

## TCP INJECTION - Je Internet opravdu ohrožen?

Nedávno proběhla hrozivá zpráva o jakési fatální chybě ohrožující samu existenci Internetu. Z původního krátkého vysvětlení, oč vlastně jde, vzniklo delší (a nepříliš záživné) povídání, jehož první část vám tímto nabízím

Onehdy jsem úplně zbytečně strávil hromadu času tím, že jsem s jedním člověkem mluvil o oné "život ohrožující chybě protokolu TCP", o níž psal mimo jiné [Pů tady](#) a o níž v různých novinách napsali spoustu různých pitomých blábolů. A protože takové bláboly vycházející z úplného nepochopení věci nemám rád, zkusím do celé záležitosti vnést trochu světla.

Protokol TCP (Transmission Control Protocol, definovaný v [RFC 793](#)) představuje jeden ze základních komunikačních protokolů Internetu. Prakticky všechny vyšší služby jej využívají jako svůj transportní protokol. Než se ponoříme do podrobnějšího popisu jeho fungování (abychom si následně mohli ukázat, kde tkví ona "chyba"), zaměříme se na dva důležité pojmy, které s problémem souvisejí a na něž budeme narážet. Půjde o **sekvenční číslo** a o **potvrzovací okno**.

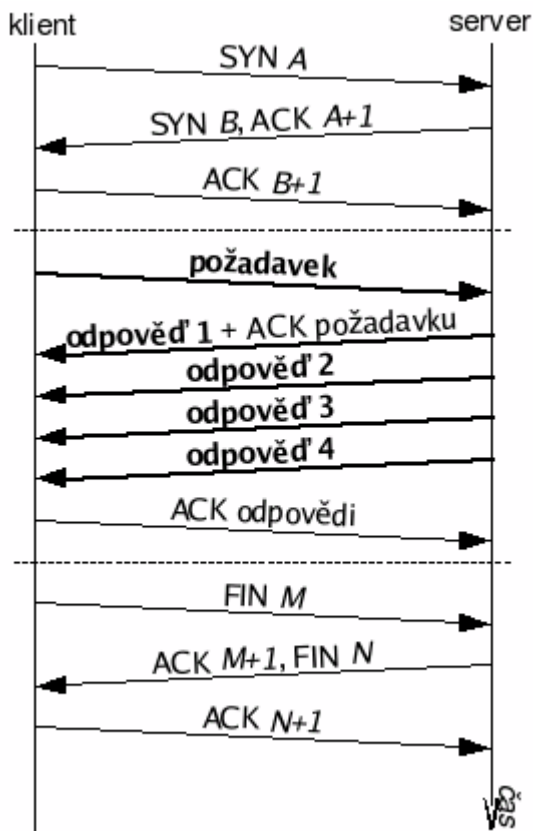
Každý bajt dat, který se protokolem TCP přenáší, má své sekvenční číslo. Smyslem sekvenčních čísel je, aby příjemce mohl detekovat ztracená nebo zduplikovaná data a aby mohl přijetí dat odesílateli potvrdit. Potvrzovací mechanismus je kumulativní, takže potvrzení bajtu se sekvenčním číslem  $X$  znamená, že byly přijaty *všechny* bajty až do  $X-1$ . Bajty se samozřejmě nepřenáší a nečíslují každý zvlášť - přenáší se v paketech a paket obsahuje pouze jediné sekvenční číslo. Například paket se sekvenčním číslem 372237 a s délkou (datové části) 1460 obsahuje bajty se sekvenčními čísly 372237 až 373696 a příjemce jej může potvrdit paketem s potvrzovacím číslem 373697. Není dokonce nutné (a ani žádoucí) potvrzovat každý přijatý paket - vedlo by to k neefektivnímu využití linky<sup>1)</sup>.

