“Nástroje monitorující a generující zprávy jednoduchých distance-vector protokolů “

Predmět: ISA

Author: Ksenia Bolshakova

Brno 2018

Obsah

1.Úvod

Cílem projektu je vytvořit komunikující aplikaci podle konkrétní vybrané specifikace pomocí síťové knihovny BSD sockets v jazyce C/C++.

První úkol byl nastudovat si směrovací protokoly RIP a RIPng.

Druhý úkol je naprogramovat sniffer RIPv1, RIPv2 a RIPng zpráv.

Třetí úkol je naprogramovat podvrhávač falešných RIPng Response zpráv.

Čtvrtý je provést úspěšný útok.

Pátý naprogramovat podvrhávač falešných RIPng Request zpráv (bonusový úkol).

2.RIP protocol

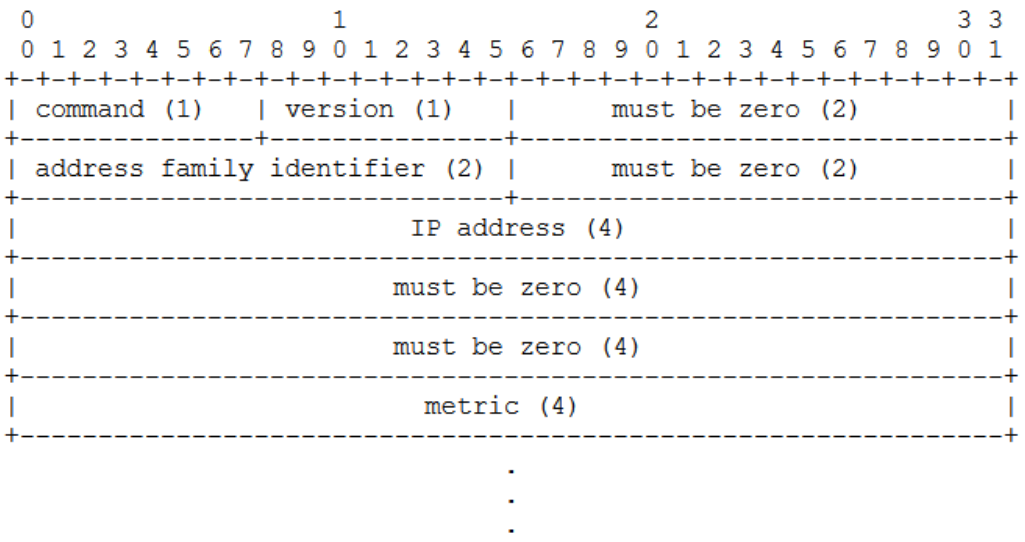
Routing Information Protocol (RIP) je směrovací protokol umožňující směrovačům (routerům) komunikovat mezi sebou a reagovat na změny topologie počítačové sítě. RIP používá UDP (User Datagram Protocol) jako transportní protokol. Na portu 520 běží RIPv1 a RIPv2, na portu 521 běží RIPng. Vzniknul dany protokol v roce 1969.

Timer

Každých 30 vteřin se process RIP probudí a pošle se nevyžádaná zpráva odpovědí, která obsahuje celou RTE.

a) RIP verze 1

Originální specifikace RIPu dle RFC 1058 používá směrování podle původních tříd IPv4 adres A, B nebo C. Periodické aktualizace směrování nezahrnují informace o masce sítě, protože podle původního systému je maska dána příslušností IP adresy do jedné ze tříd. Chybí tak podpora pro CIDR (Classless Inter-Domain Routing), což znemožňuje existenci různě velkých podsítí uvnitř jedné třídy IP adres. Všechny podsítě musí být stejně velké (tj. se stejnou maskou). Neexistuje zde podpora pro vzájemnou autentizaci routerů, a proto je protokol RIPv1 napadnutelný nejrůznějšími útoky.



b) RIP verze 2

Zahrnuje možnost přenášet informace o masce sítě, tudíž podporovala CIDR (Classless Inter-Domain Routing). K udržení zpětné kompatibility, zůstalo omezení 15 skoků (hop count). Při správné konfiguraci může být RIPv2 plně kompatibilní se starší verzí. K lepší kompatibilitě slouží vlastnost *compatibility switch*. Ve snaze vyhnout se zbytečnému zatížení na hostiteli, jež se neúčastní směrování, RIPv2 vysílá celou směrovou tabulku všem sousedním směrovačům na adrese 224.0.0.9 (multicast), čímž se liší od RIPv1, který používá broadcast). Unicastové adresování je stále povoleno pro mimořádné účely.

