

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Nástroje monitorující a generující zprávy jednoduchých distance-vector protokolů

ISA 2018

1 Abstract

In internet it is not likely to have one single protocol that is used for the entire network. Rather, network is divided into autonomous systems (AS) where each AS has its own routing technology. Although there are many types of protocol that determines communication between nodes, we can divide them into 2 groups Interior gateway protocol(IGP) and Exterior gateway protocol(EGP). IGP is a routing protocol used in AS and a EGP is used to transfer information among ASs. RIP is designed to work as an IGP in moderate-size AS.

2 Routing information protocol

RIP is a distance-vector routing protocol used in smaller networks, that sends the complete routing table out to all interfaces every 30 seconds. RIP only uses hops to determine the best way to a remote network. However there is a maximum hop count of 15 by default (16 means unreachable). RIP version 1 uses mechanism called classful routing meaning that all devices in the network must use same subnet mask. The reason is that RIPv1 doesn't send updates with subnet mask information. On the other hand RIPv2 provides classful routing and sends subnet mask with the route updates[1].

RIPv2 as a newer protocol should solve problems that appeared in version one. There is a short recapitulation of differences between RIPv1 and RIPv2.

2.0.1 Differences between protocols

RIPV1

- no authentication
- no support for discontinuous¹ networks
- classful
- broadcast based
- no support for variable long subnet mask(VLSM)

RIPV2

- allows MD5 authentication
- supports discontinuous networks
- classless
- uses multicast 224.0.0.9
- support for VLSM

2.0.2 Controls

There are more controls on RIP packet that we have to keep in mind while programming RIP.

Response must be ignored if:

- ip address is not from right UDP port
- ip source is from a valid neighbor

¹By definition, contiguous means next or together in sequence, <https://learningnetwork.cisco.com/thread/30426>

- source of datagram must be from link local address
- if the metric is greater than 16, ignore the entry but log the event

2.0.3 Common limitations

1. maximum of 15 hops
2. counting to infinity to resolve certain unusual situations (if it was immense it would require either much time or bandwidth)
3. Uses fixed metrics (no real time factors as delay, reliability, load)

2.0.4 Structure of RIP

RIP protocols are designed to have one header (4 B) and several entries (each 20 B). RIP header contains information as version of RIP and command (either request or response). In entries we find address family identifier (AFI) describing type of address (4 B), route tag (4 B), which differs external rip routes from internal or it can be also a type of authentication. If an address family identifier equals 0xFFFF, then according to the authentication type we parse authentication data (16 B). To see more information about authentication go to section 2.0.6. If AFI equals value of AF_INET, then subnet mask (4 B), destination IPv4 address (4 B), next hop² (4 B) and the cost to reach that destination (called metric, 4 B) are sent.

2.0.5 RIP header

All RIP protocols have same header that contains version, zeros and type of message. There are two types of messages Request and Response.

- **Request**
 - used to ask for response containing all OR part of a routers routing table
 - normally send as multicast (RIPv1 broadcast) but can be a situation that we need to send to just one response to just one router (possible with RIPv1) then packet is sent directly to the router from a UDP port other than the RIPv1 port
 - if there is one entry in the request, destination prefix is zero, a prefix length is zero, metric of infinity 16, then this is request to send entire routing table
- **Response** A response A message containing all or part of the sender's routing table. It can be sent in response to a request or it may be a routing update generated by the sender. There 3 reasons to send a response:
 - response to a query
 - regular update
 - triggered update caused by a route change

To conclude, RIPv2 is an advanced RIPv1 which can be used in small networks or at the edge of larger networks because of its simplicity in usage and configuration.

²address of next router along the path to the destination

2.0.6 Authentication

Security is one of the primary concerns of network designers today[8]. Our main concern in RIP protocols is to ensure that information entered into the routing table is valid or not. An attacker can easily send a malicious packet that will be sent to network and might end up in the routing table due to poor configuration. Because of this, it's necessary to authenticate the routing update process. Either we can authenticate an update with Simple or MD5 authentication (RIPv2). The difference is clear, the simple password is sent through network as a plain text for everybody to see compared to MD5 that sends encrypted data to be decoded by a specific authentication algorithm.