Přijatá data se potvrzují po větších skupinách, přičemž velikost bloku, který může zůstat najednou nepotvrzen aniž by si s tím jeho odesílatel dělal těžkou hlavu, je určena hodnotou zvanou **potvrzovací okno**. Potvrzovací okno je 16bitové číslo, takže může mít velikost maximálně 65535 bajtů - takže odesílatel může najednou (při velikosti datové části 1460 bajtů) poslat 44 paketů a teprve pak se starat o přijetí potvrzení<sup>2)</sup>. Velikost okna je mimochodem nesmírně důležitý dynamický parametr protokolu TCP - jeho hodnota se v průběhu přenosu může na základě poměrně sofistikovaných algoritmů měnit tak, aby došlo k co nejefektivnějšímu využití přenosové linky v závislosti na jejích parametrech a kvalitě<sup>3)</sup>.

Důležité je, že posloupnost sekvenčních čísel *nezačíná* od nuly. Každá strana si na počátku spojení náhodně<sup>4)</sup> zvolí své **počáteční sekvenční číslo**, sdělí ho protistraně a od této hodnoty se pak odvíjí další číslování. Toto chování je, jak dále uvidíme, významné z bezpečnostního pohledu. Dodejme, že sekvenční číslo je 32bitová hodnota.

Každé TCP spojení je jednoznačně určeno čtveřicí čísel (která samozřejmě zůstávají po celou dobu spojení neměnná). Jsou to (překvapivě) IP adresa a port jedné strany a IP adresa a port druhé strany. A teď už můžeme přejít k tomu, jak celé TCP spojení proběhne.

Takže máme klienta, který něco potřebuje po serveru. Klient pošle serveru paket s příznakem SYN (od SYNchronize) a sdělí mu v něm své počáteční sekvenční číslo  $A$ . Server na to reaguje paketem, který nese příznak ACK (od ACKnowledge) a potvrzuje klientem zvolené sekvenční číslo, a druhák nese příznak SYN a sděluje klientovi serverem zvolené počáteční sekvenční číslo  $B$ . Klient na to odpoví paketem ACK, kterým potvrdí serverem zvolené sekvenční číslo. Tato iniciální výměna se jmenuje **three-way handshaking** a po jejím proběhnutí je navázáno TCP spojení.



Následně klient posílá jeden paket s požadavkem (první datový bajt tohoto paketu bude mít sekvenční číslo odpovídající hodnotě A). Server se ve své odpovědi rozkecá a předvede nám hned několik triků najednou. V prvním paketu odpovědi zároveň pošle ACK požadavku<sup>5)</sup> klienta (potvrzovací číslo bude o jedničku větší než sekvenční číslo posledního datového bajtu v požadavku). Následně pošle tři další pakety s odpovědí, aniž by čekal na potvrzení každého paketu zvlášť. Spokojený klient ACKne poslední bajt posledního paketu odpovědi, čímž potvrdil přijetí všeho.

No a protože klient už nic víc nechce, zahájí ukončování spojení. Pošle serveru paket s příznakem FIN a se sekvenčním číslem M, které je právě na řadě. Server opět odpoví dvojjediným paketem - ACKne klientův FIN (a bere tak na vědomí ukončení spojení na straně klienta) a protože nemá důvod ve spojení pokračovat, zároveň hned pošle svůj FIN se sekvenčním číslem N. Klient tento paket ACKne (a bere na vědomí, že server rovněž ukončil spojení) a celá TCP relace tímto skončila<sup>6)</sup>.

Takže teď už víme, jak celé TCP spojení funguje, a můžeme na něj začít útočit. Podotýkám, že půjde o útok "z dálky" - útočník nemá možnost spojení odposlouchávat a/nebo do něj vstoupit (v takovém

případě by měl totiž mnohem širší možnosti a určitě by se nepokoušel o to, o čem budeme dál mluvit). Útočník nějak zjistil IP adresy a porty existujícího spojení (jak to zjistil, to ponechávám stranou, protože to může a nemusí být triviální) a nyní se snaží ze svého orlího hnízda kdesi v horách vpašovat do tohoto spojení zákeřný paket, který způsobí zhroucení serveru [seznam.cz](http://seznam.cz) a následnou nestabilitu osmi z deseti světových ekonomik.

