
Networkforyou
Welcome
To
Network for you
URPF
Networkforyou

Unicast Reverse Path Forwarding (uRPF):

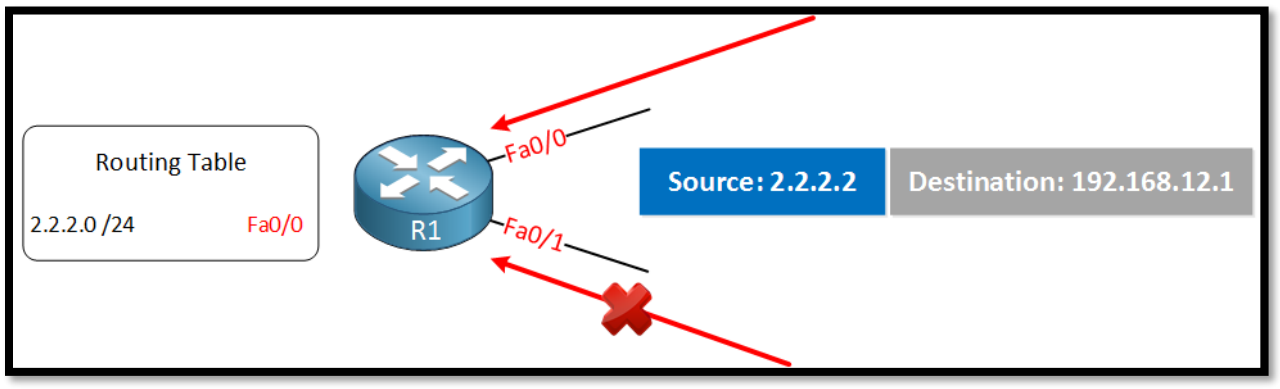
- **uRPF is a security feature that prevents these spoofing attacks**.
- Normally when our router receives unicast IP packets it only cares about.
- What is the destination IP address of this IP packet so I can forward it?
- If the IP packet has to be routed it will check the routing table for the destination IP address, select the correct interface and it will be forwarded.
- Our router really doesn't care about source IP addresses as it's not important for forwarding decisions.
- Because the router doesn't check the source IP address it is possible for attackers to spoof the source IP address and send packets that normally might have been dropped by the firewall or an access-list.
- **uRPF is a security feature that prevents these spoofing attacks**.
- Whenever your router receives an IP packet it will check if it has a matching entry in the routing table for the source IP address.
- If it doesn't match, the packet will be discarded.

uRPF has two modes:

- Strict mode
- Loose mode

Strict Mode:

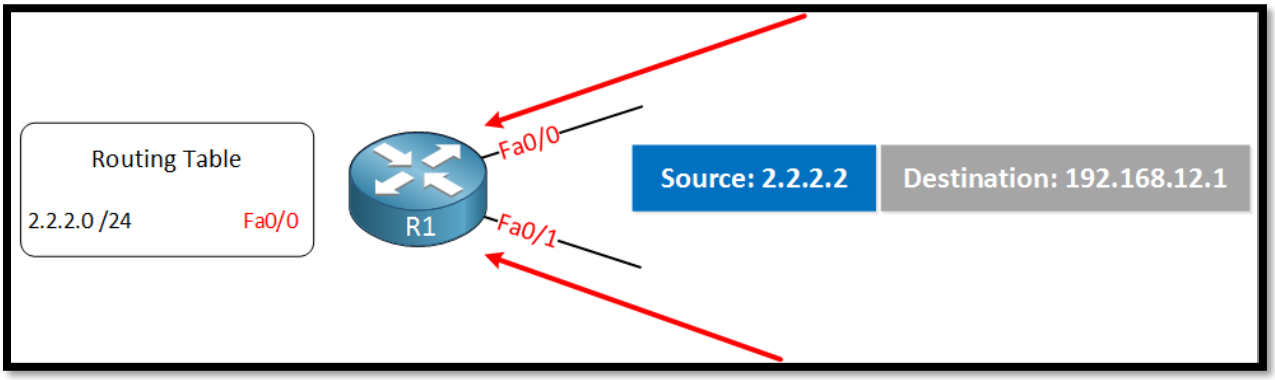
- Strict mode means that that router will perform **two checks for all incoming packets on a certain interface:**
- Do I have a matching entry for the **source in the routing table**?
- Do I use the **same interface to reach this source as where I received this packet on**?



- R1 has installed network 2.2.2.0 /24 in its routing table and in order to reach this network it will use the FastEthernet 0/0 interface.
- Suddenly this router receives an IP packet with source IP address 2.2.2.2 on both of its interfaces.
- The one it receives on the FastEthernet 0/0 will be accepted but the packet on the FastEthernet 0/1 interface will be dropped because this is not the interface we use to reach this source.

