**Seriál Bezpečné IPv6**
**Bezpečné IPv6 : směrovač se hlásí**

Seriál jsme rozdělili na několik dílů, ve kterých bychom Vás chtěli seznámit s několika typy útoků, které jsou u protokolu IPv6 víceméně novinkou. Není naším záměrem dávat do rukou útočníků jednoduché návody na provádění útoků, ale poukázat na to, že řada útoků je skutečně velmi snadno proveditelná a jejich důsledky mohou být poměrně fatální. V seriálu se zejména soustředíme na principy útoků, popíšeme, jak je realizovat, a následně se společně pokusíme najít možné způsoby, jak uvedené útoky eliminovat. V dnešní době to ale v mnoha případech nebudou úkoly zcela jednoduché, což je dáno několika faktory:

- IPv6 protokol je nekompatibilní s protokolem IPv4. V praxi to tedy znamená, že pokud je nějaké zabezpečení v síti již realizováno pro protokol IPv4 (například firewall), nic to neříká o bezpečnostních pravidlech aplikovaných na IPv6. Toto platí samozřejmě i obráceně.
- Byť jsou první specifikace protokolu IPv6 téměř dvacet let staré, je protokol v porovnání s nasazením IPv4 značně nezralým sourozencem a přirozený provozní tlak na řešení některých typů problémů je tedy zatím velice malý.
- Třetím významným faktorem je to, že řada výrobců zatím IPv6 nebere zcela vážně. Dnes je naprosto typické, že nový výrobek je uveden na trh nejdříve s podporou IPv4 a podpora IPv6 se přidává až v následných opravách. V takových případech je opět celkem logické, že přednost má základní funkčnost (aby to nějak fungovalo) a bezpečnostní prvky (aby to bylo bezpečné) se objevují až v dalších verzích firmware.

Všechny tyto aspekty dohromady nám pak poměrně znepříjemňují situaci, pokud chceme udržet alespoň stejnou míru zabezpečení jak pro IPv4, tak pro IPv6. V praxi to znamená, že se mnohdy musíme uchylovat k různým trikům a chybějící prvky v zabezpečení IPv6 řešit alternativními způsoby. Někdy ale ani tak neřešíme chybějící implementaci bezpečnostních mechanismů, jako spíše obcházíme architekturu IPv6, abychom eliminovali nově vzniklé problémy.

Při sestavování jednotlivých dílů seriálu jsme vycházeli z naších praktických zkušenosti při implementaci protokolu IPv6 v rámci počítačové sítě Vysokého učení technického v Brně, která se dle statistik sledovaných společnosti Akamai nebo statistik vedených Geoffem Hustonem celosvětově pohybuje na čelních příčkách v míře penetrace IPv6 na koncových zařízeních. Velký dík patří rovněž i kolegům z ostatních univerzit, kteří se v rámci pracovních skupin organizovaných sdružením CESNET, neváhají podělit o vlastní zkušenosti spojené s implementací protokolu IPv6 v jejich sítích.

Oznámení směrovače

Mezi snad nejznámější typ útoku v sítích IPv6 patří zneužití zpráv *Oznámení směrovače* (*Router Advertisement*, RA). *Oznámení směrovače* je integrální součást autokonfigurace koncových zařízení specifikované jako součást mechanismu objevování sousedů *Neighbor Discovery for IP version 6* - RFC 4861). Každé zařízení připojené do IPv6 sítě si prostřednictvím zprávy *Výzva směrovači* (*Router Solicitation*, RS) může požádat okolní směrovače o předání potřebných údajů pro komunikaci v síti (například adresa směrovače, prefix sítě atd.). Tyto informace se předávají prostřednictvím zprávy *Oznámení směrovače*, která je zasílaná všem zařízením v příslušné podsíti.

Vlastní zpráva *Oznámení směrovače* má relativně jednoduchý formát a v podstatě pouze ostatním uživatelům v síti říká: "Já jsem směrovač a můžeš mě použít jako cestu do Internetu (*default gateway*)." Tato samotná informace nám však často nestačí, a proto se k této zprávě přidávají další konfigurační údaje vhodné pro koncové zařízení - zejména prefix sítě. Tyto konfigurační údaje jsou pak zaslány jako ICMPv6 zpráva všem zařízením v síti. Na rozdíl od automatické konfigurace řešené prostřednictvím DHCP (jak v4, tak v6) se mohou koncová zařízení dozvědět prakticky okamžitě, že mají začít používat jiný síťový prefix (v případě přečíslování), anebo jiný směrovač (například při výpadku některého z nich).

Legitimní zpráva RA může vypadat přibližně následovně:

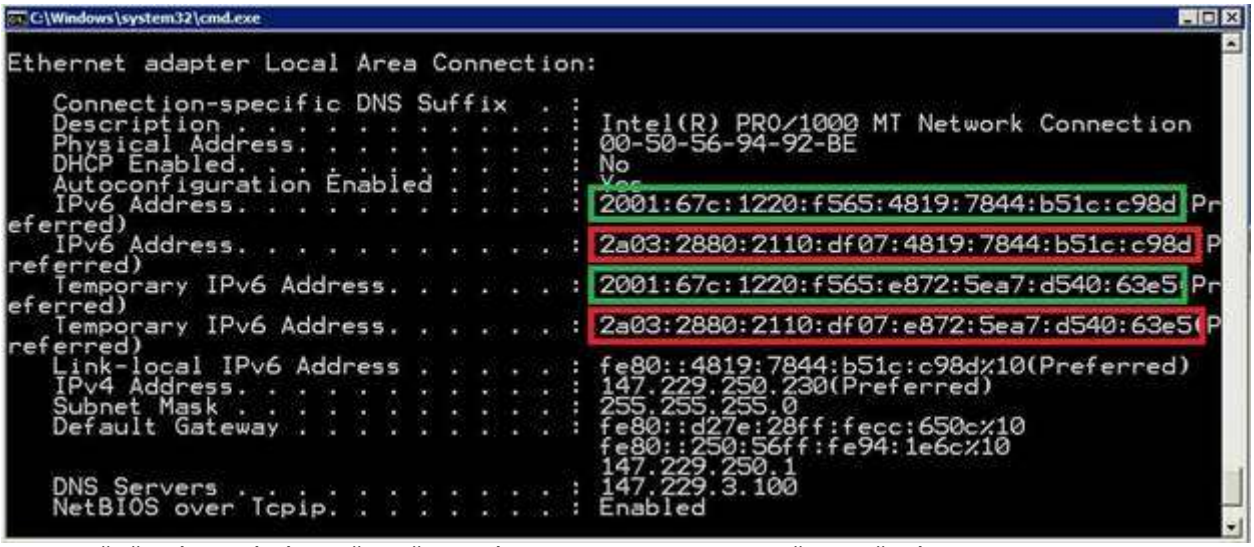
```
▷ Ethernet II, Src: 00:04:96:1d:4e:30 (00:04:96:1d:4e:30), Dst: 33:33:00:00:00:01 (33:33:00:00:00:01)
▷ Internet Protocol Version 6, Src: fe80::204:96ff:fe1d:4e30 (fe80::204:96ff:fe1d:4e30), Dst: ff02::1 (ff02::1)
▽ Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0xa4a4 [correct]
    Cur hop limit: 64
  ▷ Flags: 0x80
    Router lifetime (s): 1800
    Reachable time (ms): 30000
    Retrans timer (ms): 1000
  ▷ ICMPv6 Option (Source link-layer address : 00:04:96:1d:4e:30)
  ▷ ICMPv6 Option (Prefix information : 2001:67c:1220:80c::/64)
```

Vidíme, že směrovač posílá ze své *link-local* adresy (fe80::204:96ff:fe1d:4e30) oznámení směrovače všem uzlům v lokální síti (multicast adresa *All nodes address* ff02::1). V *Oznámení směrovače* šíří směrovač informaci o používaném prefixu IPv6 - 2001:67c:1220:80c::/64 a své adrese MAC. *Link-local* adresu směrovače (fe80::204:96ff:fe1d:4e30) si koncová zařízení uloží do své směrovací tabulky, jako výchozí bránu a informace o prefixu použijí pro vytvoření unikátní adresy.

Jak jistě tušíte, tento mechanismus přímo svádí k některým typům útoků. Za směrovač se může prohlásit jakékoliv zařízení a prostřednictvím falešného *Oznámení směrovače* "podstrčit" všem zařízením v síti nové konfigurační údaje. Ukažme si na příkladu, jak může věc fungovat v praxi. Pro náš případ využijeme nástroje THC-IPV6, který obsahuje hned několik užitečných nástrojů, které budeme používat i v některých dalších scénářích. Jedním z nástrojů je utilita fake_router6, která z libovolného PC s Linuxem vytvoří falešný směrovač IPv6, a to prostým spuštěním příkazu:

```
# ./fake_router6 -X eth0 2a03:2880:2110:df07::/64
Starting to advertise router 2a03:2880:2110:df07:: (Press Control-C to end) ...
```

Pokud se po spuštění příkazu podíváme do konfigurace některého z počítačů připojeného ve stejné podsíti, tak uvidíme v konfiguraci adres přibližně toto:




Kromě adresy přidělené regulérním směrovačem v síti (2001:67c:1220:*, zeleně vyznačené), *Link-local* adres (fe80:*) jsou taky na rozhraní nakonfigurovány námi "podstrčené" adresy (2a03:2880:2110:df07:*, červeně vyznačené).
Jestli někomu ony podvržené adresy přijdou povědomé, tak to není náhoda. Pokud se podíváte do databáze whois tak zjistíte, že tyto adresy jsou přiděleny společnosti Facebook. Skrývá se v tom další záměr. Představme si situaci, kdy útočník na svém PC spustí následující posloupnost příkazů:

```
# host facebook.com
facebook.com has address 173.252.110.27
facebook.com has IPv6 address 2a03:2880:2110:df07:face:b00c:0:1
facebook.com mail is handled by 10 msgin.t.facebook.com.
# ip address add 2a03:2880:2110:df07:face:b00c:0:1/64 dev eth0
```

Tedy nejprve si zjistí, pod jakou IPv6 adresou se skrývá doména facebook.com a tu stejnou adresu nakonfiguruje na svém rozhraní. Pro uživatele v příslušné podsíti je výsledek poměrně přímočarý. Při pokusu o přístup na Facebook se mu může zobrazit něco podobného:


Díky tomu, že koncová zařízeni v příslušné síti mají nakonfigurovanou stejnou adresu podsítě, v jaké je Facebook, pakety k Facebook serveru nejsou odeslány na adresu směrovače, ale provoz je navázán s podvrženou adresou na útočníkově PC (server útočníka se tváří, že je ve stejné síti jako uživatel). Sofistikovanější verzi tohoto útoku pak může být vytvoření MitM (*Man in the Middle*) proxy, která přesměrovává provoz na regulérní server. Útočník pak může zasáhnout do probíhající komunikace podle svých potřeb. Zprovoznění takovéto proxy se zabývá například článek Ondřeje Caletky.
Předchozí příklad měl pouze demonstrovat, jak je s využitím IPv6 poměrně snadné (spuštěním dvou příkazů) upravit konfiguraci jedné podsítě tak, aby bylo možné podvrhnout komunikaci. Celkem oprávněně lze namítnout, že tento typ útoku lze eliminovat zabezpečeným kanálem prostřednictvím SSL/TLS. To ovšem platí pouze za podmínky, že uživatelé jsou náležitě obezřetní a automaticky do svého prohlížeče neimportují jakýkoliv certifikát (tedy i útočníkův). Poněkud nepříjemné je, že v tomto případě není možné útok eliminovat ani s využitím DNSSEC, protože z pohledu překladu doménového jména je výsledná IPv6 adresa stále stejná.

Když je oznámení směrovačů o něco více

Podvrhávání komunikace není jediná záškodnická činnost, kterou lze s využitím zpráv *Oznámení směrovače* realizovat. Jak jsme si ukázali v předchozím případě, jedno IPv6 rozhraní může mít nakonfigurováno hned několik síťových prefixů a adres. To lze využít k dalšímu zajímavému útoku, který je znám pod názvem *RA Flood*. Podstatou tohoto útoku je periodické generování paketů *Oznámení směrovače* s novými, náhodnými prefixy. Operační systém pak musí každý takový paket zpracovat následujícím způsobem:

- Nakonfigurovat další IPv6 adresu na rozhraní, které paket přijalo.
- *Link-local* adresu směrovače (také náhodně generovanou) vložit do směrovací tabulky jako výchozí bránu.

Pokud pakety s *Oznámením směrovače* dokážeme generovat dostatečně rychle, způsobíme tím většině operačních systému docela velké potíže. K simulaci útoku lze opět použít utilitu z balíku THC-IPV6:

```
# ./flood_advertise6 -X eth0
Starting to flood network with neighbor advertisements on eth0
(Press Control-C to end, a dot is printed for every 100 packet):
....
```

Jak velké to jsou potíže si můžete prohlédnou na následujícím videu. Již několik vteřin po spuštění příkazu je CPU PC vytíženo na 100%. V dalším sledu je PC prakticky neovladatelné až do té míry, že přestane reagovat na uživatelské vstupy. Pokud byste podléhali přesvědčení, že je to pouze záležitost produktů z dílny firmy Microsoft tak vás můžeme uklidnit, že ne. Útokem byly (jsou) zranitelné takřka všechny platformy téměř bez výjimky.

Uvedený typ útoku je znám už od roku 2010 a oficiálně byl poprvé publikován 15. dubna 2011 prostřednictvím CVE-2010 autorem námi používaného THC-IPV6 toolkitu Marcem Heusem. I přes zjevný problém někteří výrobci, jako například Juniper, věc uzavřeli s tím, že se jedná o záležitost stran specifikace IPv6 a je nutno věc nejdříve vyřešit na půdě IETF, jak detailněji rozebírá článek (Microsoft, Juniper urged to patch dangerous IPv6 DoS hole). V mezičase ostatní výrobci implementovali alespoň nějaké formy ochrany tak, aby útok nevedl k plnému vytížení CPU. V produktech Microsoft se alespoň základní ochrana objevila až začátkem roku 2013. Nicméně i přes omezení množství zpracovávaných zpráv (*rate limiting*) *Oznámení směrovače* zůstává stále *RA Flood* závažným problémem. Mezi nahodile zahozenými zprávami může být oznámení skutečného směrovače, čímž koncové zařízení přijde o korektní IPv6 konektivitu.

Tímto však zábava s tímto druhem útoku zdaleka nekončí. Omezení (*rate limiting*) zpráv *Oznámení směrovače* pomohlo vůči základnímu útoku *RA Flood*, který pouze náhodně generuje prefixy a *link-local* adresy směrovačů. Zprávu *Oznámení směrovače* lze nicméně rozšířit o další volby. Jednou z těchto voleb je *Route Information Option* která nese podrobnější informace o směrování. Klient, který přijme zprávu *Ohlášení směrovače* s touto volbou, si informaci vloží do směrovací tabulky. Tak lze pouze jedním paketem *Ohlášení směrovače* docílit vložení několika desítek (potenciálně stovek) cest do směrovací tabulky. Některé platformy, jako např. MAC OS nebo Microsoft Server 2012, skončí po zaslání těchto paketů restartem. Pro zájemce lze doporučit stránku Sama Bowneho, kde jsou uvedeny návody a demonstrace pro různé platformy.

Selektivní útoky

Ačkoliv *Oznámení směrovače* jsou primárně určena všem zařízením připojeným v příslušné podsíti, paket s *Oznámením směrovače* je možné selektivně zaslat na libovolnou unicast adresu. Tímto můžeme předat "zajímavé" autokonfigurační údaje pouze těm zařízením, u kterých k tomu máme důvod. Z hlediska provozu sítě se tato možnost používá ve specifických případech, kterým se budeme věnovat v jednom z dalších dílů seriálu. Útočníkovi se nicméně tato vlastnost velice hodí, protože problém jednotlivce nebude jistě budit tak velkou pozornost jako to, že se všem začaly podivně chovat stránky například Google.com. Pro vytvoření takového paketu již útočníkovi nebudou stačit nástroje z THC-IPv6, ale bude se muset uchýlit k něčemu trošku sofistikovanějšímu. Jednak může zprovoznit vlastní instanci démona radvd, kterou nastaví tak, aby určitým klientům předávala požadované parametry. Může také použít nástroj Dana Lüdtkeho ratools, který umožňuje vytvořit zprávu *Ohlášení směrovače* zcela podle útočníkových potřeb. V neposlední řadě lze také použít některý generátor paketů. Výborně se pro tento účel hodí knihovna Scapy napsaná pro jazyk Python, která prostřednictvím předdefinovaných tříd umožňuje sestavení vlastního paketu a následné odeslání do sítě. Balíčky jsou k dispozici pro většinu distribucí.

Následující kód nám vytvoří paket, který zvolenému zařízení podvrhne adresy serverů Google:

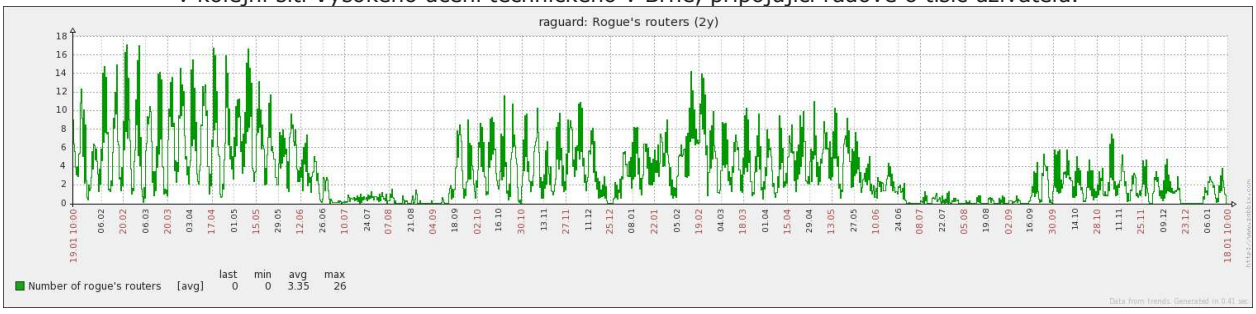
```
$ python
>>> from scapy.all import *
>>> b = ICMPv6ND_RA(chlim=64,routerlifetime=1800)
>>> c = ICMPv6NDOptPrefixInfo(prefix='2a00:2450:4014:80b::',prefixlen=64)
>>> a = IPv6(dst='fe80::6ab5:99ff:feea:d48a')/b/c
>>> send(a)
```

Oproti nástroji fake_router6 však podvržený prefix patřící Google není zaslán všem zařízením v sítí, ale pouze zařízením s adresou fe80::6ab5:99ff:feea:d48a.

Pachatel 6to4

Útoky vedené prostřednictvím falešných zpráv *Oznámení směrovače* nemusí nutně znamenat cílenou záškodnickou činnost. V síti celkem často můžeme narazit na zařízení, která běžný uživatel připojí do sítě a ta se začnou chovat jako IPv6 směrovač. Běžně se tak chovají některé MS Windows s nakonfigurovanou službou "Sdílení Internetu". Tato služba funguje pro IPv4 provoz jako NAT. Pokud zařízení nemá nativní IPv6 konektivitu, tak si, pokud může, nastaví tunelování IPv6 skrz IPv4 pomocí přechodového mechanismu 6to4, který mu IPv6 konektivitu umožní. Vytvořený prefix 6to4 je následně oznamován do sítě a ostatní zařízení se jej bez problému naučí. Důsledkem je, že takto nakonfigurované PC na sebe "stáhne" veškerý provoz IPv6 a prostřednictvím 6to4 se snaží pakety doručit na nejbližší 6to4 relay. Výsledkem tedy je, ze vaše data určená například na server ve vedlejší místnosti oběhnou půl světa.

Problém nechtěných a nevědomě (díky službě "sdílení Internetu") vznikajících směrovačů IPv6 je znám už řadu let. Na některých verzích Windows se tento problém pokouší řešit IPv6 readiness update, který implicitně sdílení vypíná. I přes tuto aktualizaci se s falešnými směrovači lze stále v síti setkat, jak je možné vidět v následujícím grafu, který zachycuje výskyt falešných směrovačů v kolejní síti Vysokého učení technického v Brně, připojující řádově 6 tisíc uživatelů.



Graf je vytvořen za poslední dva roky a lze vidět, že počet falešných směrovačů klesá, jak jsou postupně instalovány systémové aktualizace. I přesto se s danými zařízeními můžeme v síti stále běžně potkat. Detailněji se problémem rovněž zabývála prezentace "Falešné směrovače v sítích IPv6" na IPv6 semináři organizovaným sdružením CESNET.

Zamezení konektivity

Další možností útoků je pomocí zprávy *Ohlášení směrovače* odepřít všem, nebo pouze vybraným uživatelům, konektivitu do IPv6 světa. Na začátku dnešního článku jsme napsali, že pokud zařízení přijme zprávu *Ohlášení směrovače, link-local*adresu směrovače, který ji zaslal, si vloží do své směrovací tabulky jako adresu výchozí brány. Přesněji řečeno, RFC 4861 definuje datovou strukturu *Default Router List*, do které si zařízení zařazuje seznam směrovačů, kterým může zaslat data. Tato struktura může být implementovaná přímo jako směrovací tabulka daného zařízení nebo jako samostatná datová struktura, která je se směrovací tabulkou pouze propojena. U každé adresy směrovače v *Default Router List* si zařízení poznamená, po jakou dobu je směrovač dostupný. Tuto informaci šíří samy směrovače v políčku *Router Lifetime* ve zprávě *Ohlášení směrovače.* Útočník pak může zaslat podvržené *Ohlášení směrovače*, kde nastaví dostupnost směrovače na 0. Po přijetí takovéto zprávy jsou zařízení povinny vymazat daný směrovač ze svého seznamu a de facto tak přijdou o svou adresu výchozí brány. Celý útok pak může vypadat následovně:

```
$ python
>>> from scapy.all import *
>>> raddr = x
>>> a=IPv6(src=raddr, dst='ff02::1')/ICMPv6ND_RA(routerlifetime=0)
>>> send(a)
```

Útočník si nejprve zjistí adresu směrovače, který posílá legitimní *Ohlášení směrovače*. Poznamená si jeho *link-local* IPv6 adresu do proměnné raddr a vytvoří si vlastní zprávu *Ohlášení směrovače*. Daný ICMPv6 paket pak zašle všem zařízením v dané síti, která si po přijetí této zprávy adresu legitimního směrovače odstraní ze svého seznamu směrovačů a tedy i ze své směrovací tabulky. Jako cílovou adresu lze uvést pouze konkrétní adresu zařízení a provést tak selektivní útok.

Dalším způsobem, jak uživatelům odepřít IPv6 konektivitu, je zneužití zpráv *Ohlášení souseda* (Neighbor Advertisement) a *Výzva sousedovi* (Neighbor Solicitation). Tyto zprávy se používají pro zjištění mapování mezi IPv6 adresou a linkovou (MAC) adresou. Jedná se tedy o obdobu zpráv protokolu ARP - *ARP Request* a *ARP Reply*, jak jej známe ze světa IPv4. Podrobněji se celému mechanismu zjištění mapování mezi IPv6 a MAC adresou budeme věnovat v dalších dílech seriálu. Pro nynější útok nám bude stačit informace, že pokud směrovač odpovídá zprávou *Ohlášení souseda*, jaká MAC adresa odpovídá jeho IPv6 adrese, měl by do zprávy uvést, že je směrovač - nastavit příznak *Router flag*.

Pokud útočník *Ohlášení souseda* podvrhne a příznak vynuluje, zařízení si musí danou adresu směrovače opět smazat ze svého seznamu směrovačů (*Default Router List*). Přijde tedy opět o svou výchozí bránu. Princip útoku lze pak demonstrovat následujícím jednoduchým kódem.

```
$ python
>>> from scapy.all import *
>>> raddr = x
>>> b = ICMPv6ND_NA(R=0,tgt=raddr)
>>> a=IPv6(src=raddr, dst='fe80::6ab5:99ff:feea:d48a')/b
>>> send(a)
```

Útočník vytvoří falešné *Ohlášení souseda*, ve které vynuluje příznak definující, že zprávu zaslal směrovač. Zařízení s adresou fe80::6ab5:99ff:feea:d48a si po přijmutí této zprávy směrovač ze svého seznamu směrovačů odebere. Prakticky tak přijde o výchozí bránu a tedy i o IPv6 konektivitu do světa.

V čem je to jiné oproti IPv4

Čtenáře jistě napadne, že obdobné problémy musí zákonitě existovat i v protokolu IPv4. I zde jsme zvyklí, že zařízení připojíme do sítě a ono si "samo" zjistí všechny potřebné konfigurační údaje. V čem je tedy situace jiná?

Ve světě IPv4 se dnes jako prostředek autokonfigurace využívá v sítích Ethernet takřka bezvýhradně protokol DHCP (RFC 2131). Zařízení po připojení do sítě prostřednictvím požadavku na všesměrovou adresu (*broadcast*) vyzve server DHCP o přidělení konfiguračních údajů. Server DHCP odpoví a klient si na základě jeho odpovědi nastaví IP adresu na rozhraní, případně další údaje, například adresy rekurzivních jmenných serverů. Konfigurační údaje se vždy přidělují na předem stanovený čas, který se může pohybovat od několika sekund až po několik dnů. V okamžiku, kdy klientovi končí platnost přidělených údajů, znovu požádá o jejich prodloužení, ale tentokrát již nikoliv prostřednictvím dotazu na všesměrovou adresu, ale dotazem přímo na server DHCP, který konfigurační údaje poskytl. Klient si tedy musí sám aktivně vyslat žádost o přidělení, případně o prodloužení platnosti konfiguračních údajů. Protokol tedy funguje na principu dotaz - odpověď - potvrzení.

Pokud by útočník chce podvrhnout konfigurační údaje, musí nějakým způsobem zasáhnout do komunikace v okamžiku vyslání prvního požadavku klienta. Pokud útočník má štěstí a dokáže odpověď vytvořit rychleji než servery DHCP instalované správcem sítě, je klient prakticky plně pod útočníkovou kontrolou. Této zranitelnosti například zneužívají Trojan.Flush.M a Trojan:W32/DNSChanger.

V případě autokonfigurace IPv6 je logika věci obrácená. Klient neustále poslouchá na síťovém rozhraní a v okamžiku, kdy dorazí paket obsahující *Oznámení směrovače*, si ihned nakonfiguruje příslušné parametry. Změnu konfiguračních údajů lze tedy na rozdíl od IPv4 provést kdykoliv, jediným paketem a prakticky s okamžitou reakcí.

Znalci IPv6 jistě namítnou, že i v rámci protokolu IPv6 je možné použít přidělování konfiguračních údajů prostřednictvím DHCPv6. To je bezesporu pravda. Protokol DHCPv6 je nicméně řešen pouze jako rozšiřující nástavba nad bezstavovou autokonfigurací a pro své správné fungování potřebuje *Oznámení směrovače*. Zařízení připojené do sítě musí nejdříve z *Oznámení směrovače* zjistit, že má pro konfiguraci adresy použít protokol DHCPv6. Až následně je provedena procedura, kterou známe: dotaz - odpověď, případně potvrzení (podle typu použitého DHCPv6). Bez *Oznámení směrovače* nelze DHCPv6 použít, protože neexistuje způsob, jak klientovi sdělit výchozí bránu do Internetu. DHCPv6 tuto informaci sdělit neumí.

Bez *Oznámení směrovače* také nelze předat informaci o tom, v jaké je zařízení podsíti - DHCPv6 přiděluje pouze adresu, ne síťovou masku (délku prefixu). Zařízení ve stejné síti tedy nemohou komunikovat mezi sebou, protože neví o tom, že ve stejné síti jsou. V praxi se navíc rozchází jednotlivé implementace operačních systémů, kde některé berou informaci z *Ohlášení směrovače* jako autoritativní a některé pouze jako tip, jak by se v síti měly chovat. Určité platformy navíc protokol DHCPv6 nepodporují (např. Android). Tady se ale není čemu divit, jelikož dle RFC 6434 je koncové zařízení povinné implementovat pouze bezstavovou konfiguraci. Podpora protokolu DHCPv6 je pouze doporučována, nicméně není striktně vyžadována. Zkrátka a jednoduše - v IPv6 to bez bezstavové autokonfigurace a *Oznámení směrovačů* prostě nejde (pomineme-li statickou konfiguraci).

Pokud se podíváme na útok zneužívající zprávy *Výzva sousedovi* a *Ohlášení souseda* můžeme považovat za úplnou novinku u protokolu IPv6. U IPv4 jsme sice měli možnost provést útoky na protokol ARP, nicméně ani pomocí protokolu ARP nejsme schopni smazat adresu směrovače ze směrovací tabulky.

Historie se opakuje

Pokud se podíváme do minulosti, tak zjistíme, že velice podobný mechanismus bezstavové konfigurace výchozí brány, byl rovněž k dispozici v protokolu IPv4 pod názvem *ICMP Router Discovery* a specifikován v RFC 1256. I přestože jeho podpora byla implementována takřka ve všech operačních systémech včetně systémů Windows 95, patrně jste o něj v dnešní praxi nikde nezavadili. V roce 1999 bylo publikováno bezpečnostní oznámení na prakticky stejné typy problémů, které dnes známe z autokonfigurace IPv6. V té době byl uvedený problém považován za relativně závažnou hrozbu. Lze říci, že takřka všechny systémy od té doby ve výchozím stavu zprávy *ICMP Router Advertisment* podle RFC 1256 ignorují a konfigurační údaje se přebírají pouze z DHCP.

Pokud by někdo očekával, že se situace může opakovat i v případě IPv6, bude zklamán. Jak jsme zmínili dříve, bezstavová konfigurace a *Oznámení směrovače*jsou integrální součásti protokolu IPv6 bez kterých automatická konfigurace v IPv6 zkrátka nefunguje. Záležitost kolem bezstavové konfigurace je navíc kontroverzním tématem rozdělujícím IPv6 komunitu na dva nesmiřitelné tábory. První z nich preferuje, aby bylo možné IPv6 adresy přidělovat pouze přes DHCPv6, ideálně s využitím linkové adresy (MAC) a přiřazením výchozí brány pomocí DHCPv6. Tímto by mohla být bezstavová konfigurace pomocí *Ohlášení směrovače* zcela eliminována a vše by mohlo probíhat přesně tak jak známe z IPv4. Druhá skupina se ovšem domnívá, že to je nekoncepční přebírání vlastností z IPv4 porušující navíc nezávislost vrstev u TCP/IP. Pokud je totiž výchozí brána přidělována pomocí DHCP, tak aplikační protokol (DHCP) přiděluje informaci určenou pro směrování. Tuto informaci by ale měly sdělovat pouze zařízení pracující na síťové vrstvě (směrovače), které díky směrovacím protokolům jako jediné ví, jaký je reálný stav sítě. Bezstavová konfigurace pomocí *Ohlášení směrovače* s případnou kombinací DHCPv6 tak podle druhé skupiny poskytuje větší flexibilitu a zachovává vrstvovou nezávislost.

Tento ideologický rozpor se vleče již několik let a četné diskuse objevující se přibližně s měsíční periodou nejenom v rámci IETF vedou k patovému výsledku. Konsensus klíčových lidí, tvořící jádro pracovních skupin IPv6 (6man, v6ops), je jasný - žádná výchozí brána v DHCPv6, dokud nebude zdokumentována situace, která je současnými prostředky IPv6 nerealizovatelná. Poněkud rozpačitým výsledkem celého sporu je pak současné řešení, kde musíme provozovat a vhodným způsobem zabezpečit dva protokoly pro dosažení podobné míry ochrany jako u IPv4 s DHCPv4. Vrstvová závislost je navíc stejně porušená, protože informace o rekurzivních jmenných serverech je možné vložit do *Ohlášení směrovače*.

Mě se ale IPv6 netýká

V tomto díle jsme si ukázali, jak nám autokonfigurace, která je integrální a nedílnou součásti protokolu IPv6, může způsobit značné nesnáze. Pokud někdo až do této chvíle žil v přesvědčení, že se ho uvedené problémy netýkají, protože IPv6 ho zatím vůbec nezajímá, tak je na velkém omylu. Dnes mají všechny hlavní operační systémy podporu IPv6 ve výchozím stavu aktivovanou a je tedy na nich možné realizovat téměř všechny útoky využívající protokol IPv6 bez ohledu na to, zda je protokol v síti implementován správcem či nikoliv. Prostá ignorace IPv6 tedy zřejmě není nejlepším způsobem. Stojí tedy za zvážení, zda není lepší v síti raději implementovat IPv6 a mít nad věci alespoň rámcovou kontrolu, než věcem nechat zcela volný průběh. Implementace by však měla odpovídat bezpečnostní politice, která je nastavená pro IPv4. V opačném případě hrozí, že bezpečnostní opatření pro IPv4 jsou takřka bezpředmětná, pakliže je lze snadno obejít s využitím IPv6. V příštím díle si ukážeme, jak je možné se proti uvedeným útokům bránit nebo alespoň minimalizovat případné škody v případě, kdy pro plnou obranu nemáme vhodné technické prostředky.