RIPv2 včlenilo podporu pro vzájemnou autentizaci routerů. Hesla jsou však přenášena v nekódovaném textu, což je nedostatečné pro bezpečnou komunikaci v síti internet. Šifrovací (MD5) autentizace pro RIP byla představena v roce 1997. RIPv2 je Internetový Standard STD-56.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Address family identifier (2)																Route Tag (2)															
IPv4 address (4)																															
Subnet mask (4)																															
Next hop (4)																															
Metric (4)																															

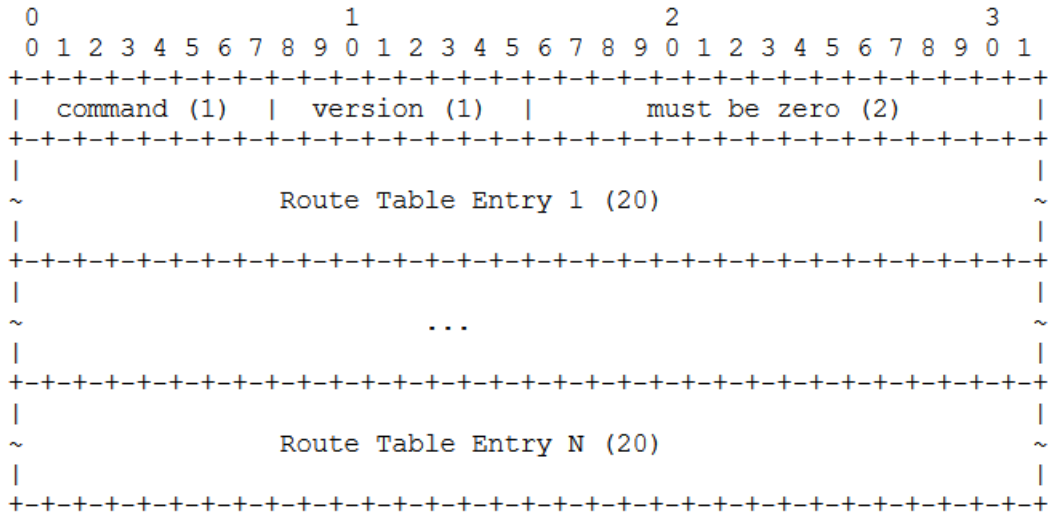
Autentifikace

c) RIPng

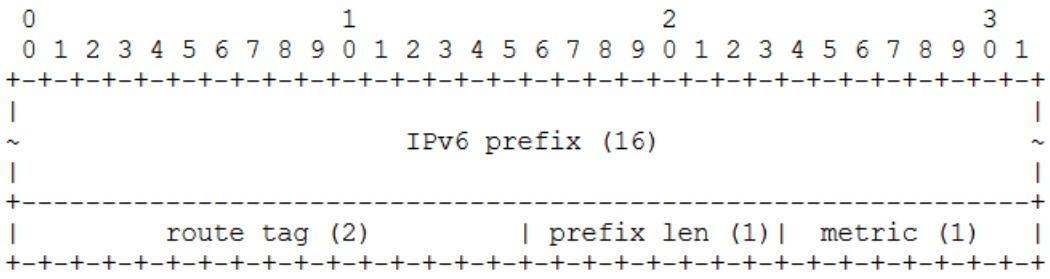
RIPng (RIP další generace), který je definován v RFC 2080, je rozšířením RIPv2 zahrnující podporu IPv6 (Internetového Protokolu další generace). Hlavní rozdíly mezi RIPv2 a RIPng jsou:

