VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

Síťové aplikace a správa sítí

2018/2019

Projekt

Varianta 3: Nástroje monitorující a generující zprávy jednoduchých distance-vectore protokolů

Pavel Chlubna (xchlub02) Brno, 10. listopadu 2018

Obsah

Úvod

Tento projekt se zabývá především protokoly RIP verze 1 a 2 a také protokolem RIPng, o těchto protokolech budu psát níže. Právě s těmito protokoly pracuje program, který jsem vytvořil.

Celý projekt se dá rozdělit do dvou částí. První z nich bylo vytvoření nástroje pro odposlech komunikace v síti, se zaměřením na již zmíněné packety nesoucí data protokolu RIPv1, RIPv2 nebo RIPng. Následně také vypsání všech dat, který tento packet nese uživateli na výstup.

Druhá část se zabývá naprogramováním podvrhávače falešných zpráv, který posílá odpovědi do sítě s daty o routování, které mu byly dány uživatelem.
Pomocí těchto nástrojů se dá realizovat útok na síť, kde se k synchronizaci směrovacích tabulek routerů používá protokol RIPng.

K tomu abych docílil odposlechu byly použity knihovní funkce libpcap a k posílání dat do sítě BSD socket. Jak byly tyto technologie použity si ukážeme až později.

Zadání

Vaším úkolem je:

1:
Nastudovat si směrovací protokoly RIP a RIPng;

2:
Naprogramovat sniffer RIPv1, RIPv2 a RIPng zpráv;

3:
Naprogramovat podvrhávač falešných RIPng Response zpráv;

4:
Za použití obou nástrojů, které jste si předpřipravili v předchozím bodě pak provést úspěšný útok;

Zadání dokumentace

V dobré dokumentaci se budou následující části: titulní strana, obsah, logické strukturování textu, výcuc relevantních informací z nastudované literatury, popis zajímavějších pasáží implementace, sekce o testování, bibliografie, popisy k řešení bonusových zadání.

Routování

V informatice používáme směrování při komunikaci na síti, jde o určení cesty datagramu, takové, aby byla co nejefektivnější. Směrování se provádí na síťové vrstvě modelu ISO/OSI, tedy na třetí vrstvě (Internetová vrstva v modelu TCP/IP). Jelikož topologie sítě bývá většinou velice složitá a mezi adresátem a příjemcem je tak velké množství prvků sítě, neřeší se cesta k příjemci celá, nýbrž se podle routovací tabulky pouze určí další krok, čili na které rozhraní se mají tyto data odeslat.

Zpravidla dělíme routování na statické a dynamické. U statického routování naplní routovací tabulku “manuálně” správce sítě a tyto data v tabulce jsou neměnná, dokud je správce opět nezmění. Nepoužíváme tedy žádné routovací protokoly.
Výhodou statického routování je bezpečnost, vyplívající z absence routovacích protokolů, které se mohou použít k útoku na síť.
Nevýhodou je však veliká náročnost na implementaci. Správce sítě musí znát detailně topologií a tato metoda se dá použít v síti s maximálně třemi až čtyřmi prvky, poté už by byla náročnost příliš veliká. V případě, že by došlo ke změně v topologií sítě, musíme opět tabulky na všech routerech ručně přepsat.

Dynamické routování nás od starosti s vyplňováním tabulek oprostí. Využíváme routovací protokoly, které si mezi sebou routery posílají a je tím tak zajištěno automatické vyplnění routovacích tabulek.
Velikou výhodou je, že routery si mezi sebou pošlou sítě, ke kterým mají přístup a ostatní pak vědí, jak se dostat k určité síti. Mimo jiné, pokud například do sítě přidáme další router, mezi ostatními se rozšíří zpráva o změně v topologii a jsou schopni na ni reagovat.
Nevýhodou je však snížená bezpečnost, jelikož je možné do sítě odeslat protokol s falešnými daty, jak jsme se o tom přesvědčili v tomto projektu.[1]

Routovací protokol

Používá se při dynamickém routování, když mezi sebou jednotlivé směrovače pomocí tohoto protokolu sdílí data, které následně pomáhají pří výběru nejlepší cesty mezi libovolnými dvěma uzly v síti, přičemž tuto cestu vybírají směrovací algoritmy.