**Poznámka:** *Uvedené příklady, nástroje nebo ukázky zdrojových kódů jsou upraveny, aby je nebylo možné použít pouhým zkopírováním do konzole a spuštěním. Slouží pouze jako prostředek k hlubšímu pochopení diskutované tématiky. Upozorňujeme, že jejich zneužití může být v rozporu přinejmenším s dobrým vychováním.*

**Bezpečné IPv6: zkrocení zlých směrovačů**

Dříve než se podíváme, jak můžeme IPv6 síť zabezpečit proti útokům z minulého dílu seriálu, seznámíme se s možnostmi, které máme pro řešení obdobných typů útoků v protokolu IPv4. Jak už jsme zmínili v předchozím článku, falešný DHCP server může v IPv4 síti napáchat pořádnou neplechu. Obecně se tento typ útoku (podvržení zpráv protokolu DHCP) označuje jako *DHCP Spoofing*. Na rozdíl od falešného IPv6 směrovače sice není možné převzít kontrolu nad celou podsítí během několika milisekund, nicméně většina správců si je rizik a nepříjemností s falešným DHCPv4 serverem vědoma.

Díky možnosti podvržení zpráv u protokolu DHCP vznikly ve světě IPv4 postupem času mechanismy na ochranu proti typickým útokům. Jedním z nejběžnějších prostředků obrany je technologie známá pod označením *DHCP Snooping*. *DHCP snooping* se konfiguruje na portu přepínače, do kterého je připojen koncový uživatel (přístupová vrstva - *access*). Uživatelské porty se označí jako nedůvěryhodné a na těchto portech jsou zahazovány všechny pakety, které by ostatní mohli interpretovat jako odpověď DHCP serveru, tj. pakety s cílovým UDP portem 68 a zprávy *DHCP Offer*, *DHCP Acknowledge* aj. Tímto vcelku efektivně zajistíme, že za daným portem nemůže běžet DHCP server, který by nebyl pod kontrolou správce sítě. *DHCP Snooping* nejen že rozhoduje, které zprávy jsou validní a které nikoli, ale u těch validních si do interní tabulky také uloží informaci o tom, která IP adresa byla přidělena zařízení připojeném k příslušnému portu a jakou MAC adresu toto zařízení používá. Vzniká tak vazební tabulka (*binding table*) IP - MAC - port, tedy jaké zařízení (MAC adresa) používající danou IP adresu je připojeno na příslušném portu.

Na přepínači může databáze vypadat například takto:

```
MacAddress     IP              VLAN Interface Time Left
-------------- --------------- ---- --------- ---------
001b38-afbacb 147.229.212.122 120  D16       6787
001f16-ada1d7 147.229.212.61  120  A21       5438
002522-8b69ee 147.229.212.153 120  A19       6972
```

Praxe však ukázala, ze pouhá ochrana před falešnými DHCP servery je v některých případech nedostatečná. Z toho důvodu umožňují některé přepínače aktivovat další stupeň ochrany, který je znám pod pojmem *Dynamic ARP Inspection* (případně u některých výrobců *Dynamic ARP Protection*). Pro své správné fungování využívá databázi vytvořenou mechanismem *DHCP Snooping*. Princip *Dynamic ARP Inspection* je pak vcelku jednoduchý. Na portech, kde je ochrana aktivována, jsou kontrolovány všechny pakety protokolu ARP, u kterých je prováděná analýza, zda se údaje v ARP odpovědích (IP a MAC adresy) shodují s příslušným záznamem v*DHCP Snooping* databázi. Tento typ kontroly znemožní připojenému zařízení provést útok typu *ARP poisoning*. Další často ceněnou vlastností tohoto mechanismu je fakt, že zařízení nedokáže rozumně používat síť, pokud předem nepožádalo o konfigurační údaje DHCP server. To prakticky znemožňuje uživatelům používat vlastnoručně nakonfigurované statické IP adresy, což je noční můra mnoha správců.

Je dobré si uvědomit, že popsaný mechanismus *Dynamic ARP Inspection* zkoumá pouze pakety protokolu ARP a mechanismus *DHCP Snooping* pouze pakety protokolu DHCP. Nic tedy nezabraňuje útočníkovi posílat data s podvrženými IP a MAC adresami, což je oblíbená kratochvíle zejména různých DDoS nástrojů. K zamezení tohoto útoku existuje třetí zbývající technika: *IP Source Guard* (*Dynamic IP Lockdown*). Využíva opět databázi vytvořenou mechanismem *DHCP Snooping*, nicméně kontrolu rozšiřuje i na hlavičky protokolů Ethernet a IP. Na příslušném portu tedy přepínač propustí pouze pakety obsahující správnou kombinaci MAC a IP adresy, čímž efektivně brání možnému podvrhávaní zdrojové IP nebo MAC adresy. Použití uvedených mechanismů tedy zajistí splnění podmínky na validaci zdrojové adresy dle doporučení BCP38 a to na nejdůkladnější možné úrovni.
Pokud se v tom všem ztrácíte, tak následující tabulka shrnuje typy útoků realizovatelné v rámci lokální IPv4 sítě a dostupné prostředky pro jejich eliminaci.

| | **DHCP snooping** | **ARP inspection** | **IP source guard** |
|---|---|---|---|
| Falešný DHCP server | ✓ | | |
| Otrava ARP tabulky | | ✓ | |
| Vynucení DHCP | | ✓ | ✓ |
| Podvržení adresy IPv4 | | | ✓ |
| Podvržení adresy MAC | | | ✓ |

Čistě teoreticky tedy můžeme být nadšení, že máme pokryty všechny výše zmíněné bezpečnostní problémy, a prostě jen stačí v celé síti tyto ochranné prvky aktivovat a je vystaráno. Aktuálně navíc nejsou známy další závažné útoky, které by bylo možné realizovat v rámci lokální sítě IPv4 a dané obranné mechanismy jim nezabránily.
V praxi však bohužel není vše tak jednoduché. Uvedené ochranné prvky vnášejí do chodu sítě složitost, která pak vyžaduje více kvalifikovaného správce, komplikují analýzu případných problémů a v neposlední řadě mohou, vlivem chyb v implementaci, vnést do sítě problémy nové. Neméně významným faktorem je rovněž skutečnost, že za přepínač podporující tyto vlastnosti si budeme muset o něco připlatit. Nutnost nasazení tedy vždy záleží na posouzení konkrétní sítě - zatímco ve vaší domácí síti nebo v malé kanceláří jsou zřejmě bezpředmětné, v síti sídlištního operátora s tisíci uživateli jsou prakticky nezbytné. Pro úplnost ještě uveďme, že uvedená zabezpečení se označují souhrnným pojmem *First-Hop-Security*.
V souvislosti s bezpečnostními prvky možná někoho napadlo, zda by nebylo snazší v síti jednoduše implementovat autentizovaný přístup prostřednictvím IEEE 802.1X. Bohužel autentizace klinetů prostřednictvím 802.1X nám v tomto případě příliš nepomůže. Protokol 802.1X na základě certifikátu nebo jména a hesla pouze zpřístupní port pro daného uživatele. Přidělení L3 informací - IP adresa aj. jde ale typicky již mimo něj. Nic tedy nezabrání uživateli si nastavit adresu vlastní, podvrhnout cizí nebo provést přeplnění či otrávení tabulky MAC adres na přepínači. Proto i v tomto případě bude nutné výše zmíněné bezpečností mechanismy použít, abychom těmto útokům zabránili.

Jak je to v IPv6

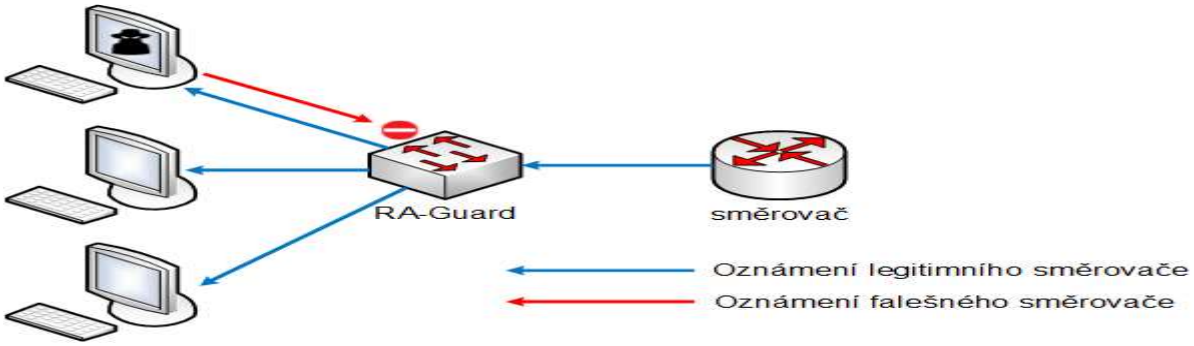
Jistě vás napadá myšlenka, že by asi nebyl problém převzít jednou ověřené postupy ze světa IPv4 rovnou do IPv6. Byť jsou signalizační protokoly z hlediska formátu zpráv v IPv6 trochu jiné, v principu fungují stejně. To částečně platí, nicméně je zde několik drobností, které nám situaci budou poněkud komplikovat:

- IPv6 používá k autokonfiguraci buď bezstavovou konfiguraci (SLAAC) s jednoduchou výměnnou zpráv: výzva směrovači, odpověď, nebo je SLAAC doplněn o protokol DHCPv6. Tím se nám množina protokolů pro zabezpečení rozšiřuje na dva.
- Podpora protokolu DHCPv6 je na koncovém systému pouze volitelná a řada zařízení ho nepodporuje (např. Android). Vazební tabulku IP - MAC - Port vytvořenou na základě protokolu DHCPv6 tedy nemůžeme využít ve všech případech.
- V případě bezstavové konfigurace je tvorba výsledné IPv6 adresy plně v moci koncového systému a při použití *Privacy Extensions* (RFC 4941 a RFC 7217) je adresa, kterou zařízení použije pro komunikaci vygenerovaná náhodile a tedy předem nepredikovatelná.
- Uvedené mechanismy musí podporovat hardware přepínače (např. v ASIC chipu), který zajišťuje připojení takto chráněného koncového systému. Pokud by je totiž zpracovával přepínač pouze procesorem, lze ho vygenerováním tisícem zpráv zahltit.

Další nemalý problém je, že díky pomalému zavádění protokolu IPv6 je obrovská asymetrie mezi útočníky a mechanismy pro ochranu sítě. Nástroje pro útočníky jsou již dávno k dispozici, nástroje pro ochranu sítě IPv6 ale stále chybí. Příkladem může být nástroj THC-IPV6, který jsme využili v minulém díle a který implementoval většinu útoků pro IPv6 v roce 2005, nicméně obranné mechanismy byly standardizované až v roce 2011 (například RA-Guard, RFC 6105). Navíc, jak si ukážeme v následujících dílech seriálu, ani jejich použití neznamená, že je problém vyřešen.
V praxi pak můžeme stát před obtížným rozhodnutím: Jaké řešení použít tak, aby konfigurace sítě současně vyhověla požadavkům na bezpečnost, spolehlivost a odpovídala definovaným RFC? Pojďme si tedy probrat reálné možnosti, které máme v dnešní době k dispozici:

RA-Guard

Směrem, který je aktuálně považován za správné řešení je použití nástroje jménem *RA-Guard*. Před několika lety tuto problematiku rozebíral Pavel Satrapa ve svémčlánku na Lupě. V mezičase práce na standardizaci nástroje *RA-Guard* pokročila a na jaře 2011 byl vydán popis nástroje ve formě informačního RFC 6105
Základní myšlenka, kterou *RA-Guard* používá je v celku jednoduchá a ilustruje ji následující obrázek.




Na portu přepínače, kde je koncové zařízení připojeno, budeme na vstupu zahazovat všechny pakety obsahující zprávy *Oznámení směrovače*. Tj., všechny ty, které lze identifikovat jako zprávu ICMPv6 typ 134. Případný útočník, který bude chtít zneužít zpráv *Oznámení směrovače* je sice může vytvořit a odeslat do sítě, tyto zprávy ale neprojdou dál než na port, kterým je útočník připojen. Ostatní uživatele tedy nebudou takovými aktivitami nijak ohrožení. Jedná se tedy téměř o stejný mechanismus jako *DHCP Snooping*, který jsme si popsali u IPv4.

Na síťových prvcích se *RA-Guard* aktivuje jednoduchou kombinací příkazů. Příklad nastavení pro zařízení Cisco:

```
Switch(config)# ipv6 nd raguard policy POLICY-NAME
Switch(config-ra-guard)# device-role {host | router}
```

Nejprve vytvoříme politiku, kterou budeme později aplikovat na síťové rozhraní. Role zařízení pro danou politiku může být vícero, nás ale zajímají pouze následující dvě.

- Host: všechny zprávy *Ohlášení směrovače* a *Přesměrování* jsou zahazovány, tato volba je implicitní
- Router: Zprávy *Ohlášení směrovače* a *Přesměrování* jsou propouštěny.

V dané politice lze nastavit ještě celou řadu dalších kontrol zprávy *Ohlášení směrovače* - zda je správná priorita směrovače, správný prefix, byla odeslána ze správné adresy aj. Tyto další kontroly se používají pro politiku určenou pro porty směrovače, podrobnosti lze nalézt v dokumentaci. Danou politiku pak aplikujeme na rozhraní pomocí následujících příkazů:

```
Switch(config)# interface INTERFACE
Switch(config-if)# ipv6 nd raguard attach-policy POLICY-NAME
```

Politiku pro koncová zařízení aplikujeme na uživatelské porty, politiku pro směrovač na port směrem k nadřazenému přepínači nebo směrovači (uplink). Konfigurace u zařízeních HP se pak liší v závislosti na použité platformě. U řady přepínačů používající firmware ProVision lze pro jednotlivé porty nastavit pouze jednoduchou filtraci:

```
HP Switch(config)# ipv6 ra-guard ports <port-list>
```

Tímto příkazem dojde k zahození zpráv *Ohlášení směrovače* a *Přesměrování*. Žádné složitější validace nastavit nelze. Pro port vedoucí ke směrovači samozřejmě filtraci nenastavujeme.

U zařízeních HP používající firmware Comware je *RA-Guard* součástí obecnějšího mechanismu *ND-Snooping*, který popíšeme podrobně později. *RA-Guard* na těchto prvcích povolíme následovně:

```
[Switch] vlan 1
[Switch-vlan1] ipv6 nd detection enable

[Switch] interface GigabitEthernet 1/0/1
[Switch-GigabitEthernet1/0/1] ipv6 nd detection trust
```

Pro zvolenou VLAN povolíme *RA-Guard*. Nesmíme také zapomenout nastavit port, který vede ke směrovači jako důvěryhodný, abychom nezahazovali validní zprávy.

ND-Snooping

*RA-Guard* řeší pouze část problému, protože dokáže filtrovat pouze zprávy *Ohlášení směrovače*. Již jsme zmínili, že IPv6 adresy mohou být přiděleny bezstavově nebo pomocí protokolu DHCPv6. V dnešní době se k adresaci lokální sítě používá primárně bezstavová konfigurace. Hlavním důvodem je kompatibilita, jelikož ne všechna zařízení podporují protokol DHCPv6. Dalším důvodem jsou chybějící vlastnosti u DHCPv6 serverů - např. zálohování jednoho serveru druhým (*High Availability*). Výjimku tvoří sítě ISP, kteří přidělují pomocí protokolu DHCPv6 prefix uživatelskému zařízení (CPE). Adresace CPE nicméně často probíhá bezstavově.

Při bezstavové konfiguraci si koncové zařízení vygeneruje samo náhodnou IPv6 adresu a správce sítě nad tím nemá kontrolu. Jakým způsobem ale zabránit podvržení adresy? Jakým způsobem zamezit útočníkovi, aby nepodvrhl záznamy v tabulce CAM přepínače?

*ND-Snooping* je technika, kterou je možné pro tyto účely použít. *ND-Snooping* se snaží na přístupovém přepínači vytvořit mapování mezi MAC a IPv6 adresou. Protože nemůže použít zprávy protokolu DHCPv6, snaží se mapování vytvořit odposlechem zpráv *Výzva sousedovi* a *Ohlášení souseda* (*Neighbor Solicitation*, *Neighbor Advertisement* - ekvivalenty zpráv *ARP Request* a *ARP Reply* pro IPv6). Před nakonfigurováním IPv6 adresy totiž zařízení ověřuje její unikátnost - zjednodušeně řečeno se ptá, zdali vygenerovanou adresu v síti již někdo nepoužívá. Pokud se mu do určité doby nikdo neozve, tak zařízení ví, že může adresu použít. Pokud tedy přepínač odchytne tuto počáteční výměnu zpráv, může si vytvořit mapování mezi vygenerovanou IPv6 adresou a MAC adresou.

Na zařízeních Cisco se mechanismus konfiguruje podobně jako *RA-Guard*: (možnosti příkazů se liší na jednotlivých platformách, tyto příkazy jsou pro 3750-X a 2960-S):

```
Switch(config)# ipv6 nd inspection policy POLICY-NAME
Switch(config-nd-inspection)# device-role {host | monitor | router}
Switch(config)# interface INTERFACE
Switch(config-if)# ipv6 nd inspection attach-policy POLICY-NAME
```

Opět vytvoříme politiku, kde definujeme typ zařízení (host je implicitní volba) a ta se pak aplikuje na jednotlivá rozhraní. Na základě odposlechu zpráv potom přepínač vytváří vazební tabulku, jak lze vidět na obrázku.

```
Switch# show ipv6 neighbors binding
<output omitted>
    IPv6 address                Link-Layer addr Interface vlan prlvl  age    state     Time
        left
ND  FE80::32E4:DBFF:FE17:EFA0 30E4.DB17.EFA0  Gi1/0/1       1  0011    8s   REACHABLE  295 s
ND  FE80::200:FF:FE00:BAD     0000.0000.0BAD  Gi1/0/4       1  0005    8s   REACHABLE  299 s
ND  FE80::200:FF:FE00:B0B     0000.0000.0B0B  Gi1/0/2       1  0005    4mn  REACHABLE  58 s
ND  FE80::200:FF:FE00:ABE     0000.0000.0ABE  Gi1/0/3       1  0005    4mn  REACHABLE  59 s
```

V případě, že by chtěl útočník připojený na portu Gi1/0/4 provést otrávení tabulky CAM například pro link local adresu FE80::200:FF:FE00:BOB, tak bude podvržená zpráva zablokována, jelikož neodpovídá adrese ve vazební tabulce.
Pro zařízení HP využívající Comware, lze ND snooping nakonfigurovat také, pomocí následujících příkazů:

```
[Switch] vlan 1
[Switch-vlan1] ipv6 nd detection enable
[Switch-vlan1] ipv6 nd snooping enable
[Switch] interface GigabitEthernet 1/0/1
[Switch-GigabitEthernet1/0/1] ip check source ipv6 ip-address mac-address
```

Zde si je ale třeba dát pozor, jelikož u zařízení HP nevytváří *ND Snooping* vazební informace z jakéhokoliv paketu *Výzva sousedovi* a *Ohlášení souseda*, nicméně pouze když se zařízení poprvé připojuje do sítě a testuje unikátnost vytvořené adresy (zdrojová IPv6 adresa je ::). Pokud tedy povolíte tento mechanismus v síti kde jsou aktuálně připojená koncová zařízení, vypnete jim IPv6 konektivitu, dokud se znovu nepřipojí do sítě nebo si nevygenerují novou adresu. Je si také třeba pohlídat správnou verzi firmware. Některé verze (release 1211 a starší) nepodporovaly vytvoření IP - MAC - port databáze z *link-local* adres, což je problém, protože klient pak není schopen zjistit MAC adresu výchozí brány, která typicky používá pouze *link-local* adresu. Release 18xx tyto chyby opravil a je možné funkčnost aktivovat pomocí příkazů:

```
[Switch] ipv6 nd snooping enable link-local
[Switch] ipv6 nd snooping enable global
```

Na první pohled by se mohlo zdát, že s použitím *ND-Snooping* lze dosáhnout podobné míry zabezpečení jak je tomu u IPv4. Bohužel realita není tak jednoduchá. Problémem tohoto mechanismu je fakt, že přepínač netuší, jestli za připojeným portem je útočník nebo regulérní uživatel. U IPv4 to bylo snadné, tam jsme si vynutili, aby všechna zařízení získala adresu od centrální autority (DHCP serveru) a následně přepínač nakonfigurovali, aby pracoval pouze s informacemi přidělenými touto autoritou. Při bezstavové konfiguraci v IPv6 to ale funguje na principu - Kdo se první přihlásí na daném portu, ten je více důvěryhodný. Nasazení tohoto obranného mechanismu paradoxně otevírá vrátka pro útočníky, kteří pak mohou ostatním uživatelům zablokovat přístup do sítě tím, že si počkají až se jejich zařízení odpojí a pak si jeho adresu "zaregistrují" na svůj port. Pokud se zařízení s původní adresou opět připojí do sítě, nebude mu to umožněno, jelikož jeho adresu má již útočník v databázi IP - MAC zaregistrovanou pro jiný port. Díky tomu, že pakety budou zahozeny, neproběhne ani kontrola duplicity adresy. Zařízení si tedy ani nemůže vygenerovat novou adresu, jelikož neví, že danou adresu již v síti někdo používá. V principu se tomu nedá zabránit, protože se využívají zcela legitimních vlastností protokolu IPv6: a) rozhraní může mít více IPv6 adres; b) IPv6 adresu lze generovat nahodile; c) přepínač nemá při bezstavové konfiguraci žádnou autoritativní odpověď, jako měl u IPv4 při použití DHCP. Přepínač tudíž nemůže rozlišit, kdo má pravdu a kdo ne. Bohužel pro tento problém není aktuálně známo žádné řešení a nasazení mechanismu *ND-Snooping* tak může útočníkovi některé útoky zkomplikovat, ale nemůže mu zcela zabránit.

DHCPv6

V sítích, kde se pro autokonfiguraci využívá primárně protokol DHCPv6 je nutné, kromě zabezpečení *Ohlášení směrovače* pomocí *RA-Guard*, dořešit ještě ochranu protokolu DHCPv6. K tomuto je možné použít *DHCPv6 Snooping*, který funguje de facto stejně jako u protokolu DHCP. Snaží se tedy získat informaci o IPv6 adrese ze zpráv, které si vyměňují DHCPv6 klient a server. Pokud se mu to podaří, vytváří si vazební tabulku, kterou jsme viděli již dříve. Tato funkcionalita je podporována na zařízeních HP používající firmware ComWare, kde ji můžeme aktivovat pro danou VLAN následujícími příkazy.

```
[Switch] ipv6 dhcp snooping enable
[Switch] vlan 1
[Switch-vlan1] ipv6 dhcp snooping vlan enable
[Switch] interface GigabitEthernet 1/0/1
[Switch-GigabitEthernet1/0/1] ip check source ipv6 ip-address mac-address
```

Nejprve povolíme *DHCPv6 snooping* globálně, potom ho aktivujeme pro vybranou VLAN. Přepínač začne odposlouchávat DHCPv6 komunikaci a z výměny zpráv si vytvoří vazební tabulku. Nesmíme také zapomenout nastavit port vedoucí k serveru DHCPv6 jako důvěryhodný, aby nedošlo k zahození odpovědí od serveru.

```
[Switch] interface GigabitEthernet 1/0/2
[Switch-GigabitEthernet1/0/2] ipv6 dhcp snooping trust
```

Opět připomínáme, že je třeba použít novější firmware a také aktivovat mechanismus *ND Snooping*. Pokud totiž zapnete pouze *DHCPv6 snooping* a povolíte kontrolu adres IPv6 a MAC, tak klienty odstřihnete od IPv6 konektivity, jelikož nebudou moci používat *link-local* adresy - pro *link-local* adresy totiž samotný *DHCPv6 snooping* nebude moct vytvořit ve vazební tabulce žádný záznam.

Cisco zařízení jsou na tom s podporou DHCPv6 snooping vcelku bídně. Aby se vytvářela vazební tabulka, musí se aktivovat podpora pro *DHCPv6 Guard*, který je zatím podporován pouze u řad 4500, 4900, 7600. Tato zařízení jsou pro přístupovou vrstvu vcelku drahá a většinou se využívají jako přepínače vrstvy distribuční. Při nastavení se pak musí DHCPv6 Guard kombinovat s mechanismem *ND Snooping*, který jsme si popsali výše. Příklad nastavení lze nalézt v oficiální dokumentaci nebo na Insinuator.net.

PACL

Jednou z dalších možností, jak vyřešit zabezpečení *Oznámení směrovače* a protokolu DHCPv6, je nastavení filtru (ACL) na vstupním portu, kde je zařízení připojeno. ACL pak může vypadat třeba následovně:

```
Switch(config)# ipv6 access-list DENY-RA-DHCP
```

```
Switch(config-ipv6-acl)# deny icmp any any router-advertisement
Switch(config-ipv6-acl)# deny udp any eq 547 any
Switch(config-ipv6-acl)# permit ipv6 any any
```

Následně pak aplikujeme na rozhraní:

```
Switch(config)# interface INTERFACE
Switch(config-if)# ipv6 traffic-filter DENY-RA-DHCP in
```

Je třeba si ale pohlídat, jestli přepínač umí rozpoznat typ zprávy ICMPv6. U zamezení DHCPv6 většinou problém není, jelikož se jedná o standardní komunikaci protokolu UDP na portu 547.

Výhodou tohoto řešení je, že filtraci typicky zpracová hardware daného přepínače (více o této problematice si řekneme v dalším díle). Nevýhodou je, že pomocí tohoto řešení nelze zamezit útočníkovi podvrhávat adresy a páchat další útoky typu Man-in-the-Middle (MitM).

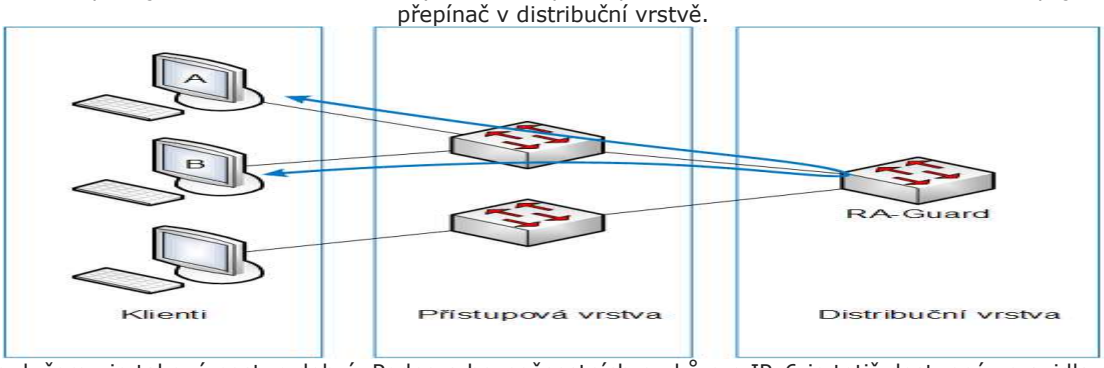
Ani tento přístup tak není zcela bez problému. Řada přístupových přepínačů navíc nepodporuje vůbec filtraci IPv6. Pokud už příslušná platforma filtry (ACL) pro IPv6 podporuje, je velice pravděpodobné, že bude rovněž podporovat *RA Guard* či základní podobu *ND-Snooping* a je tedy lepší použít přímo tyto mechanismy. V případě, že nemáme možnost nakonfigurovat obranné mechanismy na přístupových přepínačích, může přijít vhod některá z následujících možností.

Oddělení klientů

Vzhledem k tomu, že se problém týka pouze zařízení umístěných v jedné podsíti, tak se celkem logicky nabízí možnost síť dále rozdělit na více VLAN. V případě IPv6 máme adres pro vytváření podsítí relativně dost a z hlediska adresace by se tedy nemělo jednat o závážný problém. Možnosti útoku tímto sice neeliminujeme, ale výrazně tím omezujeme útočníkův manipulační prostor, resp. zmenšíme počet zařízení, které je schopen takto ovlivnit. Takový plán nám do značné míry znepříjemňuje současná koexistence sítí IPv4 a IPv6. Pokud nechceme mít dvě naprosto oddělené a nezávislé infrastruktury, musela by se rozdělit na více podsítí i síť IPv4. To by vedlo k přeadresaci, což je krok zpravidla velice bolestivý, a hlavně, rozdělení na více podsítí vede ke značnému plýtvaní s IPv4 adresami. Pokud bychom akceptovali myšlenku, že rozdělíme pouze síť IPv6, tak lze v ne-Cisco světě použít protocol-based VLAN - například tak, že na uživatelském portu si pouze IPv6 provoz přesměruji do unikátní VLAN pro daného uživatele. Zachováme tak stávající infrastrukturu pro IPv4 a klienty využívající IPv6 máme od sebe oddělené. Administrační náročnost celého procesu - hlavně pokud musíme pro dané podsítě nastavit zálohování výchozí brány např. pomocí VRRP - je extrémní. Stejně tak jsme limitování počtem VLAN, které přepínače zvládnou. Tomuto kroku se je tedy ve větší síti nejlépe vyhnout.

Použití privátní VLAN (RFC 5517) je někde na pomezí mezi rozdělením sítě na více VLAN a plnohodnotným řešením *First-hop-security* na přístupovém přepínači. V případě využití privátní VLAN degradujeme přístupový přepínač do role přeposílače paketů do nadřazené distribuční vrstvy. Koncová zařízení připojená ve stejném přepínači, byť jsou ve shodné VLAN, nedokážou potom komunikovat napřímo, ale data musí vždy putovat přes nějaký nadřazený prvek.

Tato funkčnost nicméně není podporována na čistě přístupových prvcích (např. řada Catalyst 2960), ale spíše na L3 (např. řada Catalyst 3560 a vyšší). Podobného účinku lze ale dosáhnout pomocí techniky označována jako *Private VLAN edge*, kdy uživatelským portům zamezíme komunikaci mezi sebou. Tato vlastnost navíc bývá podporována i na levných přepínačích. Funkčnost ilustruje následující obrázek. V klasické síti by komunikaci mezi klientem A a B odbavil přímo přepínač v přístupové vrstvě. Pokud ale použijeme některou ze zmíněných technik pro separaci klientů, komunikace mezi A a B půjde vždy přes přepínač v distribuční vrstvě.




Jistě si říkáte, k čemu je takový postup dobrý. Podpora bezpečnostních prvků pro IPv6 je totiž dostupná zpravidla až ve vyšších řadách přepínačů, které se spíše používají pro vrstvy distribuční nebo páteřní. Využitím *Private VLAN edge* tak můžeme koncová zařízení připojit prostřednictvím "hloupého" a tedy i levného přístupového přepínače, který všechna data předává nadřazenému "chytřejšímu" přepínači, který může zajistit náležitou filtraci procházejícího provozu.

Funkčnost lze zapnout u zařízeních Cisco pomocí následujících příkazů:

```
Cisco Switch(config)# interface INTERFACE
Cisco Switch(config-if)# switchport protected
```

Porty označené jako protected nemohou komunikovat mezi sebou napřímo. Jejich provoz je tudíž přeposlán na port, který takto označen není - typicky uplink.

U zařízeních HP využívající firmware ProVision lze obdobný filtr nastavit následovně:

```
HP Switch(config)# filter source-port 2-24 drop 2-24
```

Porty 2 - 24 nemohou komunikovat mezi sebou a jejich provoz bude přeposílán na port 1.

Je ovšem třeba ale upozornit na několik nevýhod tohoto řešení.

- Pokud zamezíme komunikaci mezi jednotlivými porty, tak zamezíme i komunikaci uživatelů mezi sebou. Řešením tohoto problému může být nastavením proxy-ARP na distribučním přepínači. Získáme tak separaci klientů na L2, kdy broadcast a multicast neprojde distribučním prvkem, nicméně unicast komunikace mezi klienty bude fungovat. U IPv4 sítě nám bude vše bez problémů fungovat, u IPv6 provozu je ale problém, jelikož např. zařízení Cisco nepodporují ND-proxy. Platforma HP s Comware tuto možnost podporuje.