- podpora IPv6 síťování
- podpora aktualizovaných autentizací (verifikace ověření identity osoby nebo procesu zabezpečovacím systémem) - RIPng nepodporuje (nahrazeno IPsec)
- připojování libovolných tagů k směrovačům (routerům) - RIPng nepodporuje
- kódování dalších skoků do každého směrovacího záznamu - RIPng vyžaduje specifické kódování na dalším skoku, pro set směrovacích záznamů

RIPng packet format:



Kde každá Route Table Entry (RTE) má následující formát:

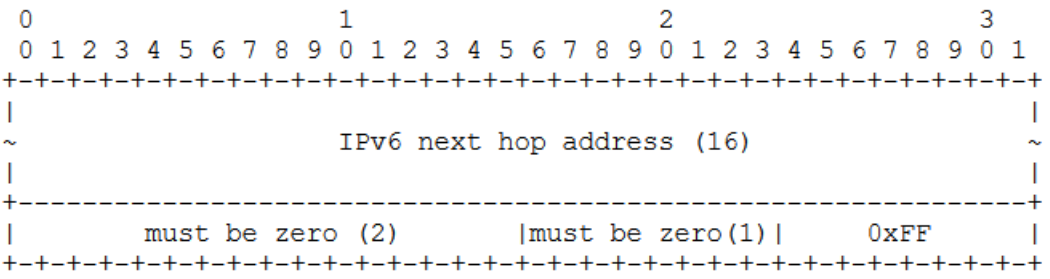


Maximální číslo RTE je definováno dolu.

Ve Version naimplementovány příkazy 1 – request (požadavek, aby reagující systém poslal celou nebo část tabulky), 2 – response (zpráva obsahující všechny nebo části odesílatele směrovací tabulky. Tato zpráva může být zaslána jako odpověď k žádosti nebo může být nevyžádaná směrování aktualizace generované odesílatelem).

RIPng umožnuje specifikovat next hop IPv6 adresu, přes kterou bude směrován paket do cíle.

Next hop Route Table Entry (RTE) format:



Implicitní format next hopu je 0:0:0:0:0:0:0:0 indikuje, že příští adresa hopu by měla být původcem vizitky RIPng. Adresa specifikovaná jako next hop musí byt link-local adresa. Účelem next hop RTE je vyloučit směrování paketů prostřednictvím extra hopů v systému.

3. Implementace Snifferu pro RIPv1, RIPv2, RIPng

Spustí se jako ./myripsniffer -i [rozhraní]. Pokud uzivatel bude chtít poradit, spustí program jako ./myripsniffer -h, a vypíše se mu nápověda.

Pro implementace snifferu byla využita knihovna libpcap.

Funkce *getopt()* používá se pro parsování command-line argumentů, která vezme na vstupu parametr -i a jeho argument – název rozhraní a uloží ho do proměnné *dev*.

Na zaklade tutoriálu o knihovně libpcap byl vytvořen sniffer.

Funkce *pcap_lookupnet()* se používá pro určení subnet a netmask rozhraní, v případě chyby vrátí -1.

pcap_open_live() získá handle pro živé zachycení.

pcap_datalink() určí formát paketů dodaných handlem.

Pomocí *pcap_compile()* řetězec filtru je kompilován do programu pseudo-machine-language.

Výsledný program může vytvořit filtr pro handle pomocí *pcap_setfilter().*

Funkce *ripInfo()* vypisuje všechny potřebné informace pro RIPv1, RIPv2 a RIPng. Podle verze IP adresy určí,jestli jsou to RIPng nebo RIPv1 a RIPv2 protokoly. Na základě formátu dat uložených v protokolu, vypíše verze protokolu, command, metriku a IP adresu.

Pokud je to RIPng nebo RIPv2 vypíše autentifikaci. Když typ autentifikace je 02 (Simple Password) u RIPv2, vypíše heslo. Když je typ autentifikace je 03 vypíše "MD5" (šifrované heslo). Pokud je na bytech 48 a 49 uloženo něco jiného, vypíše "Other". Dále vypíše zdrojovou a cílovou IP adresu.

V cyklu se vypisuje Route Table Entry pro RIPv2, RIPng.