Při spuštění směrovače je směrovací tabulka naplněna pouze sítí, ke které má router přímý přístup. Následně router odešle své informace ze směrovací tabulky sousednímu routeru a takto se informace rozšíří do celé sítě a tak každý router bude znát topologii sítě.[2]

Dělení protokolů

Protokoly můžeme dělit do několika skupin podle různých kritérií, jako je pole působnosti protokolu anebo způsob, jakým probíhá výpočet nejlepší cesty.

Pokud bychom chtěli protokoly řadit podle toho, kde se používají, můžeme je rozdělit na Interior gateway protocol (IGP) a Exterior gateway protocol (EGP).
IGP je protokol, který nasazujeme uvnitř jednoho autonomního systému, tedy uvnitř jedné skupiny routerů, které spadají do stejné sítě. Zástupcem je například OSPF, nebo RIP.
EGP naopak využíváme mimo sítě. Jeho úkolem je právě mezi těmito autonomními systémy routovat a propojit je tak. Příkladem je BGP.

Rozdělení podle způsobu výpočtu nejlepší cesty je na dva typy. Distance-vector, Link-state a Path-vector.

Jelikož RIP protokol spadá do Distance-vector, popíšeme si tento typ routovacího protokolu. Tyto protokoly plní svoji routovací tabulku informacemi o vektoru, čili vzdálenosti do dané sítě. Svoje informace v tabulce periodicky odesílá svým sousedům, kteří si aktualizují svojí tabulku a opět odešlou informace dál do sítě. Počet metrik, kterými se vypočítává nejlepší cesta, může být jedna (počet hopů do sítě u RIP protokolu) nebo více metrik (počítá se například i s propustností linky - EIGRP).[4]

Routing Information Protocol

Routing Information Protocol (RIP) patří mezi nejstarší, ačkoliv stále používaný, routovací protokoly používané při konfiguraci sítí. První nasazení protokolu proběhlo v roce 1969. Jelikož různé firmy používaly svoji vlastní verzi, došlo později k sjednocení pod jeden standard RFC 1058.

RIP se řadí do Interior gateway protocols a je typu distance-vector. Jako metriku pro výpočet nejlepší cesty do jiné sítě používá hop count, což je počet prvků (počet skoků), které na cestě do dané sítě jsou. Jelikož má však hop count omezený počet, konkrétně 15 z důvodu zabránění vzniku smyček, je použitelný pouze v malých a středních sítích.[4]

V dnešní době je stále používán, kvůli jeho jednoduchosti konfigurace a jeho jednoduchost. Avšak mezi nevýhody patří velká časová náročnost na konvergenci, tím myslíme čas potřebný k reakci a přizpůsobením se změnám v topologií sítě. Protože protokol funguje tak, že každý směrovač jednou za 30 sekund pošle svoji tabulku sousedům, je tato doba konvergence příliš dlouhá. Také dochází k velkému zatížení sítě (v dnešní době je problém řešen tím, že nedochází k rozeslání tabulek současně v jednu chvíli, ale každý router jí posílá samostatně).[3]

RIPv1

První verze protokolu specifikována ve standardu RFC 1058, která pracuje s třídami IPv4 adres. Jelikož tato verze nepracuje s beztřídním řazením adres, nepřenáší ani informace o masce, protože maska byla dána příslušností adresy do dané třídy.
Mimo jiné také protokol neumožňoval žádnou autentizaci routerů, což ulehčovalo útoky na sítě.[3]