- Data ze všech portů přístupového přepínače putují vždy na distribuční přepínač a zpět, a to i přesto, že za normalních okolností by byla odbavena v rámci jednoho přepínače. Toto řešení tedy můžeme použít v situaci, kdy je linka mezi distribučním a přístupovým přepínačem dostatečně dimenzovaná anebo charakter provozu je takový, že komunikace mezi zařízeními připojenými v jednom přepínači je minimální.

SEND

Dalším řešením, které bylo jistou dobu považováno za správné a koncepční, je použití kryptograficky podepsaných zpráv protokolu NDP - tedy *Oznámení směrovače*, *Ohlášení souseda* atd. Každý klient si potom může, na základě PKI, ověřit validitu těchto zpráv pomocí příslušného certifikátu. Samotná specifikace protokolu SEND pochází z roku 2005, přesto jsme se do dnešního dne nedočkali prakticky žádného rozšíření v existujících systémech. O protokolu se raději přestalo mluvit a dnes je sice občas zmiňován jako způsob zabezpečení lokální sítě, ale těžko bychom našli někoho, kdo by uvažoval o protokolu SEND jako o reálné možnosti pro zabezpečení současných lokálních IPv6 sítí. Je to dáno několika faktory. Jednak klade protokol SEND enormní administrativní nároky na provoz sítě - je nutná pravidelná obměna klíčů na směrovačích, distribuce certifikátů klientům, udržování PKI infrastruktury atd. Navíc se SEND nedá využít s adresami, které jsou nakonfigurovány manuálně, přiděleny pomocí DHCPv6 nebo vygenerovány s využitím privacy extensions. SEND vyžaduje vlastní formát kryptograficky generovaných adres (CGA), kterou si vytvoří dané zařízení. Pokud si ale zařízení vygeneruje vlastní CGA adresu, dostáváme se opět před klasický problém v síti IPv6, kdy nelze rozpoznat, jestli je daná IPv6 adresa útočníka nebo ne. Protokol tak dává smysl pouze pro ověření pravosti zpráv *Ohlášení směrovače*, nicméně aktuální trend je řešit tento problém spíše filtrací na úrovni síťové infrastruktury, jak jsme si popsali výše. V neposlední řadě je protokol patentován, což snižuje touhu výrobců protokol implementovat na svých zařízeních k nule. V současné době existuje implementace protokolu SEND pouze na zařízeních firmy Cisco (která patent vlastní). Edward Horley, autor knihy Practical IPv6 for Windows Administrator, nepředpokládá, že by Microsoft přidal podporu pro daný protokol a to i přes to, že za specifikací protokolu stojí lidi z firmy Microsoft. Existuje sice často odkazovaná verze WinSend, která od roku 2011 tvrdí, že přidala podporu daného protokolu pro Windows, nicméně, i přes proklamace autorů, do dnešního dne program ani kód nebyl zveřejněn. Pokud používáte systém Linux a chtěli byste si zabezpečit svou lokální síť, můžete zkusit volně dostupnou implementaci Ing. Lukáše Bezdíčka, kterou zpracoval ve své diplomové práci. V dané diplomové práci jsou také rozebrány současné implementace protokolu SEND, včetně těch, které se pouze tváří, že fungují.

Pasivní monitorování a aktivní protiútok

Pokud nemáme žádnou z uvedených možností, je více než vhodné zajistit alespoň pasivní monitorování. Jedna z možných technik je rozebrána v článku Ondřeje Caletky. Tento způsob monitorování je víceméně výhodné použít vždy, i pokud byste používali nějakou z technik popsaných výše - alespoň máte přehled, co se v síti děje. Dané nástroje pro monitorování falešných směrovačů se občas dají rozšířit o možnost "odregistrace" falešných prefixů. Síťový administrátor se tak vlastně stane útočníkem ve vlastní síti a použije techniku, kterou jsme si popsali v předchozím díle v části *Zamezení konektivity*. Tedy, pokud je detekováno falešné *Ohlášení směrovače*, zasílá se obratem zpět podvržené *Ohlášení směrovače*, ve kterém se nastaví dostupnost (*Router Lifetime*) falešného směrovače na 0. Tím se koncoví klienti donutí k odebrání falešného směrovače ze svých směrovacích tabulek.

Když dojdou všechny možnosti

V případě, že žádná z dříve popsaných možností pro vás nepřipadá v úvahu a nechcete nechat chod protokolu IPv6 ve vaší síti v rukou osudu, pak zbývají poslední dvě možnosti. První z nich je deaktivovat podporu IPv6 na koncových systémech. Tuto možnost ovšem máme pouze v případě, že dokážeme ovlivnit konfiguraci operačního systému na koncových zařízeních (například ve firemní síti). U systémů Windows to však není příliš doporučováno, protože IPv4 a IPv6 stack je u Windows zkombinován do jednoho. IPv6 tedy nelze kompletně vypnout. I přes toto doporučení se v řadě firemních prostředí setkáváme s politikou, kde maximální možné potlačování IPv6 je naprostou samozřejmostí a zatím nemáme informace, že by to mělo nějaký negativní vliv na funkčnost systémů. Výrazně složitější situace je například na mobilních platformách, kde IPv6 zpravidla není možné deaktivovat.

Druhou možností je blokace protokolu na portech přepínače tak, aby přepínač filtroval všechny pakety obsahující v hlavičce protokolu Ethernet identifikaci protokolu (EtherType) nastavenou na 0x86DD (IPv6). Tato filtrace musí být opět podporována přepínačem, což také nebývá vždy pravidlem. Pokud zařízení podporuje klasické IPv6 ACL, lze si vytvořit filtr zakazující IPv6 a aplikovat ho na jednotlivé porty přepínače.

Pro lepší orientaci v jednotlivých technikách zabezpečení lokální IPv6 sítě může pomoci následující tabulka.

| | **RA-Guard** | **ACL/ PACL** | **ND-Snooping** | **SEND** | **Deaktivace IPv6** | **Blokování IPv6** |
|---|---|---|---|---|---|---|
| Falešný DHCPv6 server | | ✓ | ✓ | | ✓ | ✓ |
| Falešné oznámení směrovče | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Otrávení ND cache | | | ✓ | ✓ | ✓ | ✓ |
| Podrvžení zdrojové IPv6 adresy | | | ✓ | | ✓ | ✓ |
| Podrvžení zdrojové MAC adresy | | | ✓ | | ✓ | ✓ |
| Vyžaduje podporu na síťových prvcích | ✓ | ✓ | ✓ | ✓[1] | | ✓ |
| Vyžaduje podporu v koncových systémech | | | | ✓ | ✓ | |

[1] Podpora protokolu SEND je nutná pouze na příslušném směrovači.

Závěr

Společně s rozšiřováním protokolu IPv6 bychom měli brát na zřetel i ochranu IPv6 sítě, která by měla odpovídat bezpečnostní politice aplikované pro IPv4. Pokud v síti nemáme oba protokoly stejně zabezpečené, jakékoliv zabezpečení je pak vůči cíleným útokům vcelku bezpředmětné, protože ho útočník dokáže obejít přes druhý protokol. Zabezpečení protokolu IPv6 ale není záležitost úplně jednoduchá a v praxi musíme velice často volit nějakou formou kompromisního řešení mezi cenou zařízení,

složitostí implementace a mírou zabezpečení. V případě existujících instalací se staršími prvky pak mnohdy skončíme u řešení s pasivním monitorováním a nadějí, že IPv6 útoky ještě po nějakou dobu nebudou dostatečně atraktivní pro páchání záškodnické činnosti. V případě nových instalací můžeme stát před relativně těžkou volbou. Buď ušetřit peníze a pořídit přístupové přepínače bez ochranných prvků IPv6 a zařízení s plnohodnotnou podporou koupit později, kdy jejich cena bude o něco příznivější. Druhá možnost je si připlatit a mít ochrané prvky pro IPv6 k dispozici už teď. Bohužel ani tato volba není v dnešní době plnohodnotnou ochranou proti některým typům cílených útoků. Na tyto problémy se ale podíváme v příštím díle.

**Bezpečné IPv6: trable s hlavičkami**

Koncept rozšířených hlaviček měl být jedním z klíčových trháků protokolu IPv6. Problematice jsme se již před několika léty věnovali v seriálu na Lupa.cz. Vzhledem k tomu, že téma rozšířených hlaviček je poměrně obsáhlé a komplikované, tak si dnes uděláme krátkou rekapitulaci jak rozšířené hlavičky fungují a na jaké problémy můžeme narazit při jejich zpracování. V dalším dílu si pak ukážeme, jak tento nový koncept může zneužít útočník a jaké jsou případně možnosti obrany.

Na rozdíl od protokolu IPv4 je základní hlavička každého IPv6 paketu tvořená minimalistickou strukturou, která obsahuje pouze naprosto nezbytné informace pro doručení paketu na cílové místo. Díky tomu je délka IPv6 hlavičky pouze dvakrát větší než základní hlavička IPv4, a to i přesto, že se zdrojové a cílové IPv6 adresy zvětšily čtyřikrát. Koncept základní IPv6 hlavičky poměrně chytře předpokládá, že takto tvořená hlavička je postačující pro naprostou většinu přenášených paketů. Aby však bylo možné protokol IPv6 rozšiřovat o další zajímavé vlastnosti, je možné základní hlavičku doplnit o další hlavičky, pro které IPv6 používá termín "rozšířené hlavičky" - *Extension Headers*. Technicky vzato lze rozšířené hlavičky přirovnat k datové struktuře v podobě jednocestného lineárního seznamu, kde první prvek tvoří základní IPv6 hlavička, rozšířené hlavičky jsou položkami seznamu a celý seznam je zakončen hlavičkou protokolu nesoucího data (TCP, UDP, ICMP, …), případně ukončující hlavičkou (*No Next Header*). Následující obrázek nám pak ilustruje několik variant uspořádání hlaviček.


U první varianty následuje za základní IPv6 hlavičkou přímo hlavička protokolu TCP. U další je mezi základní IPv6 hlavičku a protokol TCP vložena hlavička *Hop-by-Hop*, které by měly věnovat pozornost všechny uzly po cestě. U poslední varianty je vložena navíc ještě hlavička *Destination Options*, kterou by se mělo zabývat pouze koncové zařízení.

Vlastní rozšířené hlavičky můžeme rozdělit do dvou základních skupin. První z nich nesou informace určené směrovačům, které doručují pakety. Druhý typ hlaviček je určen pouze pro cílové uzly. Příkladem z první skupiny může být hlavička *Routing Header*, která se používá pro rozšíření IPv6 o mobilitu. Do druhé skupiny pak můžeme zařadit podpůrné hlavičky pro IPsec (ESP, AH), hlavičku nesoucí informaci o fragmentaci aj. Rozšířené hlavičky by měly být v paketu řazeny dle určitých pravidel. Základní pravidla jsou specifikována v RFC 2460, které je ale aktualizováno několika dalšími RFC, jak si popíšeme dále v článku. Obecně ale platí, že hlavičky zpracovávané uzly po cestě by měly být zařazeny na začátku seznamu a hlavičky týkající se koncového uzlu by měly být umístěné na konci. Na první pohled to tedy vypadá, že konceptu rozšířených hlaviček nelze nic vytknout. Pro většinu paketů se použije úsporná základní IPv6 hlavička následována hlavičkou vlastního protokolu transportní vrstvy. Současně je ale otevřená možnost budoucího rozšiřování protokolu IPv6 o nové vlastnosti. Rozšířené hlavičky ovšem přináší i některé komplikace a nové možnosti vedení útoků. Zkusme se tedy podívat, jaké záludnosti nás mohou v souvislosti s rozšířenými hlavičkami potkat.

Každý pes jiná ves

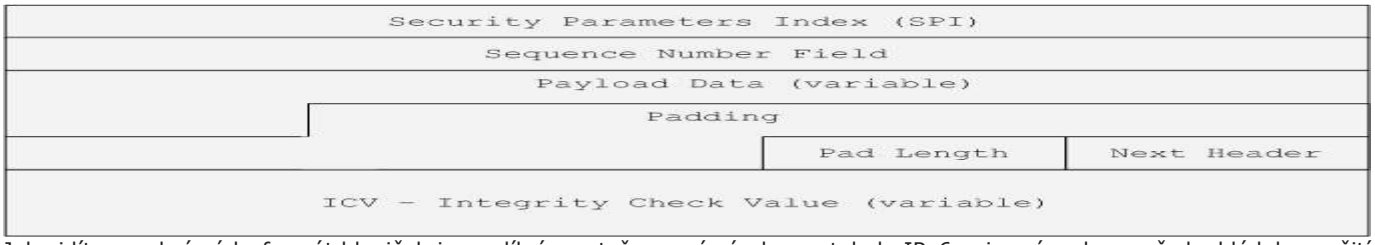
Na začátku si dovolíme menší odbočku, abychom problematiku rozšířených hlaviček lépe pochopili. Rozšířená hlavička je obecně datová struktura, která přenáší další informace, které se nevešly do základní hlavičky IPv6. U struktur obdobného rázu jakými jsou rozšířené hlavičky, bývá často u síťových protokolů zvykem používat formátování dat, které se označuje pojmem TLV (*Type - Length - Value*). Formát TLV má jasně dané rozložení, kde typicky v prvním bajtu je definován typ informace, ve druhém bajtu její délka a zbytek je již informace samotná. Výhoda této datové struktury je, že pokud zařízení danému typu nerozumí, může ho bez problémů přeskočit, protože ví jeho velikost. Položka o velikosti datové části (přenášené informaci) je totiž vždy na přesně definované pozici. Tímto je umožněno poměrně snadné rozšiřování protokolů a současně se zachovává kompatibilita se staršími verzemi a implementacemi. Protokoly, které tímto způsobem informace přenáší (např. EIGRP, ISIS, CDP) lze pak vcelku jednoduše rozšířit o nové informace a jejich stávající implementace s tím nemají problém. Nyní se vraťme zpět k rozšířeným hlavičkám a podívejme se detailněji na jejich vlastní formát.

V počáteční specifikaci protokolu IPv6 bylo definováno několik rozšířených hlaviček sloužící pro fragmentaci, šifrování aj. Vlastní formát rozšířených hlaviček ale nebyl nijak unifikován a byl zcela v rukou tvůrce příslušné rozšiřující hlavičky. Názorně si můžeme rozdílné formáty rozšířených hlaviček ukázat na příkladu hlaviček *Fragment Header*, *Hop-by-Hop Header* a IPsec ESP, které lze vidět na následujícím obrázku.

**Fragmentace**


**Hop-by-Hop**


**IPsec ESP**




Jak vidíte na obrázcích, formát hlaviček je rozdílný, protože první návrhy protokolu IPv6 ani v náznaku nepředpokládaly využití TLV. Některé hlavičky, jakými je například hlavička *Hop-by-Hop*, používají formát podobný TLV. Podobný proto, že typ (políčko *Next Header*) nenese informaci o vlastním typu, ale o typu hlavičky následující. Následuje pak délka vlastní *Hop-by-Hop* hlavičky a následně již samotná informace. Tento rozpor, kdy typ nepopisuje danou hlavičku, ale délka ano, nám pak způsobí další komplikace, jak si popíšeme dále v textu. U hlavičky fragmentace je tomu pro změnu úplně jinak. Hlavička má fixní velikost a tudíž neobsahuje políčko, které by její velikost definovalo. Hlavička IPsec ESP to má také kompletně jinak. IPsec ESP slouží k šifrování přenášených dat (políčko *Payload Data*), kde si můžete představit třeba protokol TCP. Identifikace, která data vlastně jsou takto šifrovaná, je až na konci celého paketu. Políčko *Next Header* u IPsec ESP tedy slouží jako zpětný ukazatel na začátek políčka *Payload Data*. Ostatní definované hlavičky se víceméně snaží dodržovat formát podobný TLV jako používá hlavička *Hop-by-Hop*, i když výpočet délky hlavičky (*Hdr Ext Len*) je také občas pro různé hlavičky rozdílný.

Z tohoto neustáleného formátu rozšířených hlaviček plyne poměrně nepříjemný závěr, kdy každé zařízení, které má umět zpracovat rozšířené hlavičky, musí vždy rozumět i syntaxi všech existujících rozšířených hlaviček, a to i přesto, že příslušná hlavička nenese pro dané zařízení žádné relevantní informace. Pracovní skupina IETF 6man si tento problém uvědomila a v roce 2012 vydala RFC 6564, kde zavádí jednotnou strukturu. Ta se podobá TLV formátu použitém pro *Hop-by-Hop* hlavičku. Tato nová struktura však celkem pochopitelně závazně platí pouze pro nově vytvářené hlavičky a dříve definované hlavičky již navždy zůstanou v původním formátu. Bohužel ani tato jednotná struktura neřeší celkový problém, jak si ukážeme dále.

Cesta ke sjednocení

Původní specifikace protokolu IPv6 z roku 1998 v zásadě nijak neřešila zpracování rozšířených hlaviček mezilehlými zařízeními (firewall, IDS, aj.). V té době zkrátka panovala představa, že síťová zařízení jako například směrovače budou pouze "bezmyšlenkovitě" doručovat data na základě cílové IP adresy a vyšší protokolové vrstvy je nemají co zajímat. Nepředpokládalo se, že by provoz po cestě měl být analyzován, filtrován či by do něj mělo být nějak zasahováno. Původní autoři IPv6 totiž veškerá mezilehlá zařízení (middlebox) považují za čisté zlo, které by se nemělo v síti vůbec vyskytovat. Z toho důvodu původní RFC 2640 doporučuje, aby mezilehlá zařízení všechny rozšířené hlavičky zkrátka ignorovala a nechala vše v režii koncového uzlu. Výjimku tvoří pouze hlavička *Hop-by-Hop*, která je určena pro všechna zařízení po cestě. Požadavky se však od roku 1998 značně změnily a prakticky ladění čtenáři nám jistě dají za pravdu, že v dnešní době, kdy DoS či DDoS útoky jsou takřka na denním pořádku, je možnost filtrace, například podle portů nebo příznaků v TCP, naprosto nezbytnou provozní nutností. Tento trend ostatně odrážejí i novější RFC, kde kupříkladu filtrace pro domácí sítě je aktuálními standardy přímo doporučována v RFC 4864. Vyvažování zátěže, Firewally, IDS/IPS systémy a QoS mohou být dalšími příklady služeb, kterým informace v základní IPv6 hlavičce nestačí a musí pracovat i s informacemi z transportního či dokonce aplikačního protokolu. Tento posun v nahlížení na mezilehlá zařízení reflektují i novější standardy a snaží se najít z tohoto problému únikovou cestu.

Jednou z počátečních komplikací pro správnou implementaci rozšířených hlaviček bylo vůbec zjistit, jaké rozšířené hlavičky jsou vlastně definované. Identifikátory a popis rozšířených hlaviček nebyly totiž od začátku přidělovány organizací IANA, takže nebyl nikde k dispozici celkový přehled. Jednotlivé specifikace rozšířených hlaviček tak byly "rozstrkány" v jednotlivých RFC různých pracovních skupin. Tento problém byl vyřešen v roce 2013 schválením RFC 7045. Dané RFC přesně definuje všechny rozšířené hlavičky, které musí umět zařízení zpracovat a současně předává přidělování identifikátorů organizaci IANA. Seznam všech současných a budoucích rozšířených hlaviček je tedy na jednom standardním místě. RFC 7045 rovněž diskutuje problematiku filtrování a připouští, že zpracovávání či zahazování rozšířených hlaviček může být problém. Přináší proto následující doporučení. Všechny současně definované rozšířené hlavičky musí zařízení podporovat a jejich zahození/přeposlání je pak záležitost nastavené síťové politiky. Všechny nově definované rozšířené hlavičky musí mít jednotnou strukturu dle RFC 6564. Pokud zařízení takovou nově definovanou hlavičku ještě nezná, RFC 7045 ji doporučuje "přeskočit" a pokračovat dál ve zpracování řetězce hlaviček. Jinými slovy postupovat tak, aby se nekomplikovalo zavádění nových rozšíření. Nicméně i přes tuto snahu je vyřešená pouze první část problému.

Protokol nebo rozšířená hlavička?

Druhou částí problému je totiž fakt, že identifikátor rozšířené hlavičky i čísla protokolů spolu sdílí stejné políčko (*Next Header*). Pokud zařízení identifikátor použitý v políčku *Next Header* nezná, nedokáže pak rozlišit, zda se jedná o nově definovanou rozšířenou hlavičku nebo o nově definovaný protokol. Důsledkem je, že nové rozšiřující hlavičky a protokoly nelze vlastně inkrementálně zavést. Celé si to můžeme ilustrovat na následujícím příkladu.

Představte si situaci, kdy budete chtít definovat nový protokol, například nové geniální TCP, říkejme mu třeba XTCP. Firewall nicméně nemůže tušit, že se jedná o zcela nový protokol. Pokud bude brát doporučení z RFC 7045, pokusí se data interpretovat jako neznámou rozšířenou hlavičkou a pokusí se ji "přeskočit". Na místě, kde se má nacházet identifikace navazující hlavičky a její délka jsou ale již data související s XTCP, která jsou tím pádem mylně interpretována jako informace popisující další rozšířenou hlavičku. V této fázi už lze těžko předvídat další chování systému. V ideálním případě se nezhroutí a rozhodnutí, jak naloží s paketem, provede podle své implicitní bezpečnostní politiky (propustit/zahodit), nicméně dané rozhodnutí je značně

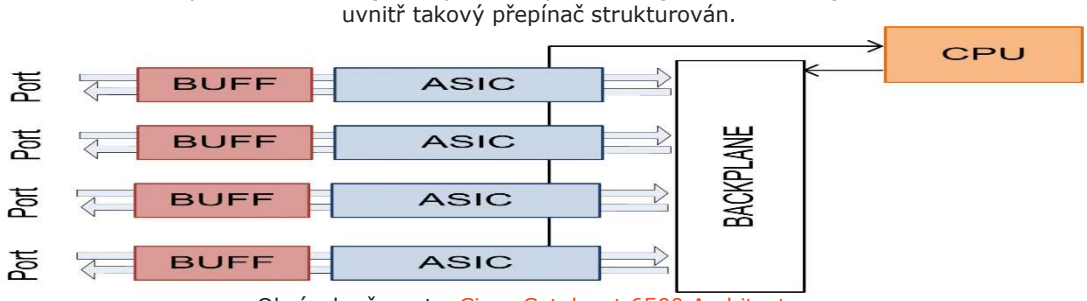
nedeterministické. V koncovém důsledku tedy nejsme schopni rozumně projít celým zřetězeným seznamem hlaviček, dostat se až na koncový transportní protokol a bezpečně ho detekovat.

Pracovní skupina IETF 6man si je i tohoto problému vědoma a nabízí částečné řešení v již zmíněném RFC 6564. Už jsme si popsali, že toto RFC definuje jednotný formát pro nové rozšířené hlavičky. Navíc také ale říká, že nová rozšířená hlavička nesmí být definovaná, pokud se pro nesení požadovaných informací dá použít již nějaká hlavička stávající. Vzhledem k tomu, že do hlavičky *Destination Options* lze zakódovat vcelku cokoliv, je přímo daným RFC doporučováno, aby se raději pro přenos případných dalších informací použila tato hlavička. Výsledkem tedy víceméně je snaha zachovat současné rozšířené hlavičky a žádné nové nedefinovat.

Efektivní zpracování hlaviček

Nepříjemným problémem rozšířených hlaviček je také jejich efektivní zpracování. Pokud hlavičky mají být zpracovány v software (např. v jádře operačního systému), je zpracování poměrně jednoduché - vše je řešitelné jednoduchým cyklem, který se zastaví v místě, kde je nalezena koncová hlavička (např. TCP). Cyklické procházení datových struktur je ovšem velká komplikace v případě, že paket má být zpracován ve specializovaném a ideálně paralelizovaném hardware, jakým je ASIC nebo FPGA čip. Zastánci striktně oddělených síťových vrstev (tj. zařízení na jedné vrstvě nezasahuje do vrstvy vyšší nebo nižší) jistě namítnou, že toto není žádný problém, protože zpracování rozšířených hlaviček je ryze záležitostí buď koncových systémů (kde se zpracovávají v SW) anebo specializovaných zařízení jako jsou firewall, IDS systém či monitorovací sonda, které mají dostatek zdrojů i pro složitější zpracování paketu. Praxe nicméně ukazuje, že tohoto striktního oddělení nejde vždy dosáhnout a někdy je třeba, aby zařízení zasahovala i do protokolových vrstev, které jim "nepřísluší". Typickým příkladem je efektivní přeposílání multicast provozu (IGMP/MLD snooping), kdy přepínač, pracující primárně na linkové vrstvě, potřebuje získat data z vrstev vyšších. Dalším příkladem mohou být mechanismy *First Hop Security*, které jsme si popsali v minulém díle. Pro efektivní fungování by měly být implementované na přístupových portech L2 přepínače, nicméně pak L2 přepínač musí zasahovat i do provozu vrstev vyšších a tedy být schopen zpracovat celý řetězec rozšířených hlaviček až k hlavičce vlastního protokolu.

V příštím díle si tento problém ilustrujeme na příkladu implementace mechanismu *RA-Guard*, který slouží jako ochranný prvek proti falešným směrovačům. Abychom si však objasnili podstatu problému, je vhodné se nejdříve krátce seznámit, jak je vlastně uvnitř takový přepínač strukturován.




Obrázek převzat z Cisco Catalayst 6500 Architecture

Ve velice zjednodušené podobě si můžeme přepínač představit, jak je zobrazeno na obrázku. Provoz na jednotlivých portech zpracovává specializovaný čip ASIC. Ten s paketem provádí na vysokých rychlostech řadu stěžejních úkolů: vyhledává, na jaký port jej poslat; aplikuje filtrovací pravidla; QoS apod. Samotný ASIC čip ovšem dokáže provádět pouze omezenou množinu základních operací. V případě, že ASIC nedokáže rozhodnout sám, nebo je třeba podrobnější analýza paketu, předává zpracování procesoru, který složitější problém vyřeší za něj. Procesor však primárně neslouží k vyhledávání cesty, přeposílání paketů nebo filtraci. Procesor obsluhuje hlavně protokoly nutné pro správný chod sítě (směrovací protokoly, Spanning Tree Protocol) a vlastní správu (management) zařízení. Pro klasické přepínání a směrování je pro dnešní rychlosti totiž relativně pomalý. Bez této dělby práce mezi CPU a hardware bychom nedokázali zpracovávat data dostatečně rychle se zachováním možnosti detailního zpracování části provozu v CPU. Pro představu dodejme, že při zpracování provozu pomocí čipu ASIC, můžeme počítat s propustnosti v řádech 10, 100, či více Gb/s, zatímco u zpracování dat v CPU stěží přesáhneme několik Gb/s, u běžných zařízení spíše stovky Mb/s.

Pro správné fungování bezpečnostních mechanismů, filtrací apod. je ovšem důležité, aby ASIC dokázal rozpoznat IPv6 paket a zpracovat ho do té míry, že je schopen identifikovat protokol přenášený v rámci IPv6 paketu (např. ICMPv6) a nemusel veškerý IPv6 provoz přeposílat do CPU. Nicméně právě tato detekce je problém. ASIC čip a paměti TCAM, které slouží pro zpracování paketu, mají většinou omezenou velikost. Záleží na architektuře přepínače, ale typicky jsou schopny přepínače zpracovat v hardware prvních 64 - 128 bajtů daného paketu. To reálně stačí pro většinu provozu, jelikož u IPv4 máme pouze hlavičku, která má 20 bajtů (bez *IP Options* o kterých si povíme ještě dále) a hned za ní následuje protokol vyšší vrstvy. U IPv6 tomu tak ale není. Regulérní IPv6 paket má jednak dvojnásobnou délku hlavičky a navíc může obsahovat vcelku libovolné množství rozšířených hlaviček. I s několika běžnými hlavičkami lze překročit velikost hlavičky, kterou jsou dnešní přepínače schopny zpracovat v hardware. Přepínač pak má ve výsledku tři možnosti, jak s daným paketem naložit:

- zahodit, čímž ale znemožňuje zavádění případných budoucích rozšíření,
- propustit, tím je ovšem možné obejít případnou bezpečnostní politiku,
- předat CPU, kde může dojít k přetížení CPU a tím pádem degradaci výkonu a ovlivnění ostatních protokolů nutných pro chod sítě.

Žádné řešení tedy není ideální.

Dosud jsme také předpokládali, že čipy TCAM a ASIC disponují určitou mírou flexibility, která umožní definovat chování pro nové typy hlaviček, protokolů atd. Je však třeba upozornit, že řada přepínačů na trhu je vybavená jednodušími čipy, které nemohou být takto přeprogramovány. Pokud se rozhodneme takovéto zařízení koupit, pak musíme počítat s tím, že nikdy nebude umět zpracovat v hardware některé funkce související s IPv6. Bohužel informace o tom, zda příslušné zařízení má plnohodnotnou podporu v hardware, je vesměs mezi výrobci obtížně dohledatelná.

A jak je to v IPv4?

Někteří si možná vzpomenou, že obdobná dynamická struktura je rovněž k dispozici v protokolu IPv4. Základní hlavičku IPv4 je možno rozšířit o položku označovanou jako *IP Options*, která, stejně jako rozšířené hlavičky v IPv6, má dynamickou velikost. Oproti rozšířeným hlavičkám v IPv6 však je možné položku *IP Options* "přeskočit" v jednom kroku a zcela jednoznačně identifikovat protokol vyšší vrstvy (TCP, UDP, ICMP, ...). Velikost položky *IP Options* je totiž možné určit přímo z velikosti hlavičky IPv4 a stejně tak jako identifikaci protokolu vyšší vrstvy (políčko Protocol). Firewall tak může bez problémů *IP Options* ignorovat. V principu však i toto rozšíření vykazuje mnohé nepříjemné vlastnosti, které můžeme najít u rozšířených hlaviček IPv6, tj. dynamická struktura s proměnlivou délkou, jejíž zpracování zpravidla nerealizuje specializovaný hardware, ale procesor. Z tohoto důvodu jsou IPv4 pakety obsahující *IP Options* většinou správců považovány za nebezpečné a na směrovačích rovnou zahazovány. Z dnešního pohledu můžeme říci, že toto rozšíření nikdy nebylo v minulosti nějak výrazně používáno a na

některých novějších L3 přepínačích není ani možné doručování IPv4 paketů s *IP Options* povolit. Další forma rozšířených hlaviček, kterou v IPv4 můžeme potkat, jsou hlavičky ESP a AH protokolu IPsec, který byl ostatně do IPv4 převzat z IPv6. Zde je situace trochu podobná nicméně ani zde nedochází k onomu nepříjemnému řetězení hlaviček.

Závěr

V dnešním dílu seriálu jsme si popsali základní koncept a některé principiální problémy rozšířených hlaviček. Do budoucna nelze vyloučit, že rozšířené hlavičky v IPv6 postihne stejný osud jako *IP Options* v IPv4 - tj. nebudou se prakticky využívat a na směrovačích se pakety s rozšířenými hlavičkami budou rovnou zahazovat. Na půdě IETF, ale zatím převažuje snaha udržet rozšířené hlavičky v co nejotevřenější a tedy i nejflexibilnější podobě i do budoucna. V dalším díle seriálu se podrobně podíváme, jaké je vlastně reálné nasazení rozšířených hlaviček v dnešních IPv6 sítích, jak může celý koncept rozšířených hlaviček zneužít útočník a současně se také pokusíme najít cestu, jak se těmto útokům dá zabránit.

**Bezpečné IPv6: vícehlavý útočník**

Dnešní díl bude bez dalších úvodů navazovat na díl předchozí. Před čtením tohoto dílu tedy doporučujeme nejdříve prostudovat díl předchozí, ve kterém byla obecná problematika rozšířených hlaviček vysvětlena.

Vkládání hlaviček

Úplně nejzákladnějším způsobem zneužití rozšířených hlaviček je jejich cílené vkládání do těla paketu. Smyslem tohoto útoku je doručit koncovému zařízení data, která by za normálních okolností byla odfiltrována. Jak tento útok může fungovat v praxi si ukážeme na našem oblíbeném mechanismu *RA-Guard*.