Aby mohl útočník svým paketem něčemu ublížit, musí splnit několik podmínek:

1. Vytvořit paket tváříci se, že patří do probíhajícího spojení.
2. Dopřít paket na místo.
3. Zajistit, aby spojení paket "skouslo".

Ani jeden z úkonů není triviální.

Za účelem bodu 1 musí útočník znát IP adresy obou komunikujících stran a porty, které obě strany pro probíhající relaci používají. Následně musí vytvořit paket určený pro (řekněme) server a port, na němž běží spojení na serveru, jako odesílatele musí v paketu uvést (rozuměj, zfalšovat) IP adresu klienta a port, který používá klient. Pokud by kterákoliv z těchto čtyř hodnot nesouhlasila, paket nebude posouzen jako náležící do spojení, na něž útočník útočí, a tedy se nic nestane.

V rámci druhého bodu to má útočník nejjednodušší - prostě vytvořený paket odešle svým internetovým připojením a bude doufat, že paket dorazí na místo určení (tedy na server). Jak ale posléze uvidíme, právě na tomto bodě drtivá většina podobných pokusů o útok zákonitě ztroskotá.

Bod tři se konečně potkává s problematikou oné mediálně profláknuté "chyby". Už víme, že každý TCP paket v sobě nese nějaké to sekvenční číslo. Zatím jsme si ale neřekli, že pokud se v rámci spojení vyskytne paket se sekvenčním číslem "mimo pořadí" (tedy s číslem, které do probíhající posloupnosti nezapadá), bude paket tiše ignorován. Aby tedy útočník mohl svůj paket do spojení vecpat, musel by se trefit do sekvenčního čísla, které je teď zrovna na řadě. A teď tedy konečně to skandální odhalení.

Sekvenční číslo má délku 32 bitů a "všichni si vždycky mysleli", že úspěšně do spojení něco vpašovat prakticky nelze, protože šance na uhodnutí správného sekvenčního čísla je 2 na minus 32 (tedy děsně malá). Jenže "teď se přišlo na to", že ono to s tím sekvenčním číslem není až tak striktní. Spojení samozřejmě toleruje rozmezí sekvenčních čísel odpovídající šířce potvrzovacího okna a tedy při velikosti okna 64 KB se stavový prostor akceptovatelných sekvenčních čísel

zvětšuje na 2 na 16, při ještě větším okně je ještě větší - a šance útočníka tak nezanedbatelně rostou. BINGO! Útočník nepotřebuje miliardy pokusů, stačí mu řádově tisíce. Internet se hroutí v základech a kdyby [dr. Postela](#) blahé paměti už jednou šlak netrefil, trefil by jej jistě právě teď.

Dokonce i autorům této fámy o "chybě" (respektive o vlastnosti, která je známá od roku 1981, kdy byl protokol TCP navržen) bylo zjevné, že výše uvedená hrozba k dostatečné panice nestačí, takže bylo nutné to obohatit ještě o další skrytá rizika. Je totiž poměrně jasné, že útočník, který *nezná* obsah probíhající komunikace, má poměrně mizivou šanci vytvořit paket s daty a poslat jej ve správný okamžik na správnou adresu a se správným sekvenčním číslem tak, aby paket zapadl do probíhající komunikace a způsobil nějakou škodu vyšší síťové vrstvě. Koneckonců, tento typ útoku se již léta označuje jako **TCP Injection** a bez podrobnějších znalostí probíhající komunikace je vzdálenému útočníkovi na nic. Proto se upozorňuje na možnost **DoS** útoku (Deny of Service, tedy zrušení/zablokování/odepření služby, jak chcete). Útočník totiž může vyrobit (bez znalosti obsahu komunikace) paket FIN a trefí-li sekvenční číslo, komunikaci zabije.