RIPv2

Vznikla v roce 1993, jako reakce na zavedení Classless Inter-Domain Routing (beztřídní směrování) a je definována standardem RFC 2453. Druhá verze tohoto protokolu přináší několik vylepšení oproti té původní, jako je podpora beztřídního směrování, autentizace routerů nebo způsob rozšiřování informací z routing tabulky. Byl navržen tak, aby byl kompatibilní se starší verzí.
Jelikož se přešlo na systém beztřídního směrování, tak protokol této verze nese také informace o masce sítě.
Další změnou je adresa, na kterou jsou posílány informace z tabulky. Dřívější verze totiž využívala ke sdílení informací adresu broadcastu, čímž docházelo ke zvýšení vytížení sítě. Druhá verze k výměně dat routovací tabulky používá adresu 224.0.0.9 (multicast).
Začala se také používat autentizace mezi routery, avšak tyto hesla nejsou zakódována, takže bezpečnost stále nebyla dostatečná. Autentizaci zabezpečenou šifrováním přineslo až použití MD5, které bylo zavedeno v roce 1997.[3]

RIPng

RIP next generation je další verzí, u které byla zavedena podpora IPv6 adres. Protokol je definován ve standardu RFC 2080.
Rozdíl oproti starším verzím je kromě podpory IPv6 také ve způsobu zabezpečení. U této verze je totiž zabezpečení zajištěno technologií IPsec.[3]

Implementace

RIP sniffer

Program slouží na odposlech síťové aplikace, z které následně odfiltruje komunikaci na portu 520 a 521, tedy pakety nesoucí informace s RIP protokolem. Správné funkčnosti bylo docíleno pomocí funkcí z knihovny libpcap, které tento odposlech umožňují.

Abychom správně odposlouchávali na daném rozhraní, je nutno zkontrolovat, že je možné k rozhraní získat potřebné informace, jako je IP adresa a maska. Tímto krokem také potvrdíme, že rozhraní je validní (funkce *pcap_lookupnet*). Následně již můžeme toto rozhraní otevřít pro odposlech (funkce *pcap_open_live*).

Pro správné filtrování je třeba nastavit filtr na protokol UDP pro porty 520 a 521. Ve výsledném programu vypadá nastavení filtru takto *"portrange 520-521 and udp"* a tento filtr je poté nutno zkompilovat (funkce *pcap_compile*). Byl-li tento krok úspěšný, výsledný zkompilovaný filtr již můžeme nastavit (funkcí *pcap_setfilter*).

Pokud všechno nastavení proběhlo úspěšně, můžeme již začít odposlouchávat komunikaci na rozhraní. V knihovně libpcap je k tomu vhodná funkce *pcap_loop*. Druhý parametr této funkce určuje počet paketů, které mají být zpracovány, pokud mu však zadáme na toto místo -1, bude se jednat o nekonečnou smyčku. Pokaždé když zachytíme packet, který odpovídá parametrům daným ve filtru, je zavolána funkce, ve které už packet ošetříme.

Abychom se dostali k informacím obsaženým v RIP paketu, musíme nejdříve odstranit ethernetovou, IP a udp hlavičku. Následně už máme k dispozici data routovacího protokolu. Jeho hlavička je pro všechny verze stejná a obsahuje informace o příkazu a verzi protokolu (RIPng podle toho však nepoznáme, je nutno rozeznat podle verze adresy v IP hlavičce). Dále se už dostáváme k Entry table protokolu, kde jsou obsaženy informace o síti a také případná autentifikace.

Jelikož program běží do nekonečna a je třeba ho ukončit signálem (kombinace ctrl + c), bylo také implementováno ošetření tohoto signálu. V tomto ošetření se volá funkce *pcap_breakloop*, která ukončí smyčku a poté už dochází k vyčištění paměti a program je řádně ukončen.

RIP response

Tato část má za úkol provést útok na síť tím, že podvrhne falešný RIP paket. K docílení musíme vytvořit strukturu a naplnit ji daty ze vstupu, následně utvořit socket a odeslat přes něj do sítě tento paket.

Pro vytvoření paketu je nutno alokovat do paměti strukturu hlavičky a entry tabulky a naplnit ji daty. Velikost alokovaného místa záleží na tom, jestli byl na vstup zadán parametr next hop adresy. V tom případě musíme alokovat entry tabulku dvakrát.