Podstatou útoku je vložení několika rozšířených hlaviček tak, aby filtrační mechanismus přístupového portu přepínače nebyl schopen identifikovat protokol, který je v IPv6 paketu přenášen. Vzhledem k tomu, že paket obsahující více rozšířených hlaviček je z hlediska specifikace protokolu IPv6 naprosto v pořádku, koncové zařízení jej normálně zpracuje. Pokud si tedy vytvoříme specifický ICMPv6 paket *Ohlášení směrovače*, koncové zařízení bez problémů provede na základě informací v paketu konfiguraci či rekonfiguraci. Podrobně se tímto jednoduchým útokem zabývá RFC 7113. Paket, do kterého jsme si vložili tři *Destination Options*hlavičky pak může vypadat například následovně.

```
▸ Frame 1: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0
▷ Ethernet II, Src: CadmusCo_f5:4b:d0 (08:00:27:f5:4b:d0), Dst: IPv6mcast_01 (33:33:00:00:00:01)
▽ Internet Protocol Version 6, Src: fe80::a00:27ff:fef5:4bd0 (fe80::a00:27ff:fef5:4bd0), Dst: ff02::1 (ff02::1)
    ▷ 0110 .... = Version: 6
    ▷ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
      .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
      Payload length: 72
      Next header: IPv6 destination option (60)
      Hop limit: 64
      Source: fe80::a00:27ff:fef5:4bd0 (fe80::a00:27ff:fef5:4bd0)
      [Source SA MAC: CadmusCo_f5:4b:d0 (08:00:27:f5:4b:d0)]
      Destination: ff02::1 (ff02::1)
    ▽ Destination Option
        Next header: IPv6 destination option (60)
        Length: 0 (8 bytes)
      ▷ IPv6 Option (PadN)
    ▽ Destination Option
        Next header: IPv6 destination option (60)
        Length: 0 (8 bytes)
      ▷ IPv6 Option (PadN)
    ▽ Destination Option
        Next header: ICMPv6 (58)
        Length: 0 (8 bytes)
      ▷ IPv6 Option (PadN)
▽ Internet Control Message Protocol v6
      Type: Router Advertisement (134)
      Code: 0
      Checksum: 0xb8fc [correct]
```

Tento paket je pak již dostatečně velký, aby obešel filtrační jednotku některých zařízení. Pokud byste chtěli útok otestovat ve vaší síti, můžete použít následující rozšířený kód z prvního dílu, který jsme použili pro generování falešných zpráv *Oznámení směrovače*.

```
$ python
>>> from scapy.all import *
>>> h = IPv6ExtHdrHopByHop(len=0)
>>> a =
IPv6(dst='fe80::6ab5:99ff:feea:d48a')/h/ICMPv6ND_RA(chlim=64,routerlif
etime=1800)/ICMPv6NDOptPrefixInfo(prefix='2a00:2450:4014:80b::',prefixlen=64)
>>> send(a)
```

Na třetím řádku přibyl kód, který zajistí vytvoření jedné *Hop-by-hop* rozšířené hlavičky v celkové délce 8 bajtů. Ta je pak následně vložená mezi základní IPv6 hlavičku a ICMPv6. Tím jsme vložili pouze jednu rozšiřující hlavičku. Větší počet si můžeme naskládat například takto:

```
h = IPv6ExtHdrHopByHop(len=0)/IPv6ExtHdrHopByHop(len=0)
```

Přestože je tento útok znám již delší dobu, současná zařízení jsou vůči němu pořád zranitelná. Příkladem může být poslední verze přepínače určeného pro přístupovou vrstvu Cisco Catalyst 2960-S. I s nejnovější verzi IOS, která je dnes dostupná (15.2.2E), je přepínač vůči tomuto útoku zranitelný. Tento přepínač navíc nelze nakonfigurovat tak, aby zahazoval pakety, které nerozpozná (*undetermined transport* v ACL, jak si popíšeme dále), tedy je vůči tomuto útoku kompletně bezbranný.

Na straně operačních systémů je situace o něco lepší. Obrana je postavená na jednoduchém principu. Pakliže je operačnímu systému doručen paket *Oznámení směrovače* a současně tento paket obsahuje rozšířené hlavičky, tak je zahozen. Mnohé operační systémy již tuto obranu mají implementovanou. Příkladem mohou být poslední verze Windows. U Linuxu záleží na distribuci - např. Ubuntu se rozšířeným hlavičkám v *Ohlášení směrovače* brání, CentOS ne. Tato obrana však není definována v žádném RFC, jedná se tedy víceméně o porušení standardu.

Demonstrace útoku prostřednictvím *RA-Guard* je jednoduchá díky tomu, že se jedná o jednosměrnou komunikaci. Trošku složitější situace nastává v případě, že chceme klasifikační/filtrační jednotku obelstít při použití interaktivní komunikace - například u protokolu TCP. Princip útoku zůstává stále stejný. Mezi základní IPv6 hlavičku a hlavičku protokolu TCP vložíme jednu či několik rozšířených hlaviček a zašleme protistraně. Realizace toho útoku, pokud nechceme ve Scapy programovat celý TCP stack, je ale trochu složitější. Pro případ, že chcete otestovat, jestli firewall na vašem síťovém zařízení je vůči takovému typu útoku odolný, můžete zkusit použít jednoduchý modul pro linuxové jádro, který zajistí, že do každého odeslaného paketu bude vloženo několik rozšířených hlaviček.

## Zneužití Fragmentace

Další typ útoků je možné realizovat díky hlavičce fragmentace. Tento útok je v principu hodně podobný útokům zneužívající fragmentaci v IPv4. Princip spočívá v tom, že celý paket je účelově rozdělen do fragmentů. Vzhledem k tomu, že většina jednoduchých paketových filtrů (například těch realizovaných v ASIC čipu) neumí provádět rekonstrukci fragmentů (fragment reassembling), lze tento útok také s úspěchem použít k obcházení filtračních pravidel v síti.

Pro otestování lze použít následující kód:

```
$ python
>>> from scapy.all import *
>>> a =
IPv6(dst='fe80::6ab5:99ff:feea:d48a')/IPv6ExtHdrFragment()/ICMPv6ND_RA
(chlim=64,routerlifetime=1800)/ICMPv6NDOptPrefixInfo(prefix='2a00:2450:4014:80b::',
prefixlen=64)chlim=64,routerlifetime=1800)/ICMPv6NDOptPrefixInfo(prefix='2a00
:2450:4014:80b::',prefixlen=64)
>>> send(fragment6(a,x)
```

V kódu jsme za IPv6 hlavičku vložili rozšířenou hlavičku fragmentace, zbytek paketu může zůstat tak, jak je. O fragmentaci samotnou se postaráme pozměněním kódu pro odesílání, kde zavoláme funkci fragment6(), která se za nás postará o vlastní fragmentaci. Funkce přijímá dva argumenty - první argument jsou data k odeslání a druhý argument je maximální velikost fragmentu. Tu si zvolíme tak, aby nám vyhovovala a paket se skutečně rozfragmentoval. V praxi pak výsledný paket může vypadat jak na následujícím obrázku.

```
▷ Frame 2: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▷ Ethernet II, Src: 08:00:27:f5:4b:d0 (08:00:27:f5:4b:d0), Dst: 33:33:00:00:00:01 (33:33:00:00:00:01)
▽ Internet Protocol Version 6, Src: fe80::a00:27ff:fef5:4bd0 (fe80::a00:27ff:fef5:4bd0), Dst: ff02::1 (ff02::1)
  ▷ 0110 .... = Version: 6
  ▷ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 24
    Next header: IPv6 fragment (44)
    Hop limit: 64
    Source: fe80::a00:27ff:fef5:4bd0 (fe80::a00:27ff:fef5:4bd0)
    Destination: ff02::1 (ff02::1)
  ▽ Fragmentation Header
      Next header: ICMPv6 (58)
      Reserved octet: 0x0000
      0000 0000 0000 0... = Offset: 0 (0x0000)
      .... .... .... .00. = Reserved bits: 0 (0x0000)
      .... .... .... ...1 = More Fragment: Yes
      Identification: 0x0c2168a8
    Reassembled IPv6 in frame: 4
▽ Data (16 bytes)
    Data: 860054754008070800000000000000000
    [Length: 16]
```

Pokud by chtěl firewall zpracovat daný paket, nevidí do samotného obsahu - nemůže tedy ihned rozhodnout jestli má paket propustit nebo zahodit. Vidíme však, že firewall dokáže poznat protokol uvnitř paketu. Toto však může útočník obejít tím, že zkombinuje tento typu útoku s předchozím a vytvoří za hlavičkou fragmentace dostatečně dlouhý řetězec dalších rozšířených hlaviček. Hlavička vlastního protokolu se tak nedostane do prvního fragmentu. Je to tedy další cesta, jak můžeme v praxi vcelku jednoduše obejít paketový filtr.

## Filtrování hlaviček a vzdálený DoS s využitím fragmentace

V předchozím díle jsme si řekli, že zpracování rozšířených hlaviček v hardware je relativně komplikovaná záležitost. Z toho důvodu mnohá zařízení často IPv6 paket i s pouhou jednou rozšířenou hlavičkou přeposílají do CPU. Tím dramaticky degradují výkonnost a propustnost celého zařízení zpracovávající takový provoz. Díky těmto problémům mnozí ISP raději pakety s rozšířenými hlavičkami rovnou zahazují, než aby riskovali případné ohrožení stability infrastruktury. Touto problematiku se podrobněji zaobírá pracovní skupina IETF v6ops, která dochází kzajímavým zjištěním. Například pro jednoduchou hlavičku *Hop-by-Hop* je šance na zahození kolem 40 %. V dnešní době, kdy se v praxi intenzivně řeší ztrátovost paketů větší jak několik málo procent, je tato ztrátovost extrémní a posouvá to rozšířené hlavičky téměř do pozice nefunkční technologie. Problémem také je, že filtrování provozu obsahující rozšířené hlavičky se často neděje až v koncové síti, ale někde po cestě. To znamená, že zahazování není součástí nastavené bezpečnostní politiky koncového autonomního systému, ale pakety zahazuje často některý z tranzitních operátorů. To je podstatně složitější problém k vyřešení, protože díky dynamičnosti směrování vlastně dopředu nevíme, kudy nám provoz do cílové destinace poteče a koho případně kontaktovat. Pokud by potenciálně někdo chtěl využít koncept rozšířených hlaviček pro nějakou novou aplikaci, tak nemůže, jelikož nedokáže ani zajistit, že daný paket vůbec dojde Internetem na koncové zařízení. Tohoto stavu může navíc využít útočník pro vcelku zajímavý, i když poněkud kuriózní, typ DoS útoku.

Jelikož se útok opírá ještě o některé další vlastnosti protokolu IPv6, o kterých jsme se zatím v článcích nezmiňovali, nejprve si krátce je vysvětlíme.

IPv6 protokol vyžaduje, aby minimální délka paketu, kterou koncové uzly musí umět zpracovat, byla 1280 bajtů. Tím se posouvá hranice velikosti v bajtech, od které pakety mohou být fragmentovány nad tuto hodnotu. IPv6 také kompletně mění způsob fragmentace - již se neděje po cestě, jak tomu bylo zvykem u IPv4, ale fragmentují pouze koncová zařízení. Aby koncové zařízení mohlo zjistit, že paket, který odeslalo, je pro nějaký směrovač po cestě příliš velký, využívá se protokolu ICMPv6 a jeho zprávy *Packet Too Big*. Pokud tedy velký paket dorazí po cestě k cíli na úzké hrdlo, směrovač ho zahodí a pošle zpět koncovému zařízení zprávu *Packet Too Big*. Poslední informaci, kterou budeme pro správné pochopení útoku potřebovat, jsou tzv. Atomické fragmenty (*Atomic Fragments*). Jedná se o běžné, nefragmentované pakety, které však v sobě nesou fragmentační hlavičku. Byly definovány již v původní specifikaci IPv6 a podrobně se jimi zabývá RFC 6946. Pokud se zamýšlíte nad tím, k čemu je vůbec dobrá fragmentační hlavička v nefragmentovaném paketu, tak jí využívají některé přechodové mechanismy v okamžiku, kdy je třeba doručovat IPv6 pakety do IPv4 sítě.

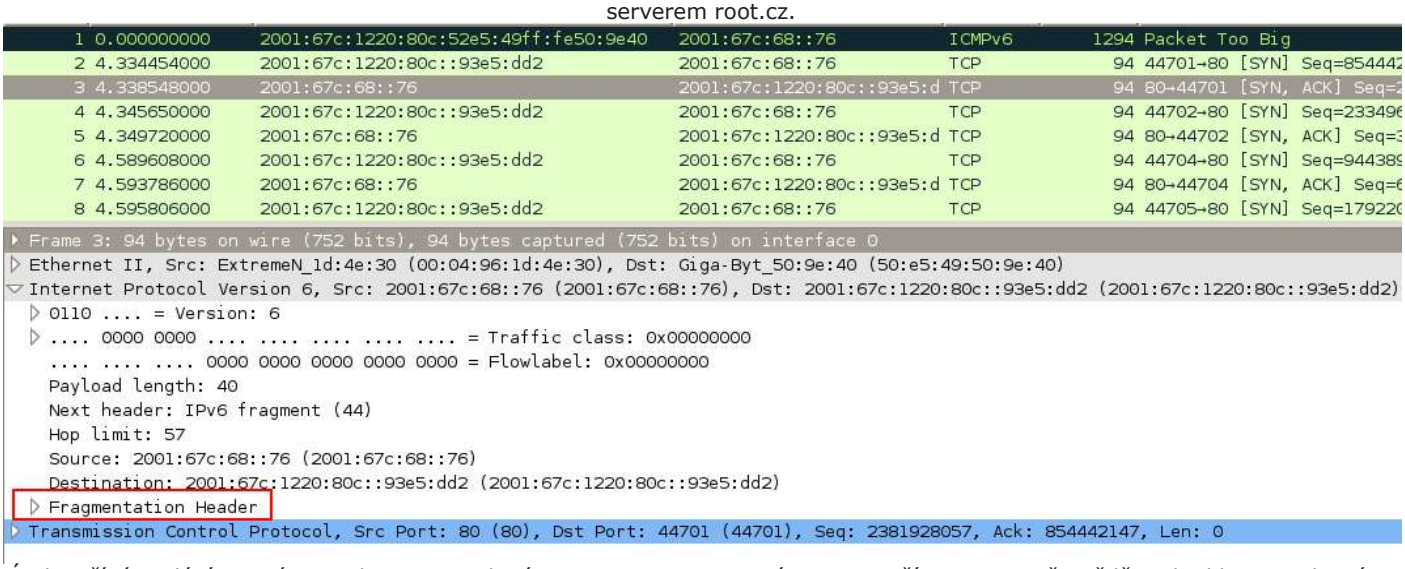
Jak potom probíhá vlastní útok? Útočník, který chce narušit komunikaci mezi klientem a serverem, zašle serveru zprávu *Packet Too Big* ve které server informuje, že cesta ke klientovi má MTU menší jak 1280. Server si tuto informaci poznamená, a pokud pak klient server kontaktuje, začne mu server v souladu se specifikací zasílat odpovědi s vloženou hlavičkou *Fragmentace*. Pokud jsou pakety s rozšířenými hlavičkami zahazovány po cestě, a v předchozí části jsme si ukázali, že tomu tak je v nemalém

množství případů, útočníkovi se podaří znemožnit IPv6 komunikaci mezi klientem a serverem. Pokud data s rozšířenou hlavičkou zahazována nejsou, stále je tu problém. Prohlížeče často odpověď obsahující hlavičku *Fragmentace* neinterpretují jako odpověď na svůj pokus o navázání spojení, a pokud prohlížeč nepoužívá nějakou variantu Happy Eyeballs, spojení trvá nemalou dobu, než se přepne zpět na IPv4.

Pro jednoduchou realizaci útoku si můžeme ICMPv6 paket poskládat sami pomocí Scapy, nebo lze využít další z penetračních nástrojů ipv6toolkit. Realizaci pak provedeme spuštěním příkazu:

```
icmp6 --icmp6-packet-too-big -d IPv6:adresa:serveru --peer-addr
IPv6:adresa:klienta --mtu 1000 -o 80 -v
```

Na adresu serveru se zašle zpráva ICMPv6 *Packet Too Big* tvářící se tak, že klient, specifikovaný parametrem peer-addr, má problém s přijetím dat od serveru, protože používá MTU pouze 1000 bajtů. Server pak začne vkládat do odpovědi klientovi hlavičku *Fragmentace* a útok je úspěšný. Na následujícím obrázku lze vidět, jak reálný útok vypadá, mezi naším zařízením a serverem root.cz.




Útok začíná zasláním zprávy *Packet too Big* danému serveru. Touto zprávou se snažíme server přesvědčit, aby klientovi, který je specifikován v těle této zprávy, začal posílat data s hlavičkou *Fragmentace.* Když se pak klient pokouší navázat TCP spojení (2. paket), server mu dle specifikace odpovídá s hlavičkou fragmentace (3. paket). Vidíme však, že klient tuto zprávu neinterpretuje jako odpověď na svůj pokus o navázání spojení a zkouší ho navázat znovu (4. paket). Toto se opakuje, dokud to klienta nepřestane bavit nebo nezasáhnou Happy Eyeballs. Tento příklad je také demonstrován na síti, kde nedojde k zahození paketů s rozšířenými hlavičkami. Pokud by ale odpověď od serveru putovala skrz tranzitního operátora, který takové pakety zahazuje, došlo by k efektivnímu útoku DoS. Odpovědi od serveru by totiž ke klientovi vůbec nedorazily.

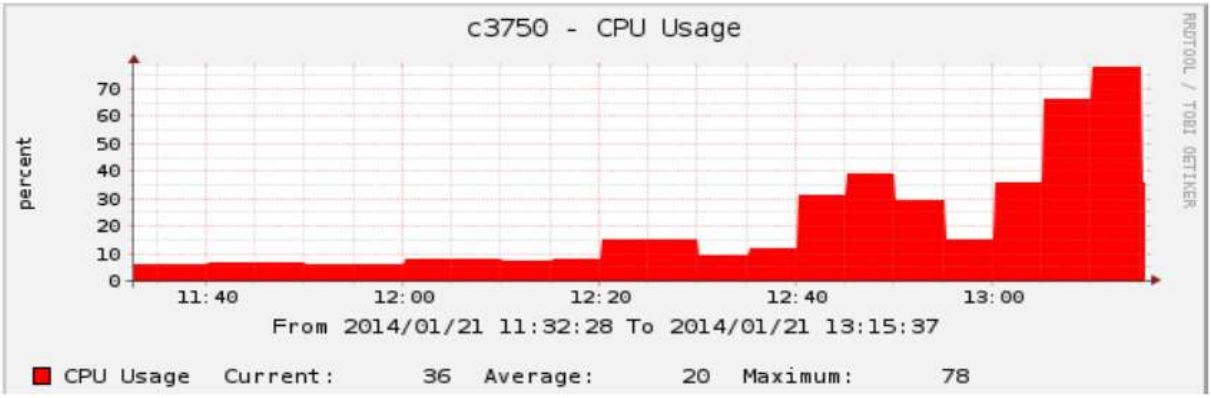
Pokud si to celé shrneme, tak pro úspěšnou realizaci útoku musí být splněno několik předpokladů.

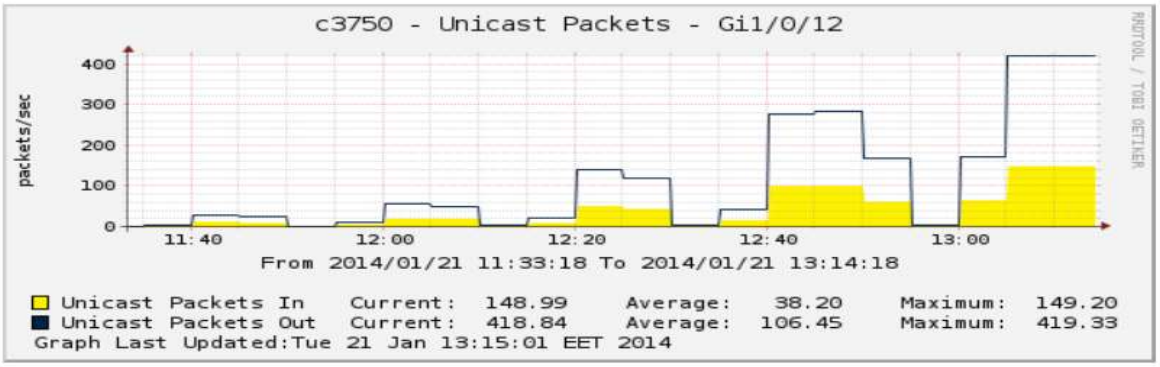
1. Útočník potřebuje znát IPv6 adresu klienta a serveru. To nicméně není až tak problém a RFC 5157 popisuje několik způsobů, jak dané adresy získat.
2. IPv6 pakety s atomickými fragmenty jsou po cestě filtrovány. Pokud se pakety nefiltrují, stále je zde možnost útoku, protože odpověď obsahující hlavičku *Fragmentace* často není interpretována jako odpověď na zaslaný paket.
3. Server musí při přijetí zprávy ICMPv6 *Packet Too Big* začít generovat atomické fragmenty (vložit do odpovědi hlavičku *Fragmentace*).
4. Server musí akceptovat podvrženou zprávu ICMPv6 *Packet Too Big*
5. ISP útočníka musí akceptovat paket, ve kterém je podvržená adresa.

Zdálo by se, že podmínek pro to, aby útok byl úspěšný, je hodně, nicméně většina z nich je vcelku bez problémů splnitelná. Pro zastánce přístupu: "Problém je vyřešen, když existuje RFC nebo BCP ( *Best Current Practice*), které mu zamezuje" mohou být lehce kontroverzní body 4. a 5. Pro bod 4. existuje RFC 5927, které podobné útoky popisuje a navrhuje určité validace zpráv. Pokud by servery používaly tato doporučení, byl by takový útok daleko obtížnější. U bodu 5. existuje dokument BCP 38, který doporučuje operátorům provádět validaci a filtraci prefixů, aby se zamezilo odeslání paketů s podvrženou zdrojovou IP adresou. Jednoduše řečeno, operátor by měl předávat pouze pakety, kde zdrojová adresa spadá pod prefix, který je danému operátorovi přidělen. Pokud by všichni operátoři BCP 38 zavedli, útočník by pak nemohl podvrhnout svou adresu a byl by lépe dohledatelný. Zdálo by se, že tedy řešení problému dávno existuje, nicméně realita je však taková, že se tyto kontroly moc neprovádí. Proč tomu tak je, by vydalo na samostatný seriál. Spokojíme se zatím s tím, že útočníkovi v dnešním Internetu vcelku nic moc nebrání zasílat pakety s podvrženou IP adresou a podvrženými údaji v ICMP paketu.

Přetížení CPU síťových prvků

Je třeba si dát také pozor na vytížení procesoru na síťových zařízeních. Přepínače se většinou snaží zpracovávat pakety pomocí hardware, nicméně ne vždy je to možné. Příkladem paketu, který vždy zpracovává procesor je IPv6 paket obsahující rozšířenou hlavičku *Hop-by-Hop*. Ostatní rozšířené hlavičky se zařízení většinou snaží přeposlat pomocí hardware, jelikož se jimi moc zabývat nemusí. Problém nicméně je, když od zařízení požadujeme právě kontroly a validace těchto paketů. Potom se může stát, že paket již v hardware zpracovat nelze a musí se předat CPU. Tématu vytížení procesoru při aktivních mechanismech First Hop Security se věnuje například práce „The impact of using SAVI on processing resources". V následujícím grafech Ovidiu Strugaru například ukazuje, o kolik se zatíží CPU, pokud ověřování pro IPv6 zapneme.




Autoři práce postupně zvedali počet paketů za vteřinu, které musel přepínač validovat. Pro 80% zatížení procesoru stačilo pouze 150 paketů za vteřinu. Dané vytížení bylo dáno pouze při podvržení zdrojové IPv6 adresy a MAC adresy. Vůbec tedy není zohledněno další vytížení při vložení rozšířených hlaviček, kde se dá předpokládat, že bude vyšší.
Ideální přepínač by měl omezovat počet paketů, které jsou zasílány z hardware do CPU (*rate-limiting*). Cisco povětšinou ve svých materiálech uvádí, že ho má implementovaný, nicméně typicky pouze pro platformy 6500 a 7200, tedy ne pro zařízení určené pro přístupovou vrstvu. U ostatních zařízení je *rate-limiting* většinou implementován pouze pro jednotlivé protokoly (např. limit na broadcast pakety u ARP aj.), nicméně chybí nějaká lepší podpora pro IPv6. Ostatně to lze vidět i v dané ukázce, kde byli schopni vcelku výkonný přepínač 3750 hodně zatížit.

Částečná obrana

Pro obranu proti výše zmíněným útokům využijeme zpravidla nějakou formu filtrace paketů s rozšířenými hlavičkami. Lze vytvořit např. následující ACL, které bude zahazovat hlavičky *Destination Options* a *Hop-by-Hop*.

```
Router(config)# ipv6 access-list DENY-EXTHEADERS
Router(config-ipv6-acl)# deny 60 any any
Router(config-ipv6-acl)# deny 0 any any
Router(config-ipv6-acl)# permit ipv6 any any
```

Takováto filtrace je ale poměrně drastické opatření, které může poškodit i legitimní provoz. Není také vždy pravidlem, že zařízení podporuje danou filtraci. Například u L2/L3 přepínačů 2960s nebo 3750x tyto ACL podporovány nejsou. Přepínač vám je sice dovolí nakonfigurovat, nicméně provozní pakety s rozšířenými hlavičkami bez problémů přepošle dále.
Abychom zbytečně nezahazovali legitimní provoz, lze použít méně striktní variantu. Filtry nastavíme tak, aby zahazovaly pouze provoz, u kterého zařízení nedokáže rozpoznat transportní protokol - tedy když zařízení není schopné paket zpracovat natolik, aby našlo, kde transportní protokol vůbec začíná. Tyto pakety mají navíc do legitimního provozu většinou dost daleko, tedy jejich zakázáním se toho tolik nestane. Zneužití fragmentace nebo řetězce hlaviček, který je příliš dlouhý, že ho zařízení není schopno zpracovat, jsme demonstrovali na útoku, pomocí kterého je možné obejít ACL přepínače. Ve spolupráci se společností HP se nakonec podařilo najít a odladit řešení, které tento typ útoku eliminuje. Výsledkem je nově přidaná funkcionalita, kterou můžete na nejnovějším firmware Comware aktivovat pomocí jednoduchého příkazu:

```
[HP-Comware] ipv6 option drop enable
```

Cisco na některých svých zařízeních podporuje ACL, kterým lze dosáhnout podobného efektu. Můžeme tak rozšířit ACL, které jsme si ukázali v předchozím díle následovně:

```
Switch(config)# ipv6 access-list DENY-RA
Switch(config-ipv6-acl)# deny icmp any any router-advertisement
Switch(config-ipv6-acl)# deny ipv6 any any undetermined-transport
Switch(config-ipv6-acl)# permit ipv6 any any
```

Klíčový je zde třetí řádek, kdy pomocí klíčového slova *undetermined-transport*přikážeme zařízení, aby nerozpoznaný provoz zahodil. ACL tedy zahodí všechny IPv6 pakety *Oznámení směrovače* a pakety, které není schopno zpracovat natolik, aby rozpoznalo, co je v nich vlastně přenášeno. Je třeba ale poznamenat, že podpora tohoto příkazu zcela závisí na tom, jestli je podporován v hardware daného přepínače. U přepínačů určených pro přístupovou vrstvu (např. řada 2900) tomu tak většinou není.
Pokud zařízení nepodporuje *undetermined-transport* u ACL, lze ještě využít přístup negace. Místo toho, abychom zakázali to, co přepínač nerozpozná, tak povolíme pouze to, co rozpozná. Konfigurace bude sice o dost složitější, ale pro některé platformy je to jediná možnost jak těmto útokům zabránit. Příkladem by pak mohlo být následující ACL.

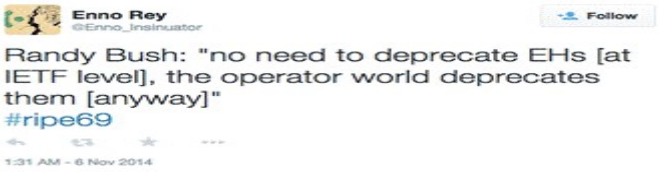
```
Switch(config)# ipv6 access-list DENY-UNDETERMINED-TRANSPORT
Switch(config-ipv6-acl)# permit 0 any any
Switch(config-ipv6-acl)# permit 1 any any
Switch(config-ipv6-acl)# permit 3 any any
...
...
Switch(config-ipv6-acl)# permit 254 any any
Switch(config-ipv6-acl)# permit 255 any any
```

Nenechte se zmást faktem, že v daném ACL vlastně nic nezakazujeme a vše naopak povolujeme. Toto ACL totiž využíváme pouze proto, abychom povolili provoz, který zařízení je schopno rozpoznat. Jelikož se přepínač snaží zpracovat paket až na transportní protokol, pokud transportní protokol nerozpozná, tak ho nemůže propustit, protože ACL povoluje jenom pro něj známé protokoly.

Pokud se u síťových zařízení nemůžeme spoléhat na filtraci nebo mechanismy *First Hop Security*, zbývá nám pouze nějak zabezpečit koncové operační systémy. Existují dvě RFC, které se danou problematikou zabývají. RFC 7112 doporučuje, aby při použití fragmentace, byly všechny rozšířené hlavičky v prvním fragmentu. Zařízení pak je schopno rozpoznat, jaký protokol se ve fragmentovaném paketu nachází a zařídit se podle toho. RFC 6980 pak zakazuje použití hlavičky fragmentace u zpráv *Ohlášení směrovače, Ohlášení souseda* atd. a některé koncové systémy ho již implementovaly. Některé operační systémy zašly ještě dále a filtrují všechny zprávy protokolu NDP, které obsahují nějakou rozšířenou hlavičku. Jedná se sice o porušení standardu, nicméně zas tak to nevadí. U protokolu NDP zatím neexistuje jediný důvod, proč by nějaké rozšířené hlavičky měl obsahovat, tedy když se zahodí, nic se nestane. Ideální je tedy udržovat koncové systémy aktualizované, což asi nemusíme příliš připomínat.

Neradostné vyhlídky rozšířených hlaviček

V současné době neexistuje jednoznačná cesta k řešení problému rozšířených hlaviček. Z pohledu IETF je celkem pochopitelná snaha zachovat možnosti rozšířených hlaviček pokud možno v co nejvíce flexibilní podobě. Proti tomu ovšem stojí implementační možnosti současných technologií a požadavky operátorů. Realitu pak vystihuje glosa Randyho Bushe z posledního meetingu RIPE69:




Díky problémům s neefektivním zpracováním rozšířených hlaviček se nelze divit, že řada ISP realizuje ve svých sítích striktní politiku a pakety s rozšířenými hlavičkami zahazuje - většinou kromě hlavičky fragmentace. Organizace IETF, která je jako jediná schopna vyřešit věc na úrovni standardu, nicméně nemá na problematiku jednotný názor. Jedna snaha vedla směrem k vytvoření ještě univerzálnější rozšířené hlavičky než té z RFC 6564, kterou jsme popisovali minule. Změna formátu rozšířených hlaviček však nebyla podpořena.

Jiným směrem se ubírá aktuálně vznikající draft, který detailně rozebírá, který typ rozšířených hlaviček je přípustné filtrovat a který nikoliv. Otázka je, jestli získá podporu a bude schválen. Navíc, i pokud by schválen byl, nejedná se o standard, ale pouze o doporučení (Best Current Practice), takže i jeho reálný dopad je omezený.

Výsledkem je, že se v reálné síti rozšířené hlavičky téměř nepoužívají. Situaci v síti VUT popisuje následující tabulka, která zobrazuje počet jednotlivých protokolů a rozšířených hlaviček za jeden týden provozu. Data jsou získané z linky připojující VUT v Brně do sítě CESNET.

| Hlavička/Protokol | Paketů |
|---|---|
| TCP (6) | 11 123,4 mln |
| UDP (17) | 507,3 mln |
| ICMP6 (58) | 43,2 mln |
| Fragment (44) | 21,2 mln |
| OSPF (89) | 0,16 mln |

Vidíme, že kromě TCP, UDP, ICMPv6 a hlavičky fragmentace se ostatní hlavičky nepoužívají. Je třeba ale vzít v potaz, že tato statistika je trošku zkreslená, protože se jedná o páteřní linku - není zde tedy vidět například hlavička Hop-by-Hop, kterou běžně používá IPv6 multicast.