A protože ani toto není žádná smrtící tragédie, bylo třeba najít ještě i vhodný cíl, který bude takovýmto útokem ohrožen. Pokud vám někdo zabije spojení mezi vaším prohlížečem a webovým serverem, stáhne si prohlížeč danou stránku znovu. Pokud někdo zabije přenos mailu, přenesení jej mailserver po chvíli znovu. Ovšem takový protokol BGP (Border Gateway Protocol, [RFC 1163](#) a [RFC 1164](#)), to je lákavý cíl a touha každého útočníka. Jednak jde o klíčový protokol, na němž je založena distribuce směrovacích údajů po celém Internetu, a druhák pracuje na základě trvalého TCP spojení mezi sousedícími směrovači. Zabij toto spojení a je zle. Tedy, samozřejmě že není, spojení se naváže znovu (a s jiným číslem portu), takže by bylo nutné neustále spojení zabíjet, aby to mělo nějaký efekt - ale to nevádí. Hrozba je zde!

Původně jsem se chtěl ještě rozepsat o tom, proč se útočníkovi ve většině případů nezdaří onen bod dvě ve výše uvedeném seznamu - konkrétně jsem se chtěl zmínit o věcech jako je reverse-path filtr, ingress filter a (konkrétně v případě BGP) TTL hack. Což jsou všechno běžně používané bezpečnostní mechanismy, přes něž podobný typ útoku nemá šanci projít. Ale už teď je to děsně dlouhé, takže popis těchto ochranných mechanismů odkládám na příště.

1) Lze snadno nahlédnout, že například v případě Ethernetové LAN, kde je round-trip řekněme 0,5 ms a velikost datové části TCP paketu 1460 bajtů bychom při potvrzování způsobem paket dat - potvrzení - paket dat - potvrzení mohli přenášet rychlostí maximálně  $1460 \cdot (1/0,0005)$  B/s, tedy nějakých 2,8 Mb/s, což je na 100 Mb/s síti nic moc. Na rychlé síti je proto nezbytné, aby bylo potvrzovací okno velké.

2) Na rychlých linkách s větší latencí (řekněme nějaká transoceánská optika) nebude stačit pro efektivní využití linky ani okno 64 KB, můžeme potřebovat okno klidně i 1 MB. Proto [RFC 1072](#) zavádí **Window Scale Options**, což je rozšíření TCP protokolu umožňující oběma stranám dohodnout si okno větší než 64 KB.

3) Viz [RFC 813](#) a cokoliv, co vám Google vrátí na "TCP Window" :-).

4) S náhodou je to u počítačů vždycky problém, takže přesnější bude říct, že počáteční sekvenční číslo se volí nějakým (implementačně závislým) co možná nejnepřesněji predikovatelným mechanismem. Mimochodem, hodnota počátečního sekvenčního čísla u několika po sobě jdoucích TCP spojení je jeden z významných faktorů tzv. "TCP fingerprintu" - jinak řečeno, jde o jednu z důležitých informací, na jejichž základě lze vzdáleně detekovat operační systém.

5) Tento trik spojení potvrzení požadavku a odpovědi do jednoho paketu se jmenuje **piggybacking** a použije se v případě, že server stihne zpracovat požadavek a vygenerovat odpověď do cca 200 ms. Pokud by to nestihl, pošle nejprve samostatně ACK požadavku a teprve ve druhém paketu by poslal první část odpovědi.

6) Variant ukončení TCP spojení je mnohem více. V našem případě inicioval ukončení klient, stejně

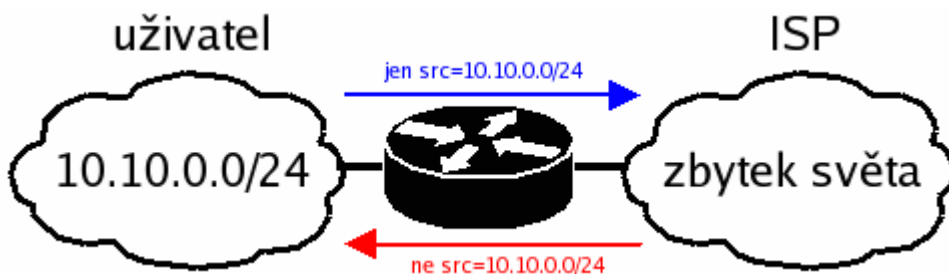
dobře ale může první paket FIN poslat server. Dokonce je možné, aby spojení ukončila jen jedna strana a druhá ho zatím neukončila (spojení bude ve stavu **half-open**).