Dále je nutno vytvořit socket, skrze který proběhne odeslání dat. Socket musíme nastavit tak, aby používal IPv6 a protokol UDP. Port je nastaven na 521, tedy RIPng a do adresy je dáno *ff02::9* (multicast pro RIPng). Mimo jiné je nutno socketu nastavit také počet hopů na 255. Následně je již možno data odeslat, uzavřít socket a uvolnit alokovanou paměť.

Zajímavé pasáže

Následující kód ukazuje převod hesla při autentizaci MD5. Pomocí funkce *snprintf* formátujeme řetězec na hexadecimální podobu, aby byl čitelný pro uživatele.

```
if(ntohs(entry->route_tag) == 1)
{
	char pswd[32] = {0};

	for(int i = 0; i < 16; ++i)
	{
		snprintf(pswd + (i * 2), 3,
			"%02x",entry->data.authentication[i]);
	}

	printf("MD5 Password:\t%s\n",pswd);
}
```

Jelikož v protokol může obsahovat více entry tabulek, tento for iteruje, dokud se v paketu nachází další. Následně volá funkce podle toho, zdali se jedná o tabulku s informacemi o síti, nebo o tabulku s autentifikací.

```
for(rip_length -= HEADER_LEN; rip_length >= ENTRY_LEN; rip_length -= ENTRY_LEN)
{
            if(entry->addr_faml == 0xFFFF)
            {
                  //this entry contains authentication
                  rip_authentication_print(entry);
            }
            else
            {
                  rip_entry_print(entry, rip_hdr->version);
            }

            entry = (rip_entry_table *) ((u_char *) entry + ENTRY_LEN);
}
```

Funkčnost

Zde jsou přiloženy snímky obrazovky, které zobrazují přeložitelnost na testovacím virtuálním počítači isa2015, serveru Merlin a také na Fedoře. Dále jsou také přiloženy snímky výsledků valgrind z obou programů.

```
xchlub02@merlin: ~/isa$ make
gcc -std=gnu11 -Wall -Wextra -Werror -pedantic -o myripsniffer myripsniffer.c -lpcap
gcc -std=gnu11 -Wall -Wextra -Werror -pedantic -o myripresponse myripresponse.c -lpcap
xchlub02@merlin: ~/isa$
```

```
[chlubnap@localhost ISAlock]$ make
gcc -std=gnu11 -Wall -Wextra -Werror -pedantic -o myripsniffer myripsniffer.c -lpcap
gcc -std=gnu11 -Wall -Wextra -Werror -pedantic -o myripresponse myripresponse.c -lpcap
[chlubnap@localhost ISAlock]$
```

```
isa2015@isa2015:~/Downloads/isa$ make
gcc -std=gnu11 -Wall -Wextra -Werror -pedantic -o myripsniffer myripsniffer.c -lpcap
gcc -std=gnu11 -Wall -Wextra -Werror -pedantic -o myripresponse myripresponse.c -lpcap
isa2015@isa2015:~/Downloads/isa$
```

Valgrind pro *myripresponse*

```
[chlubnap@localhost ISAlock]$ sudo valgrind --leak-check=full ./myripresponse -i enp2s0 -r 2001:db8:0:abcd::/64
==3255== Memcheck, a memory error detector
==3255== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3255== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==3255== Command: ./myripresponse -i enp2s0 -r 2001:db8:0:abcd::/64
==3255==
==3255==
==3255== HEAP SUMMARY:
==3255==     in use at exit: 0 bytes in 0 blocks
==3255==   total heap usage: 1 allocs, 1 frees, 24 bytes allocated
==3255==
==3255== All heap blocks were freed -- no leaks are possible
==3255==
==3255== For counts of detected and suppressed errors, rerun with: -v
==3255== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[chlubnap@localhost ISAlock]$
```

Valgrind pro *myripsniffer*

```
==3253== HEAP SUMMARY:
==3253==     in use at exit: 0 bytes in 0 blocks
==3253==   total heap usage: 72 allocs, 72 frees, 53,655 bytes allocated
==3253==
==3253== All heap blocks were freed -- no leaks are possible
==3253==
==3253== For counts of detected and suppressed errors, rerun with: -v
==3253== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[chlubnap@localhost ISAlock]$
```

V testovacím virtuálním stroji isa2015 se bohužel nachází starší verze knihovny libpcap a ve funkci *pcap_compile* může docházet k memory leak (viz https://sourceforge.net/p/libpcap/bugs/107/).