MD5 authentication

Configuration of MD5 consists of:

1. defining a key chain with a name that determines the set of keys that can be used on the interface
2. defining the key or keys on the chain (related to number point number one)
3. defining the key string to be used in the key, needs to be same on remote routers, the actual password
4. enable authentication on the interface and specify the key chain + type of authentication (MD5)

Apart from type number 3 in Key Keyed Message Digest, routing update contains:

1. 16 bit authentication data
2. 8 bit field contains Key Identifier or Key id
3. 8 bit field contains the length of data field
4. 32 bit sequence number that must be non-decreasing for all message with same key ID

Verification uses MD5 hash algorithm using a password which routing updates does not carry (that makes is authenticate). Rather, it carries a 16 bit message, generated by the MD5 algorithm on the password.

2.1 RIPng

Protocol RIPng is a routing protocol implemented for IPv6 networking. In comparison with RIPv2 there is no authentication, RIPng relies on the IP Authentication Header[7] and the IP Encapsulating Security Payload[6][5]. Structure is similar to RIPv2 described in 2.0.4, but instead of sending subnet mask, a prefix length is sent.

2.1.1 Process of setting a route to routing table

1. set destination prefix and length in RTE (routing table entry)
2. set metric to the newly calculated metric
3. set the next hop address to be the address from which the datagram came or is specified by a next hop in RTE
4. initialized the timeout
5. set the route change flag
6. signal the output process to trigger an update

If there is an existing router, compare next hop, If next hop is from the same router as the existing router, update timer.

If metric is lower, put the next metric in, adjust IP address and set the route change flag

If the new metric is infinity, start the deletion process[5].

3 Implementation details

Our main task was to study routing protocols RIP, RIP2 and RIPng. After that, we implemented sniffer RIPv1, RIPv2 and RIPng messages, faked that sends a RIP response message to router and perform a success attack.

3.1 Sniffer

There is an example how to run the program after compilation (after make). Program sniffs all RIP packets and formats them on output.

```
format:  ./myripsniffer -i <interface>
example:  ./myripsniffer -i eth0
```

3.1.1 Structure

`sniffer.c` - contains functions to parse and print RIP packets (RIPv1, RIPv2, RIPng)
`headers.h` - contains RIP packets headers
`sniffer_h.h` - contains headers of functions belonging to RIP sniffer
`myripsniffer.c` - main file that starts capturing packets

For capturing packets I was using functions from libpcap library. Firstly it is important to set an interface, open a handle and apply a filter. Chosen filter was udp port 520 or udp port 521 that corresponds to RIPv1, RIPv2 and RIPng protocols. In an infinite loop (function `pcap_loop`) I was parsing packets and printing all important information (function `print_packet`). Thanks to libraries as `ip.h`, `udp.h`, I did not have to implement structure of udp and ip protocol. I implemented RIP structures described in 2.0.4 in file `headers.h` and had to distinguish IPv4 packet from IPv6 packet (functions `ipv6_process` and `ipv4_process`). Due to a fact that we know a routing table entry length and length of RIP packet, we can loop till actual parsing length is equal or greater than rip length. In this loop I parse RIP entry, for IPv4 we must not forget types of authentication (see more in section 2.0.6).

3.2 RIP response

There is an example how to run the program sending one rip packet after compilation (after make). Program sends RIP packet according to information given on input.

```
format:  ./myripresponse -i <interface> -r <IPv6>/[16-128] -n <IPv6> -m [0-16]
-t [0-65535]
example:  ./myripresponse -i eth0 -r 2001:db8:0:abcd::/64
```

3.2.1 Structure

`myripresponse.c` - main file for rip response, parse and process all arguments
`response.c` - file to create and send rip packet
`response.h` - contains used response structures

This part of project is shorter than the first one. It is composed of two parts - parsing arguments and sending a RIPng packet. This packet consists of one RIP header and two routing table entries. In header we set response command and version one. First entry contains information given on input (route tag, prefix, number of hops, metric) and the second contains next hop with metric of value of `0xFF[5]`. Next hop is set according to input or default a next hop is used (`: :`). While creating an UDP packet, we set a receiver's address to multicast group

FF02::9, RIPv2 port 521, interface index, hop limit of 255 (maximum hop limit), AFI and receivers IPv6 address to any (in6addr_any).