## II.

**Posledně** jsme se dozvěděli, v čem tkví podstata **ohrožení Internetu** - díky toleranci TCP spojení na akceptované sekvenční číslo (která je dána velikostí potvrzovacího okna) může útočník vytvořit podvržený paket, jímž dokáže TCP spojení ukončit. Tvrdil jsem ale, že riziko takového útoku je značně přeceněné - zejména proto, že útočník bude mít málokdy šanci zajistit, aby paket se zfalšovanou IP adresou odesílatele dorazil k cíli. Dnes bych tedy rád ukázal, proč takové pakety nemají mnoho šancí.

Budeme hovořit o metodách, které obecně zabraňují přenosu paketů se zfalšovanou IP adresou odesílatele (takové falšování se označuje termínem **IP spoofing**). Tyto metody chrání nejen před výše zmíněným útokem, ale obecně před jakýmkoliv aktivitami, které jsou na falšování adres založeny<sup>1</sup>). A asi hned na úvod bude dobré optimisticky říct, že jde o metody běžně používané a většina seriózních ISP je ve své síti implementuje.

Začněme jednoduchým mechanismem, který se označuje termíny **ingress** a **egress** filtr - česky to neznámá nic jiného, než příchozí a odchozí. To je taky první a poslední věc, kterou si může (a měl by) nastavit normální smrtelník, která se netýká jen ISP. Tyto filtry se nastavují na hraničních směrovačích - hraničním směrovačem přitom rozumíme zařízení oddělující od sebe dvě administrativně nezávislé sítě (například síť koncového uživatele a síť jeho ISP, nebo síť ISP od jeho "upstream" ISP a podobně). Vše vysvětlí následující obrázek, kde máme směrovač oddělující síť nějakého uživatele (s IP adresami 10.10.0.0/24) a síť jeho ISP (která se chová jako "zbytek světa").



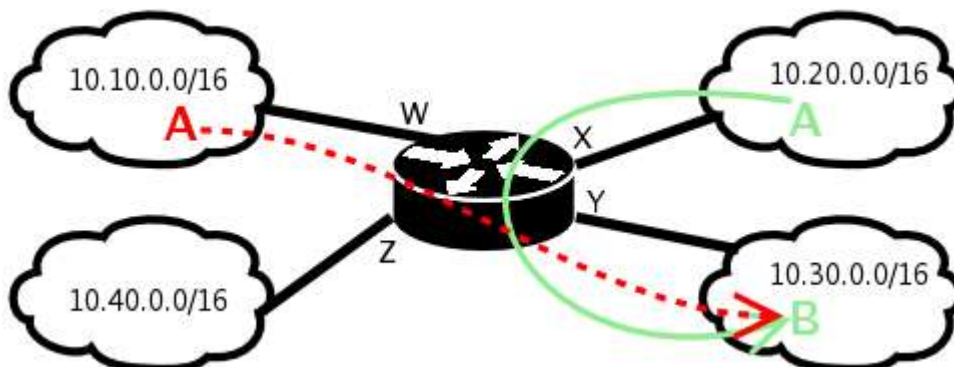
Ingress filtr je znázorněn červenou šipkou - dovnitř nepropouštíme pakety se zdrojovou adresou odpovídající vnitřní síti. Takové pakety totiž nemohou (pocitivou cestou) ve zbytku světa vzniknout. Tímto prostým opatřením se chráníme před všemi útoky založenými na tom, že nám do sítě vpašují paket s naší zdrojovou adresou. Ingressův kámoš egress (modře) pro změnu nepustí ven nic, co nemá zdrojovou adresu z naší sítě. Ten nechrání nás, ale je naším příspěvkem k blahu celého Internetu - nedovolí našim uživatelům vyrábět pakety se zfalšovanými adresami.