Testování

Sniffer

Výstup programu jsem vypsal do souboru sniffer, který je zahrnut v adresáři

IPv6 sítě

IPv6 Prefix: fd00::
Route tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:ce:2df0::
Route tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:10d:2d3f::
Route tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:8dc:63::
Route tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:960:1450::
Route tag: 0
Prefix Length: 64
Metric: 1

IPv4 sítě

IP Address: 10.0.0.0
Mask: 255.255.255.0
Route tag: 0
AFI: 2 (IP)
Next Hop: 0.0.0.0
Metric: 1

IP Address:	10.48.50.0
Mask:	255.255.255.0
Route tag:	0
AFI:	2 (IP)
Next Hop:	0.0.0.0
Metric:	1

IP Address:	10.104.117.0
Mask:	255.255.255.0
Route tag:	0
AFI:	2 (IP)
Next Hop:	0.0.0.0
Metric:	1

IP Address:	10.108.207.0
Mask:	255.255.255.0
Route tag:	0
AFI:	2 (IP)
Next Hop:	0.0.0.0
Metric:	1

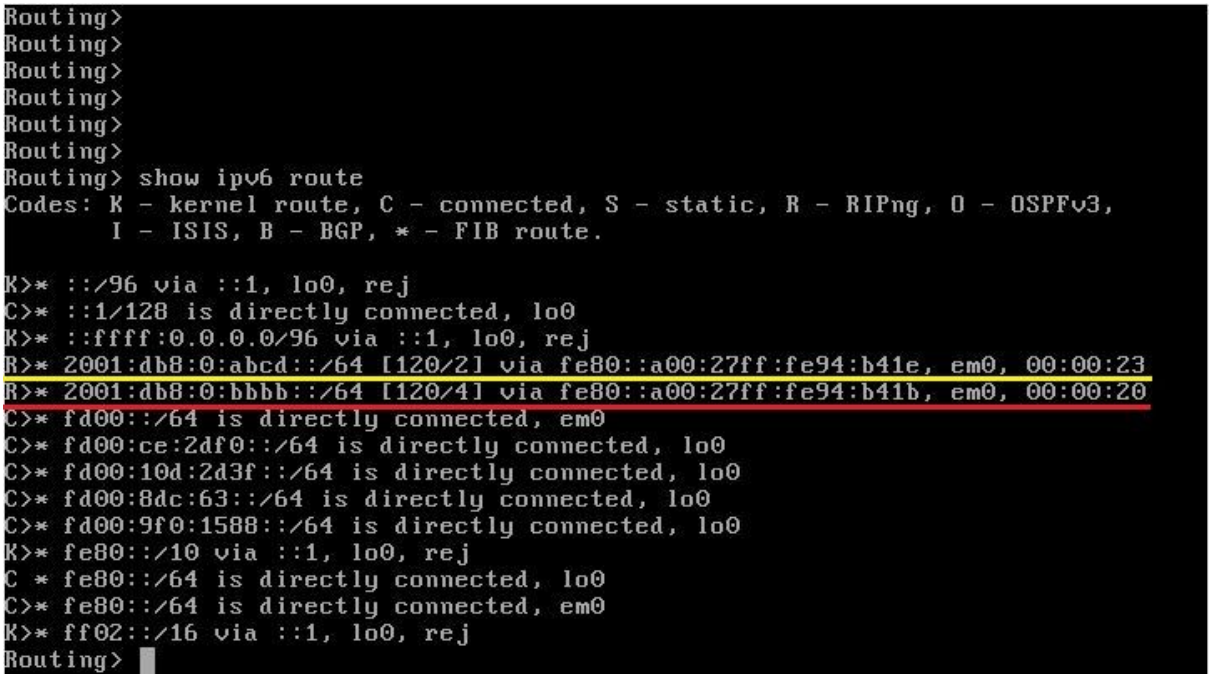
IP Address:	10.221.98.0
Mask:	255.255.255.0
Route tag:	0
AFI:	2 (IP)
Next Hop:	0.0.0.0
Metric:	1

Autentifikace

Zpráva RIPv2 byla zabezpečena jednoduchým heslem: **ISA>28612f48698**

Response

Na virtuální router jsem poslal pomocí programu dvě sítě, jednu podle zadání a druhou pro znázornění zprávy, která obsahuje nexthop.



