Vysoké učení technické v Brně
Fakulta informačních technologií

Dokumentácia k projektu z predmetu ISA

Tomáš Lapšanský, xlapsa00

19. novembra 2018

Obsah

1 Úvod

Dokumentácia sa zaoberá aplikáciami, ktoré slúžia na prácu a monitorovanie siete, v ktorej sa využívajú smerovacie protokoli RIPv1, RIPv2 a RIPng. Ďalej sa zaoberá návrhovými a implementačnými detailami aplikácií a taktiež uvádza základné informácie o finálnych aplikáciách a predvádza ich demonštráciu na jednoduchom útoku.

2 Inštalácia

Aplikáciu je možné jednoducho nainštalovať pomocou priloženého makefilu príkazom `make`. Následne sa v hlavnom priečinku aplikácie vytvoria aplikácie myripsniffer a myripresponse.

3 RIP protokol

`Routing information protocol` je smerovací protokol ktorý sa používa v lokálnych sieťach. Štandard RIPv1 ktorý je definovaný podľa `RFC 1058`[1] bol definovaný v roku 1988. Táto verzia nepodporovala prenášanie páru adresa + maska, a teda bolo potrebné aby všetky adresy v smerovacej tabuľke mali masku rovnakú. Tento nedostatok odstraňuje RIPv2 vydaný v roku 1994 (`RFC 2453`[2]). Novšia verzia taktiež prináša podporu autentizácie (plain text, MD5,...). Neskôr v roku 1997 prichádza RIPng (`RFC 2080`[3]) s podporou IPv6.

RIP tak ako aj RIPng sú protokoly založené na UDP komunikácií, sú pre nich rezervované porty 520 a 521. RIP ako metriku používa počet skokov (hops), štandardne obmedzených na 15. Podľa štandardu každých 30 sekúnd posiela celú svoju smerovaciu tabuľku okolitým zariadeniam, ktorú udržiava 180 sekúnd.

4 RIP sniffer

4.1 Problematika aplikácie

Sniffer je aplikácia príkazového riadku, pomocou ktorej je užívateľ schopný sledovať prevádzku na predom užívateľsky definovanom sieťovom rozhraní. Aplikácia je schopná zachytiť RIP (v1 a v2) a RIPng pakety, z ktorých je schopná rozanalyzovať Ethernet hlavičku, IP/IPv6 hlavičku, UDP hlavičku a následne samotnú RIP/RIPng správu.

4.2 Implementačné detaily

Základom aplikácie je použitá knižnica pcap library, vďaka ktorej je možné odchytávať pakety na definovanom sieťovom rozhraní, ktoré užívateľ zadáva pri spustení aplikácie. Pri inicializácií snifferu nadväzujeme komunikácie pomocou funkcie `pcap_open_live` a ďalej jej nastavujeme parametre podľa toho, aký typ prevádzky chceme zachytávať. Pre odchytávanie paketov je použitá funkcia `pcap_loop`, ktorá vždy pri odchytení paketu volá funkciu `packet_handler`, čo je nami zadefinovaná funkcia pre výpis informácií o pakete. Handler rozanalyzuváva paket, kde najprv vypisuje informácie z ethernet hlavičky. Pre štandardné hlavičky ako je UDP, IP, atď využívame štandardné funkcie `netinet`. V ďalšom kroku je na základe informácií uložených v ethernet hlavične schopný odlíšiť, či sa jedná o IPv4 alebo IPv6 hlavičku a vypísať jej detaily, na čom je neskôr závyslé ďalšie fungovanie. Následne aplikácia vypíše detaily z UDP hlavičky a dostáva sa do najdôležitejšej fázy, a to je výpis detailov z protokolu RIP. Pre protokol RIP bolo potrebné vytvoriť štruktúry na základe definícií v `RFC dokumentoch`, ktoré sa nachádzajú v súbore `lib/rip.h` (v zdrojovom súbore sa nachádzajú odkazy na konkrétne časti dokumentácie, z ktorých sme čerpali).Na základe použitej IPv4 alebo IPv6 komunikácie rozlišuje RIP a RIPng protokol, kedže RIPng využíva práve novší štandard IPv6. Vrámci RIP hlavičky sniffer rozlišuje verziu RIP protokolu a následne vypíše detaily z hlavičky a jej správ. Sniffer vrámci implementácie rozlišuje aj spôsob zabezpečenia v prípade použitia RIPv2 a je schopný vypísať správy pre `plain text` a `MD5` zabezpečenie správy.

4.3 Spustenie aplikácie

Aplikácia sa dá po nainštalovaní jednoducho spustiť nasledujúcim príkazom
`./myripsniffer -i [env]`
kde parameter `[env]` udáva názov rozhrania, na ktorom chceme sledovať prevádzku. Názvy rozhraní vieme jednoducho zistiť pomocou, napr. pomocou nástroja `ifconfig`.

4.4 Demonštrácia aplikácie

```
1. packet:

ETHERNET HEADER
src:        08:00:27:65:40:01
dest:       33:33:00:00:00:09
type:       0x86dd

IP HEADER
version:    6
src:        fe80::a00:27ff:fe65:4001
dest:       ff02::9

UDP HEADER
src port:   521
dest port:  521
len:        112
checksum:   0xd94f

RIP HEADER
command:    response
version:    RIPng
    IPv6 Prefix: fd00::/64 Metric: 1 Tag: 0x0000
    IPv6 Prefix: fd00:d1:2d78::/64 Metric: 1 Tag:
    0x0000
    IPv6 Prefix: fd00:104:3084::/64 Metric: 1 Tag:
    0x0000
    IPv6 Prefix: fd00:540:6c::/64 Metric: 1 Tag: 0x0000
    IPv6 Prefix: fd00:900:1230::/64 Metric: 1 Tag:
    0x0000
```

Aplikácia zobrazuje odchytený RIPng paket.