Vzhledem k tomu, že hraniční směrovač mezi uživatelem a ISP obvykle plní i roli firewallu, neměl by být problém na něm takováto pravidla zavést (a máte-li firewall nastaven slušně, určitě na něm taková nějaká pravidla jsou - považují se totiž za základ). Ještě poznámka k ingressu - filtrovat pakety s našimi zdrojovými adresami je pouze základní filtr. Rozumné je také filtrovat pakety se zdrojovými adresami z **bogonských** rozsahů. Bogoni jsou adresní rozsahy, které nemají na Internetu co pohledávat. Patří tam například privátní adresní rozsahy ([RFC 1918](#)), adresní rozsahy se speciálním použitím ([RFC 3330](#)), rozsahy [IANAou](#) rezervované a tak všelijak<sup>2</sup>). Viz třeba [tady](#), anebo ještě lépe - aktuální seznam bogonů v různých formátech je k dispozici [tady](#).

Podobné, ale složitější filtry se obvykle nastavují i na hraničních směrovačích například mezi různými ISP. Tam se ale zohledňuje spousta věcí - alokované adresní rozsahy, peeringové politiky a podobně. Princip je ale stejný. Použitelnost tohoto druhu filtrace je ale omezena pouze na směrovače, kde je poměrně jasně definováno, co může a nemůže být na které straně. V případě různých směrovačů tvořících infrastrukturu sítě by byla filtrační pravidla neúnosně složitá, nehledě k tomu, že takové směrovače obvykle používají dynamické směrovací protokoly, takže zítra tam klidně může znamenat už včera a nikdy nevíte, co odkud přijít může a nesmí.

Proto existuje další trik, označovaný jako **reverse-path** filtr. Ten nespolehá na žádná nastavená

filtrační pravidla a řídí se pouze tím, co směrovač vždy ví, protože to vědět musí - filtr se řídí směrovací tabulkou. Mějme směrovač, který má ke svým čtyřem rozhraním (W, X, Y a Z) nějak připojeny adresní rozsahy 10.10.0.0/16, 10.20.0.0/16 atd.



Jak vlastně takový směrovač funguje? Má směrovací tabulku, která mu říká "když přijmu rozhraním X paket ze zdrojové adresy A určený pro cílovou adresu B, pošlu jej dál rozhraním Y<sup>3)</sup>". No a RP-filtr toto rozhodování doplňuje o jednoduchou úvahu: "Mám tady paket z adresy A pro adresu B, přijatý rozhraním X. Podle tabulky bych jej měl poslat rozhraním Y. *Co bych dělal, kdyby mi rozhraním Y přišel paket z adresy B pro adresu A? Poslal bych jej rozhraním X? Ano, poslal! OK, předám paket dál.*" (Viz zelená šipka.) Ovšem může se stát také "Mám tady paket z adresy A pro adresu B, přijatý rozhraním W. Podle tabulky bych jej měl poslat rozhraním Y. *Co bych dělal, kdyby mi rozhraním Y přišel paket z adresy B pro adresu A? Poslal bych jej rozhraním W? Kdepák, poslal bych jej rozhraním X. Takže paket zahazuju.*" (Viz červená šipka.)

Všimněte si, že na rozdíl od ingress/egress filtrů se v tomto případě nenastavují žádná pravidla říkající, co odkud kam může a co ne. Veškeré rozhodování se odvíjí jen od obsahu směrovací tabulky - takže i pokud se směrování dynamicky mění, RP-filtr bude stále fungovat správně. Jako každý elegantní, vtipný a účinný mechanismus má RP-filtr i své úskalí. Jeho fungování je totiž založeno na jednom předpokladu: že totiž cesta z A do B je stejná jako cesta z B do A. A tento předpoklad platí jen v případě **symetrického** směrování.