4 Testing

4.1 Settings

On my computer I have two virtual machines with FreeBSD (given from school) and Ubuntu that I run on VirtualBox. Both machines are set on internal network with same name of the network to be able to see each other. While testing, I run Wireshark on Ubuntu and compare results with my program. As a result I can validate displayed data. For testing MD5 I use a tool tcpreplay with given pcap file because no MD5 authentication is implemented on FreeBSD 3.

4.2 Sniffer

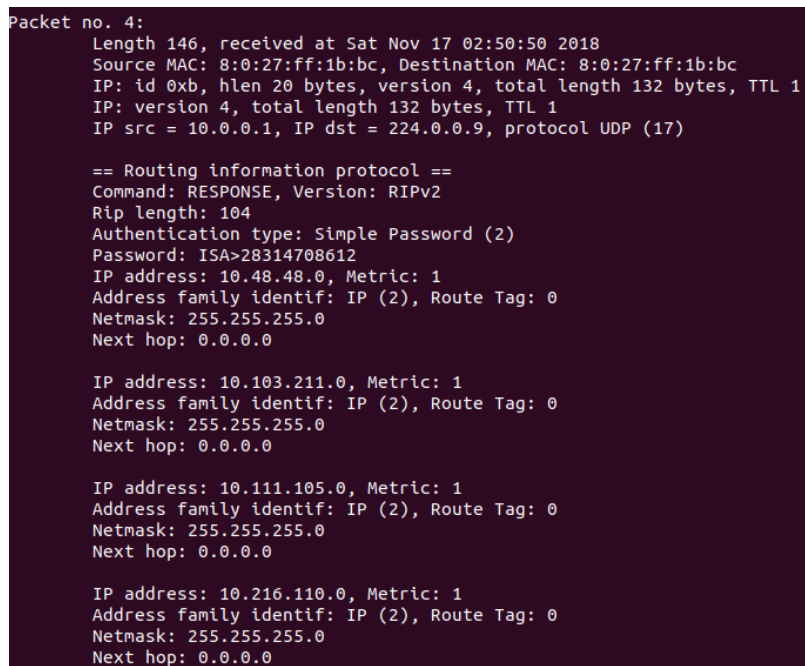
Sniffer needs to be compatible with RIPv1, RIPv2 and RIPv2. Because of it I use a filter that captures packets on UDP port 520 and 521.

4.2.1 RIPv2 with simple password

In Figure 1 there is an example of captured RIPv2 packet with a simple password. RIPv2 part start with a header == Routing information protocol ==, we find there all information related with RIPv2 as command, IP address, authentication, version, AFI, route tag, netmask and nexthop. Each entry is separated by an empty row. Authentication password given from school is: ISA>28314708612.

4.2.2 RIPv2

In Figure 2 there is an example of captured RIPv2 packet. It is very similar to RIPv1 and RIPv2 packet described in section 4.2.



```
Packet no. 4:
  Length 146, received at Sat Nov 17 02:50:50 2018
  Source MAC: 8:0:27:ff:1b:bc, Destination MAC: 8:0:27:ff:1b:bc
  IP: id 0xb, hlen 20 bytes, version 4, total length 132 bytes, TTL 1
  IP: version 4, total length 132 bytes, TTL 1
  IP src = 10.0.0.1, IP dst = 224.0.0.9, protocol UDP (17)

  == Routing information protocol ==
  Command: RESPONSE, Version: RIPv2
  Rip length: 104
  Authentication type: Simple Password (2)
  Password: ISA>28314708612
  IP address: 10.48.48.0, Metric: 1
  Address family identif: IP (2), Route Tag: 0
  Netmask: 255.255.255.0
  Next hop: 0.0.0.0

  IP address: 10.103.211.0, Metric: 1
  Address family identif: IP (2), Route Tag: 0
  Netmask: 255.255.255.0
  Next hop: 0.0.0.0

  IP address: 10.111.105.0, Metric: 1
  Address family identif: IP (2), Route Tag: 0
  Netmask: 255.255.255.0
  Next hop: 0.0.0.0

  IP address: 10.216.110.0, Metric: 1
  Address family identif: IP (2), Route Tag: 0
  Netmask: 255.255.255.0
  Next hop: 0.0.0.0
```