Žlutou čárou je podtržena síť, která je poslána dle zadání, bez zadaného nexthopu.
Červeno síť, které byla navíc přidána adresa nexthop *fr80::a00:27ff:fe94:b41b*.

V odevzdaném adresáři se navíc nachází soubor obsahující výpis snifferu, který zachytil tyto odesílané zprávy (název souboru: response).

Návod

Projekt je nejdříve nutno přeložit. To učiníme pomocí příkazu *make*, který přeloží projekt a vytvoří dva spustitelné soubory, jejichž spuštění je popsáno níže.

RIP Sniffer

./myripsniffer -i <rozhraní>, kde význam parametru je následující:
* *-i*: <*rozhraní*> udává rozhraní, na kterém má být odchyt paketů prováděn.

V případě že budeme program spouštět na fyzickém rozhraní (ve virt. stroji isa2015 je to eth0), je nutné provést spuštění jako root, čili vložit před příkaz sudo.
Veškeré odchycené pakety přenášející RIP protokol budou vypsány na standardní výstup.

RIP Response

./myripresponse -i <rozhraní> -r <IPv6>/[16-128] {-n <IPv6>} {-m [0-16]} {-t [0-65535]}
kde význam parametrů je následující:
* *-i*: <*rozhraní*> udává rozhraní, ze kterého má být útočný paket odeslán;
* *-r*: v <*IPv6*> je IP adresa podvrhávané sítě a za lomítkem číselná délka masky sítě;
* *-m*: následující číslo udává RIP Metriku, tedy počet hopů, implicitně 1;
* *-n*: <*IPv6*> za tímto parametrem je adresa next-hopu pro podvrhávanou routu, implicitně ::;
* *-t*: číslo udává hodnotu Router Tagu, implicitně 0.

Parametry -n, -m a -t jsou nepovinné. V případě že posíláme přes fyzické rozhraní, je opět nutno použít sudo
Odešle packet se zadanými daty.

Závěr

Obě části programu se podařilo naimplementovat s dobrými výsledky.
K těžším částem implementace patřilo správné zobrazení hesla u MD5 autentifikace, kdy je třeba změnit formát na hexadecimální. Další problém nastal, když bylo třeba vypsat cílovou adresu z IP hlavičky. Tento problém jsem původně dával za vinu špatné struktuře, avšak později se ho podařilo vyřešit odlišnou prací s pamětí (bylo třeba použít funkci strcpy).
V případě podvrhnutí falešné zprávy bylo těžké vytvořit a nastavit socket tak, aby fungoval správně. Pro správnou komunikaci multicastem bylo třeba socketu nastavit správný multicast hop count a také ho správně nabindovat na zadané rozhraní.
Výsledné dva programy byly testovány na fyzickém rozhraní eth0 virtuálního počítače isa2015 a fungují podle zadání. Problém jsem našel při testování snifferu ze souboru, který byl formátu .pcapng, kdy program nefungoval správně. Avšak správné zpracování MD5 jsem otestoval pomocí jiného souboru formátu .pcap.

Citace

1. Směrování – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Sm%C4%9Brov%C3%A1n%C3%AD

2. Směrovací protokol – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Sm%C4%9Brovac%C3%AD_protokol

3. Routing Information Protocol – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Routing_Information_Protocol

4. TCP/IP - Routing - směrování < články -> SAMURAJ-cz.com. SAMURAJ-cz.com - počítačové sítě, Cisco, Microsoft, VMware, administrace [online]. Copyright © 2005 [cit. 18.11.2018]. Dostupné z: https://www.samuraj-cz.com/clanek/tcpip-routing-smerovani/