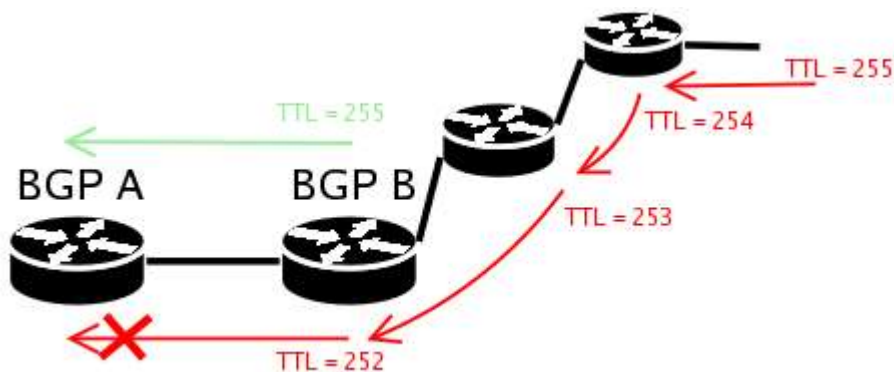
Již zmíněný protokol BGP je ovšem typickým představitelem **asymetrického** směrování. To, že cesta z A do B vede někudy nám neříká nic o tom, kudy vede cesta z B do A<sup>4)</sup>. Nelze tedy použít RP-filtr a i jakoukoliv jinou filtraci je třeba pečlivě zvažovat - světový routing se totiž může převrtět všelijak a není mnoho jistot, kterých se lze držet :-). BGP směrovače nicméně představují poněkud elitářský svět a očekává se, že je chrání různí ti poskoci kolem nich. Protože ale výše popsaný útok představuje údajně hrozbu zejména pro BGP směrovače, je načase zmínit se o posledním a mimořádně vtipném ochranném triku, který tyto směrovače používají.

Řeč bude o triku zvaném **BGP TTL Hack**, nejprve ale trocha teorie. BGP směrovače mají dvě základní charakteristiky - nekomunikují každý s každým (nebo kdekdo s kdekým, jak chcete), baví se pouze se svými partnery. Druhá důležitá věc je ta, že partneři jsou spolu ve většině případů přímo propojeni<sup>5)</sup>. No a teď malá odbočka k fungování protokolu IP.

Jedním z údajů v hlavičce IP paketu je hodnota **TTL** - Time-to-live, česky životnost. Když se odněkud někam vydá na cestu paket, má nastavenou nějakou počáteční hodnotu TTL, řekněme 64. Každý směrovač, jímž paket prochází, pak má následující povinnost: snížit hodnotu TTL o jedničku a v případě, že výsledkem bude nula, paket zahodit a nesměrovat jej dále<sup>6) 7)</sup>. Smyslem hodnoty TTL je ochrana před směrovacími smyčkami - pokud by díky chybně nastavenému směrování někde v síti vznikla směrovací smyčka, cirkulovaly by v ní pakety věčně. Díky TTL ale budou v takém případě po nějakém počtu hopů zahozeny. Poslední důležitá informace je ta, že TTL je 8bitový údaj, má tedy hodnotu maximálně 255.

A teď už zpátky k BGP TTL Hacku. Je založen na tom, že administrátoři partnerských směrovačů se domluví na následujícím chování: Směrovač A bude při komunikaci se svým partnerem B ve svých paketech nastavovat TTL vždy 255. Protože po cestě k B žádný další směrovač není, musí B přijmout rovněž paket s TTL 255. Pokud by měl paket (bavíme se samozřejmě o paketech směrovacího protokolu BGP, kterým si partneři povídají mezi sebou) jiné TTL než 255, směrovač B jej bude ignorovat. Pokud by chtěl do spojení proniknout nějaký útočník, musel by tak učinit "zvenčí", přes směrovač A či B. A i pokud falešný paket přes některý ze směrovačů projde, bude mu sníženo jeho TTL - takže nikdy nebude rovno 255, což je maximální možná počáteční

hodnota:



Zelená je oprávněná komunikace ze směrovače B na směrovač A s TTL rovným 255. Červený útočník někde vzadu sice také posílá své útočné pakety s TTL 255, nemůže ale ovlivnit to, že po trase jsou další směrovače, které TTL postupně snižují, takže na směrovač A už paket dorazí s TTL menším a bude zahozen. (Mimochodem, při správně nastaveném filtru na směrovači B by (bez ohledu na TTL) došlo k zahazení útočného paketu už na tomto směrovači.)