Závěr

Rozšířené hlavičky jsou často opomíjenou součásti protokolu IPv6. Často je na ně pohlíženo jako benefit protokolu IPv6. Tím částečně také jsou. Umožňují potenciální rozšíření protokolu do budoucna a dodávají mu flexibilitu. Je třeba ale také poznamenat, že nic není zadarmo. Za zvýšenou flexibilitu platíme daň v podobě složitějšího zpracování paketů. Jak jsme si ukázali, použití rozšířených hlaviček v reálném provozu je nyní zatím spíše omezené. Při implementaci protokolu IPv6 do sítě bychom však jejich existenci neměli ignorovat a jejich zpracování řešit jako součást bezpečnostní politiky. Zejména kvůli faktu, že se reálně dají použít pro obcházení filtrovacích pravidel nebo IDS systémů, jak bylo ukázáno na konferenci BlackHat. S jejich pomocí může také útočník v dnešní době většinou vcelku bez problémů obejít mechanismy *First Hop Security*, které jsme si popsali v předchozích dílech. Při nákupu zařízení, zejména těch řešících bezpečnost, je tedy dobré se informovat na schopnosti zařízení správně zpracovat rozšířené hlavičky. U některých zařízeních může výrobce v budoucnu podporu pro efektivní zpracování rozšířených hlaviček přidat v podobě aktualizace firmware. Někdy ale, pokud tuto funkcionalitu budeme vyžadovat, bude nutné vyměnit celé zařízení nebo jeho části.

**Poznámka:** *Uvedené příklady, nástroje nebo ukázky zdrojových kódů jsou upraveny, aby je nebylo možné použít pouhým zkopírováním do konzole a spuštěním. Slouží pouze jako prostředek k hlubšímu pochopení diskutované tématiky. Upozorňujeme, že jejich zneužití může být v rozporu přinejmenším s dobrým vychováním.*

**Bezpečné IPv6: trable s multicastem**

Multicast je obecně zasílání dat z jednoho zdroje více příjemcům, ideálně tak, aby byla co nejefektivněji využita síťová infrastruktura. Data by ideálně měla sítí téct každou linkou maximálně jednou a „větvit" se pouze tam, kde je to nezbytně nutné - nejlépe až na přístupové (*access*) vrstvě. Multicast se používá především pro online přenos multimediálních dat, kdy nechceme zbytečně zatížit síťovou infrastrukturu.

Obecnému popisu multicast vysílání v IPv6 jsme se již před lety zabývali v sedmém dílu jiného seriálu. V rámci tohoto seriálu se podíváme, jak nám zavádění protokolu IPv6 do větších sítí zamávalo s úmysly využít pro všechny případy multicast provoz. Pro začátek si ale udělejme krátkou exkurzi do historie a zopakujme základní informace, které budeme potřebovat pro pochopení zbytku článku.

Trocha historie

Protokol IPv6 byl navrhován jako nekompatibilní s IPv4. Proto byla snaha ho navrhnout od začátku správně a eliminovat určité nedostatky, kterými se tehdy potýkal protokol IPv4. Jedním z problémů, který v 90. letech trápil síťové administrátory, bylo příliš mnoho broadcast provozu. Sítě byly tehdy propojené pomocí hubů či jednoduchých přepínačů a v takových sítích je příliš mnoho broadcast zpráv problém. Pro tento jev se vžilo označení *broadcast storm* a dokázal správcům způsobit celou řadu problémů. Výpočetní čas pro zpracování zprávy na zařízení byl v té době ještě drahý (CPU nebyla tak výkonná), a problémem je, že broadcast zprávu musí zařízení zpracovat vždy a až podle obsahu se rozhodnout, jestli ji budou ignorovat, nebo ne.

Jedním z důvodů vzniku velkého množství broadcast zpráv může být zapojení velkého množství zařízení. Každé zařízení totiž občas potřebuje zaslat zprávu všem ostatním v dané síti a více zařízení vede logicky k většímu počtu broadcast zpráv.

Dalším důvodem je samotná architektura protokolu Ethernet. Ethernet totiž sám o sobě nemá žádný prostředek pro eliminaci smyček, jako například IP v podobě TTL, tedy, jakmile je síť omylem zakruhována, broadcast provoz je přeposílán stále dokola. Síť je v takovém případně nestabilní, musí se dohledat zařízení, na kterém zakruhování vzniklo a kruh rozpojit. Tento problém eliminuje protokol STP (*Spanning Tree Protocol*), nicméně jednoduché přepínače ho v té době nepodporovaly a nedokázaly tedy smyčku v síti automaticky přerušit.

Snahou tedy od začátku bylo u protokolu IPv6 broadcast eliminovat. Ač IPv6 nemůže vyřešit problém zakruhování topologie, může odstranit nutnost zasílání broadcast zpráv na síťové vrstvě. Jelikož je ale občas potřeba zaslat nějaké informace více zařízením, nelze používat pouze unicast. Multicast se tedy nabízel jako správná volba, jelikož je to obecně nejefektivnější doručení dat více zařízením skrz síťovou infrastrukturu. Navíc za prvotním návrhem IPv6 stojí Steve Deering, autor celého konceptu multicast vysílání.

Při návrhu protokolu IPv6 byl tedy multicast použit téměř pro všechny obslužné protokoly a činnosti. Jedná se zejména o bezstavovou konfiguraci, kontrolu duplicity adresy, zjištění mapování mezi IPv6 a MAC adresami aj.

Multicast - základní princip

Pojďme se nyní podívat na základní pravidla fungování multicast vysílání. Pokud je třeba data efektivně doručit skrz síť, zdroj dat je zašle do příslušné multicast skupiny - tedy jako cílovou IPv6 adresu nastaví IPv6 adresu dané multicast skupiny. Klienti, pokud o daná data mají zájem, se do dané multicast skupiny přihlásí a začnou data odebírat. Vidíme tady trochu odlišné chování od unicast komunikace. Klienti si o multicast data musí nejprve říci, než jim jsou zaslaná. Toto projevení zájmu provedou pomocí protokolu MLD (*Multicast Listener Discovery*).

Protokol MLD využívá tři základní zprávy - *Query*, *Report*, *Done*, které jsou součástí ICMPv6. Zpráva *Query* slouží k detekci zájemců o multicast provoz. Zpráva *Report*slouží k vyjádření zájmu o multicast a zprávou *Done* zařízení informuje, že o daný multicast provoz již zájem nemá.

Princip fungování protokolu MLD je pak následující. Směrovač (*Querier*) se pravidelně prostřednictvím zprávy *Query* ptá, jestli jsou v síti nějací zájemci o multicast provoz. Koncová zařízení mu odpovídají zprávou *Report*, ve které sdělují, o jaké skupiny mají zájem. Multicast provoz jim je pak následně přeposílán. Aby byl směrovač informován, že zprávám protokolu MLD má věnovat pozornost, obsahují všechny MLD zprávy rozšířenou hlavičku *Hop-by-Hop* s nastavenou volbou *Router Alert*.

U multicast vysílání také rozlišujeme, zda-li se jedná o multicast přeposílaný mezi směrovači, kde směrování multicast provozu zajišťuje převážně protokol PIM (*Protocol Independent Multicast*) nebo multicast v lokální síti. My se v tomto dílu budeme věnovat pouze multicast provozu v lokálních sítích.

V lokální síti platí pro multicast provoz trochu jiná pravidla než mezi směrovači, protože se pro jeho doručení nepoužívá žádný směrovací protokol. Zařízení v lokální síti (přepínače, uživatelské počítače) musí tedy nějakým způsobem sama zjistit, že daný provoz je opravdu multicast provozem. Na síťové vrstvě je to vcelku jednoduché - IPv6 multicast má vyhrazený prefix ff00::/8. V lokální síti ale rámce adresujeme pomocí linkových (MAC) adres. K odlišení multicast provozu od unicast komunikace nám zde poslouží speciální MAC adresa, jejíž prvních 16 bitů má hodnotu 33:33. Zbývajících 32 bitů dané MAC adresy se doplní spodními 32 bity multicast IPv6 adresy. Například IPv6 muticast skupina FF02::1:FF68:12CBse tak namapuje na Ethernet multicast 33:33:FF:68:12:CB. Díky tomuto mapování všechna zařízení pracující na linkové vrstvě (přepínače, síťové karty klientů) poznají, že se jedná o multicast provoz a zachází s ním odpovídajícím způsobem. Pokud se klient přihlásí do IPv6 multicast skupiny, síťová karta si poznamená odpovídající multicast MAC adresu a příchozí data z této skupiny bude předávat operačnímu systému. Zbytek multicast provozu může odfiltrovat, aby se operační systém nezabýval jiným multicast provozem, než o který má opravdu zájem.

O jaké skupiny má zájem třeba vaše zařízení, si můžete v Linuxu zobrazit příkazem ip maddr. Daným příkazem vypíšete, k jakým multicast skupinám je vaše zařízení přihlášeno. Například:

```
$ ip maddr
2:    eth0
link  33:33:ff:ea:d4:8a
inet6 ff02::1:ffea:d48a
```

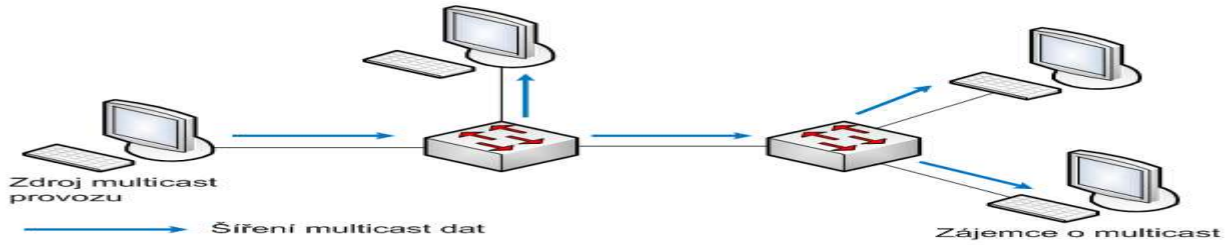
Z tohoto zkráceného výpisu, vidíme, že zařízení je přihlášeno, na rozhraní eth0, do multicast skupiny ff02::1:ffea:d48a. Z výpisu můžeme také vyčíst odpovídající linkovou multicast adresu (33:33:ff:ea:d4:8a) k dané IPv6 multicast skupině (ff02::1:ffea:d48a).

Pokud byste hledali podobný příkaz pro Windows, tak lze použít netsh interface ipv6 show joins.

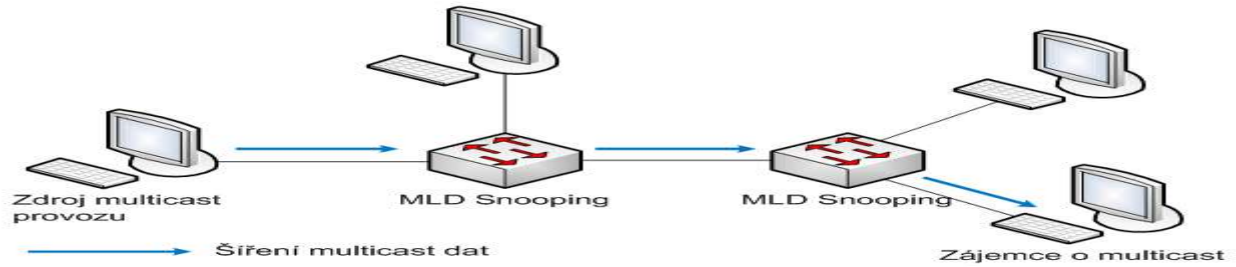
Efektivní doručení multicast provozu

Jakým způsobem můžeme v lokální síti zajistit distribuci multicast provozu? Ten nejjednodušší způsob je s multicast provozem zacházet jako s broadcast provozem. Jelikož nevíme, kdo o něj má zájem, pošleme ho prostě po celé síti. Koncová zařízení se

pak sama rozhodnou, zdali data zahodí nebo předají operačnímu systému. Tento způsob ilustruje následující obrázek, na kterém vidíme, že se multicast provoz šíří i na zařízení, která o něj zájem nemají.




Tento způsob nicméně není příliš efektivní. Ne všechny síťové karty dokáží odfiltrovat multicast provoz, který je nezajímá. Pokud do sítě vysíláme např. větší množství multimediálních dat ve vysokém rozlišení, bitrate často dosahuje desítek až stovek Mb/s. Takovým množstvím dat již můžeme koncová zařízení nepříjemně zatížit. Aby se tomuto problému zabránilo, používá se technika označovaná jako *MLD Snooping*. Zjednodušeně řečeno, když zařízení informuje síť o svém zájmu o určitou multicast skupinu zprávou *Report*, přepínač si u sebe tuto informaci poznamená. Díky tomu přepínač ví, na kterém portu je jaký zájemce o jaký multicast. Trochu tím porušujeme vrstvový model, jelikož přepínač (L2 zařízení) se dívá do vrstev vyšších, protože potřebuje zpracovat zprávy protokolu MLD, nicméně je to vyváženo efektivitou zasílaného provozu. Pokud tedy v síti aktivujeme *MLD Snooping*, zajistíme tak efektivní přeposílání multicast provozu a tok multicast dat pak může vypadat jako na následujícím obrázku.




Tabulka na zařízeních Cisco může vypadat následovně:

```
Switch#show ipv6 mld snooping address
  Vlan      Group                Type        Version     Port List
------------------------------------------------------------------------------
  1         FF1E::1              mld         v2          Fa0/11, Fa0/23
    1         FF02::FB             mld         v2          Fa0/23
```

Z daného výpisu je patrné, že např. provoz směřující do multicast skupiny FF1E::1bude zasílán pouze na porty Fa0/11 a FA0/23. Ostatní zařízení v síti nebudou provozem obtěžována. V praxi tyto optimalizace multicast provozu provádí pouze operátoři, kteří využívají svou infrastrukturu pro přenos větších objemů multimediálních dat - typicky televizní vysílání. Většina sítí tyto optimalizace u IPv6 a často ani u IPv4 neprovádí a provoz je tak rozesílán na všechna zařízení jako u prvního příkladu.

Broadcast = multicast?

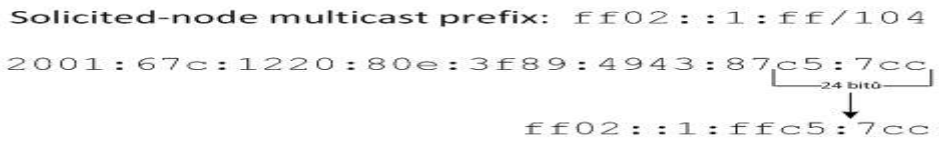
Přestože byla u IPv6 snaha kompletně eliminovat broadcast, někdy prostě potřebujeme zaslat informace všem klientům v jedné síti - například zprávu *Ohlášení směrovače*. Jak to udělat v IPv6, když broadcast nemáme? Pravdou je, že IPv6 nemá broadcast jenom na první pohled. Broadcast totiž představuje speciální případ multicast provozu, kdy se všechny uzly v síti přihlásí k příjmu stejné multicast skupiny. Daná multicast skupina *all-nodes* má adresu FF02::1 a podleRFC 4861 se do ní musí povinně přihlásit všechny uzly. V tomto případě ale nedává smysl, aby zařízení signalizovala pomocí MLD, že o danou skupinu mají zájem. Proto RFC 3810 definuje, že se pro danou skupinu žádné zprávy protokolu MLD neposílají. Přepínače s podporou *MLD Snooping* si tedy nemusí pro danou skupinu udržovat stavovou informaci a data rozešlou na všechny porty. Reálně je tedy tato skupina implementována naprosto totožně jako broadcast u IPv4. Fakticky tedy u IPv6 broadcast máme a záleží pouze na vás, jestli zprávy poslané s cílovou adresou FF02::1 nazvete broadcast nebo multicast provozem.

IPv6 překlad IPv6 - MAC

Víme už jak multicast obecně pracuje a jak lze provést jeho efektivní doručení v síti. Pojďme se nyní ještě podívat na část protokolu IPv6, která se podstatně změnila oproti IPv4 - překlad mezi IP a MAC adresou. IPv4 využívá k tomuto účelu protokol ARP definovaný v RFC 826. Dotazy protokolu ARP využívají broadcast a jsou jednoduše rozposílány na všechna zařízení v lokální síti. Odpovědi využívají unicast a jsou tedy zasílány zpět přímo žadateli.

IPv6 protokol používá pro překlad mezi IPv6 a MAC adresou protokol ND (*Neighbor Discovery protocol*) definovaný v RFC 4861, kterým jsme se již zabývali v předchozích dílech seriálu. Z protokolu ND se využívá zpráv *Ohlášení souseda*(*Neighbor Advertisement*) a *Výzva sousedovi* (*Neighbor Solicitation*). Tyto zprávy jsou zasílané do speciální IPv6 multicast skupiny *solicited-node*.

Jakým způsobem to pak celé funguje? Každé zařízení si nejprve pro každou svou IPv6 adresu vytvoří vlastní multicast skupinu *solicited-node*. Multicast adresa dané skupiny vzniká tak, že k multicast prefixu FF02:0:0:0:0:1:FF00::/104 se přidá spodních 24 bitů IPv6 adresy, kterou si chce zařízení nakonfigurovat. Ilustrujme si celý postup na příkladu na následujícím obrázku.



Zařízení si chce nakonfigurovat IPv6 adresu 2001:67c:1220:80e:3f89:4943:87c5:7cc. Spodních 24 bitů této adresy, tedy v hexa zápisu 0xC507CC se přidají k multicast prefixu určený pro *solicited-node* skupinu a vznikne tak multicast adresa ff02::1:ffc5:7cc. Tato IPv6 multicast skupina se přemapuje podle již nám známých pravidel na multicast MAC adresu 33:33:ff:c5:07:cc.
Pokud chce nějaké zařízení zjistit, jaká MAC adresa odpovídá IPv6 adrese 2001:67c:1220:80e:3f89:4943:87c5:7cc, pošle dotaz *Neighbor Solicitation*právě do multicast skupiny ff02::1:ffc5:7cc. Jelikož zařízení odebírá zprávy zasílané do této skupiny, může mu zprávou *Neighbor Advertisement* odpovědět, jaká je jeho MAC adresa. Zařízení si danou informaci poznamenají do cache sousedů a mohou spolu začít komunikovat. Použitím *solicited-node* multicast skupiny je zajištěno, že dotazem nebudou obtěžována všechna zařízení v síti, ale pouze dané konkrétní zařízení. Může se stát, že se spodních 24 bitů bude u několika IPv6 adres shodovat. V tom případě se budou shodovat i jejich multicast skupiny. To ale vcelku ničemu nevadí, jelikož síť zajistí doručení oběma zařízením. Obě předají dotaz svému operačnímu systému, ale pouze to zařízení, které danou IPv6 adresu opravdu má, odpoví. Oproti protokolu IPv4 tady můžeme nalézt vylepšení v tom, že dotaz na překlad adresy nejde všem zařízením, ale pouze těm konkrétním, které IPv6 adresu opravdu mají.
Je důležité ale připomenout následující požadavky. Pokud si zařízení konfiguruje IPv6 adresu, vytvoří si k ní odpovídající *solicited-node* multicast skupinu, jak jsme si popsali výše. Zařízení se pak ale také musí do této skupiny přihlásit pomocí protokolu MLD. To je právě z důvodu, aby byly informovány přepínače v lokální síti, že na daném portu je zařízení, které je členem dané multicast skupiny. Pokud chceme aby správně fungoval překlad IPv6 - MAC adres, musíme tedy zajistit i správné fungování multicast provozu.

Zneužití MLD

Protokol MLD je zajímavý také pro útočníky, protože se jedná o protokol v lokální síti, který v principu nemůže být moc filtrován a neobsahuje žádné bezpečnostní mechanismy. Protokol je také zajímavý tím, že je u operačních systémů implicitně aktivován. Pro útočníka se tak otevírá další kanál, kterým může se zařízením komunikovat. Protokol MLD se tedy dá použít k několika neplechám:

1. Mapování sítě

Pokud si útočník dělal zálusk na nějakou IPv4 síť, proskenoval ji adresu po adrese, aby zjistil aktivní zařízení. Protokol IPv6 ale používá pro adresaci zařízení v lokální síti typicky 64 bitů. To je potenciálně natolik velké množství adres, že je jejich skenování adresu po adrese nemyslitelné. Existuje nicméně pár triků, které může útočník použít. Pokud je připojený v lokální síti, lze vyzkoušet ping na adresu všech zařízení (ff02::1). Tento způsob nicméně naráží na problém, že některá zařízení (typicky Windows) na něj neodpoví. Pro oskenování sítě lze ale vcelku triviálně použít protokol MLD. Pokud útočník do sítě pošle zprávu MLD *Query*, zařízení v lokální síti se budou moci přetrhnout, aby ho informovala do jakých multicast skupin jsou přihlášené. Útočník tak zjistí nejenom jejich *link-local* adresy, ale také lze z odpovědi zjistit nebo alespoň vytušit operační systém - např. Windows se přihlašuje do multicast skupiny FF02::1:3 (Link Local Multicast Name Resolution). Pokud znáte *link-local* adresy, můžete se začít pokoušet zjišťovat, jaké služby zařízení provozuje, protože implicitní nastavení serveru je, že naslouchá na všech IPv6 adresách - tedy i na *link-local*. Tento problém se týká hlavně datových center, kde často nemají poskytovatelé oddělenou síť pro jednotlivé virtuální stroje, jak popisuje například tento článek.

2. Odepření příjmu multicast provozu

Pokud je v síti šířen např. televizní stream pomocí skupinového vysílání, může se útočník pokusit odepřít příjem těchto dat ostatním zařízením. Realizace tohoto útoku je vcelku jednoduchá a velice podobná jako u protokolu IGMP pro IPv4. Útočník se pokusí přesvědčit ostatní zařízení, že je *Querier* (tj. směrovač odpovědný za přeposílání multicast provozu). Podle RFC se *Querier*stává ten, kdo má nejmenší IP adresu. Díky tomu, že IPv6 adresa se většinou u směrovačů generuje na základě linkové adresy podle algoritmu EUI-64, je vcelku triviální si nakonfigurovat adresu nižší a stát se tak *Querier* pro daný segment. Pokud by vás tato problematika zajímala podrobněji, můžete se na detaily podívat například v prezentaci MLD Considered Harmful prezentované na konferenci DeepSec.

3. Přetížení síťových zařízení

Největší bolení hlavy síťovým operátorům ale způsobuje zneužití protokolu MLD pro vytížení jejich infrastruktury. Při tomto typu útoku se snaží útočník zneužít právě optimalizací, které přepínače musí provádět, aby byly efektivně schopny doručovat multicast. Již jsme si popsali, že de facto jediným způsobem, jak zajistit opravdové efektivní doručení multicast provozu v lokální síti, je nutnost aktivovat *MLD Snooping* na přepínačích. Aktivováním si zařízení jsou schopny zjistit informace v jakých skupinách a na kterých portech jsou klienti přihlášeni. Pro zařízení to znamená, že všechny MLD zprávy *Report* a *Done* musí přeposílat do svého CPU, aby je byl schopen zpracovat.
Jak takový útok může vypadat? Lze použít Scapy pro vytvoření MLD Query zprávy, kterou je pak možno zaslat do sítě. Například:

```
$ python
>>> from scapy.all import *
>>> a = IPv6(dst='ff02::1',hlim=1)
>>> b = IPv6ExtHdrHopByHop(nh=58,options=RouterAlert())
>>> c = ICMPv6MLQuery(mrd=0)
>>> mldquery = a/b/c
>>> send(mldquery, inter='x')
```

Jako cílovou adresu nastavíme "broadcast", tedy všechny klienty v síti. Dle RFC mají mít multicast zpráv nastavený Hop Limit (TTL) na 1. Vložíme rozšířenou hlavičku Hop-by-Hop s volbou Router Alert, kterým informuje směrovač, že by danému paketu měl věnovat pozornost. Pak stačí doplnit zprávu ICMPv6 MLD Query a zaslat do sítě. Na tuto zprávu odpoví všechna zařízení v síti připojená a nahlásí tazateli své multicast skupiny do kterých jsou přihlášená. Stačí pak tuto zprávu poslat v cyklu a zařízení se mohou dost zapotit. Pro představu, cca 3000 těchto paketů za vteřinu zcela vytíží Cisco Catalyst 3650 a udělá z něj vcelku nepoužitelné zařízení. V reálné síti bude dopad takovéto zprávy ještě větší. Zařízení se totiž přihlašují do několika skupin a zpráva pro přihlášení do multicast skupiny se dle RFC má zaslat dvakrát. Amplifikace takového útoku je pak značná.
Řešením daného problému může být filtrace zpráv MLD Query na přístupovém portu, pokud vám to vaše zařízení podporuje. ACL může pak vypadat například takto:

```
Switch(config)# ipv6 access-list DENY-MLD
Switch(config-ipv6-acl)# deny icmp any any mld-query
```

Dané ACL zabrání nicméně pouze výše zmíněnému útoku. Útočník se ale může přihlašovat do náhodně generovaných multicast skupin a potom je filtrace podstatně obtížnější. Pokud použije prefix pro *solicited-node* multicast skupiny, tak správce navíc nemůže rozhodnout, která adresa je validní a která ne. Velký počet multicast skupin však nemusí být jen známkou útoku, ale zcela normálním stavem. Pojďme se podívat, kolik takových skupin můžeme v reálné síti očekávat při běžném provozu.

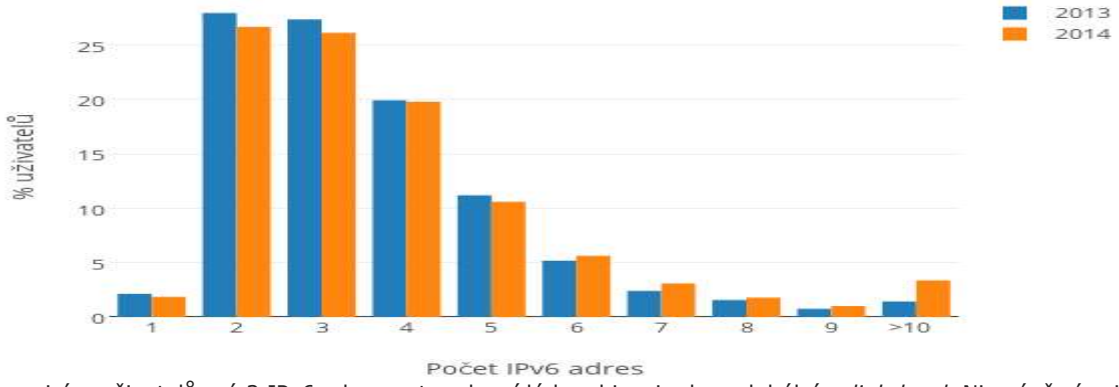
Kolik skupin?

Počet multicast skupin v IPv6 síti primárně závisí na způsobu adresace koncových zařízení. IPv6 protokol vyžaduje, aby každé zařízení mělo *link-local* adresu pro komunikaci s ostatními zařízeními na lince. Koncept *link-local* adres je u IPv6 vcelku novinkou. Protokol IPv4 tyto adresy sice měl také (169.254.0.0/16) nicméně ty se používají pouze když selžou jiné způsoby konfigurace IPv4 adresy. Oproti tomu, *link-local* adresy jsou u IPv6 opravdu třeba, jelikož se používají jako adresa výchozí brány, směrovače je používají pro výměnu směrovacích informací aj. Jelikož je *link-local* IPv6 adresa jako každá jiná, zařízení si tedy pro ni musí vytvořit odpovídající multicast adresu.

Pouze s *link-local* adresou bychom ale daleko nedošli, jelikož dosah *link-local*adresy je omezen pouze k nejbližšímu směrovači. Nelze se tedy pomocí těchto adres dostat na zdroje v Internetu. K tomu potřebujeme, aby zařízení mělo přidělenou ještě globální IPv6 adresu, tedy další multicast adresu. Pokud používáme v síti ULA adresy (*Unique Local IPv6 Unicast Addresses)* pro přístup k lokálním službám, máme další multicast adresu. Počet adres nám může narůst také při použití více prefixů, nebo když kombinujeme automatickou konfiguraci s DHCPv6. Vždy tedy musíme počítat s tím, že v síti budeme mít **minimálně**dvojnásobek multicast skupin jako je připojených zařízeních v daném segmentu sítě, často více.

Historicky tomu tak ale nebylo a zařízení měla typicky pouze jednu multicast adresu. Proč? Původní návrh protokolu IPv6 totiž počítal s tím, že zařízení si IPv6 adresu vytvoří na základě prefixu sítě a své MAC adresy. Všechny adresy, které by si koncové zařízení přidělilo (globální, *link-local*, ULA) by tedy měly stejný koncový identifikátor. Díky tomu by byla stejná i multicast adresa. Koncept vytvoření identifikátoru z MAC adresy se ale znelíbil, jelikož nezaručuje dostatek privátnosti a vznikly tak náhodně generované IPv6 adresy (RFC 4941 a RFC 7217). Každá IPv6 adresa má tedy v dnešní době spodních 32 bitů rozdílných, tedy pro každou adresu se vytvoří unikátní multicast skupina.

Jaká je realita v koncových sítích? Kolik adres a tedy multicast skupin můžeme očekávat? Podívali jsme se podrobněji na přibližně 5500 unikátních uživatelů v naší síti. Následující graf nám pak zobrazuje kolik má jeden uživatel typicky unikátních IPv6 adres za jeden den. Měření jsme provedli v letošním roce a pro porovnání je zobrazen i rok minulý.




Vidíme, že nejvíce uživatelů má 2 IPv6 adresy - to odpovídá kombinaci adres globální + *link-local*. Nicméně více jak 70% uživatelů má adres více. Proč? Každý restart operačního systému nebo reasociace ve WiFi síti u mobilního telefonu vede typicky k vygenerování nové IPv6 adresy. Průměr je pak přibližně 3.6 adres na uživatele za den v roce 2013 a víc jak 4 adresy na uživatele za den v roce 2014. Podstatně také narůstá maximální počet IPv6 adres. Zatímco před rokem jsme se většinou nepřehoupli přes 20 adres na uživatele, dnes není problém nalézt uživatele, který má za den více jak 60 unikátních IPv6 adres. Z pohledu síťových zařízeních je tedy kladen stále větší důraz na velikost paměti (cache sousedů - *Neighbor cache*), která je určena k uchování těchto záznamů. Této problematice se budeme věnovat podrobněji v příštím díle seriálu. Nás z pohledu dnešního článku spíše zajímá, kolik můžeme čekat multicast skupin a zde nám z dat vyplývá, že je to přibližně 4x více než je uživatelů v lokální síti. Pokud tedy máte 1000 uživatelů v jedné L2 síti, můžete počítat s přibližně 4000 unikátními multicast adresami.

Škálovatelnost MLD

Jak se protokol IPv6 začíná více prosazovat, vychází najevo také několik provozních komplikací. O zábavu se síťovým operátorům v nedávné době postarala společnost Intel a její síťové karty viz. Intel fórum, Cisco mailing list nebo blog. Problémem bylo, že síťové karty, po přepnutí do sleep mode, posílaly do sítě větší množství IPv6 MLD zpráv a došlo tak k přetížení přepínačů. Zde byl na vině pouze špatně naprogramovaný driver pro síťovou kartu a společnost vydala v zápětí opravenou verzi. Dalo by se tedy říci, že to byla pouze chyba v implementaci, nicméně to není zas tak jednoduché.

Pokud by se IPv6 multicast používal pouze pro přeposílání například televizního provozu, nebyl by to problém, protože situace by byla stejná jako u IPv4. Protokol IPv4 sice multicast v lokálních sítích využívá také a jsou protokoly, které dokážou zaslat velké množství IPv4 multicast zpráv (např. mDNS nebo Link-local Multicast Name Resolution), nicméně pokud je pro zařízení problém, lze je jednoduše zablokovat. Multicast je ale u IPv6 použit téměř všude, zejména pro zjištění mapování mezi IPv6 a MAC adresou, jak jsme si popsali výše. Zde ale leží zakopaný pes, jelikož použití pro tyto účely přináší ve větších sítích komplikace. Mohou tak nastat situace, kde zařízení nejsou schopna zpracovat tak velké množství multicast skupin. V dnešní době totiž typická topologie v enterprise nebo kampus sítích vypadá tak, že klienti jsou svedeni na distribuční vrstvu, kde probíhá samotné směrování. Zařízení, které se o dané směrování starají nicméně nejsou a v principu ani nemohou být tak výkonná, aby dokázala zvládnout potenciálně všechny IPv6 *solicited-node* multicast skupiny. Filtrovat dané skupiny nelze, protože IPv6 adresy jsou náhodně generované, tedy nelze daný prostor (16 777 216 adres) nikterak omezit. Příklad z praxe pak lze nalézt třeba u výpadku MIT sítě způsobené právě velkým množstvím IPv6 multicast ND provozu - viz. The network nightmare that ate my week.

