

Remote shell using raw socket

Distributed System

Group 9

University of Science And Technology of Hanoi

ICT Department

April 1, 2019

1 Introduction

2 Raw Socket

Introduction

Introduction

1 Remote Shell

- The remote shell (rsh) is a command line computer program that can execute shell commands as another user, and on another computer across a computer network.

Raw Socket

Raw Socket

- Can determine every section of packet, either header or payload
- Simply put raw sockets provide a way to bypass the whole network stack traversal of a packet and deliver it directly to an application.

```
# L3 socket , Network Layer Protocol = IPv4  
socket(AF_INET, RAW_SOCKET, ...)
```

```
# L3 socket , Network Layer Protocol = IPX  
socket(AF_IPX, RAW_SOCKET, ...)
```

```
# L3 socket , Network Layer Protocol = IPv6  
socket(AF_INET6, RAW_SOCKET, ...)
```

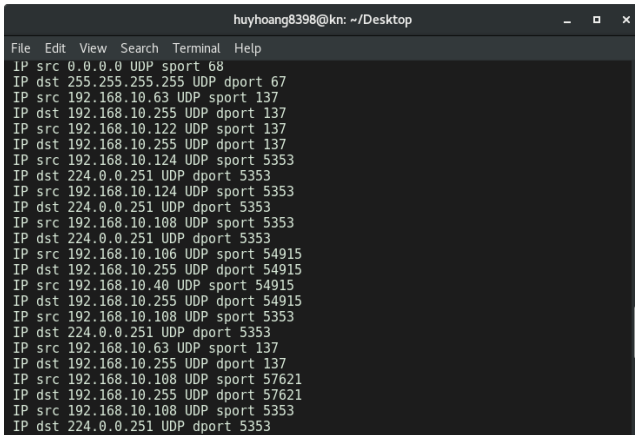
```
# L2 socket , Data-link Layer Protocol = Ethernet  
socket(AF_PACKET, RAW_SOCKET, ...)
```

Raw Socket

| Type header | IP header | Your header | payload |

- More
- Linux manual page - RAW

Example

A screenshot of a terminal window titled 'huyhoang8398@kn: ~/Desktop'. The terminal displays a list of network traffic logs, each line representing a packet capture. The logs show source and destination IP addresses, source and destination ports, and the protocol used (UDP). The traffic appears to be a series of requests and responses between various IP addresses, including 0.0.0.0, 255.255.255.255, 192.168.10.63, 192.168.10.255, 192.168.10.122, 192.168.10.124, 224.0.0.251, 192.168.10.108, 192.168.10.106, 192.168.10.40, 192.168.10.100, and 192.168.10.101. The ports used are 68, 137, 5353, 54915, 57621, and 5353. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

```
huyhoang8398@kn: ~/Desktop
File Edit View Search Terminal Help
IP src 0.0.0.0 UDP sport 68
IP dst 255.255.255.255 UDP dport 67
IP src 192.168.10.63 UDP sport 137
IP dst 192.168.10.255 UDP dport 137
IP src 192.168.10.122 UDP sport 137
IP dst 192.168.10.255 UDP dport 137
IP src 192.168.10.124 UDP sport 5353
IP dst 224.0.0.251 UDP dport 5353
IP src 192.168.10.124 UDP sport 5353
IP dst 224.0.0.251 UDP dport 5353
IP src 192.168.10.108 UDP sport 5353
IP dst 224.0.0.251 UDP dport 5353
IP src 192.168.10.106 UDP sport 54915
IP dst 192.168.10.255 UDP dport 54915
IP src 192.168.10.40 UDP sport 54915
IP dst 192.168.10.255 UDP dport 54915
IP src 192.168.10.108 UDP sport 5353
IP dst 224.0.0.251 UDP dport 5353
IP src 192.168.10.63 UDP sport 137
IP dst 192.168.10.255 UDP dport 137
IP src 192.168.10.108 UDP sport 57621
IP dst 192.168.10.255 UDP dport 57621
IP src 192.168.10.108 UDP sport 5353
IP dst 224.0.0.251 UDP dport 5353
```

Figure 1: Server listening for all data



Code Implementation

- Assign Destination, Source MAC address, protocol

```

sudo python3 client.py
~/Desktop/final
> sudo python3 client.py
ls
b'commandls'
b'output \nclient.py\nlistener.py\n_pycache\nRawReceive.py\nserver.py\nstringexec.py\nstringexec.pyc\ntest.txt\ntest.zip\n'
ls
b'commandls'
b'output \nclient.py\nlistener.py\n_pycache\nRawReceive.py\nserver.py\nstringexec.py\nstringexec.pyc\ntest.txt\ntest.zip\n'

sudo python server.py
test.txt
test.zip
client.py
listener.py
_pycache
RawReceive.py
server.py
stringexec.py
stringexec.pyc
test.txt
test.zip
client.py
listener.py
_pycache
RawReceive.py
server.py
stringexec.py
stringexec.pyc
test.txt
test.zip

sudo tcpdump -i lo
0x0050: 7374 7269 6e67 6578 6563 2e70 7963 0a74 stringexec.pyc.t
0x0060: 6573 742e 7478 740a 7465 7374 2e7a 6970 est.txt.test.zip
0x0070: 0a
10:12:04.028296 08:00:27:8e:75:44 (oui Unknown) > 08:00:27:dd:d7:43 (oui Unknown), ethertype Unknown (0x88b5), length 23:
0x0000: 636f 6464 616e 646c 73 commandls
10:12:04.033454 08:00:27:dd:d7:43 (oui Unknown) > 08:00:27:8e:75:44 (oui Unknown), ethertype Unknown (0x88b5), length 127:
0x0000: 6f75 7470 7574 200a 636c 6965 6e74 2e70 output..client.p
0x0010: 790a 6c69 7374 656e 6572 2e70 790a 5f5f y.listener.py._
0x0020: 7079 6361 6368 655f 5f0a 5261 7752 6563 pycache._RawRec
0x0030: 6569 7665 2e70 790a 7365 7276 6572 2e70 eive.py.server.p
0x0040: 790a 7374 7269 6e67 6578 6563 2e70 790a y.stringexec.py.
0x0050: 7374 7269 6e67 6578 6563 2e70 7963 0a74 stringexec.pyc.t
0x0060: 6573 742e 7478 740a 7465 7374 2e7a 6970 est.txt.test.zip
0x0070: 0a
10:12:04.038231 08:00:27:dd:d7:43 (oui Unknown) > 08:00:27:8e:75:44 (oui Unknown), ethertype Unknown (0x88b5), length 127:
0x0000: 6f75 7470 7574 200a 636c 6965 6e74 2e70 output..client.p
0x0010: 790a 6c69 7374 656e 6572 2e70 790a 5f5f y.listener.py._
0x0020: 7079 6361 6368 655f 5f0a 5261 7752 6563 pycache._RawRec
0x0030: 6569 7665 2e70 790a 7365 7276 6572 2e70 eive.py.server.p
0x0040: 790a 7374 7269 6e67 6578 6563 2e70 790a y.stringexec.py.
0x0050: 7374 7269 6e67 6578 6563 2e70 7963 0a74 stringexec.pyc.t
0x0060: 6573 742e 7478 740a 7465 7374 2e7a 6970 est.txt.test.zip
0x0070: 0a

```