Funkce pcap_loop(handle, 0, ripInfo, NULL); spouští do nekonečna funkci ripInfo(), dokud uživatel neukončí ten program. Druhý parametr "0" umožňuje toto nastavení.

4. Implementace myripresponse

Program se spouští:

./myripresponse -i <interface> -r <IPv6>/[16-128] {-n <IPv6>} {-m [0-16]} {-t [0-65535]}

Kde

* -*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.

Nebo se spouští:
./myripresponse -h

Aby se vypsala nápověda.

V souboru ripresponse.cpp:

Pomocí funkce getopt() načte argumenty z command-line, rozparsuje je a uloží do odpovídajících proměnných, zkontroluje, aby byli všechny povinné argumenty nastaveny.

Okamžitě se kontroluje, jestli jsou zadané hodnoty, jsou v zadaném rozsahu.

Pomocí funkce inet_pton() konvertujeme zadanou uživatelem IP adresy podvrhávané sítě a next hop do struktur in6_addr, která se používá pro IPv6 adresy.

Funkce if_nametoindex(optarg) konvertuje uživatelem zadaný interface na index.

Do struktury source_addr nastaví se všechny parametry pro source address (adresu, port, address family), to stejné se udělá pro strukturu dest_addr, která bude přijímat socket.

Funkce socket(); vytvaří socket s určitými parametry.

Funkce *setsockopt(socket_of_client, IPPROTO_IPV6, IPV6_MULTICAST_IF, &index, sizeof(index))* nastavuje parametry pro vytvořený socket, kde index je konvertované rozhraní.

Funkce *bind(socket_of_client, (struct sockaddr *) &source_addr, sizeof(source_addr))* napojuje socket na zadaný port.

Funkce *setsockopt(socket_of_client, IPPROTO_IPV6, IPV6_MULTICAST_HOPS, &max_hop, sizeof(max_hop))* nastavuje maximalnípočet hopů pro daný socket.

Potom se vytváří nový RIPng packet s parametry, zadanými uživatelem.

Funkce *sendto(socket_of_client, ripngPacket->packet, ripngPacket->length, 0, (struct sockaddr*)&dest_addr, sizeof(dest_addr))* odesílá vytvořený socket.

V souboru ripngpacket.cpp

V tomto souboru se vytvaří RIPng packet. Používá se a i pro ripresponse, a i pro riprequest.

Na začátku se alokuje pamět pro packet, zakládají se pole pro ukládaní adresy next hopu, proměnná pro ukládání délky packetu.

Do packetu nejprve uložíme Command, pomocí funkce memset(), pro Verzi, 3 a 4 bity musí mít hodnotu “0”. Dále nakopírujeme IPv6 adresu, pomocí memcpy(), kterou zadal uživatel. Zatím se uloží Route Tag, délka prefixu, a metrika. Dále se uloží next hop a 0xff uložené na konci.

5. Útok

Pomocí aplikace myripresponse byl podvrhnout SW směrovači zprávou RIP Response routu 2001:db8:0:abcd::/64. Úspěch podvrhnutí byl ověřen (ve virtuálním počítači pomocí příkazu *show ipv6 route*, byla zobrazena aktuální směrovací tabulka, kde byla zobrazena poslána zpráva.

Útok byl úspěšně proveden, výsledek je vidět v sekci Výstup programu.

6. Implementace myriprequest

Implementace je podobná jako u myripresponse - jediné, co se změní, je Command.