Figure 1: Sample of captured RIPv2 packet with simple password

```

Packet no. 3:
  Length 166, received at Sat Nov 17 02:50:49 2018
  Source MAC: 8:0:27:ff:1b:bc, Destination MAC: 8:0:27:ff:1b:bc

  == Routing information protocol ==
  Command: RESPONSE, Version: 1
  Rip length: 104
  Route Tag: 0
  IPv6 address: fd00::
  Prefix Length: 64
  Metric: 1

  Route Tag: 0
  IPv6 address: fd00:d5:3390::
  Prefix Length: 64
  Metric: 1

  Route Tag: 0
  IPv6 address: fd00:108:2c4c::
  Prefix Length: 64
  Metric: 1

  Route Tag: 0
  IPv6 address: fd00:4d4:6c::
  Prefix Length: 64
  Metric: 1

  Route Tag: 0
  IPv6 address: fd00:900:14d0::
  Prefix Length: 64
  Metric: 1

```

Figure 2: Sample of captured RIPng packet

```

Packet no. 44:
  Length 146, received at Sat Nov 17 02:36:16 2018
  Source MAC: aa:bb:cc:0:1:0, Destination MAC: aa:bb:cc:0:1:0
  IP: id 0x0, hlen 20 bytes, version 4, total length 132 bytes, TTL 2
  IP: version 4, total length 132 bytes, TTL 2
  IP src = 10.0.0.1, IP dst = 224.0.0.9, protocol UDP (17)

  == Routing information protocol ==
  Command: RESPONSE, Version: RIPv2
  Rip length: 104
  Authentication type: Key Message Digest (3)
  Digest Offset: 84, Key ID: 1
  Auth Data Len: 20, Seq num: 20
  IP address: 10.1.11.0, Metric: 3
  Address family identif: IP (2), Route Tag: 0
  Netmask: 255.255.255.0
  Next hop: 0.0.0.0

  IP address: 10.1.12.0, Metric: 3
  Address family identif: IP (2), Route Tag: 0
  Netmask: 255.255.255.0
  Next hop: 0.0.0.0

  IP address: 10.1.13.0, Metric: 3
  Address family identif: IP (2), Route Tag: 0
  Netmask: 255.255.255.0
  Next hop: 0.0.0.0

  Authentication data: 6946f20b014d173031dd1d2e48e9af37

```

Figure 3: Sample of captured RIPv2 packet with MD5 password

4.3 Captured routes from FreeBSD

In tables below there is a list of routes that I captured. The same IP address we can also see in Figure 1 and Figure 2.

RIPv2					
IPv4 address	Mask	NextHop	AFI	Route Tag	Metric
10.48.48.0	255.255.255.0	0.0.0.0	IP	0	1
10.103.211.0	255.255.255.0	0.0.0.0	IP	0	1
10.111.105.0	255.255.255.0	0.0.0.0	IP	0	1
10.216.110.0	255.255.255.0	0.0.0.0	IP	0	1

RIPng

IPv6 address	Prefix	Metric	Route Tag
fd00:d5:3390::	64	1	0
fd00:108:2c4c::	64	1	0
fd00:4d4:6c::	64	1	0
fd00:4d4:6c::	64	1	0

4.4 Sending a fake RIP response packet

I implemented a program to send RIP response packet with malicious IP address, then by connecting to router demon (127.0.0.1 2601) I could verify that my fake IP address was added to attacked router's routing table (see in Figure 5). In Figure 4 we can find structure of sent RIPng Response packet visualized in wireshark.