Topologie v datových center, kde je velké množství virtuálních strojů svedené do několika málo fyzických tzv. Top of the Rack (ToR) přepínačů je ještě více problematická. Na jednom fyzickém prvku pak může být připojeno několik tisíc virtuálních strojů a daný ToR přepínač pak může být multicast provozem dost vytížen.

Řešení problému

Síťový administrátor má několik možností, jak se poprat s daným problém, nicméně ideální řešení není. Možné řešení jsou následující:

1. Pokud chceme zachovat výhody multicast provozu, musíme mít zapnutý MLD snooping. Jelikož přepínač nedokáže zpracovat tolik multicast skupin, může ignorovat stav multicast skupiny pro *Solicited-node* adresy a bude je přeposílat klasicky jako broadcast. Pokud má síťová karta koncového zařízení dostatečně velkou hardware kapacitu pro odfiltrování nezajímavých dat, pořád je tu výhoda, že operační systém nebude obtěžován provozem, který mu není určen. Toto zjednodušení používají již v dnešní době někteří výrobci. HP nevytváří MLD tabulku pro jakékoliv *link-local* skupiny - tedy skupiny s prefixem FF02::/16 Cisco například stav pro *Solicited-node* skupiny nevytváří, nicméně je stejně musí vyhodnotit, tedy i s tímto vylepšením ho lze jednoduše vytížit. Problém nicméně zůstal u virtuální infrastruktury, kdy Hyprevisor stále musí zpracovávat velké množství skupin a filtry síťových karet jsou v tomto zatím nedostatečné.
2. Nepoužívat MLD snooping je další variantou, jak problém vyřešit. Zcela tím eliminujeme výhody efektivního doručení paketů, nicméně infrastruktura bude o trochu stabilnější. Toto řešení ale není vhodné použít, pokud je multicast využíván pro šíření většího objemu dat.
3. Upravit specifikaci protokolu, aby nevyužívala multicast pro překlad IPv6 adresy na MAC adresu. Toto řešení nicméně není na pořadu dne. I pokud by se v budoucnu použilo, potrvá několik let, než by se tato optimalizace dostala do koncových systémů. Reálně tedy v dnešní době nic neřeší.
4. Omezit množství multicast/broadcast dat, které je zasláno do CPU přepínače, pokud to zařízení podporuje. Nevýhodou zde je, že může dojít k zahození relevantního provozu.
5. Rozdělit architekturu sítě na menší celky použitím více směrovačů/L3 přepínačů. Toto může pomoci proti problémům s velkým množstvím klientů, nicméně to nevyřeší problém cíleného útoku.

Multicast a WiFi

Zcela samostatnou kapitolou je problematika multicast vysílání IPv6 v bezdrátových sítích. Připomeňme si základní charakteristiky bezdrátových sítí.

Zásadním rozdílem oproti drátové nebo optické infrastruktuře je, že bezdrátové sítě fungují v režimu half-duplex, tedy pouze jedno zařízení může vysílat, ostatní v danou chvíli mohou pouze naslouchat. Zařízení jsou také typicky v různých vzdálenostech od AP (Access Point) a používají tak rozdílné rychlosti a způsob kódování. Zjednodušeně řečeno, čím je zařízení dál od AP, tím používá robustnější kódování a tedy i nižší přenosovou rychlost.

Pokud AP zasílá uživatelům multicast rámce, snaží se je zaslat co nejspolehlivěji tak, aby došla na všechna, i vzdálená, zařízení - tedy zasílá je co nejmenší rychlostí. Přenosová rychlost je pak typicky 1Mbps nebo 6Mbps. Zaslání jednoho multicast rámce tak časově blokuje médium po dlouhou dobu a může zabrat tolik času, jako odeslání několika unicast rámců. Jelikož ve WiFi sítích nejsou multicast a broadcast rámce potvrzovány, může také dojít k tomu, že zařízení data nepřijme a je pak záležitost vyššího protokolu, aby zajistil jejich případné přeposlání.

Dalším charakteristickým znakem bezdrátových sítí je, že jsou používány primárně mobilními zařízeními, případně různými senzory, napájenými baterií, kterou se snaží co nejvíce šetřit. Standard 802.11 pak umožňuje, aby zařízení sdělilo AP, že se uspí. Pokud AP registruje pro uspané zařízení nějaká unicast/multicast data, uchová si je a přepošle je danému zařízení, až se znovu probudí. Tento mechanismus může značně prodloužit životnost daného zařízení na baterii. Spotřeba zařízení se pak může pohybovat okolo 10 mA při uspání a něco mezi 100 až 150 mA při přijetí multicast rámce, jak zdokumentoval Yoann Desmouceaux v draftu Power consumption due to IPv6 multicast on WiFi devices u zařízení Samsung i9195.

Protokol IPv6, který opírá většinu své signalizace o multicast, pak dokáže v bezdrátové síti způsobit pár nepříjemností. Této signalizační komunikace navíc není zcela málo. Zařízení si totiž generuje více IPv6 adres, kde pro každou musí ověřit její unikátnost a přihlásit se do odpovídajících multicast skupin. Jedná se typicky o desítky multicast paketů s každým připojením zařízení do WiFi sítě. Problém také je, že zprávy *Ohlášení směrovače* jsou zasílané všem zařízením v síti. Jelikož typický scénář při připojení zařízení je, že si pomocí zprávy *Výzva směrovači*vyžádá konfigurační parametry, daných zpráv může být ve velké síti velké množství. Protokol NDP, který zaštiťuje všechny tyto konfigurační IPv6 zprávy navíc není příliš připraven na to, že se některé pakety ztratí. Ztrátovost ale ve WiFi sítích není zas tak výjimečná věc a takové ověření unikátnosti IPv6 adresy pak může poněkud "drhnout".

Již zmíněný draft, také analyzuje dopad IPv6 multicast vysílání na zařízení ve WiFi sítích a dochází k závěrům, že už u 30 zařízení v síti se spotřeba baterie vlivem IPv6 multicast dat zdvojnásobuje. U větších sítích (600 uživatelů) může být až 15x větší.

Jak z této šlamastyky ven? V současné době není příliš mnoho kontrolérů, které by podporovaly nějaké formy optimalizace IPv6 multicast komunikace. Cisco umožňuje aktivovat techniku *RA throttling*, která omezuje množství zpráv *Ohlášení směrovače*. Těch totiž může být ve velkých WiFi sítí opravdu hodně, protože si každé WiFi zařízení žádá o zaslání konfiguračních parametrů při každém připojení. Princip *Ra throttling* je pak vcelku jednoduchý - AP propustí pouze zprávu co definovaný interval. Další optimalizace - NDP proxy na AP, zasílání *Ohlášení směrovače* pomocí unicast zpráv atp., popisuje draft Reducing Multicast in IPv6 Neighbor Discovery. Tématem se také zabývá například Eric Vyncke ve svéprezentaci na IPv6 Council Belgium, případně Christopher Werny ve své prezentaci.

Tyto optimalizace jsou zatím většinou ve stádiu diskuzí a nenajdete moc zařízení, které by je podporovalo. Některé optimalizace také vyžadují změnu standardu, což je sám o sobě velice dlouhý proces. V případě, kdy se sahá na integrální součásti IPv6 jako navrhují nynější optimalizace, tak diskuze bude o to delší. Reálné implementace tak jsou zatím v nedohlednu.

Závěr

IPv6 multicast měl zefektivnit komunikaci klientů a zabránit vytížení koncových zařízení, jak byli navrhovatelé protokolu svědky v 90. letech. Proto se multicast komunikace u IPv6 použila pro celou řadu obslužných činností. Poněkud se ale zapomnělo na fakt, že efektivní doručení multicast dat musí podporovat samotná infrastruktura. Optimalizace přeposílání multicast dat stojí ale přepínače a směrovače určitý výkon, což ve větších sítích nebo datacentrech může vést k problémům s jejich přetížením. Vzniká zde totiž velký počet IPv6 multicast skupin, jelikož IPv6 zařízení si typicky adres konfiguruje několik. Síťový administrátor si tedy musí dát pozor zejména na to, aby znemožnil útočníkovi využít protokol MLD pro vedení DoS útoků a také aby vytvořil dostatečně robustní infrastrukturu, která bude schopna pojmout velké množství IPv6 multicast adres. V současné době připadá na koncové zářízení minimálně dvě, průměrně čtyři multicast adresy. Částečně by daný počet mohla snížit implementace RFC 7217, která doporučuje vytvářet náhodné identifikátory s vazbou na síťový prefix. Zařízení by si tedy ve stejné síti mělo vždy vygenerovat stejnou náhodnou adresu. Počet multicast skupin by se pak zmenšil, nicméně implementací zatím nedisponuje žádný operační systém. Implementaci lze nicméně vyzkoušet v podobě userspace aplikace dhcpcd. I v tomto případě je ale třeba počítat s několika IPv6 adresami na jedno zařízení, protože dané RFC neřeší problematiku dočasných IPv6 adres. Zcela novou sérii problémů přinesl multicast do WiFi komunikace. Z bezpečnostního hlediska se však nejedná o nové bezpečnostní hrozby. Přemíra multicast komunikace v IPv6 se tak projevuje pouze v neefektivní komunikaci a rychlejším "vysáváním" baterek.

**Bezpečné IPv6: když dojde keš**

Dříve než si popíšeme vlastní útoky a pokusíme se najít způsoby jejich eliminace se podíváme, co je to cache sousedů (*Neighbor Cache*) a jak v ní záznamy vůbec vznikají. Cache sousedů je datová struktura, kterou si udržuje každý IPv6 uzel. Obsahuje, mimo jiné, dvě klíčové položky, a to IPv6 adresu a jí odpovídající linkovou (MAC) adresu. Pro zjištění tohoto mapování mezi IPv6 a MAC adresami se používá dvojice zpráv *Výzva sousedovi* (*Neighbor Solicitation*) a *Ohlášení souseda*(*Neighbor Advertisement*),

které jsou součástí protokolu NDP, se kterým jsme se už potkali v předchozích dílech seriálu. V naprosté většině případů vznikají záznamy v cache sousedů automaticky a z pohledu komunikující aplikace zcela transparentně.

Připomeňme si, k čemu toto mapování vůbec potřebujeme. Pokud chce IPv4 nebo IPv6 uzel komunikovat s jiným uzlem, musí si v první řadě zodpovědět na následující otázky.

1. Můžu druhý uzel kontaktovat přímo? Nachází se ve stejné L2 síti jako já?
2. Pokud ne, znám směrovač, přes který mu můžu data poslat?

Ve světě IPv4 se zařízení všechny potřebné informace dozvědělo statickou konfigurací, nebo protokolem DHCPv4. Zejména pak IPv4 adresu, prefix sítě a výchozí bránu. Zjištění, jestli se adresa dalšího zařízení nachází ve stejné síti, se pak určí vcelku jednoduše tak, že se použije binární operace AND a musí platit, že zdrojová-IP AND prefix sítě == cílová-IP AND prefix sítě. Pokud rovnice platí, lze zařízení kontaktovat přímo a pro zjištění jeho linkové adresy se použije protokol ARP. Pokud rovnice neplatí, zařízení, které chceme kontaktovat, se nachází v jiné síti. Pakety se tedy pošlou na adresu výchozí brány, jejíž linkovou adresu zjistí zařízení také pomocí ARP.

Protokol IPv6 na to jde nicméně trochu jinak. Rozděluje zařízení na ta, která se nachází na stejné lince (*on-link*) a ostatní (*off-link*). Pod pojmem linka si zde můžete představit jakékoliv médium, přes které se dá komunikovat na linkové úrovni - například Ethernet nebo PPP.

Jak se určí, že jsou zařízení připojená na stejné lince? Tuto informaci zařízením sdělí směrovač ve své zprávě *Ohlášení směrovače*. Jak jsme si popsali v prvním díle seriálu, v této zprávě se zasílá informace o IPv6 prefixu, který se v síti používá, a ze kterého si zařízení mají vygenerovat IPv6 adresu. Pokud je pro daný IPv6 prefix nastaven příznak *on-link*, zařízení připojená do dané sítě (s daným prefixem) spolu mohou komunikovat přímo. Tedy pomocí zpráv *Výzva sousedovi* a *Ohlášení souseda* zjistit mapování mezi svými IPv6 a linkovými adresami. Pokud u prefixu příznak nastaven není, předpokládá se, že zařízení je *off-link*. Provoz pak musí zařízení zaslat na výchozí bránu.

Pokud byste dumali nad tím, kdy má smysl do sítě šířit IPv6 prefix a zakázat komunikaci zařízení mezi sebou, tak to dává smysl třeba v NBMA (Non-broadcast multiple-access) sítích, kde komunikace probíhá vždy přes centrální směrovač. Tento koncept lze také teoreticky použít pro separaci klientů v lokální síti - provoz musí totiž jít vždy přes směrovač. Zde je nicméně problém v implementacích, kdy ne všechny na tento příznak reagují správně. Zájemce o podrobnější informace lze odkázat na RFC 5942 a RFC 4943.

My se nyní podíváme na ty nejklasičtější varianty komunikace, které můžeme v IPv6 síti dnes nalézt. Když spolu chtějí komunikovat dvě zařízení na stejné lince, nebo chcete-li, ve stejné síti, a když zařízení posílá data do Internetu.

Při komunikaci v rámci stejné linky odesílající uzel zašle do sítě zprávu *Výzva sousedovi*. Pokud vše funguje jak má, cílový uzel zprávu zpracuje a tázajícímu uzlu odpoví prostřednictvím zprávy *Ohlášení souseda*. Kromě dotazované IPv6 adresy do odpovědi přibalí ještě svou adresu linkové vrstvy (MAC adresu). Na základě této zprávy si tázající uzel zařadí příslušnou IPv6 adresu a jí odpovídající MAC adresu do cache sousedů, kde si je ponechá po určitý čas.

V případě, že chce uzel zaslat data mimo svou síť, vyhledá si podle směrovací tabulky IPv6 adresu odpovídajícího směrovače nebo výchozí brány a dále postupuje stejným způsobem jak v předchozím případě. Výsledkem je, že se v cache sousedů objeví IPv6 a MAC adresa směrovače. Na následujícím obrázku je zachycen právě takový průběh komunikace koncového zařízení v případě, že cache sousedů je na samotném zařízení i na připojujícím směrovači prázdná. Na koncovém zařízení je spuštěn příkaz ping6 www.cesnet.cz. Před vlastním vysláním ICMPv6 paketu *Echo Request* musí ale ještě proběhnout převod doménového jména na IPv6 adresu (pakety č. 3 a 4). Tomu však předchází zjišťování linkové adresy směrovače (pakety 1 a 2). Pro pakety s odpovědí již směrovač nemusí zjišťovat linkovou adresu cílového uzlu, protože všechny informace, které potřebuje, již získal z předchozí komunikace. Pro jistotu však po několika vteřinách provede ověření, zda je příslušná IPv6 adresa skutečně dostupná (pakety 7 a 8).

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 2001:67c:1220:f777::2 | ff02::1:ff75:b3b4 | ICMPv6 | 86 | Neighbor Solicitation for fe80::d27e:28ff:fe75:b3b4 from 00:50:56:94:b2:e1 |
| 2 | 0.003055 | fe80::d27e:28ff:fe75:b3b4 | 2001:67c:1220:f777::2 | ICMPv6 | 86 | Neighbor Advertisement fe80::d27e:28ff:fe75:b3b4 (rtr, sol, ovr) is at d0:7e:28:75:b3:b4 |
| 3 | 0.003074 | 2001:67c:1220:f777::2 | 2001:67c:1220:e000::100 | DNS | 93 | Standard query 0x7cf9 AAAA www.cesnet.cz |
| 4 | 0.003353 | 2001:67c:1220:e000::100 | 2001:67c:1220:f777::2 | DNS | 293 | Standard query response 0x7cf9 AAAA 2001:718:1:101::4 |
| 5 | 0.003516 | 2001:67c:1220:f777::2 | 2001:718:1:101::4 | ICMPv6 | 118 | Echo (ping) request id=0x6c63, seq=1 |
| 6 | 0.007288 | 2001:718:1:101::4 | 2001:67c:1220:f777::2 | ICMPv6 | 118 | Echo (ping) reply id=0x6c63, seq=1 |
| 7 | 4.730436 | fe80::d27e:28ff:fe75:b3b4 | 2001:67c:1220:f777::2 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777::2 from d0:7e:28:75:b3:b4 |
| 8 | 4.730478 | 2001:67c:1220:f777::2 | fe80::d27e:28ff:fe75:b3b4 | ICMPv6 | 78 | Neighbor Advertisement 2001:67c:1220:f777::2 (sol) |

V případě, že zjišťování linkové adresy sousedního uzlu nebo směrovače neproběhne úspěšně do určitého času, je do tabulky sousedů uložen záznam s příznakem, že uvedený cílový uzel nebo směrovač je nedostupný.

Pokud bychom hledali obdobný mechanismus v IPv4 tak odpovídajícím ekvivalentem cache sousedů je ARP tabulka a zprávy *Neighbor Solicitation* a *Neighbor Advertisement* odpovídají zprávám *ARP Request* a *ARP Reply* protokolu ARP. Je zde však několik odlišností. První jsme již zmínili - rozdělení na *on-link*, *off-link*. Mezi další odlišnosti patří, že zprávy *výzva sousedovi* nejsou posílány na broadcast, ale zasílány do multicast skupiny a že samotný protokol NDP je součásti ICMPv6 a nikoliv samostatným protokolem nad linkovou vrstvou, jak tomu bylo v případě protokolu ARP. Pro názornost zachycují následující obrázky obsah části tabulky ARP a cache sousedů na tomtéž zařízení.

```
<irf-kn>display arp
                Type: S-Static     D-Dynamic     M-Multiport
IP Address       MAC Address     VLAN ID  Interface                Aging  Type
147.229.188.165  0050-56ad-78af  300      BAGG1                    20     D
147.229.196.2    0050-56ad-78ee  220      BAGG1                    19     D
147.229.191.167  0050-568f-0001  200      BAGG1                    20     D
147.229.200.2    0050-56ad-78ee  310      BAGG1                    20     D
147.229.205.2    0050-56ad-78ee  320      BAGG1                    20     D
```

U ARP tabulky vidíme mapování mezi IPv4 adresou a MAC adresou, na jakém rozhraní, případně v jaké VLAN je daná IP naučena a jak dlouho již v tabulce je. Cache sousedů u protokolu IPv6 vypadá o trochu jinak.

```
<irf-kn>display ipv6 neighbors all
                Type: S-Static     D-Dynamic
IPv6 Address                      Link-layer     VID  Interface       State T Age

FE80::250:56FF:FE93:11            0050-5693-0011 230  BAGG2           STALE D 1322
2001:67C:1220:C1B2:250:56FF:FE    0050-5693-0011 230  BAGG2           STALE D 1322
93:11
FE80::883C:1982:63A4:3F0A         90e6-ba40-4f67 220  BAGG109         REACH D 0
FE80::5CEB:5298:235C:5816         206a-8a5c-888a 210  BAGG109         REACH D 0
FE80::9A4B:E1FF:FE4E:C603         984b-e14e-c603 220  BAGG109         STALE D 30
2001:67C:1220:C1B1:70BA:84D3:1    0026-187f-fb7e 220  BAGG109         PROBE D 3637
E22:C162
FE80::E1EC:3178:F25:8886          1803-736d-5753 220  BAGG109         DELAY D 56
FE80::FCC1:7DD4:BC92:F268         28d2-440c-3cca 220  BAGG109         DELAY D 51
```

Vidíme, že většina informací zůstala zachována. Přibylo ale další políčko, které popisuje stav dané adresy. Z výpisu vidíme, že se adresy v cache mohou nacházet v několika stavech - dosažitelná (*Reachable*), prošlá (*Stale*), odložená (*Delay*) a testovaná (*Probe*).

Pokud daná adresa odeslala v poslední době nějaká data, je brána jako dosažitelná. Pokud vyprší doba, po kterou lze adresu považovat za dosažitelnou, adresa přejde do stavu prošlá, nicméně v cache sousedů stále zůstává. Pokud je adresa ve stavu prošlá a je třeba ji odeslat nějaká data, data se odešlou, adresa přejde do stavu odložená a čeká se, zda dosažitelnost nepotvrdí vyšší vrstva. Pokud ne, adresa přejde do stavu testovaná a směrovač se aktivně snaží dosažitelnost ověřit.

Detailně je celý proces objevování sousedů popsán v knize Pavla Satrapy.

Seznámili jsme se s cache sousedů - k čemu je a jak se plní. Nyní se podíváme, jak může být zneužitá k záškodnické činnosti.

Lokální útok

První varianta útoku předpokládá, že je útočník ve stejné podsíti jako oběť. Útočník má v tomto případě mnohem snažší přístup k signalizačním protokolům - zejména k mechanismu objevování sousedů. Princip útoku lze jednoduše popsat takto: Postupným generováním dotazů a podvržených odpovědi vytvoříme některému z okolních uzlů (zpravidla připojujícímu směrovači) iluzi, že v síti je připojeno velké množství koncových zařízení. To způsobí, že uzel vyčerpá veškeré zdroje určené pro cache sousedů (bude ji mít zaplněnou) a to v konečném důsledku způsobí, že do této tabulky nebude možné zařazovat adresy nově připojených zařízení. V úplně základní podobě si vystačíme s velice jednoduchou variantou útoku, která využívá prostého generování zpráv *Výzva sousedovi*. V případě, že zařízení obdrží zprávu *Výzva sousedovi*, zařadí si příslušnou adresu do tabulky sousedů ve stavu *DELAY*(odložená) a počká, zda vyšší vrstva provede skutečné ověření dosažitelnosti příslušné adresy.

Pro simulaci útoku lze použít nástroj *flood_solicitate6* z balíku THC-IPV6, který zajistí generování zpráv *Výzva sousedovi*.

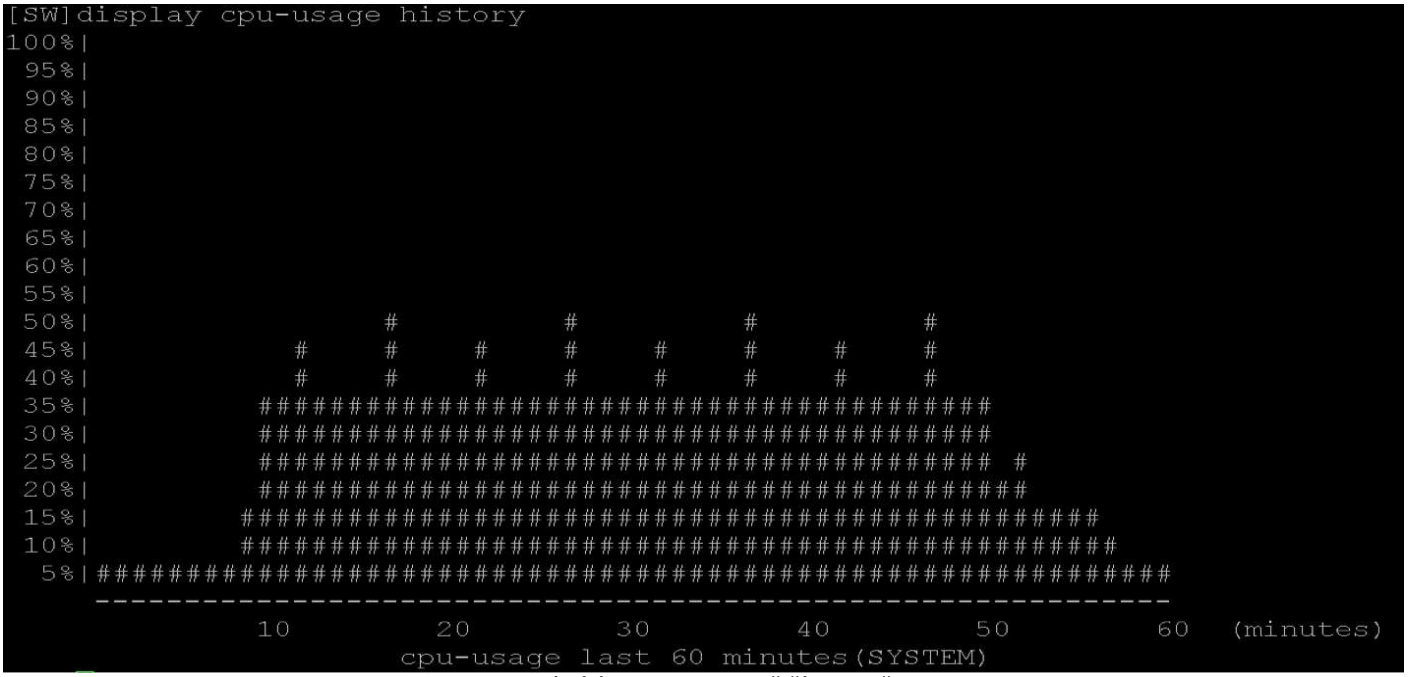
```
[root@test8 thc-ipv6-2.5]# ./flood_solicitate6 -X eth1
FE80::D27E:28FF:FE75:B3B4
Starting to flood network with neighbor solicitations on eth1 (Press
Control-C to end, a dot is printed for every 1000 packets):
.....
.....
..
```

Jak probíhá vlastní výměna paketů je zachyceno na následujícím obrázku:

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | fe80::218:21ff:fee6:d09d | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:21:e6:d0:9d |
| 2 | 0.000008 | fe80::218:19ff:fed8:1a96 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:19:d8:1a:96 |
| 3 | 0.000015 | fe80::218:5fff:fee3:c687 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:5f:e3:c6:87 |
| 4 | 0.000022 | fe80::218:72ff:fe0f:d15d | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:72:0f:d1:5d |
| 5 | 0.000029 | fe80::218:fbff:fecf:d52d | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:fb:cf:d5:2d |
| 6 | 0.000036 | fe80::218:a4ff:fe5b:e8db | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:a4:5b:e8:db |
| 7 | 0.000043 | fe80::218:f6ff:fe25:eea0 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:f6:25:ee:a0 |
| 8 | 0.000050 | fe80::218:e4ff:fef8:e305 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:e4:f8:e3:05 |
| 9 | 0.000057 | fe80::218:deff:feb4:a2f7 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:de:b4:a2:f7 |
| 10 | 0.000065 | fe80::218:8cff:febc:8deb | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:8c:bc:8d:eb |
| 11 | 0.000072 | fe80::218:a0ff:fe53:7312 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:a0:53:73:12 |
| 12 | 0.000079 | fe80::218:62ff:fe44:6f5e | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:62:44:6f:5e |
| 13 | 0.000086 | fe80::218:14ff:fe44:8bb8 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:14:44:8b:b8 |
| 14 | 0.000093 | fe80::218:a0ff:fe73:9496 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:a0:73:94:96 |
| 15 | 0.000101 | fe80::218:98ff:fe82:367c | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:98:82:36:7c |
| 16 | 0.000113 | fe80::218:7aff:fe1a:8158 | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:7a:1a:81:58 |
| 17 | 0.000127 | fe80::218:ceff:fe24:4f5a | ff02::1 | ICMPv6 | 86 | Neighbor Solicitation for 2001:67c:1220:f777:: from 00:18:ce:24:4f:5a |

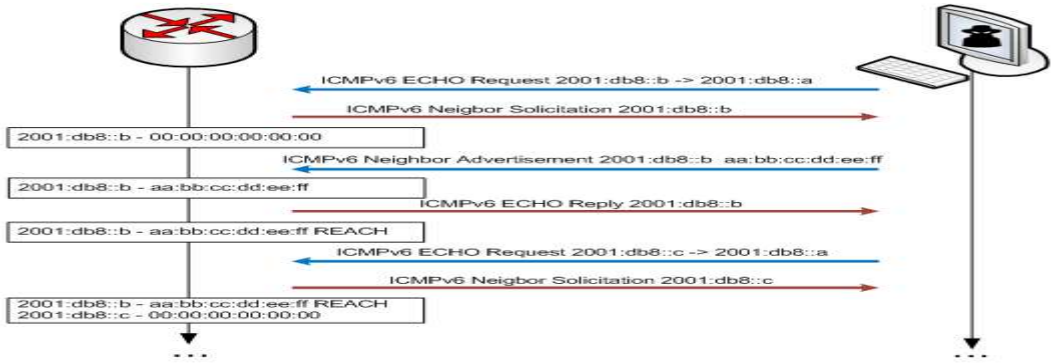
Všechy zprávy jsou zasílány na cílovou adresu ff02::1, tedy "broadcast" (nebo chcete-li multicastová skupina, kterou musejí přijímat všechny IPv6 uzly). Každý paket obsahuje nahodile vygenerovanou zdrojovou IPv6 a MAC adresu, čímž je zajištěno, že s každým paketem teoreticky vznikne nový záznam v cache sousedů. Znalci IPv6 mohou namítnout, že zpráva *Výzva sousedovi* se má správně zasílat na multicastovou adresu uzlu vytvořenou speciálně pro tento účel, jak jsme si popisovali v předchozím díle. Většina implementací IPv6 však neprovádí žádnou následnou kontrolu a takovou zprávu normálně zpracují. Tímto útokem lze tedy zasáhnout všechna zařízení v lokální síti.

Tímto útokem na většině zařízení ovšem nedojde k zaplnění cache sousedů. Je to způsobeno tím, že záznamy zůstanou ve stavu *DELAY* (odložená) případně *PROBE*(testovaná). U záznamů v tomto stavu operační systém po určitém čase, zpravidla po několika desítkách vteřin, provede prověření dostupnosti sousedů zasláním výzvy. Vzhledem k tomu, že na takovou výzvu nedorazí odpověď, příslušný záznam je vyřazen. Vlastní útok se tedy neprojeví v samotném znepřístupnění služeb po protokolu IPv6, ale pouze zvýšenou zátěží všech zařízení připojených v příslušném síťovém segmentu. Následující obrázek zachycuje histogram se zátěží L3 přepínače, který je právě pod výše popsaným útokem. Po spuštění útoku cca v desáté minutě se zvýší zátěž CPU z klidových 5 % cca na 35-40 %.




Lokální útok v rafinovanější podobě

Tímto však útočníkovy možnosti nejsou vyčerpány. Značně zákeřnější formou útoku je varianta, kdy se útočník pokusí přesvědčit zařízení, že vygenerovaná adresa je opravdu dostupná, tedy, že záznam je ve stavu *REACHABLE* (dosažitelná) nebo *STALE* (prošlá). Tento stav indikuje, že dostupnost příslušné adresy v cache sousedů byla již ověřená. Tímto je zaručeno, že záznam zůstane v tabulce po hodně dlouhou dobu, v praxi zpravidla řádově hodiny. Průběh vlastního útoku je zachycen na následujícím obrázku:




Útočník zašle směrovači (2001:db8::a) zprávu, například *ICMPv6 ECHO Request*, kde na pozici zdrojové adresy vloží například adresu 2001:db8::b. Není nezbytně nutné, aby útočník komunikoval přímo se směrovačem, na který hodlá útok vést. Obecně postačí jakékoliv zařízení za směrovačem, které dokáže vygenerovat paket s odpovědí. V našem případě se ovšem přímo směrovač bude pokoušet odpovědět zprávou *ICMPv6 ECHO Reply*. Dříve než bude schopen odeslat paket s odpovědí, musí zjistit linkovou (MAC) adresu souseda a tu si zařadit do cache sousedů. Vytvoří si tedy dočasný záznam pro adresu 2001:db8::b, kde ještě nemá vyplněnou MAC adresu a odešle *Výzvu sousedovi*. Útočník na tuto zprávu odpoví *Ohlášením souseda*. Na základě této výměny zpráv si směrovač vloží do cache sousedů informaci, že k IPv6 adrese 2001:db8::b přísluší linková adresa aa:bb:cc:dd:ee:ff a poznačí si, že tato adresa je dosažitelná (REACH). Tímto je zajištěno, že záznam bude u většiny zařízení ponechán v cache sousedů po výrazně delší dobu. V dalším kroku útočník celou operaci opakuje pro adresu 2001:db8::c, 2001:db8::d, atd.

