: opnoemen wat geblokkeerd dient te worden en dan al het overige doorlaten (default allow) of opnoemen wat toegestaan is en al het overige blokkeren (default deny). Omdat het moeilijk te voorspellen is of een gegeven communicatiesessie plaats zal vinden over IPv4 of IPv6 (als beide beschikbaar zijn) wordt een consistente gebruikerservaring het best gediend door ofwel default allow ofwel default deny voor zowel IPv4 als IPv6 te gebruiken. Dus als voor IPv4 default deny gebruikt wordt, doe dit dan ook voor IPv6, als voor IPv4 default allow gebruikt wordt, doe dan hetzelfde voor IPv6.

5.9. Filteren op prefixlengte in BGP

In de meeste opzichten functioneren IPv6-routingprotocollen nagenoeg hetzelfde als IPv4-routingprotocollen. Een belangrijk verschil is dat het bij IPv4 gebruikelijk is dat alleen /24 en kortere prefixen via BGP doorgegeven worden. Voor IPv6 is er niet een vergelijkbare de-facto maximale prefixlengte. De meest gebruikelijke IPv6-prefixlengte die in BGP voorkomt is /32, dat is de grootte van prefixen die aan middelgrote ISPs uitgedeeld wordt. Grote ISPs hebben korteren prefixen. Daarnaast kunnen eindgebruikersorganisaties /48 "provider independent" prefixen krijgen. Het is dus mogelijk prefixen langer dan /48 uit te filteren zonder dat dit problemen oplevert.

Bij IPv4 wil het wel eens gebeuren dat een ISP met bijvoorbeeld een /16 de 256 /24s waaruit die /16 bestaat lekt. Bij IPv6 zou een vergelijkbare gebeurtenis dramatische consequenties hebben, aangezien een /32 die aan een ISP gegeven is bestaat uit 65000 /48s, wat waarschijnlijk genoeg is om veel BGP-routers in geheugenproblemen te brengen. Het is dus noodzakelijk om naast de geëigende inkomende en uitgaande prefixfilters en het implementeren van MD5-wachtwoorden op BGP-sessies ook een limiet op het maximale aantal prefixen op BGP-sessies te zetten om deze eventualiteit te voorkomen.

6. IPV6 UITSCHAKELEN

Mocht u willen voorkomen dat IPv6 op uw netwerk gebruikt wordt, dan zijn er verschillende manieren om dit voor elkaar te krijgen. Deze zijn echter niet allemaal even effectief.

Een voor-de-handliggende optie is om simpelweg IPv6 niet aan te zetten in routers en firewalls IPv6-pakketten niet te laten doorgeven. In dit geval kunnen hosts op hetzelfde subnet nog steeds IPv6-pakketten uitwisselen, en IPv6-pakketten kunnen getunneld worden van/naar het internet. Sommige van deze getunnelde pakketten zijn makkelijk te filteren (protocol 41 voor 6to4, UDP poort 3544 voor Teredo), maar er zijn veel verschillende tunnelmechanismen en gebruikers kunnen proberen hun getunnelde pakketten te versleutelen of alternatieve poortnummers gebruiken.

Een andere optie is IPv6 uitzetten in de netwerkconfiguratie van alle hosts die met het netwerk verbonden zijn. Let wel op dat recente versies van Mac OS X het niet langer mogelijk maken IPv6 volledig uit te schakelen via het grafische gebruikersinterface. En uiteraard is deze optie alleen mogelijk wanneer alle apparaten die aan het netwerk gekoppeld worden onder de controle zijn van systeembeheerders.

Als laatste is er de optie om switches en Wi-Fi-basisstations het IPv6 ethertype, 0x86dd, uit te filteren, waardoor het onmogelijk wordt om IPv6-pakketten te versturen via de betreffende switches en basisstations. Filter in dit geval ook IPv6-tunnelprotocollen zoals protocol 41 (6to4, ISATAP) en UDP poort 3544 (Teredo). Zie RFC 7059 voor meer informatie.

ⓘ Sommige Windows-componenten of -applicaties hebben IPv6 nodig om te kunnen functioneren. Microsoft "beveelt aan dat u IPv6 aan laat staan, zelfs als u geen IPv6 draait in uw netwerk". Zie de Technet IPv6 for Windows FAQ.

7. COLOFON

Rogier Spoor, SURFnet, gaf opdracht voor dit document.

Christiaan Ottow, Pine Digital Security, heeft het document geëvalueerd en veel verbeteringen gesuggereerd.

Roel Hoek, Universiteit Twente, heeft het document geëvalueerd en verbeteringen gesuggereerd.

Iljitsch van Beijnum heeft de tekst geschreven.

8. BIBLIOGRAFIE

SURFnet, "Een IPv6 nummerplan opstellen". http://www.surf.nl/kennis-en-innovatie/kennisbank/2013/whitepaper-ipv6-nummerplan-opstellen.html

Christiaan Ottow et al. "The Impact of IPv6 on Penetration Testing." Information and Communication Technologies. Springer Berlin Heidelberg, 2012.