run command: `sudo ./myripresponse -i enp0s3 -r 2001:db8:0:abcd::/64`

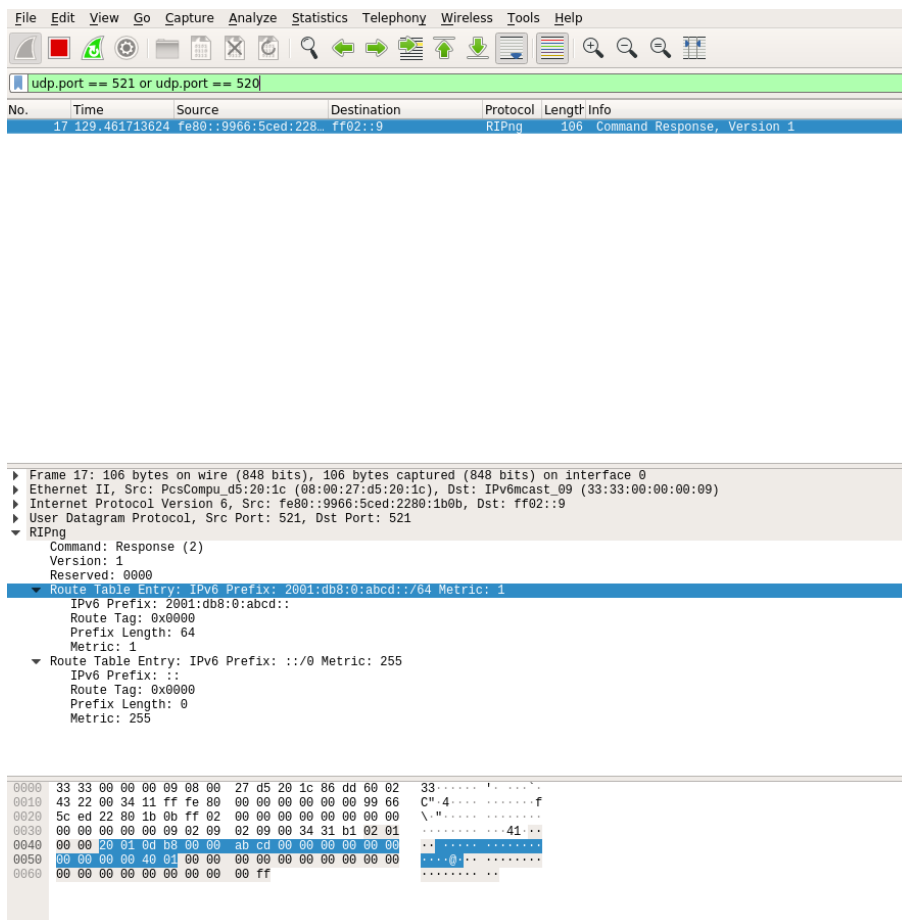


Figure 4: Sample of sent RIPng packet in Wireshark


```

Hello, this is Quagga (version 0.99.16).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
Routing> show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPv6, O - OSPFv3,
       I - ISIS, B - BGP, * - FIB route.

K>* ::/96 via ::1, lo0, rej
C>* ::1/128 is directly connected, lo0
K>* ::ffff:0.0.0.0/96 via ::1, lo0, rej
R>* 2001:db8:0:abcd::/64 [120/2] via fe80::9966:5ced:2280:1b0b, em0, 00:01:27
C>* fd00::/64 is directly connected, em0
C>* fd00:d5:3390::/64 is directly connected, lo0
C>* fd00:108:2c4c::/64 is directly connected, lo0
C>* fd00:4d4:6c::/64 is directly connected, lo0
C>* fd00:900:14d0::/64 is directly connected, lo0
K>* fe80::/10 via ::1, lo0, rej
C>* fe80::/64 is directly connected, lo0
C>* fe80::/64 is directly connected, em0
K>* ff02::/16 via ::1, lo0, rej
Routing>

```

Figure 5: Shown routing table on router demon where we can notice a malicious IP address 2001:db8:0:abcd::/64

5 Conclusion

To summarize, routing information protocols are used to communicate in smaller networks because they have a lot of limitations. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path and uses UDP as a transport protocol on port number 520 and 521. It is based on sending routing table every 30 minutes in a Response message. The only RIP protocol that contains authentication is RIPv2. As we could see, it is not hard to add fake IP address to a routing table.

References

- [1] Jeff Kellum, Mary Ellen Schutz, Patrick J. Conlan, and Christine O'Connor *Cisco Certified Entry Networking Technician STUDY GUIDE*. Neil Edde, Joseph B. Wikert, and Richard Swadley, Indianapolis, Indiana, 2008. ISBN 978-0-470-24702-0.
- [2] RFC 1058: RIP Version 1
<https://www.ietf.org/rfc/rfc1058.txt>
- [3] Lipcap library
<http://www.tcpdump.org/>
- [4] RFC 2453: RIP Version 2
<https://www.ietf.org/rfc/rfc2453.txt>
- [5] RFC 2080: RIPng for IPv6
<https://www.ietf.org/rfc/rfc2080.txt>
- [6] RFC 1827: IP Encapsulating Security Payload (ESP)
<https://www.ietf.org/rfc/rfc1827.txt>
- [7] RFC 1826: IP Authentication Header
<https://www.ietf.org/rfc/rfc1826.txt>
- [8] Sample Configuration for Authentication in RIPv2
<https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13719-50.html>