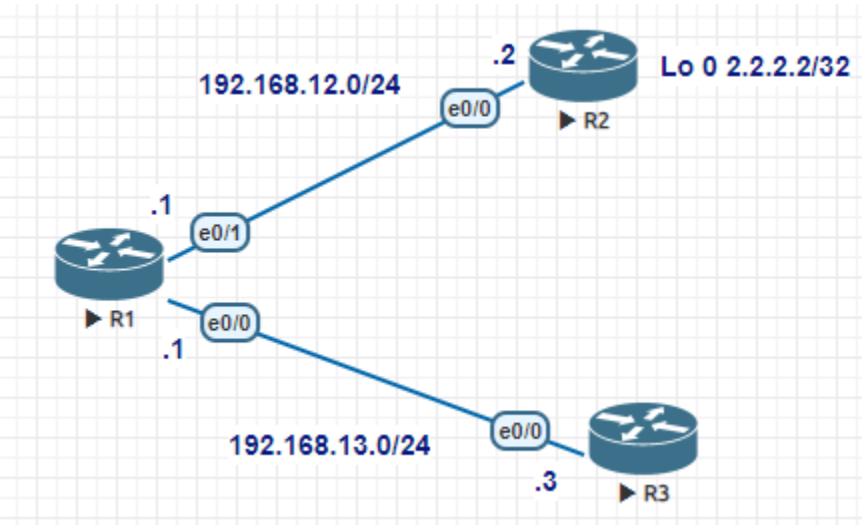
Loose Mode:

- Loose mode means that the router will perform **only a single check** when it receives an IP packet on an interface:
- **Do I have a matching entry for the source in the routing table?**
- When it passed this check, the packet is permitted. It doesn't matter if we use this interface to reach the source or not. Loose mode is useful when you are connected to more than one ISP and you use asymmetric routing.



- R1 has been configured for uRPF loose mode and receives an IP packet with source IP address 2.2.2.2 on both interfaces.
- Since it has an entry for this source in its routing table it will accept both packets.
- It doesn't care where it came from, as long as there is an entry in the routing table.

Lab time uRPF:



R1 Configuration:	R2 Configuration:
en config t hostname R1 int e0/0 ip add 192.168.13.1 255.255.255.0 no sh int e0/1 ip add 192.168.12.1 255.255.255.0 no sh	en config t hostname R2 int e0/0 ip add 192.168.12.2 255.255.255.0 no sh int lo 0 ip add 2.2.2.2 255.255.255.255 no sh
R3 Configuration:	
en config t hostname R3 int e0/0 ip add 192.168.13.2 255.255.255.0 no sh	

We will configure R1 with a static route so it can reach the loopback0 interface of R2:

R1(config)#ip route 2.2.2.2 255.255.255.255 192.168.12.2

This is what the routing table looks like now:

```
R1#show ip route | begin Gateway
Gateway of last resort is not set

      2.0.0.0/32 is subnetted, 1 subnets
S        2.2.2.2 [1/0] via 192.168.12.2
      192.168.12.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.12.0/24 is directly connected, Ethernet0/1
L        192.168.12.1/32 is directly connected, Ethernet0/1
      192.168.13.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.13.0/24 is directly connected, Ethernet0/0
L        192.168.13.1/32 is directly connected, Ethernet0/0
R1#
```

Now we'll configure uRPF strict mode on both interfaces:
R1(config)#interface Ethernet 0/0 R1(config-if)#ip verify unicast source reachable-via rx R1(config)#interface Ethernet 0/1 R1(config-if)#ip verify unicast source reachable-via rx
We can verify that it has been enabled on the interface like this:
R1#show ip interface Ethernet 0/0 | include verify **IP verify source reachable-via RX** R1#show ip interface Ethernet 0/1 | include verify **IP verify source reachable-via RX**

To test uRPF we'll send some pings from R2 first, these should be accepted:

To test uRPF we'll send some pings from R2 first, these should be accepted:
R2#ping 192.168.12.1 source loopback 0 Type escape sequence to abort. Sending 5, 100-byte ICMP Echos to 192.168.12.1, timeout is 2 seconds: Packet sent with a source address of 2.2.2.2 .!!!! **Success rate is 80 percent (4/5),** round-trip min/avg/max = 1/1/1 ms

As expected, this ping works. Now we will create a new loopback interface on R3 with the 2.2.2.2 IP address on it so that we can spoof this IP address:

```
R3(config)#interface loopback 0
R3(config-if)#ip address 2.2.2.2 255.255.255.255
```

Now I'll send some pings from this loopback:

```
R3#ping 192.168.13.1 source loopback 0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.13.1, timeout is 2 seconds:
Packet sent with a source address of 2.2.2.2
.....
Success rate is 0 percent (0/5)
```

The packets will make it to R1 but they will be dropped there, we can verify this as following:

```
R1#show ip interface Ethernet 0/0 | include drops
  4 verification drops
  0 suppressed verification drops
```


Above you see that the spoofed packets on the FastEthernet 0/0 interface have been dropped.