Wikipedia, "PF (firewall)". http://en.wikipedia.org/wiki/PF_(firewall)

Microsoft Technet, "IPv6 for Microsoft Windows: Frequently Asked Questions". http://technet.microsoft.com/en-us/network/cc987595.aspx

Saku Ytti, "IPv6 ACL bypass". http://blog.ip.fi/2011/08/ipv6-acl-bypass.html

Marc Heuse, "IPv6 Insecurity Revolutions". Hack in the Box, Kuala Lumpur, 2012. http://www.youtube.com/watch?v=t9d7p3zxoiM, http://conference.hitb.org/hitbsecconf2012kul/materials/

RFC 1918 Address Allocation for Private Internets. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear. February 1996. (Format: TXT=22270 bytes) (Obsoletes RFC1627, RFC1597) (Updated by RFC6761) (Also BCP0005) (Status: BEST CURRENT PRACTICE)

RFC 2827 Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. P. Ferguson, D. Senie. May 2000. (Format: TXT=21258 bytes) (Obsoletes RFC2267) (Updated by RFC3704) (Also BCP0038) (Status: BEST CURRENT PRACTICE)

RFC 3756 IPv6 Neighbor Discovery (ND) Trust Models and Threats. P. Nikander, Ed., J. Kempf, E. Nordmark. May 2004. (Format: TXT=56674 bytes) (Status: INFORMATIONAL)

RFC 3971 SEcure Neighbor Discovery (SEND). J. Arkko, Ed., J. Kempf, B. Zill, P. Nikander. March 2005. (Format: TXT=123372 bytes) (Updated by RFC6494, RFC6495, RFC6980) (Status: PROPOSED STANDARD)

RFC 3964 Security Considerations for 6to4. P. Savola, C. Patel. December 2004. (Format: TXT=83360 bytes) (Status: INFORMATIONAL)

RFC 4193 Unique Local IPv6 Unicast Addresses. R. Hinden, B. Haberman. October 2005. (Format: TXT=35908 bytes) (Status: PROPOSED STANDARD)

RFC 4294 IPv6 Node Requirements. J. Loughney, Ed.. April 2006. (Format: TXT=39125 bytes) (Obsoleted by RFC6434) (Updated by RFC5095) (Status: INFORMATIONAL)

RFC 4620 IPv6 Node Information Queries. M. Crawford, B. Haberman, Ed.. August 2006. (Format: TXT=31134 bytes) (Status: EXPERIMENTAL)

RFC 4861 Neighbor Discovery for IP version 6 (IPv6). T. Narten, E. Nordmark, W. Simpson, H. Soliman. September 2007. (Format: TXT=235106 bytes) (Obsoletes RFC2461) (Updated by RFC5942, RFC6980) (Status: DRAFT STANDARD)

RFC 4864 Local Network Protection for IPv6. G. Van de Velde, T. Hain, R. Droms, B. Carpenter, E. Klein. May 2007. (Format: TXT=95448 bytes) (Status: INFORMATIONAL)

RFC 4890 Recommendations for Filtering ICMPv6 Messages in Firewalls. E. Davies, J. Mohacsi. May 2007. (Format: TXT=83479 bytes) (Status: INFORMATIONAL)

RFC 4941 Privacy Extensions for Stateless Address Autoconfiguration in IPv6. T. Narten, R. Draves, S. Krishnan. September 2007. (Format: TXT=56699 bytes) (Obsoletes RFC3041) (Status: DRAFT STANDARD)

RFC 4942 IPv6 Transition/Co-existence Security Considerations. E. Davies, S. Krishnan, P. Savola. September 2007. (Format: TXT=102878 bytes) (Status: INFORMATIONAL)

RFC 5095 Deprecation of Type 0 Routing Headers in IPv6. J. Abley, P. Savola, G. Neville-Neil. December 2007. (Format: TXT=13423 bytes) (Updates RFC2460, RFC4294) (Status: PROPOSED STANDARD)

RFC 5952 A Recommendation for IPv6 Address Text Representation. S. Kawamura, M. Kawashima. August 2010. (Format: TXT=26570 bytes) (Updates RFC4291) (Status: PROPOSED STANDARD)

RFC 6092 Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service. J. Woodyatt, Ed.. January 2011. (Format: TXT=91729 bytes) (Status: INFORMATIONAL)

RFC 6105 IPv6 Router Advertisement Guard. E. Levy-Abegnoli, G. Van de Velde, C. Popoviciu, J. Mohacsi. February 2011. (Format: TXT=20817 bytes) (Status: INFORMATIONAL)

RFC 6145 IP/ICMP Translation Algorithm. X. Li, C. Bao, F. Baker. April 2011. (Format: TXT=76484 bytes) (Obsoletes RFC2765) (Updated by RFC6791) (Status: PROPOSED STANDARD)

RFC 6146 Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. M. Bagnulo, P. Matthews, I. van Beijnum. April 2011. (Format: TXT=107954 bytes) (Status: PROPOSED STANDARD)

RFC 6296 IPv6-to-IPv6 Network Prefix Translation. M. Wasserman, F. Baker. June 2011. (Format: TXT=73700 bytes) (Status: EXPERIMENTAL)

RFC 6434 IPv6 Node Requirements. E. Jankiewicz, J. Loughney, T. Narten. December 2011. (Format: TXT=64407 bytes) (Obsoletes RFC4294) (Status: INFORMATIONAL)

RFC 6724 Default Address Selection for Internet Protocol Version 6 (IPv6). D. Thaler, Ed., R. Draves, A. Matsumoto, T. Chown. September 2012. (Format: TXT=74407 bytes) (Obsoletes RFC3484) (Status: PROPOSED STANDARD)

RFC 6936 Client Link-Layer Address Option in DHCPv6 G. Halwasia, S. Bhandari, W. Dec. May 2013. (Format: TXT=14739 bytes) (Status: PROPOSED STANDARD)

RFC 6980 Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery. F. Gont. August 2013. (Format: TXT=20850 bytes) (Updates RFC3971, RFC4861) (Status: PROPOSED STANDARD)

RFC 7059 A Comparison of IPv6-over-IPv4 Tunnel Mechanisms. S. Steffann, I. van Beijnum, R. van Rein. November 2013. (Format: TXT=98886 bytes) (Status: INFORMATIONAL)

SURFnet

Radboudkwartier 273
3511 CK Utrecht

Postbus 19035
3501 DA Utrecht

030 – 2 305 305
www.surfnet.nl