7. Výstup programů:

Myripsniffer

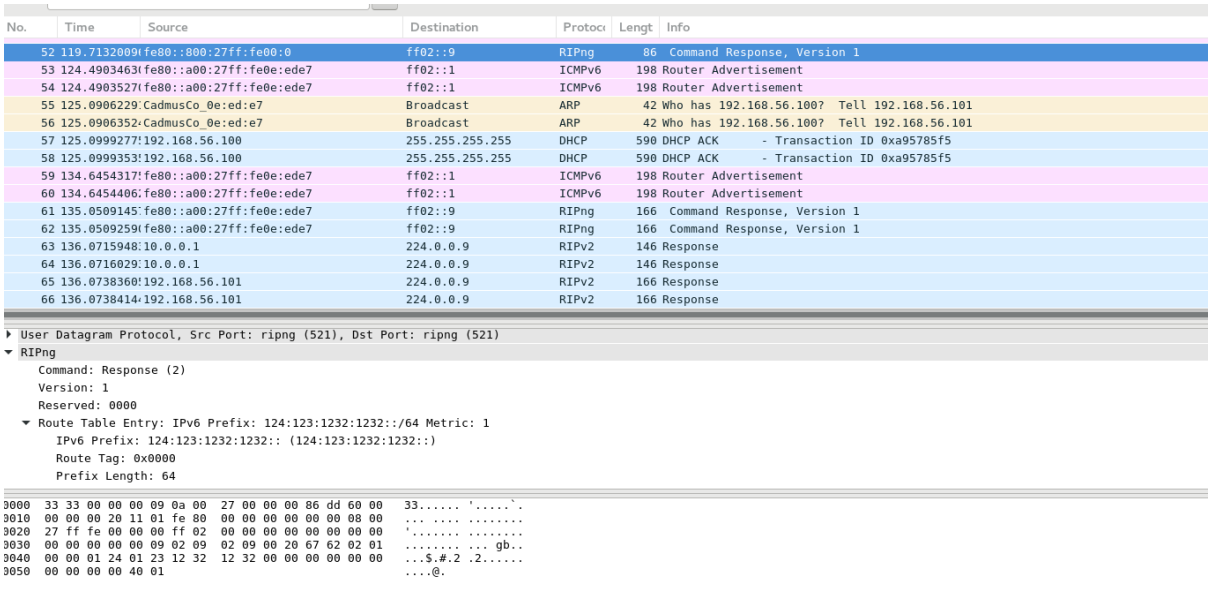
RIPv1, RIPv2

```
Version: RIPv2
Authentication type: Simple password(2).
Authentication: ISA>28813508880

Route Entry.
Address Family: 2
Route tag: 0
IP address: 10:0:0:0
Netmask: 255:255:255:0
Next hop: 0:0:0:0
Metric: 1

Address Family: 2
Route tag: 0
IP address: 10:48:48:0
Netmask: 255:255:255:0
Next hop: 0:0:0:0
Metric: 1

Address Family: 2
Route tag: 0
IP address: 10:108:206:0
Netmask: 255:255:255:0
Next hop: 0:0:0:0
Metric: 1

Address Family: 2
Route tag: 0
IP address: 10:111:115:0
Netmask: 255:255:255:0
Next hop: 0:0:0:0
Metric: 1

Address Family: 2
Route tag: 0
IP address: 10:226:104:0
Netmask: 255:255:255:0
Next hop: 0:0:0:0
Metric: 1
```

RIPng

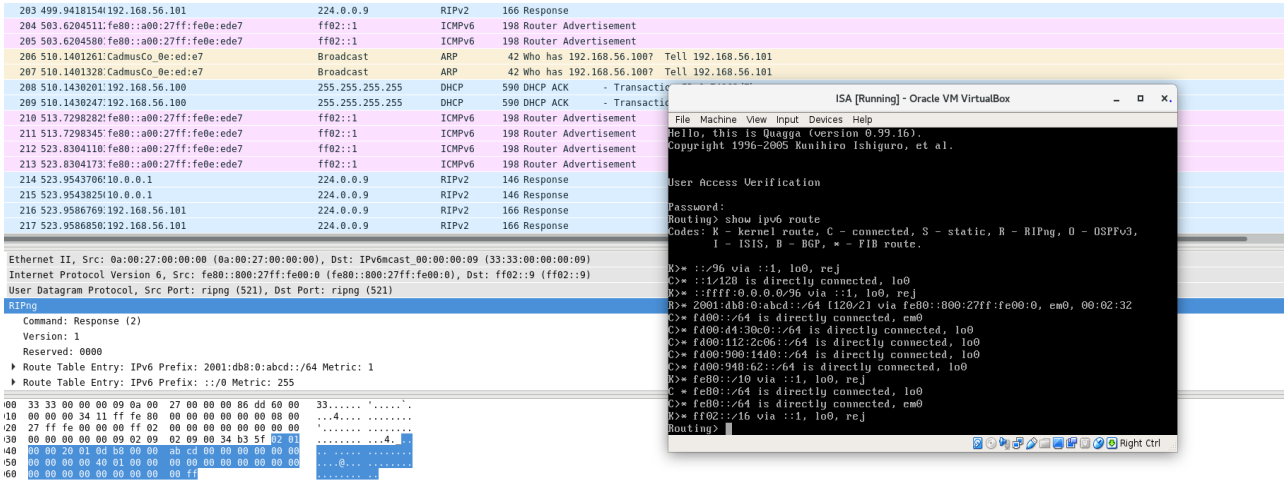
```
RIPpacket
Version: RIPng
Command: Response (2)

Route Table Entry:
IPv6 Prefix: fd00:0000:0000:0000:0000:0000:0000:0000
Route Tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:00d4:30c0:0000:0000:0000:0000:0000
Route Tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:0112:2c06:0000:0000:0000:0000:0000
Route Tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:0900:14d0:0000:0000:0000:0000:0000
Route Tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: fd00:0948:0062:0000:0000:0000:0000:0000
Route Tag: 0
Prefix Length: 64
Metric: 1

IPv6 Prefix: 0000:0000:0000:0000:0000:0000:0000:0000
Route Tag: 0
Prefix Length: 0
Metric: 0

Source IPv6 address: fe80:0000:0000:0000:0a00:27ff:fe0e:ede7
Destination IPv6 address: ff02:0000:0000:0000:0000:0000:0000:0009
```

