Outline

• Multiple Access Point Connections
• Testing
• Routing Tables
• Redirecting Traffic
• Selecting Routing Tables
• Testing Routing Tables
Multiple Access Points Connections (1)

- Linux provides ‘virtual managed interfaces’ [1]
- Enables multiple access point connections
- Wireless interface card was an Atheros card AR928X
- To be tested on other wireless interface cards

Multiple Access Points Connections (2)

Example[1]:

Virtual interface 1 (managed0):

```
 iw phy phy0 interface add managed0 type managed
 ip link set managed0 address 12:34:56:78:9A:BC
 ifconfig managed0 up
 iwconfig managed0 essid "tsunami" ap 00:07:50:d5:a2:3a
 ifconfig managed0 10.0.0.5 netmask 255.255.255.0
```

Virtual interface 2 (managed1):

```
 iw phy phy0 interface add managed1 type managed
 ip link set managed1 address AA:AA:BB:BB:BB
 ifconfig managed1 up
 iwconfig managed1 essid "tsunami2" ap 00:07:50:d5:a7:f6
 ifconfig managed1 192.168.1.3 netmask 255.255.255.0
```


Testing

- Different SSID Same Channel (11) ✓
- Different SSID Different Channel (1,11) ✗
- Same SSID Different Channel (1,11) ✗
Limitations

- My current work does not support WPA or encryption
  - WPA requires `wpa_supplicant`[1]
  - `wpa_supplicant` ‘appears’ to only allow one connection at a time
  - Needs further investigation and/or perhaps extending `wpa_supplicant` to support multiple connections

Routing Tables

- The routing tables needed to be changed on both the Laptop and PC
  - `iproute2`[1] was used to provide multiple routing tables
  - Creating multiple tables (`iproute2`)
  - Mark packets based on the flow id value (`fwmark`)[2]
  - Redirect marked packets to nominated routing table (`iptables`)[2]

Changing Routing Tables (iproute2)

- **iproute2** creates the multiple routing tables
  - tsunami table has a value of 1
  - tsunami2 table has a value of 2

First step:

```
/etc/iproute2/rt_tables
1 tsunami
2 tsunami2
```

Kernel Options needed

- To enable this feature (marking of packets), the following options needs to be enabled in the kernel*

```
IP: advanced router (CONFIG_IP_ADVANCED_ROUTER) [Y/n/?]
IP: policy routing (CONFIG_IP_MULTIPLE_TABLES) [Y/n/?]
IP: use netfilter MARK value as routing key (CONFIG_IP_ROUTE_FWMARK[Y/n/?])
```

*Kernel Options for 3.5.0-22-generic

Redirecting traffic (fwmark)

Second Step:

Laptop:

1. `ip rule add fwmark 1 table tsunami`
2. `ip route add default via 10.0.0.6 table tsunami`
3. `ip rule add fwmark 2 table tsunami2`
4. `ip route add default via 192.168.1.6 table tsunami2`

PC (Router):

```
ip rule add fwmark 1 table tsunami
ip route add default via 10.0.0.5 table tsunami
ip rule add fwmark 2 table tsunami2
ip route add default via 192.168.1.3 table tsunami2
```

Selecting Routing Tables (iptables) (1)

- Iptables has 4 different type of tables (raw, nat, filter and mangle)
- NAT table should only be used for Network Address Translation on different packets
- Raw table should only be used when no connection tracking system is desired
- Filter table uses 3 different type of chains:
  - Input chain – Processes packets that are destined for the host (not needed)
  - Output chain - Processes packets that are sent by the host (needed for Laptop and Mobile Node)
  - Forward chain – Processes packets that are received by the host and destined to another host (needed for wired router)
- Mangle table is very important – without it the marking of packets would not be possible!
Selecting Routing Tables (iptables) (2)

Insert a rule

The protocol to match is udp only

If a packet matches this rule, then mark the packet with a value of 2 – this means routing table 2

iptables -I OUTPUT -p udp -t mangle -j MARK --set-mark 2

Use the Output chain

The table that will be used is mangle (needed for marking of packets!)

Testing the Route Tables (1)

Example:

iptables -I OUTPUT -p udp -t mangle -j MARK --set-mark 2
iptables -I OUTPUT ! -p udp -t mangle -j MARK --set-mark 1

View Entries:

iptables -v -L -t mangle

Delete ALL Entries

iptables -F -t mangle
Reverse Path Filtering

- Reverse Path Filtering prevents source address spoofing
- Checks the source IP of each packet received on that interface against its routing table
- If the route is not in the routing table, the packet will be dropped

- Reverse path filtering will cause packets redirected via the ‘wrong’ interface to be dropped!

Turn it off by doing [1]:

```
for i `ls /proc/sys/net/ipv4/conf`; do echo “0” > /proc/sys/net/ipv4/conf/$i/rp_filter; done;
```

Priority flow Redirection

- Extend generic rule to match specific flows
- Command line script which takes five arguments:
  - (source ip:source port:destination ip:destination port:protocol)
- Based on these arguments the \texttt{priority\_add} script and \texttt{priority\_delete} script were tested, and will be integrated with the DIFFUSE system.

Currently working on

DIFFUSE
Fake Classifier Node
(exporter)

\rightarrow

DIFFUSE
Action Node
(collector)

\uparrow

Flow information
(from DIFFUSE)

\downarrow

Scripts:
- priority\_add
- priority\_delete

\rightarrow

iptables
Conclusion

• Successful
  • Multiple access point connections (from single NIC)
  • Changing the routing tables to redirect traffic to a different path
  • Selecting routing tables based on arguments to prioritise a particular flow

• Ongoing Work
  • (Integrating DIFFUSE to do work and building/testing the classifier)

What I’ve gained from the Internship?

• Basic introduction via coursework enabled me to expand my experience whilst doing the internship, which improved my confidence

• More familiar with FreeBSD and Linux operating systems
What I’ve gained from the Internship?

• Using my theoretical and practical knowledge from my course units enabled me to develop and test various possible solutions needed to achieve the projects criteria

• It enabled me to experience more challenging tasks that can occur and how to accomplish them appropriately

Thank you

• A special thanks to, Grenville Armitage, Philip Branch and Jason But for giving me the opportunity to work at CAIA for my internship.