Tento ochranný mechanismus je samozřejmě založen na vzájemné spolupráci a důvěře mezi partnerskými směrovači. A zatímco životní partnerce lze důvěřovat jen nepřímo úměrně v závislosti na velikosti vaší životní pojistky, k BGP partnerovi lze mít důvěru výrazně vyšší, protože bezproblémová činnost protokolu BGP je ve vašem společném zájmu.

A to je všechno o různých ochranných mechanismech. Existuje sice ještě třeba [RFC 2385](#), které zavádí ochranu BGP relací prostřednictvím MD5 podpisu TCP hlaviček - nicméně všeobecně nepanuje příliš velká důvěra k tomu, že jde o dobrou věc. A určitě existuje i celá spousta jiných věcí, o kterých nevím nebo na které jsem zapomněl, ale to snad nevádí :-). Případné dotazy a připomínky vítám, třeba mě časem vyprovokují k dalšímu článku.

- 1) Falšování IP adresy odesílatele je docela vtipný trik, který se dá použít ke spoustě zábavných operací - jeho jedinou nevýhodou je to, že útočník nemá šanci dostat na své pakety žádnou odpověď, což ale mnohdy nevádí. Zkusme si třeba představit SMTP server, který přijímá spojení jen z určitých IP adres - útočník může takovou falešnou adresu použít a daného serveru s klidem zneužít k odesílání spamů. Komunikace protokolem SMTP je totiž tak "profláklá", že máte velmi dobrou šanci správně mail odeslat, aniž byste dostávali odpovědi poštovního serveru.
- 2) Trochu opatrně je nutné postupovat při filtraci privátních rozsahů ([RFC 1918](#)). Někteří ISP je ve svých sítích používají a jejich odfiltrování byste si mohli dost ublížit. (Koneckonců, i v tom obrázku uvádím jako klientské adresy rozsah 10.10.0.0/24, což jsou adresy z privátního rozsahu.)
- 3) Ve většině případů je rozhodování směrovače ještě jednodušší - omezuje se na "mám paket pro adresu B, pošlu jej rozhraním Y". Zdrojová adresa a zdrojové rozhraní obvykle nehraje roli. Za určitých okolností ji ale hrát může (pak se mluví o **source policy routing**, což je trik, kterým je například možné relativně jednoduše zajistit třeba load-balancing mezi dvěma ISP).
- 4) Asymetrické směrování má své specifické kouzlo - zejména pro masochisty. V tomto světě totiž termín "diagnostika problémů" získává úplně jinou dimenzi. Moc vám nepomohou ani skalní nástroje jako traceroute - ten vám poví trasu od "vás" k "nim", o opačném směru cudně mlčí.
- 5) To neznamená, že bydlí líčko na líčko ve stejném racku, znamená to jen, že sdílejí stejnou propojovací IP podsít - jinak řečeno, není mezi nimi žádný další směrovač, baví se přímo.
- 6) V takovém případě navíc tento směrovač musí poslat ICMP zprávu "TTL Exceeded" původnímu zdroji paketu. Díky ní funguje třeba právě již zmíněný traceroute. <sup>7)</sup> Abychom byli přesní, TTL původně definuje životnost v sekundách a každý směrovač snižuje jeho hodnotu o počet sekund,

které mu zpracování paketu trvalo - *nejméně však* o jedničku. Vzhledem k rychlosti současných zařízení ovšem není přehnaně smělé předpokládat, že každý směrovač sníží hodnotu TTL o 1.