Pro realizaci takového útoku není potřeba žádných složitých prostředků a lze jej prakticky realizovat na každém linuxovém systému s využitím základních příkazů, tj. ip (ifconfig) a ping6 například takovýmto způsobem:

```
# ip address add 2001:db8::b/64 dev eth1
 # ping6  -I 2001:db8::b -c 1 2001:db8::a
# ip address add 2001:db8::c/64 dev eth1
 # ping6  -I 2001:db8::c -c 1 2001:db8::a
# ip address add 2001:db8::d/64 dev eth1
 # ping6  -I 2001:db8::d -c 1 2001:db8::a
```

Na směrovači pak vzniknou následující záznamy:

```
[SW]display ipv6 neighbors vlan 777
Type: S-Static   D-Dynamic
IPv6 Address              Link-layer     VID  Interface     State  T  Age
FE80::250:56FF:FE94:B2E0  0050-5694-b2e0 777  GE1/0/2       DELAY  D  129
2001:DB8::D               0050-5694-b2e0 777  GE1/0/2       REACH  D  9
2001:DB8::C               0050-5694-b2e0 777  GE1/0/2       REACH  D  9
2001:DB8::B               0050-5694-b2e0 777  GE1/0/2       REACH  D  9
```

## Praktické důsledky útoku

Jaké může mít takový útok praktické důsledky? Vlivem ponechání příslušného záznamu v tabulce se cache sousedů postupně zaplní až do té míry, kdy do ní není možné přidávat další záznamy. Pokud je tento stav vyvolán na připojujícím směrovači, tak v příslušném síťovém segmentu již nebude možné zařadit do cache sousedů záznam pro toto zařízení a tímto se stane IPv6 konektivita pro příslušné zařízení nedostupná. Pokud je navíc na příslušném směrovači, nebo alespoň slotu směrovače, paměť pro cache sousedů sdílená mezi všemi rozhraními (což ve většině případů je), tak jsou současně stejným způsobem postiženy i segmenty připojené na rozhraní sdílející tuto paměť. Pokud by se někdo domníval, že dostatečná velikost cache sousedů, která by byla schopná pojmout všechny možné záznamy, je pouhou implementační drobností, tak jenom připomínáme, že paměť pro cache by musela pojmout více než $2^{64}$ záznamů, což i v minimalistické podobě mnohonásobně přesahuje řády PB. Je tedy zřejmé, že každé zařízení musí mít nějaký konečný limit počtu záznamů v cache sousedů a musí implementovat odpovídající strategii, jak se zachovat v případě, že je tento počet záznamů překročen.

Chování jednotlivých systémů a zařízení se pochopitelně liší výrobce od výrobce. Na několika příkladech si ukážeme, jak diametrálně se mohou jednotlivé implementace lišit. Například L3 přepínač HP řady 5800 při počtu záznamů 8192 oznámí, případně zašle SNMP trap, že cache sousedů je plná a tímto je pro něj věc vyřízená. Všechna nově připojená IPv6 zařízení mají prostě smůlu:

```
<SW>display ipv6 neighbors all count
#May 7 14:30:19:562 2000 SW TPMB/4/ND TABLE FULL:
1.3.6.1.4.1.25506.2.38.1.5.4.1: Neighbor table is full and number
of items is 8192.

Total entry(ies):  8192
```

U L3 přepínače Cisco 3560 je průběh trochu jiný. Při počtu 2000 záznamů L3 přepínač poinformuje o tom, že další záznamy se nevejdou do TCAM a že pakety mohou být zpracovány v software.

```
Switch#
*Mar 1 00:24:13.393: %PLATFORM_IPv6_UCAST-6-PREFIX:  One or more, more specific
prefixes could not be programmed into TCAM and are being covered by a less
specific prefix, and the packets may be software forwarded
```

Celkem logicky se začne postupně zvyšovat vytížení CPU a při cca 20 000 záznamech začne docházet k chybám v alokacích, dojde k deaktivaci CEF (Cisco Express Forwarding) a L3 přepínač je výkonostně v koncích.

```
Switch#
*Mar 1 00:27:57.209: %SYS-2-MALLOCFAIL: Memory allocation of 65536 bytes failed from
0x2D84A28, alignment 85 Pool: Processor  Free: 111680  Cause: Memory
fragmentation6 Alternate Pool: None  Free: 0  Cause: No Alternate pool
7 -Process= "IPv6 Input", ipl= 0, pid= 3328 -Traceback= 2025520z
2D6A00Cz 2D70BCCz 2D84A2Cz 330B7D8z 330B93Cz 26E8EF0z 26B610Cz 26705C8z
2670768z 273FFA0z 275F0ACz 23C1870z 23C1E50z 23C2258z 23C416Cz9
*Mar 1 00:27:57.209: %COMMON_FIB-3-NOMEM: Memory allocation failure for fib entry
insertion in IPv6 CEF [0x0265A098] (fatal) (0 subsequent failures).10
*Mar 1 00:27:57.209: %COMMON_FIB-4-DISABLING: IPv6 CEF is being disabled due to a
fatal error.

Switch#sh processes cpu history

100 ************************************************************
90  ************************************************************
80  ************************************************************
70  ************************************************************
60  ************************************************************
50  ************************************************************
40  ************************************************************
30  ************************************************************
20  ************************************************************
10  ************************************************************
          0....5....1....1....2....2....3....3....4....4....5....5....6
              0    5    0    5    0    5    0    5    0    5    0
                        CPU% per second (last 60 seconds)
```

Pokud se podíváme na situaci u operačních systémů, tak situace je o něco lepší. Je to celkém logické. Operační systém má zpravidla k dispozici výkonnější procesor a není omezen specializovaným hardware jako u směrovačů a přepínačů. Není tedy problém, aby cache sousedů v takovém případě obsahovala například stovky tisíc záznamů. Implementace se ale liší systém od systému. Zatímco FreeBSD žádný limit pro počet záznamů nemá, jiné operační systémy (Windows, Linux, MAC OS X) mají typicky nastavený maximální počet záznamů, který je možné do cache sousedů uložit. Například Linux má ve výchozím stavu nastaven limit na hodnotu 1024 a je ovlivnitelný parametrem sysctl:

```
net.ipv6.neigh.default.gc_thresh3
```

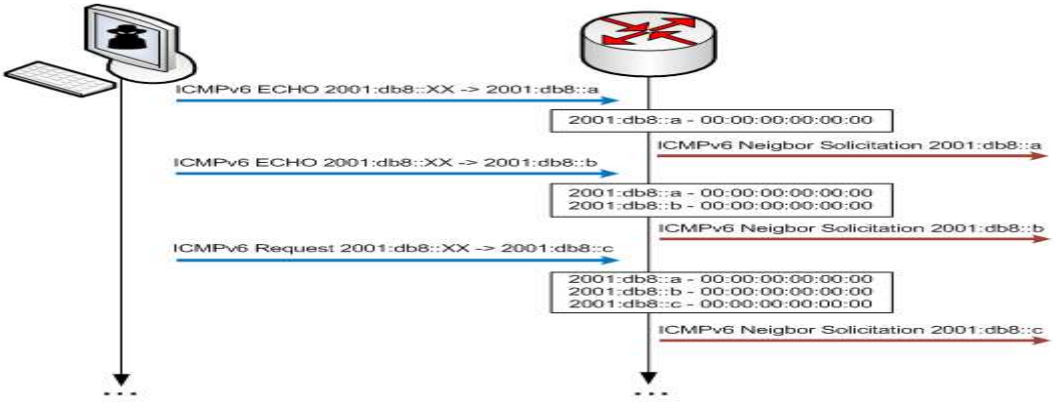
Pokud by se vám 1024 záznamů jevilo jako poměrně malé číslo i pro běžný provoz, tak v praxi zřejmě nebude představovat problém. Je to dáno tím, že Linuxové systémy, stejně jako Windows, uplatňují upravenou strategii uvolňování záznamů v tabulce sousedů. Podrobněji se této strategii budeme věnovat v příštím díle.

Asi není překvapením, že každá platforma reaguje trošku jinak a chování se mezi jednotlivými výrobci, operačními systémy či dokonce verzemi firmware liší. Poněkud nepříjemné je, že odhalení přesného chování a nalezení limitů u jednotlivých zařízení je zpravidla věcí čistě experimentální. Podle našich zkušeností, často jedinou cestou, jak se k objektivní informaci o počtu záznamů v cache sousedů dostat, je podrobit zařízení testům v laboratoři.

*Poznámka: Ačkoliv se v článcích snažíme vždy v maximálně otevřené podobě demonstrovat realizace útoků, tak v tomto případě jsme se rozhodli udělat výjimku. Na jednu stranu by postup pro zaplnění cache sousedů byl jistě užitečný pro ty, kteří si případně chtějí v testovacím prostředí ověřit limity svých zařízení. Zkušenosti nám však ukázaly, že pro řadu zařízení jsou důsledky zaplnění cache sousedů poměrně fatální a to s přesahem na chod IPv4 infrastruktury. Z toho důvodu jsme se v tomto případě rozhodli přesný postup zcela záměrně nezveřejňovat.*

Vzdálený útok

Trochu jinou variantou útoku na vyčerpání cache sousedů je jeho vzdálená varianta. V tomto případě se útočník může vyskytovat kdekoliv na Internetu. Princip útoku se dá zjednodušeně popsat takto: Postupným zasíláním paketů do cílové sítě na různé cílové adresy donutíme hraniční směrovač příslušné sítě vygenerovat zprávu *Výzva sousedovi* a zařadit si do cache sousedů záznam o nedostupnosti cílové adresy. Tím může dojít k její vyčerpání, ke zvýšení zátěže směrovače a nárustu objemu komunikace signalizačních protokolů a multicast provozu v koncové síti.




V praxi však tento útok zpravidla vede ke stejným výsledkům, jako první forma dříve popisovaného lokálního útoku. Nedojde tedy k faktickému zaplnění celé cache sousedů, ale ke značně zvýšené zátěži procesoru. Na útoku jsou ovšem nepříjemné dvě věci. Jednak může být proveden z jakéhokoliv místa na Internetu a navíc, cílem útoku nemusí být nutně pouze koncová síť připojující servery nebo uživatele. Obětí může být například kterákoliv propojovací síť spojující jeden nebo více směrovačů. Tímto se útok řadí do pozice, kdy jej lze zneužít nejen pro útočení na koncové systémy, ale i na samotnou infrastrukturu Internetu, tj. například na spojovací sítě mezi směrovači, nebo peerovací sítě v propojovacích uzlech (IXP). Na následujícím obrázku vidíme, jak může taková situace vypadat.

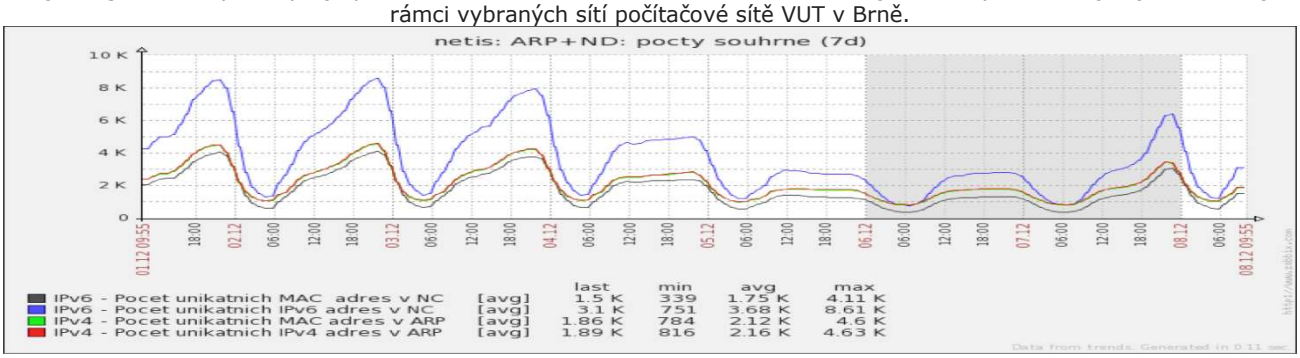

Dva směrovače jsou propojené sítí s prefixem 2001:db8::/64. Pokud útočník zašle libovolný paket na nějakou cílovou adresu z této sítě, musí směrovač vyvolat proces vyhledání souseda. K operaci využije své CPU a vyšle do sítě zprávu *Výzva sousedovi*. Druhý směrovač musí tuto zprávu nějak zpracovat, protože jen tak zjistí, že se jej netýká. K tomu opět využije CPU, takže útočník může každým paketem "obtěžovat" hned dva či více směrovačů najednou.

Útok bez útočníka

Pokud směrovač ve vaší síti zahlásí informaci o vyčerpané cache sousedů, či se začne chovat neobvykle, nemusí to nutně znamenat, že se ve vaší síti nachází záškodník. Tento jev může nastat i za normálních okolností a napomáhá tomu několik nových vlastnosti protokolu IPv6:

- Každé IPv6 zařízení má nakonfigurovánou kromě globální IPv6 adresy ještě dalších několik adres jako například *link-local* nebo *unique-local*. Minimálně je třeba počítat alespoň se dvěma IPv6 adresami na každé zařízení.
- V případě, že zařízení využívá Privacy Extensions dle RFC 4941 nebo RFC 3041, je nutno počítat s tím, že zařízení si bude průběžně generovat na rozhraní nové IPv6 adresy, přičemž dříve používané adresy si ponechá po nějakou dobu stále ještě dostupné. Skutečnou lahůdku pak představují některé operační systémy, které si novou IPv6 adresu vytvoří s každou reasociací k WiFi síti.

Pokud přihlédneme k oběma skutečnostem, můžeme bez problémů narazit na zařízení, které má na IPv6 rozhraní nakonfigurováno například 10 nebo více IPv6 adres, jak jsme si ostatně ukázali v předchozím díle. A zde pak už začíná být poměrně těsno. Pokud uvážíme, že v síti máme připojených k jednomu směrovači například 800 takových zařízení, velice snadno se můžeme přiblížit k limitu celkového počtu záznamů v tabulce sousedů. Jak to vypadá v praxi, si můžeme ukázat na následujícím grafu, který zachycuje počet unikátních MAC, IPv4 a IPv6 adres během jednoho týdne, tak, jak je monitorujeme v rámci vybraných sítí počítačové sítě VUT v Brně.



Z grafu můžeme vidět, že zelený a červený průběh, které představují počet unikátních IPv4 a MAC adres v ARP tabulce, se prakticky kryjí. Na každou MAC adresu tedy připadá jedna IPv4 adresa. Počet záznamů v ARP tabulce tudíž odpovídá počtu zařízení připojených v síti. V případě IPv6 je situace trošku jiná. Černý průběh zachycuje počet unikátních MAC adres, modrý průběh počet unikátních IPv6 adres v cache sousedů. Zde vidíme, že počet záznamů v sítích, které podporují IPv6 protokol, je více než dvoj až trojnásobný oproti počtu zařízení.

Závěr

Z vlastní zkušenosti doporučujeme na problém dostatečné velikosti cache sousedů myslet, zejména při návrhu větších sítí. V provozním prostředí je pak také vhodné průběžně monitorovat míru jejího zaplnění. Není asi třeba zdůrazňovat, že v případě zaplnění cache sousedů je diagnostika takového stavu značně problematická. Část uživatelů totiž začne hlásit, že se jim občas špatně načítá Google a při řešení takového problému vás zpravidla nenapadne, že problém může být způsoben tím, že je zaplněná cache sousedů - ať už vlivem velkého počtu řádných záznamů anebo v důsledku cíleně vedeného útoku.

Zatímco prostor pro ARP záznamy je na většině zařízeních v řádech desítek tisíc a jedná se zpravidla o velice dobře zdokumentovanou informaci, v případě IPv6 je prostor pro cache sousedů často výrazně menší a výrobci zařízení raději tento údaj v dokumentaci nezmiňují či všemožně zatemňují. Údaj o velikosti cache sousedů je přitom poměrně stěžejní zejména proto, že na mnohých platformách je velikost vázána na hardwarové prostředky, které má zařízení k dispozici. Je tedy možné, že změna limitu nebude řešitelná jinak než výměnou hardware. V příštím díle se podíváme na možné principy obrany proti cíleným útokům na cache sousedů a na konkrétní způsoby, které lze k obraně použít.

**Bezpečné IPv6: když dojde keš – obrana**

Dnešní díl bude bez dalších úvodů navazovat na díl předchozí. Pro seznámení se s problematikou útoků na cache sousedů tedy doporučujeme nejdříve prostudovat předchozí díl. Cest pro eliminaci útoků na cache sousedů existuje několik, přičemž o žádné z nich nemůžeme říct, že řeší problém komplexně a bezezbytku, či alespoň nevyvolává rozporuplné postoje v rámci IPv6 komunity.

Zvětšení délky prefixu

Jedním z prvních řešení může být **zvětšení délky prefixu** na velikost zaručující, že část prefixu (host ID) bude obsahovat pouze takové množství IPv6 adres, aby je bylo možné uložit do cache sousedů. V praxi to tedy znamená, že namísto standardní délky prefixu (/64) určeného pro koncové sítě, použijeme prefix delší, například 120 bitů. Tímto zůstane 8 bitů pro adresaci 255 koncových zařízení, tj. jsme ve stejném stavu, jako kdybychom použili délku prefixu /24 u IPv4. Tento přístup ovšem naráží na několik problémů. Při návrhu sítě musíme zohledňovat celkový počet zařízení v koncové sítí a ten promítnout do vhodně volených délek prefixů. Co je ovšem mnohem větší problém, je to, že mechanismy bezstavové autokonfigurace, stejně jako celá řada dalších standardů, předpokládají, že délka prefixu pro koncovou síť bude vždy 64 bitů. S jinou délkou prefixu bezstavová autokonfigurace zkrátka nebude fungovat. Na některých systémech může fungovat kombinace DHCPv6 a delšího prefixu, nicméně tento způsob koliduje například s RFC 4291 - *IPv6 Addressing Architecture*. Řada výrobců zařízení navíc spoléhá na to, že délka prefixu nebude delší než oněch 64 bitů a byť je delší prefix na zařízení možné nakonfigurovat, tak může dojít k neoptimálnímu zpracování provozu (např. v CPU namísto hardware). V neposlední řadě postup v principu nijak neřeší situaci, kdy je cache sousedů zaplněná a nově připojená zařízení nemohou být zařazena do cache sousedů. Celou problematiku fixní délky prefixu pro koncové sítě pak shrnuje nově vydané RFC 7421 - Analysis of the 64-bit Boundary in IPv6 Addressing.

Zvětšení délky prefixu je tedy postup, který může mít význam spíše jako ochrana vlastních zařízení před útoky cílenými na vyčerpání cache sousedů vzdáleným útočníkem. Lze jej tedy s výhodou použít například pro sítě, které propojují směrovače. Touto situací se speciálně zabývá RFC 6164, které pro takováto spojení doporučuje použít prefix s délkou 127 bitů, tedy takový, který umožňuje připojení právě dvou zařízení. Situace je zachycena na následujícím obrázku:


Na rozdíl od situace s prefixem délky 64 bitů, kterou jsme si popisovali v předchozím díle, jsou záškodníkovy pakety zneškodněny hned na prvním směrovači, a to prostřednictvím tzv. blackhole směrovacího záznamu.

Používání prefixů v jiné délce než 64 bitů je jedno z dalších velice kontroverzních témat, rozdělující IPv6 komunitu. Část komunity považuje vyčlenění 64 bitů pro koncové sítě za velké plýtvání a uvítala by volnější pravidla, druhá skupina v tom vidí přebírání negativních vlastností protokolu IPv4 a dodává, že IPv6 adres máme dostatek, tedy si můžeme dovolit takto "plýtvat" a použít všude prefix délky 64 bitů.

Statické záznamy v cache sousedů

Pro kritické aplikace běžící například na serverech v datových centrech, může být jistým řešením zavedení statických záznamů do cache sousedů. Tím je zaručeno, že v případě, kdy útočník zajistí vyčerpání cache sousedů, budou v ní stále obsaženy námi zařazené statické záznamy. Toto řešení je ovšem vykoupeno neskutečnou pracností v podobě vytváření a udržování statických záznamů a v praxi tedy použitelné pouze ve velmi specifických prostředích. Je také nutné, aby koncové zařízení využívalo striktně statickou konfiguraci IPv6 adres na svých rozhraních. Vytvoření statického záznamu u platformy HP používající firmware Comware může vypadat například takto:

```
[SW]ipv6 neighbor 2001:67C:1220::10 0050-5694-b2e0 interface
Vlan-interface 1

[SW]display ipv6 neighbors 2001:67C:1220::10 verbose
Type: S-Static    D-Dynamic

                IPv6 Address     : 2001:67C:1220::10
Link-layer      : 0050-5694-b2e0     VID : N/A    Interface   : N/A
State           : INCMP              Type: S      Age         : -
                Vpn-instance     : [No Vrf]
```

U zařízení Cisco lze stejného efektu dosáhnout zadáním následujících příkazů.

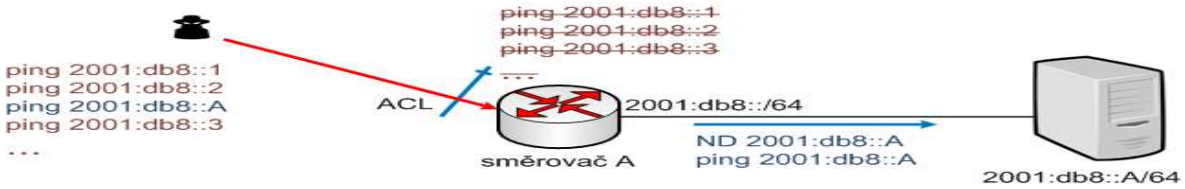
```
Switch(config)#ipv6 neighbor 2001:67C:1220::10 vlan 1 0050.5694.b2e0
```

```
Switch#sh ipv6 neighbors
IPv6 Address                Age Link-layer Addr State Interface
2001:67C:1220::10             - 0050.5694.b2e0  REACH Vl1
```

Vytváření takových statických záznamů není nezbytně nutné pro všechna zařízení v příslušné síti. Záznamy můžeme vytvořit pouze pro kritická zařízení a ostatní ponechat v režimu automatického učení prostřednictvím Neighbor Discovery.

Filtrace podle adresy

Další možností, jak se bránit vzdálené formě útoku, je filtrace adres. Logicky se nabízí filtrace podle zdrojové adresy útočníka v době útoku. Zde lze nicméně předpokládat, že útočník bude jednak vést útok z více míst, a také, že zdrojová adresa v paketech bude nejspíše podvržená. Nabízí se tedy přístup přesně obrácený - propouštět provoz pouze pro cílové adresy, o kterých víme, že na nich jsou skutečně umístěná zařízení s IPv6 adresou.




Připomeňme si, že podstatou útoku je zasílání paketů do cílové sítě na náhodně generované IPv6 adresy. Princip obrany je pak velice jednoduchý - pakety, u kterých víme, že směřují na neexistující IPv6 adresy, zahodíme hned na vstupu. Nevznikne tak důvod provádět operaci objevování sousedů a tedy nedojde k zaplnění tabulky sousedů ani k případnému přetížení CPU směrovače.

Pro realizaci obrany tohoto typu je možné využití paketových filtrů (ACL), které má většina zařízení implementována přímo v hardware. V základní podobě si tedy můžeme vystačit například s takovýmto jednoduchým filtrem, který bude aplikován na vstupu vnějšího rozhraní směrovače:

```
acl ipv6 number 3000 name nd-filter
rule 0 permit ipv6 destination 2001:DB8::A/128
rule 10 deny ipv6
```

V praxi si však s tímto jednoduchým filtrem nevystačíme, protože v koncové sítí bude zřejmě více aktivních IPv6 adres. Tím se může údržba takového filtru stát poměrně složitou záležitostí. Jistého zjednodušení lze dosáhnout například tím, že vstupní filtr je definován tak, aby propouštěl pakety s cílovou adresou patřící do společně popsatelného prefixu například délky 120 bitů. Tímto se vyhneme zařazování záznamu do ACL s každým nově připojeným zařízením. Stejně jako v předchozím případě platí, že metoda je aplikovatelná pouze na staticky přidělené adresy a tedy použitelná maximálně pro ochranu sítí, ve kterých jsou připojené servery.

*Upozornění: Možná vás napadlo, že by se problém dal pojmout komplexněji. Proč rovnou nezařadit do filtru i cílové porty příslušné služby, když už jednou provádíme filtraci na cílovou adresu? Pokud například víme, že na příslušné adrese běží pouze web server na portu 80, tak není žádný důvod, proč by měly být dostupné další porty. Zde si však dovolíme opět upozornit na problém filtrace paketů s rozšířenými hlavičkami. Pokud zařadíme filtraci pro příslušný TCP/UDP port, donutíme zařízení provádět inspekci obsahu i na vyšších protokolových vrstvách, čímž potenciálnímu útočníkovi umožníme provést útok, ať už s využitím zřetězení rozšířených hlaviček nebo hlavičky fragmentace, jak jsme popisovali v díle Trable s hlavičkami.*

IPv6 destination guard

Jakousi automatizovanou podobu předchozího způsobu představuje technika jménem *Destination guard*. Princip je velice jednoduchý. Pakliže na směrovač dorazí paket s cílovou adresou, kterou cache sousedů neobsahuje, je takový paket ihned zahozen. Technika má jedno úskalí a tím je počáteční naplnění cache sousedů. K tomuto se právě využívá vazební tabulky (*neighbor binding database*), kterou jsme popisovali v druhém díle Zkrocení zlých směrovačů. Je tedy součástí technik, které se souhrně nazývají *First-Hop-Security* a bez aplikace těchto technik, a tedy i omezení z nich vyplývajících, nebude fungovat.

Praktickou demonstraci ukazující použití této techniky je možné shlédnout na krátkém videu.

Omezení počtu IPv6 adres v cache sousedů na rozhraní

Doposud jsme si popisovali možné způsoby ochrany, které jsou použitelné pro sítě, ve kterých jsou umístěné servery. Taková koncová síť je z podstaty věci mnohem více chráněná proti lokálním útokům a tedy má smysl věnovat pozornost spíše obraně proti vzdálenému vyčerpání cache sousedů. Odlišná situace ovšem nastává v sítí, kde jsou umístěná koncová zařízení běžných uživatelů. Zde je koncová síť přístupná prakticky každému, kdo se do ni připojí a útočník má mnohem lepší podmínky pro vedení lokálního útoku na cache sousedů.

Některé platformy umožňují nastavit omezení počtu záznamů v cache sousedů příslušející jednomu rozhraní. Tato volba nám zaručí, že útočník v jedné síti nevyčerpá na směrovači veškerou kapacitu cache sousedů. Vlastní síťový segment je útokem sice postižen, ale nejsou ohroženy ostatní sítě na stejném směrovači.

Na platformě Cisco lze k tomu použít následující příkaz, který omezí počet záznamů na každém logickém rozhraní na 1000.:

```
Switch(config)# ipv6 nd cache interface-limit 1000
```

U platformy HP lze podobný limit nastavit na jednotlivých rozhraních následovně.

```
[SW]interface vlan 210
[SW-Vlan-interface210] ipv6 neighbors max-learning-num 1000
```

Omezení doby záznamu v cache sousedů

Poměrně důležitým krokem, ke kterému budete muset v některých případech sáhnout, je zkrácení doby, po kterou je záznam v cache sousedů držen. Jak jsme si řekli v předchozím díle, mnohá zařízení se na WiFi sítí chovají tak, že si s každou reautentizací vygenerují novou IPv6 adresu. Tímto může dojít poměrně k rychlému zaplnění tabulky sousedů. Záznamy, které se takto vygenerovaly, se vztahují k adresám, které se ale reálně již nepoužívají a tedy pouze zabírají místo. Účinným řešením tohoto problému je právě snížení doby, po kterou je záznam v cache sousedů uchován.

Snížení doby platnosti záznamu může posloužit rovněž jako obrana proti cílenému útoku. Víceméně pak s útočníkem hrajeme hru na přetahovanou, jejíž cílem je uvolňovat podvržené záznamy rychleji, než útočník stíhá vytvářet nové. Tuto strategii používá

například Linuxové jádro nebo novější verze systému Windows, kde ve výchozí konfiguraci je záznam v cache sousedu udržován pouhých 70 sekund, bez ohledu na to, zda je příslušná adresa stále na síti dostupná či nikoliv. Jak to funguje v praxi. Za normálních okolností, resp. dle RFC 4861 by operační systém měl před vyřazením adresy nejdříve provést ověření její dostupnosti (Neighbor Unreachability Detection - NUD). Teprve až na základě výsledku tohoto testu provést případné vyřazení adresy z cache sousedů. Oba zmíněné operační systémy věc řeší jednoduše. Ověření dostupnosti (NUD) zkrátka neprovádějí a příslušný záznam rovnou vyřadí. V minulém díle jsme zmínili, že linuxové systémy mají ve výchozím stavu cache sousedů omezenou na 1024 záznamů. Díky rychlému uvolňování záznamů nejsou v cache sousedů adresy všech zařízení v podsítích, ale pouze těch, které s uzlem komunikovaly během posledních 70 vteřin. V praxi tedy limit v podobě 1024 záznamů není tak omezující.

Přístup rychlého uvolňování není ovšem zcela bezproblémový. Jeho důsledkem je nutnost mnohem častěji provádět zjištění mapování mezi IPv6 a linkovými adresami. To v některých případech může vést na pomalejší odezvu v případě navazování nových spojení a zejména pak zvýšený provoz na úrovni signalizačních protokolů - tedy zejména multicast provozu, který je značně problematický zejména ve WiFi sítích, jak jsme si ostatně popisovali v díle Trable s multicastem. Přístup tedy představuje jakési vyhánění čerta ďáblem, který je rovněž v rozporu s původním záměrem, jak jej definuje RFC 4861 a tedy i kontroverzní z pohledu skalních zastánců bezpodmínečného dodržování RFC. I přes tyto vady na kráse se výrobci operačních systému přiklánějí spíše k variantě rychlejšího uvolňování záznamů než k implementaci celého algoritmu NUD.

U aktivních prvků je ovšem situace jiná. Zde spíše převládá strategie pozvolnějšího uvolňování záznamů. Na většině zařízení se jedná o konfigurovatelný parametr, který je možné za běhu modifikovat. Zde opět platí, že každý výrobce k problému přistupuje trošku jinak. Na zařízeních Cisco je možné nastavit dobu expirace záznamu v sekundách příkazem:

```
SW-Cisco(config)# ipv6 nd cache expire <čas v sekundách>
```

Na zařízeních HP používající firmware Comware lze použit příkaz:

```
[SW-Comware]ipv6 neighbor stale-aging <čas v hodinách>
```

Zde je trochu omezení v tom, že se čas nastavuje v hodinách a minimální doba expirace je 1 hodina. Úplně jinak je to na zařízeních stejného výrobce u řady Procurve, kde se čas nastavuje společně s dobou expirace záznamu v tabulce ARP. Tentokrát se čas nastavuje v minutách.

```
SW-Procurve(config)# ip arp-age <čas v minutách>
```

Stejně jako způsoby konfigurace se také liší i použité výchozí hodnoty - od několika desítek minut až po dny. Parametry konkrétního zařízení je tedy třeba vždy konzultovat s dokumentací nebo ověřit experimentálně.

Tímto jsme vyčerpali základní výčet obranných prostředků proti útokům vedeným na vyčerpání cache sousedů. Vzhledem k tomu, že se jedná o celkem složitou oblast, nabízíme pro přehlednost tabulku se souhrnem jednotlivých obranných technik a vhodnosti jejich použití:

| **Technika** | Eliminace lokálního útoku | Eliminace vzdáleného útoku | Nevýhoda |
|---|---|---|---|
| **Zvětšení délky prefixu** | NE | ANO | Nelze použít s autokonfigurací |
| **Statické záznamy v cache sousedů** | ANO | ANO | Náročný management |
| **Filtrace podle adresy** | NE | ANO | Vyžaduje vytváření ACL. Vhodné pouze pro servery. |
| **Omezení počtu IPv6 adres na rozhraní** | částečně | NE | Pouze minimalizuje dosah útoku |
| **Omezení doby záznamu v cache sousedů** | částečně | částečně | Zmírňovací technika, může vést na vyšší zátěž CPU |

Závěr

Dnes jsme si ukázali některé základní techniky, které můžeme použít, ať už pro eliminaci útoků na cache sousedů, nebo alespoň pro zmírnění jejich následků. V současné době, kdy je IPv6 protokol zpravidla implementován v režimu dual-stack, tj. společně s IPv4, nemusí být důsledky útoků na cache sousedů až tak fatální. Řada prohlížečů podporuje automatické přepnutí na protokol IPv4 v případě, že je protokol IPv6 nedostupný. Případné problémy s cache sousedů tedy nemusí být pro uživatele na první pohled viditelné. Automatické přepnutí ovšem funguje většinou pouze pro prohlížeč a už ne například pro emailového klienta, či jiné aplikace využívající IPv6. Výrazně horší situace také nastane v případě, kdy je síť implementována jako IPv6 only, například s využitím DNS64/NAT64, jak před časem popisoval Ondřej Caletka. V takovém případě uživatel pocítí jakýkoliv problém na IPv6 protokolu okamžitě.