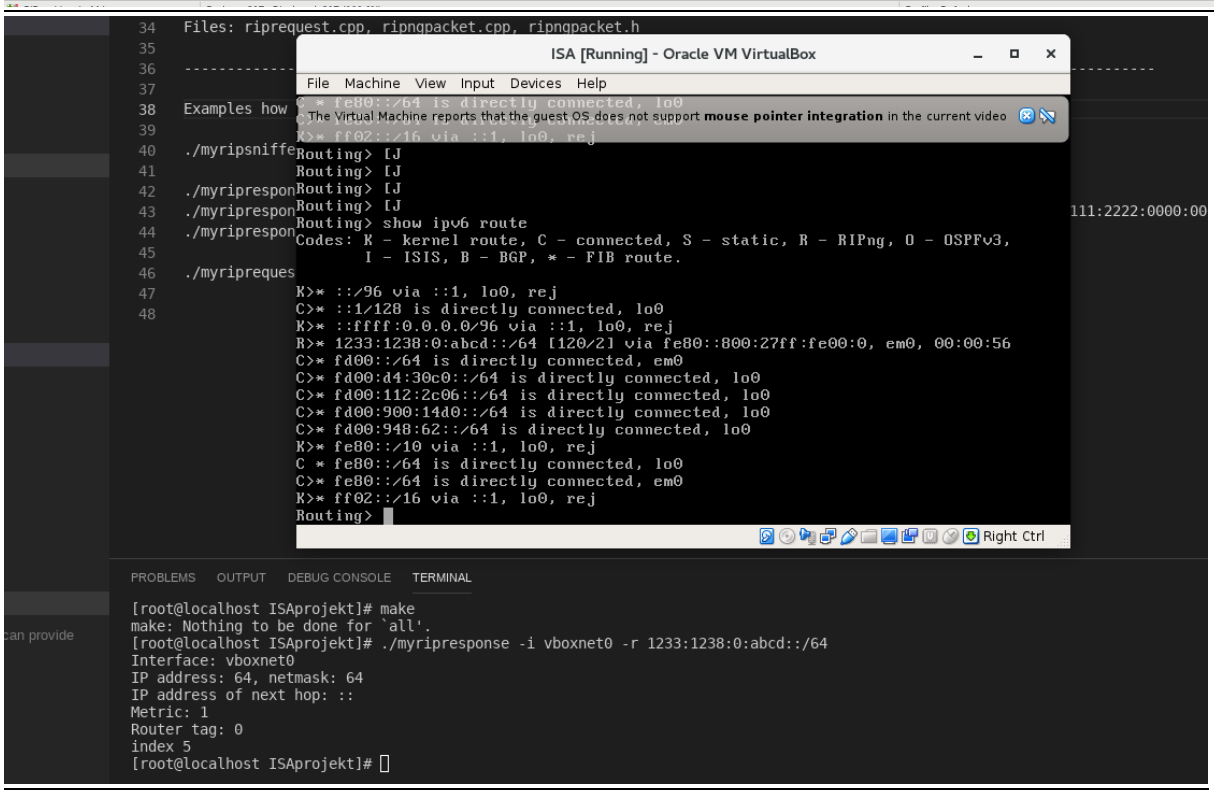
Myripresponse



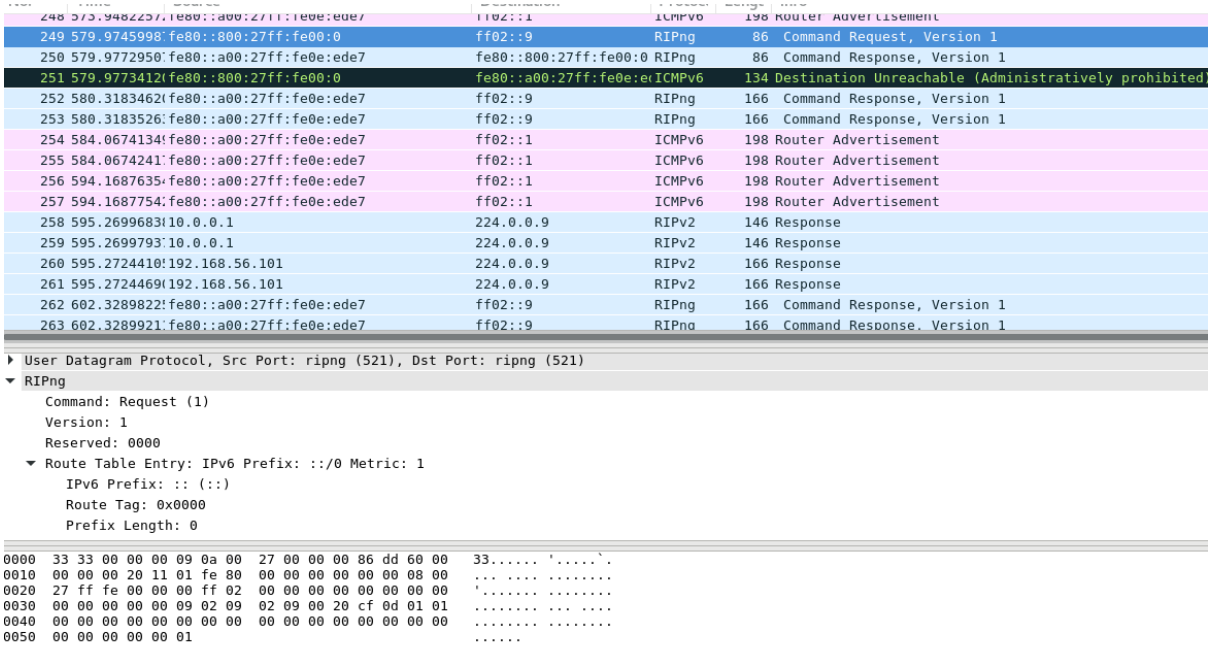
```
Reserved: 0000
▾ Route Table Entry: IPv6 Prefix: 124:123:1232:1232::/64 Metric: 6
    IPv6 Prefix: 124:123:1232:1232:: (124:123:1232:1232::)
    Route Tag: 0xe8fd
    Prefix Length: 64
    Metric: 6
▾ Route Table Entry: IPv6 Prefix: 1212:1313:7878:0:1111:2222:0:1/0 Metric: 255
    IPv6 Prefix: 1212:1313:7878:0:1111:2222:0:1 (1212:1313:7878:0:1111:2222:0:1)
    Route Tag: 0x0000
```

Utok





Myriprequest



8.Literatura

https://cs.wikipedia.org/wiki/Routing_Information_Protocol

https://tools.ietf.org/html/rfc1058

https://tools.ietf.org/html/rfc2453

https://tools.ietf.org/html/rfc2080

http://www.tcpdump.org/