Now we'll configure uRPF loose mode on both interfaces:

```
R1(config)#interface Ethernet 0/0
R1(config-if)# no ip verify unicast source reachable-via rx
R1(config-if)# ip verify unicast source reachable-via any
R1(config)#interface Ethernet 0/1
R1(config-if)# no ip verify unicast source reachable-via rx
R1(config-if)# ip verify unicast source reachable-via any
```

We can verify that it has been enabled on the interface like this:

```
R1#show ip interface Ethernet 0/0 | include verify
IP verify source reachable-via ANY
R1#show ip interface Ethernet 0/1 | include verify
IP verify source reachable-via ANY
```

To verify that it actually works we'll enable a debug on R1:

```
R1#debug ip packet
IP packet debugging is on
```

Now we will send some pings from R2 and R3 using 2.2.2.2 as the source:

```
R2#ping 192.168.12.1 source loopback 0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.1, timeout is 2 seconds:
Packet sent with a source address of 2.2.2.2
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

---------------------------------------------------------------------------------

R3#ping 192.168.13.1 source loopback 0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.13.1, timeout is 2 seconds:
Packet sent with a source address of 2.2.2.2
.....
Success rate is 0 percent (0/5)
```


The pings from R2 will make it as this is the valid entry. The pings from R3 won't work because we don't have a valid route but the packets will be accepted by R1…take a look below:

```
R1#
*Nov 14 19:20:24.394: IP: s=2.2.2.2 (Ethernet0/1), d=192.168.12.1, len 100, input feature, uRPF(57),
rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Nov 14 19:20:24.394: IP: s=2.2.2.2 (Ethernet0/1), d=192.168.12.1, len 100, input feature, MCI
Check(101), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Nov 14 19:20:24.394: IP: tableid=0, s=2.2.2.2 (Ethernet0/1), d=192.168.12.1 (Ethernet0/1), routed via
RIB
*Nov 14 19:20:24.394: IP: s=2.2.2.2 (Ethernet0/1), d=192.168.12.1 (Ethernet0/1), len 100, rcvd 3
*Nov 14 19:20:24.394: IP: s=2.2.2.2 (Ethernet0/1), d=192.168.12.1, len 100, stop process pak for forus
packet
*Nov 14 19:20:24.394: IP: s=192.168.12.1 (local), d=2.2.2.2, len 100, local feature, Logical MN local(14),
rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Nov 14 19:20:24.394: IP: s=192.168.12.1 (local), d=2.2.2.2 (Ethernet0/1), len 100, sending
```

```
*Nov 14 19:20:24.394: IP: s=192.168.12.1 (local), d=2.2.2.2 (Ethernet0/1), len 100, sending full packet
*Nov 14 19:20:24.399: IP: s=2.2.2.2 (Ethernet0/1), d=192.168.12.1, len 100, input feature, uRPF(57),
rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
R1#
*Nov 14 19:20:36.194: IP: s=192.168.13.1 (local), d=2.2.2.2, len 100, local feature, Logical MN local(14),
rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Nov 14 19:20:36.194: IP: s=192.168.13.1 (local), d=2.2.2.2 (Ethernet0/1), len 100, sending
*Nov 14 19:20:36.194: IP: s=192.168.13.1 (local), d=2.2.2.2 (Ethernet0/1), len 100, sending full packet
R1#
*Nov 14 19:20:38.196: IP: s=2.2.2.2 (Ethernet0/0), d=192.168.13.1, len 100, input feature, uRPF(57),
rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Nov 14 19:20:38.196: IP: s=2.2.2.2 (Ethernet0/0), d=192.168.13.1, len 100, input feature, MCI
Check(101), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Nov 14 19:20:38.196: IP: tableid=0, s=2.2.2.2 (Ethernet0/0), d=192.168.13.1 (Ethernet0/0), routed via
RIB
*Nov 14 19:20:38.196: IP: s=2.2.2.2 (Ethernet0/0), d=192.168.13.1 (Ethernet0/0), len 100, rcvd 3
*Nov 14 19:20:38.196: IP: s=2.2.2.2 (Ethernet0/0), d=192.168.13.1, len 100, stop process pak for forus
packet
R1#
*Nov 14 19:20:38.196: IP: s=192.168.13.1 (local), d=2.2.2.2, len 100, local feature, Logical MN local(14),
rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Nov 14 19:20:38.196: IP: s=192.168.13.1 (local), d=2.2.2.2 (Ethernet0/1), len 100, sending
*Nov 14 19:20:38.196: IP: s=192.168.13.1 (local), d=2.2.2.2 (Ethernet0/1), len 100, sending full packet
```