5 RIP response

5.1 Problematika aplikácie

Response je aplikácia príkazového riadku, ktorá je schopná generovať pakety RIPng response podľa štandardu `RFC2080`. Pakety sú do istej miery nastevené podľa vstupných parametrov aplikácie. Užívateľ dokáže nastaviť adresu podvrhávanej siete, RIP metriku, adresu nexthopu a router tag.

5.2 Implementačné detaily

Aplikácia pracuje s využitím knižnice `sys/socket.h`. Po overení všetkých vstupných parametrov pomocou funkcie `getopt` aplikácia vytvorí `socket`, ktorému nastaví všetky potrebné parametre pomocou preddefinovaných funkcií. Pre vytvorenie paketu sme zadefinovali funkciu `set_packet`, ktorá alokuje miesto pre vytváraný paket pomocou funkcie `malloc`, a následne do alokovanej pamäte kopíruje pomocou `memcpy` nami definované štruktúry. Samozrejme, pre korektnost dát je alokovaný priestor "vynulovaný"pomocou funkcie `bzero`. Po vytvorení a nastavení paketu aplikácia odosiela paket na multicast pomocou funkcie `sendto`.

5.3 Spustenie aplikácie

Aplikácia sa dá po nainštalovaní jednoducho spustiť nasledujúcim príkazom
`./myripresponse -i <env> -r <IPv6>/[16-128] {-n <IPv6>}`
`{-m [0-16]} {-t [0-65535]}`
kde majú jednotlivé parametre nasledujúci význam
* -i: udává rozhranie, z ktorého má byť paket odoslaný
* -r: IP adresa podvrhávanej siete a jej dĺžka prefixu
* -m: RIP metrika (tzv. počet hopov), implicitne 1
* -n: adresa next-hopu, implicitne ::
* -t: hodnota router tagu, implicitne 0

5.4 Demonštrácia aplikácie

Aplikácia vypisuje iba chyby na `stderr`.

6 Útok

6.1 Sniffovanie

Pomocou nami naimplementovanej aplikácie sniffer sa nám podarilo na referenčnom FreeBSD routery za pomoci virtuálnej topológie v prostredí virtualbox zachytiť niekoľko paketov nesúcich RIPv2 response správi s jednoduchým zabezpečením. Správy náš sniffer dešifroval do nasledujúcej podoby (pre zjednoduhšenie uvádzame už len výpis z RIP headeru)

```
RIP HEADER
command:        response
version:        RIPv2
    ------------------------
    AFI:    0xFFFF (authentication)
    type:   2
    msg:    ISA>28812008650
    ------------------------
    AFI:    IP
    tag:    0
    IP:     10.48.48.0
    mask:   255.255.255.0
    n-hop:  0.0.0.0
    metric: 1
    ------------------------
    AFI:    IP
    tag:    0
    IP:     10.97.115.0
    mask:   255.255.255.0
    n-hop:  0.0.0.0
    metric: 1
    ------------------------
    AFI:    IP
    tag:    0
    IP:     10.112.220.0
    mask:   255.255.255.0
    n-hop:  0.0.0.0
    metric: 1
    ------------------------
    AFI:    IP
    tag:    0
    IP:     10.212.97.0
    mask:   255.255.255.0
    n-hop:  0.0.0.0
    metric: 1
    ------------------------
```

Z uvedeného paketu je jasne vidno, že heslo je uvedené v `plain text` forme (jasne nam to ukazuje type: 2), a teda je jednoduché ho prečítať. Pre úplnosť, heslo je:
`ISA>28812008650`

Ďalej je možné z paketu vyčítať, na ktoré siete má router priamy prístup (metrika o veľkosti 1), ich ip adresy sú (pre každú z adries je uvedená maska `255.255.255.0`)

`10.48.48.0`
`10.97.115.0`
`10.112.220.0`
`10.212.97.0`

6.2 Podvrhávanie routy

V rámci našej virtuálnej topológie sme taktiež boli schopný simulovať útok na router pomocou našej rip response aplikácie. Ako vstupné parametre sme zvolili rozhranie a podvrhovanú routu (keďže tieto parametre sú povinné) a spustili sme útok. Úspešnosť aplikácie sme overili pomocou nástroja `telnet` a `show ipv6 route`. Po dokončení útoku sa nám v routovacej tabuľke pre ipv6 pridal nový záznam v nasledujúcom tvare

```
R>* 2001:db8:0:abcd::/64 [120/2] via
fe80::a00:27ff:fe94:b41e, em0, 00:00:51
```

čo bolo očakávané. Adresa ktorá je umiestnená za `via` je link:local adresa nášho rozhrania. Ďalším testovaním sme zistili, že v prípade použitia parametra `-n` (nastavenie adresy pre nexthop), sa adresa za `via` nahradí nami zvolenou adresou (samozrejme v prípade, že je zvolená adresa link:local).

7 Zdroje

[1] **RFC 1058**: RIPv1: `https://tools.ietf.org/html/rfc1058`
[2] **RFC 2453**: RIPv2: `https://tools.ietf.org/html/rfc2453`
[3] **RFC 2080**: RIPng: `https://tools.ietf.org/html/rfc2080`