**Bezpečné IPv6: příliš mnoho sousedů**

Detekce výskytu duplicitních IPv6 adres je jednou z dalších novinek protokolu IPv6. Uvedený mechanismus jsme už kdysi popisovali v jiném seriálu, takže si nyní provedeme jen velmi krátkou rekapitulaci. V rámci přidělování IPv6 adresy koncovému zařízení se od začátku předpokládalo, že adresa zařízení nebude přidělená externí autoritou, jakou je například DHCP server, ale že se na tvorbě IPv6 adresy bude podílet samo koncové zařízení. Z tohoto důvodu byla celá IPv6 adresa o celkové délce 128 bitů rozdělená na dvě stejně velké části - adresu sítě a identifikátor síťového rozhraní zařízení v dané síti. Síťová část adresy (horních 64 bitů) je společná pro všechny uzly na stejném síťovém segmentu a identifikátor síťového rozhraní (spodních 64 bitů) je unikátní pro každé síťové rozhraní IPv6 uzlu. Zatímco síťová část adresy je předem dána konfigurací sítě, předpokládá se, že identifikátor rozhraní si uzel nějakým vhodným algoritmem vytvoří sám.

Úplně první návrhy protokolu IPv6 počítaly s tím, že identifikátor síťového rozhraní hostitele bude odvozen od linkové adresy síťové karty (MAC adresy). Tento přístup však začal narážet na problémy stran ochrany soukromí - jakékoliv IPv6 zařízení by bylo sledovatelné napříč Internetem, ať už se vyskytne v jakékoliv síti. Z toho důvodu se v roce 2011 objevilo RFC 3041, později nahrazené RFC 4941, zavádějící mechanismus, kterému se zkráceně říká *Privacy Extensions*. Ten ve zjednodušené podobě

funguje tak, že identifikátor síťového rozhraní hostitele je generován zcela nahodile a v pravidelných časových intervalech se musí měnit. V konečném důsledku je tedy výsledná IPv6 adresa koncového uzlu náhodně vytvořená a předem nepredikovatelná. Skutečnost, že IPv6 adresy nejsou přidělovány centrální autoritou, která má přehled, komu jakou adresu přidělila, vede k tomu, že v přidělování adres mohou teoreticky vzniknout kolize. I když je pravděpodobnost vzniku takové kolize velmi malá, nelze ji se 100% jistotou vyloučit. Aby byly ošetřeny jednak tyto krajní případy a také situace, kdy jsou v rámci jedné sítě nedopatřením manuálně nakonfigurovány totožné IPv6 adresy, disponuje protokol IPv6 detekcí duplicitních adres (DAD - *Duplicate Address Detection*).

Celá detekce duplicitních adres funguje následovně. Představme si situaci, že váš laptop je nově připojen do sítě. Na základě *Výzvy směrovači* (*Router Solicitation*) obdrží *Oznámení směrovače* (*Router Advertisement*), kde se dozví, že má pro síťovou část adresy použít například prefix 2001:67c:1220:f777::/64. Pokud má aktivovanou podporu *Privacy Extensions*, vygeneruje si náhodně 64 bitovou posloupnost pro identifikátor síťového rozhraní, kterou připojí za přidělený síťový prefix a tím získá výslednou IPv6 adresu. Dříve než však tuto adresu nakonfiguruje na svém rozhraní, vyšle do sítě zprávu *Výzva sousedovi* (*Neigbor Solicitation*), kde jako adresu vyzývaného souseda použije právě vytvořenou IPv6 adresu. Vtip je v tom, že pokud by na síti již uzel s uvedenou adresou existoval, tak odpoví prostřednictvím zprávy *Ohlášení souseda* (*Neigbor Advertisement*). V naprosté většině případů však odpověď nedorazí a uzel tak ví, že může takto zvolenou adresu použít. Mechanismus si můžeme přirovnat k situaci, kdy vstoupíte do místnosti, kde je umístěno 1,8 * $10^{19}$ ($2^{64}$) židlí a vy víte, že místnost nemá žádný zasedací pořádek. Náhodně si tedy zvolíte židli, ale před usednutím se ještě zdvořile zeptáte, zda židle, na kterou se hodláte usadit, je volná. Pokud se do určité doby nikdo neozve, tak se prostě posadíte.

V reálné sítí pak může celý proces vypadat tak, jak je zachyceno na následujícím obrázku:

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | :: | ff02::1:ff79 | ICMPv6 | 78 | Neighbor Solicitation for fe80::c844:ce09:2179:f914 |
| 2 | 0.000038 | fe80::c844:ce09:2179:f914 | ff02::2 | ICMPv6 | 70 | Router Solicitation from 00:50:56:b5:13:23 |
| 3 | 0.000374 | fe80::250:56ff:fe94:b2e1 | ff02::1 | ICMPv6 | 110 | Router Advertisement from 00:50:56:94:b2:e1 |
| 4 | 0.498643 | :: | ff02::1:ff79 | ICMPv6 | 78 | Neighbor Solicitation for 2001:67c:1220:f777:c844:ce09:2179:f914 |
| 5 | 0.498660 | :: | ff02::1:ff11 | ICMPv6 | 78 | Neighbor Solicitation for 2001:67c:1220:f777:87f:fb67:b211:aeb2 |
| 6 | 0.997689 | fe80::c844:ce09:2179:f914 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::c844:ce09:2179:f914 (ovr) is at 00:50:56:b5:13:23 |
| 7 | 1.497410 | 2001:67c:1220:f777:c844:ce | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:c844:ce09:2179:f914 (ovr) is at 00:50:56:b5:13:23 |
| 8 | 1.497431 | 2001:67c:1220:f777:87f:fb67 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:87f:fb67:b211:aeb2 (ovr) is at 00:50:56:b5:13:23 |

Nejdříve koncový uzel vyšle do sítě požadavek na ověření dostupnosti souseda s adresou fe80::c844:ce09:2179:f914. Jedná se o jeho *link-local* adresu, kterou by chtěl nadále používat (paket č. 1). Pokud usoudí, že vytvořená *link-local*adresa s nikým nekoliduje, využije ji k zaslání následné zprávy *Výzva směrovači*(paket č. 2). Po obdržení zprávy *Oznámení směrovače* (paket č. 3) zařízení ví, jaký IPv6 prefix se v síti používá. Z daného prefixu si zařízení vytvoří dvojici IPv6 adres - globální a dočasnou a celý proces pro tyto adresy zopakuje (pakety č. 4 a 5). Vzhledem k tomu, že nedorazila reakce od žádného okolního uzlu, systém si obě globální IPv6 adresy nakonfiguruje na svém rozhraní a začne je používat.

Jak už asi tušíte, zde jsou přímo ideální podmínky pro to, aby do procesu mohl velmi jednoduchým způsobem vstoupit útočník. Ten pak na každý dotaz, zda v síti již existuje příslušná adresa, jednoduše odpoví, že existuje. V praxi pak průběh útoku vypadá tak, jak je zachyceno na následujícím obrázku:

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | :: | ff02::1:ff79 | ICMPv6 | 78 | Neighbor Solicitation for fe80::c844:ce09:2179:f914 |
| 2 | 0.000028 | fe80::c844:ce09:2179:f914 | ff02::2 | ICMPv6 | 70 | Router Solicitation from 00:50:56:b5:13:23 |
| 3 | 0.000108 | fe80::c844:ce09:2179:f914 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::c844:ce09:2179:f914 (ovr) is at 00:50:f4:4d:fd:19 |
| 4 | 0.000334 | fe80::250:56ff:fe94:b2e1 | ff02::1 | ICMPv6 | 110 | Router Advertisement from 00:50:56:94:b2:e1 |
| 5 | 0.001229 | fe80::c844:ce09:2179:f914 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::c844:ce09:2179:f914 (ovr) is at 00:50:f4:4d:fd:19 |
| 6 | 0.545801 | :: | ff02::1:ffaa | ICMPv6 | 78 | Neighbor Solicitation for fe80::a455:3c20:faaa:ff1a |
| 7 | 0.545828 | :: | ff02::1:ffaa | ICMPv6 | 78 | Neighbor Solicitation for 2001:67c:1220:f777:a455:3c20:faaa:ff1a |
| 8 | 0.545844 | :: | ff02::1:ffff | ICMPv6 | 78 | Neighbor Solicitation for 2001:67c:1220:f777:d900:ed:cfff:e54d |
| 9 | 0.545930 | fe80::a455:3c20:faaa:ff1a | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::a455:3c20:faaa:ff1a (ovr) is at 00:50:7a:14:32:24 |
| 10 | 0.546362 | 2001:67c:1220:f777:a455:3c20:faaa:ff1a | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:a455:3c20:faaa:ff1a (ovr) is at 00:50:96:20:b3:40 |
| 11 | 0.546761 | fe80::a455:3c20:faaa:ff1a | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::a455:3c20:faaa:ff1a (ovr) is at 00:50:7a:14:32:24 |
| 12 | 0.546791 | 2001:67c:1220:f777:d900:ed:cfff:e54d | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:d900:ed:cfff:e54d (ovr) is at 00:50:e4:fd:cb:d7 |
| 13 | 0.547382 | 2001:67c:1220:f777:d900:ed:cfff:e54d | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:d900:ed:cfff:e54d (ovr) is at 00:50:e4:fd:cb:d7 |
| 14 | 0.548051 | 2001:67c:1220:f777:a455:3c20:faaa:ff1a | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:a455:3c20:faaa:ff1a (ovr) is at 00:50:96:20:b3:40 |
| 15 | 0.998912 | :: | ff02::1:fff8 | ICMPv6 | 78 | Neighbor Solicitation for fe80::3c74:b13c:f0f8:6b06 |
| 16 | 0.999017 | fe80::3c74:b13c:f0f8:6b06 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::3c74:b13c:f0f8:6b06 (ovr) is at 00:50:9e:c5:67:06 |
| 17 | 0.999980 | fe80::3c74:b13c:f0f8:6b06 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::3c74:b13c:f0f8:6b06 (ovr) is at 00:50:9e:c5:67:06 |
| 18 | 1.669399 | :: | ff02::1:fff8 | ICMPv6 | 78 | Neighbor Solicitation for fe80::d3a:34de:c4f8:9380 |
| 19 | 1.669430 | 2001:67c:1220:f777:a455:3c20:faaa:ff1a | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:a455:3c20:faaa:ff1a (ovr) is at 00:50:56:b5:13:23 |
| 20 | 1.669439 | 2001:67c:1220:f777:d900:ed:cfff:e54d | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement 2001:67c:1220:f777:d900:ed:cfff:e54d (ovr) is at 00:50:56:b5:13:23 |
| 21 | 1.669507 | fe80::d3a:34de:c4f8:9380 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::d3a:34de:c4f8:9380 (ovr) is at 00:50:12:58:0d:0e |
| 22 | 1.670406 | fe80::d3a:34de:c4f8:9380 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::d3a:34de:c4f8:9380 (ovr) is at 00:50:12:58:0d:0e |
| 23 | 1.996555 | :: | ff02::1:ff78 | ICMPv6 | 78 | Neighbor Solicitation for fe80::9592:a5b:b078:e57e |
| 24 | 1.996621 | fe80::9592:a5b:b078:e57e | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::9592:a5b:b078:e57e (ovr) is at 00:50:e9:6c:4e:2d |
| 25 | 1.999924 | fe80::9592:a5b:b078:e57e | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::9592:a5b:b078:e57e (ovr) is at 00:50:e9:6c:4e:2d |
| 26 | 2.511700 | :: | ff02::1:ff30 | ICMPv6 | 78 | Neighbor Solicitation for fe80::31fe:4261:5f30:e7f7 |
| 27 | 2.511778 | fe80::31fe:4261:5f30:e7f7 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::31fe:4261:5f30:e7f7 (ovr) is at 00:50:be:de:b8:b2 |
| 28 | 2.512689 | fe80::31fe:4261:5f30:e7f7 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::31fe:4261:5f30:e7f7 (ovr) is at 00:50:be:de:b8:b2 |
| 29 | 2.995215 | :: | ff02::1:ff8b | ICMPv6 | 78 | Neighbor Solicitation for fe80::8159:d58d:18b:ac11 |
| 30 | 2.995309 | fe80::8159:d58d:18b:ac11 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::8159:d58d:18b:ac11 (ovr) is at 00:50:2c:b5:cb:a6 |
| 31 | 2.996220 | fe80::8159:d58d:18b:ac11 | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::8159:d58d:18b:ac11 (ovr) is at 00:50:2c:b5:cb:a6 |
| 32 | 3.666086 | :: | ff02::1:ff95 | ICMPv6 | 78 | Neighbor Solicitation for fe80::5980:4a36:9895:64da |
| 33 | 3.666196 | fe80::5980:4a36:9895:64da | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::5980:4a36:9895:64da (ovr) is at 00:50:ca:fd:ca:60 |
| 34 | 3.668167 | fe80::5980:4a36:9895:64da | ff02::1 | ICMPv6 | 86 | Neighbor Advertisement fe80::5980:4a36:9895:64da (ovr) is at 00:50:ca:fd:ca:60 |
| 35 | 3.993636 | :: | ff02::1:ffcb | ICMPv6 | 78 | Neighbor Solicitation for fe80::64a3:2a94:a8cb:4128 |

Stejně jako v předchozím případě uzel vyšle do sítě *Výzvu sousedovi*. Útočník ale pohotově zareaguje zprávou *Ohlášení souseda*. Uzel se tedy začne domnívat, že příslušná IPv6 adresa se už používá. V případě, že má uzel aktivovanou podporu pro *Privacy*

*Extensions*, vznikne na straně uzlu domněnka, že náhodou došlo k vygenerování stejného identifikátoru síťového rozhraní. Vygeneruje si tedy nový identifikátor - tedy i novou IPv6 adresu, doufaje, že tentokrát již kolize nenastane. Opět zkontroluje, zda v síti taková adresa existuje a opět od útočníka obdrží odpověď, že ano. Takto se děj opakuje, dokud koncový uzel nevyhodnotí situaci tak, že jakékoliv další pokusy jsou zbytečné. Pokud bychom se vrátili do našeho přirovnání k místnosti s židlemi, situace je obdobná, jako kdyby se po každém vašem dotazu, zda je příslušná židle volná, odněkud ozval hlas "tato židle je obsazená".

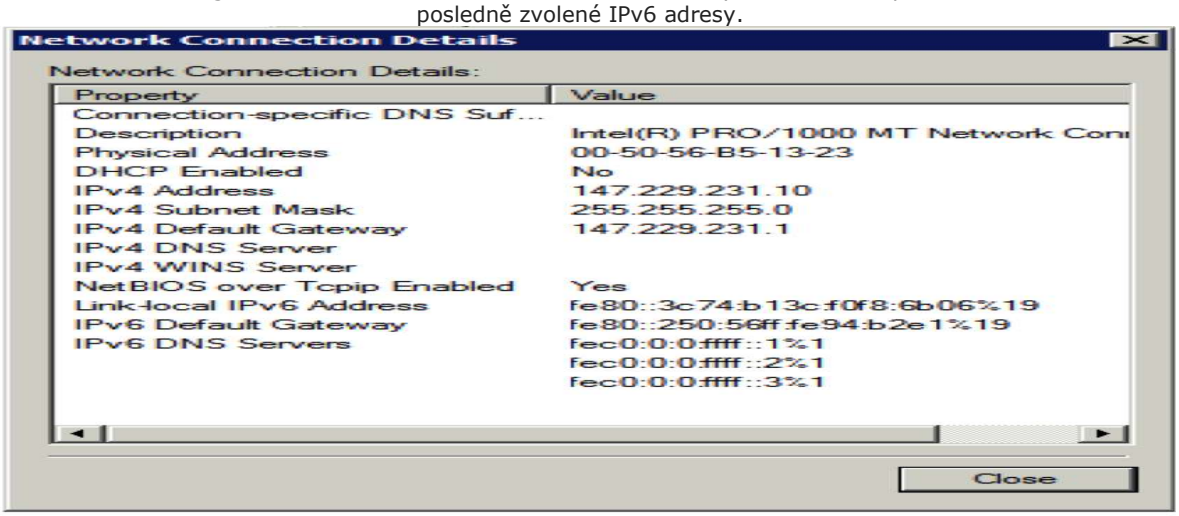
Duplicitní adresy a standardy

Dříve než si řekneme, jaké je v těchto situacích chování jednotlivých implementací, podívejme se co o detekci duplicitních adres hovoří standardy. Specifikace bezstavové autokonfigurace (RFC 4862) jasně říká, že v případě detekování duplicitní IPv6 adresy se má příslušné IPv6 rozhraní deaktivovat a podat o tom systému zprávu (například do logu). V té době specifikace předpokládala, že identifikátor síťového rozhraní dané IPv6 adresy bude vždy odvozený od adresy linkové vrstvy (MAC adresy). Výskyt duplicitní IPv6 adresy by tedy indikoval duplicitní MAC adresu v síti. S příchodem *Privacy Extensions* tj. RFC 3041 a RFC 4941, bylo toto chování upraveno. Pakliže je IPv6 adresa vytvořena s využitím *Privacy Extensions* a je detekována jako duplicitní, systém musí vytvořit novou IPv6 adresu a proces detekce duplicitní adresy opakovat. Počet opakování by pak měl být záležitostí nastavení operačního systému.

V roce 2006 doznal mechanismus detekce duplicitních adres dalšího vylepšení v podobě RFC 4429 - *Optimistic Duplicate Address Detection (DAD) for IPv6*. Původní návrhy totiž předpokládaly, že adresa, pro kterou probíhá detekce duplicity, se nesmí používat pro komunikaci. Použít ji lze až poté, co je potvrzená její unikátnost (resp. je nepotvrzena její duplicita). To ovšem může trvat v některých případech až 2 vteřiny. Výskyt kolize při použití 64 bitových identifikátorů je však velice vzácný jev, a tak RFC 4429 upravuje chování tak, že adresu je možné používat ihned a případná kolize se řeší až v následujícím kroku. Další změny v mechanismu detekce duplicitních adres přináší draft Enhanced Duplicate Address Detection, který řeší některé speciální situace, kdy koncový uzel současně přijímá data, a tedy sám sebe detekuje jako duplicitní uzel.

DAD a MS systémy

Jaké jsou praktické důsledky útoku? Zde opět bude zejména záležet na tvůrci implementace kódu detekce duplicitních adres v příslušném operačním systému. Systémy z dílny Microsoft mají ve výchozím stavu aktivovanou podporu *Privacy Extensions* a *Optimistic DAD*. Vygenerují si tedy *link-local* adresu a ihned ji použijí k odeslání zprávy *Výzva směrovači*. Pokud by došlo k detekci duplicitní IPv6 adresy - ať už náhodou nebo díky útočníkovi, systém se pokusí ještě o dalších 9 pokusů. Pokud jsou všechny vyhodnoceny jako duplicitní, tak v generování dalších adres se už nepokračuje. Dobrou zprávou nicméně je, že pokud systém nemá nakonfigurovanou IPv6 adresu, tak se v cca 3 minutových intervalech pokouší znova otestovat duplicitu posledně zvolené IPv6 adresy.




Pokud je IPv6 adresa na rozhraní nakonfigurována staticky, je sice provedeno ověření duplicitní adresy, ale adresa zůstane na rozhraní nakonfigurována bez ohledu na výsledek tohoto testu. Adresu je tedy možno využívat ke komunikaci.

DAD v Linuxu

V případě Linuxu se chování trochu odlišuje. *Link-local* adresa zůstane nakonfigurována na rozhraní bez ohledu na výsledek testu duplicity. V případě globální adresy odvozené od MAC adresy (EUI64) k nakonfigurování takové adresy nedojde. Systém pak s minutovou periodou provádí další testování duplicity této IPv6 adresy. V okamžiku, kdy už není duplicita detekována, systém IPv6 adresu na rozhraní nakonfiguruje. V případě, že je v linuxovém jádře aktivována podpora pro *Privacy Extensions*, systém provede 5 pokusů s náhodně vytvořenými IPv6 adresami. Pokud všechny testy duplicity dopadnou negativně, linuxové jádro vzdá generování dalších IPv6 adres až do restartu síťového rozhraní. V případě, že je IPv6 adresa konfigurována manuálně, nastaví se na rozhraní vždy, bez ohledu na výsledek testu duplicity.

***Poznámka:*** *Záleží na distribuci jestli náhodně generované adresy má ve výchozím stavu zapnuté nebo ne. Enterprise distribuce (např. RedHat, CentOS) používají většinou adresy odvozené od MAC adresy. Uživatelské distribuce pak zase většinou mají Privacy Extensions aktivované. Pokud byste chtěli nastavení na svém systému zkontrolovat, případně upravit, můžete chování systému ovlivnit prostřednictvím volby net.ipv6.conf.ethX.use_tempaddr. Volba use_tempaddr pak může nabývat několika voleb:*

- *0: Privacy Extensions jsou vypnuté*
- *1: Privacy Extensions jsou zapnuté, ale pro komunikaci nejsou preferované*
- *2: Privacy Extensions jsou zapnuté a preferované pro komunikaci*

V některých případech se může stát (např. při manuální konfiguraci), že test duplicity dopadne negativně a IPv6 adresa přesto bude nakonfigurovaná na příslušném rozhraní. Adresy totiž zůstanou ve stavu "tentative dadfailed" a operační systém tyto adresy nebude používat. V tomto případě je nutné nepoužívat příkaz ifconfig (který je stejně označen za zastaralý), protože se žádným způsobem nedozvíte, že adresa je v tomto stavu - jinými slovy ve výpisu příkazu ifconfig vypadá vše normálně. Prozradí vám to ale příkaz ip:

```
[root@test9 ~]# ip -6 addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
```

```
inet6 2001:67c:1220:f777::2/64 scope global tentative dadfailed
valid_lft forever preferred_lft forever
inet6 fe80::250:56ff:fe94:8a43/64 scope link tentative dadfailed
valid_lft forever preferred_lft forever
```

Poznamenejme jen, že v tomto "mrtvém" stavu už adresy zůstanou až do restartu síťového rozhraní nebo odkonfigurování a zpětném nakonfigurování IPv6 adres.

DAD a FreeBSD

Chování detekce duplicitních adres v systémech FreeBSD zřejmě nejvíce odpovídá tomu co požadují RFC. V případě, že je na rozhraní detekována duplicitní adresa je příslušné IPv6 rozhraní trvale deaktivováno. Zatím se nám však nepodařilo najít nějaký elegantní způsob, jak tento stav změnit, kromě restartu celého systému:

```
test-freebsd: # ifconfig em1
em1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
ether 00:50:56:b5:e4:3f
inet6 fe80::250:56ff:feb5:e43f%em1 prefixlen 64 duplicated scopeid 0x2
inet6 2001:67c:1220:f777::12 prefixlen 64 tentative
nd6 options=2b<PERFORMNUD,ACCEPT_RTADV,IFDISABLED,AUTO_LINKLOCAL>
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
```

Útočník na sítí - IPv6 se nekoná

Výskyt útočníka zneužívající mechanismu detekce duplicitních adres má tedy v síti poměrně nepříjemné důsledky. Takřka pro všechny systémy to znamená, že nezískají IPv6 konektivitu. Jak jsme si ukázali, některé systémy, jako například Linux nebo Windows implementují jisté opravné procedury, které umožní zpětné získaní IPv6 konektivity alespoň po odeznění útoku. Je zde opět vidět, že tvůrci operačních systémů upřednostňují větší robustnost před striktním dodržováním standardů. V praxi je pak chování každé implementace trošku jiné a zpravidla ne zcela dokonale zdokumentované. Pokud chceme znát přesně chování mechanismu detekce duplicitních adres příslušného systému, nezbývá nám zpravidla jiná možnost než analýza chování v testovacím prostředí.

K otestování je možné opět využít nástroj z balíku thc, který vytvoří z libovolného PC útočníka. Použití je opět velice jednoduché:

```
# ./dos-new-ip6 -X eth1
Started ICMP6 DAD Denial-of-Service (Press Control-C to end) ...
```

Jak je to v IPv4

Pokud se podíváme do světa IPv4 tak i tam je k dispozici podobný mechanismus pro detekci duplicitních adres. Pro podobné účely se používá mechanismus nazvaný "*Gratuitous ARP*". Nevyžádané (*Gratuitous*) ARP pakety v tomto případě značí ARP pakety, které pro normálním fungování protokolu ARP nejsou třeba. Princip mechanismu pro detekce duplicit je pak velice podobný jako u IPv6. Uzel vyšle do sítě zprávu *Gratuitous ARP Request*, kde jak cílovou, tak zdrojovou IP uvede svou vlastní. Pokud do určitého času nedorazí odpověď v podobě zprávy ARP Reply, adresa je považována za nekonfliktní. Uvedený mechanismus je dnes implementován ve všech běžně používaných operačních systémech. Má ovšem několik úskalí. Tento mechanismus totiž zasílá pouze jeden dotaz, když si zařízení konfiguruje adresu a už neprovádí žádné další detekce v průběhu. Pokud se tedy vyskytnou v síti dva počítače se stejnou adresou, později připojený počítač zahlásí chybu. První zařízení si problému často ale ani nevšimne. Zařízení pak zkouší adresu použít tak jak tak, což vede k problémům s konektivitou, jelikož si navzájem stále resetují TCP spojení. Proto bylo vydáno RFC 5227, které doplňuje detekci duplicitní adresy i pro IPv4. Způsob chování je pak stejný jako u protokolu IPv6.

Detekce duplicitních adres v IPv4 je však stále pouhou volitelnou nástavbou mající za úkol snadněji odhalit chyby v konfiguraci sítě. Nejsou zde tak striktní pravidla pro akci, kterou je nutné vykonat v případě detekce konfliktní adresy. V případě, že se do sítě připojují uživatelé, IPv4 adresy jsou typicky pod jednotnou autoritou v podobě DHCPv4 serveru, kde server eliminuje případné přiřazení konfliktní adresy. Na druhou stranu, DHCPv4 server nedokáže vyřešit situaci, kdy konfliktní adresa je nakonfigurována například staticky.

Je třeba mít také na paměti, že detekce duplicitních adres byla u protokolu IPv6 také původně zamýšlená jako pomocník pro snadnější odhalení chyb v síti. Vše se ale radikálně změnilo s příchodem *Privacy Extensions*, kde je mechanismus detekce duplicitních adres součásti běžného procesu volby adresy pro rozhraní. Pravidla jsou tu však vcelku striktní. Dle specifikací musí být příslušné IPv6 rozhraní trvale deaktivováno (tj. do restartu systému nebo rekonfigurace). Jak jsme si ukázali, tak operační systémy (Windows, Linux) po nějakém čase provádějí opětovné ověření adresy. Pokud se časem tato implementace dostane i na další zařízení tak se nejspíše vyhnete hypotetickému katastrofickému scénáři, kdy po DAD útoku budete muset oběhnout všechna PC, tiskárny, čtečky a kamery a provést jejich reboot.

Smysluplnost detekce duplicitních adres

Pomineme-li původní záměr detekce duplicitních adres, tj. eliminace konfiguračních chyb, tak vás jistě napadne, zda má vůbec smysl testovat náhodně generované adresy podle *Privacy Extensions*. Adresní prostor o velikosti $2^{64}$ je dostatečně velký, aby pravděpodobnost výskytu kolizí byla naprosto minimální. Touto otázkou se zabývá již dříve zmíněné RFC 4429, které uvádí, že v sítí s pěti tisíci uzly je pravděpodobnost výskytu kolize 5.4e-12. Otázkou tedy zůstává, zda při takto nízké pravděpodobnosti má vůbec smysl nějakou detekci duplicitních adres provádět. Ve prospěch kritiků hovoří i další argument, že mechanismus detekce duplicitních adres vyžaduje, aby všechna zařízení připojená v síti byla schopna na případnou výzvu reagovat. To je ovšem velký problém pro mobilní zařízení, která ve snaze šetřit baterie nemají trvalé zprovozněné síťové rozhraní. Zařízení připojené například k WiFi síti "uspí" IP stack, aby šetřilo baterii. Jak jsme si ukázali v jednom z předchozích článků zabývajícím se multicastem, některé zařízení konfiguraci IPv6 adresy řeší tak, že s každým probuzením se znovu provádí celý proces tvorby a ověření adresy. To ovšem s sebou nese další problémy v podobě vytváření nových záznamů v cache sousedů a v konečném důsledku zvýšenému multicast provozu na síti.

Obrana

V současné době nám není známo, že by existoval nějaký obranný prostředek proti tomuto typu útoku na straně síťové infrastruktury. Princip, který se používá pro zamezení obdobnému útoku v IPv4, tj. *ARP inspection*, o kterém jsme hovořili v druhém díle, není možné v tomto případě využít. Je to opět dáno tím, že IPv6 adresy jsou tvořeny nepredikovatelně na straně klienta. Ke zmírnění dopadu útoku je možné použít některé techniky *First-Hop-Security* (například větší segmentace sítě), které jsme popisovali již dříve.

Účinnou formu obrany proti tomuto útoku můžeme použít na straně samotného zařízení. Princip obrany je velmi jednoduchý - výsledek detekce duplicitních adres se zkrátka ignoruje a příslušná adresa je nakonfigurována vždy. Postup je sice v rozporu se standardy, ale v praxi by neměl narážet na problémy. V unixových systémech je možné chování zpravidla nastavit parametrem sysctl.

Linux:

```
# sysctl -w net.ipv6.conf.eth1.accept_dad=0
net.ipv6.conf.eth1.accept_dad = 0
```

V případě FreeBSD pak:

```
# sysctl -w net.inet6.ip6.dad_count=0
net.inet6.ip6.dad_count: 1 -> 0
```

U systémů Windows lze podobně celý mechanismus deaktivovat pomocí příkazu.

```
C:\Windows\system32>netsh interface ipv6 set privacy maxdadattempts=0
```

Zajímavé může být nastavení této vlastnosti i na směrovači připojující celou síť. U platformy Cisco nastavením dané volby na rozhraní:

```
SW-Cisco(config-if)# ipv6 nd dad attempts 0
```

Podobně tak u HP Comware

```
[SW-Comware-Vlan-interface220]ipv6 nd dad attempts 0
```

Případně u platformy HP ProCurve

```
SW-Procurve(config)# ipv6 nd dad attempts 0
```

Závěr

V dnešním díle jsme si ukázali nepříjemnosti, které nám může způsobit útočník zneužitím mechanismu detekce duplicitních adres. Původní záměr o vytvoření mechanismu, který by měl pomáhat eliminovat zejména konfigurační chyby, se díky použití *Privacy Extensions* začal postupně používat k trochu jinému účelu. Dnes mechanismus detekce duplicitních adres představuje pro útočníka prostředek, jak efektivně deaktivovat IPv6 stack při startu nebo rekonfiguraci koncového systému. Stejně jako v předchozích případech, v dnešní době nemusí být důsledky útoku nijak fatální, protože řada aplikací, zejména prohlížečů, je schopná přepnutí na protokol IPv4. Ve snaze vytvořit robustnější implementaci protokolu IPv6, pak operačních systémy, například Linux nebo Windows, implementují alespoň základní ochrané prostředky, které umožní získat zpět IPv6 konektivitu po odeznění útoku. Dochází tak nicméně k velkým odlišnostem daného mechanismu v reálných implemetacích a k rozporu se specifikacemi.

Obranné prostředky proti tomuto typu útoku jsou na straně síťové infrastruktury zatím spíše omezené. Částečně mohou pomoci některé techniky *First-Hop-Security*. Na straně koncových systémů je možné zpravidla konfiguračně deaktivovat celý mechanismus detekce duplicitních adres a tím udělat systém proti tomuto útoku imunní. Na druhou stranu ale pak může nastat problém při (byť velmi nepravděpodobném) opravdovém výskytu duplicitní adresy.

Tímto dílem jsme současně uzavřeli náš exkurs do bezpečnostních novinek protokolu IPv6. Jak jste si mohli všimnout, přestože protokol IPv6 je tu s námi již takřka 20 let, stále ještě zbývá mnoho problému k dořešení - od standardizace přes implementace v zařízeních až po konfigurační zvyklosti (Best Practices). Doufáme, že seriál přispěl k detailnějšímu porozumění některých vlastností protokolu IPv6 a doufáme, že vás neodradil od dalšího seznamování se s tímto protokolem. Jak jsme si popsali, řada útoků je velice dobře realizovatelná bez ohledu na to, zda je protokol v síti implementován či nikoliv. Dobré porozumění protokolu je tedy dnes nutností, bez které se kvalifikovaný správce neobejde.