Tricks of the MiTM Trade

Using the Linux networking stack to your advantage

cmm – Chris McCoy (@chris_mccoy)

16 May 2020

RED TEAM VILLAGE
MAY'HEM SUMMIT

About: cmm

Chris McCoy is a principal engineer in Cisco's Advanced Security Initiatives Group (ASIG). He has over 25 years of experience in the networking and security industry. He has a passion for computer security, finding flaws in mission-critical systems, and designing mitigations to thwart motivated and resourceful adversaries.

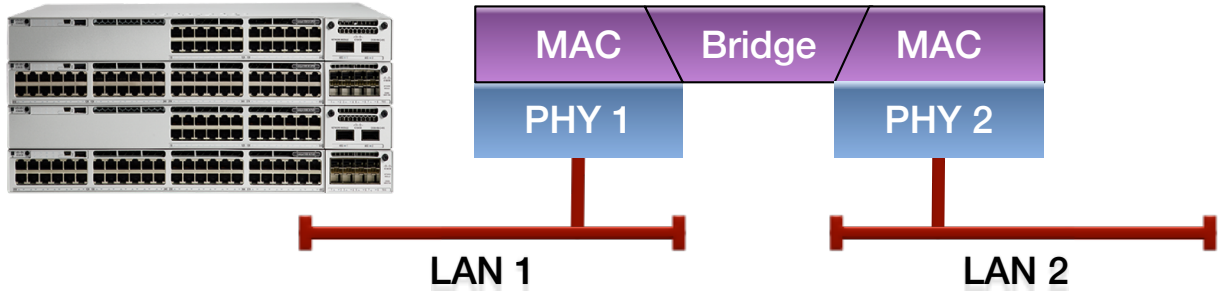
He was formerly with Spirent Communications and the U.S. Air Force. Chris is CCIE certified in the Routing & Switching and Service Provider (Emeritus) tracks, which he has held for over 10 years. You can follow Chris on Twitter **@chris_mccoy**.

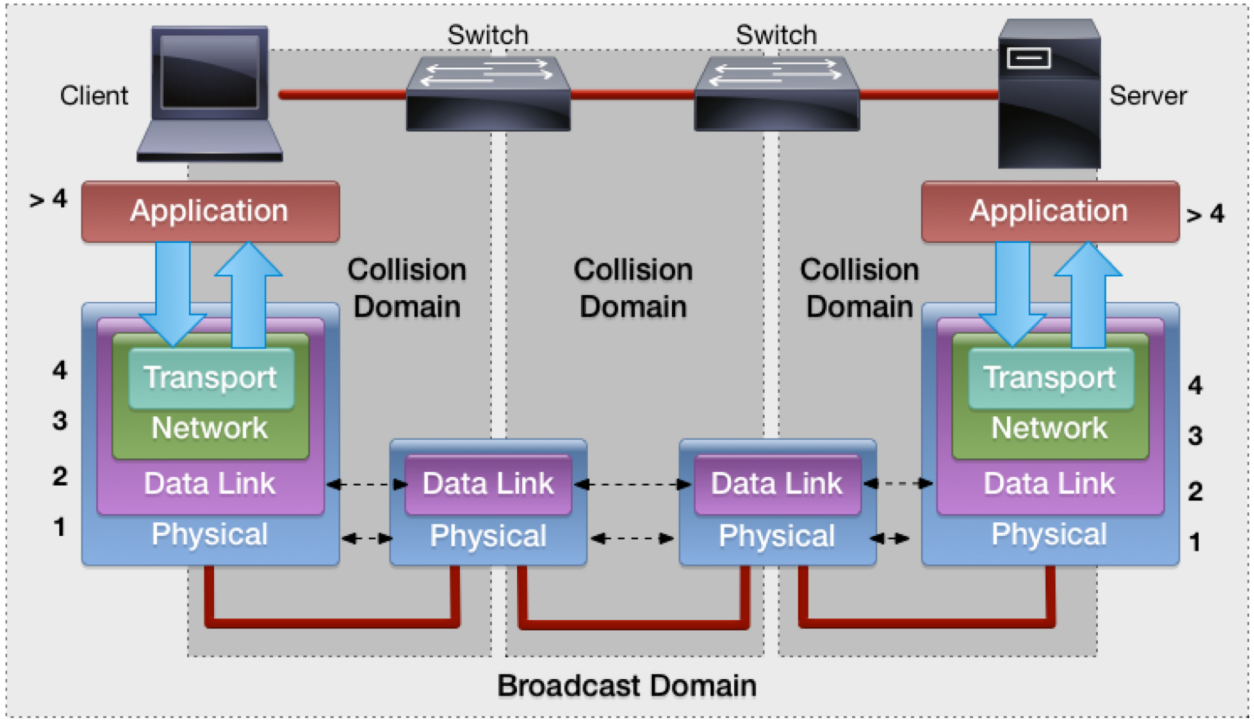
Agenda

- Introduction to Man-in-the-Middle using Linux as a Router
- Isolating the MiTM/Attacker Machine
- Using Linux as a “One-armed” Router
- Using Linux as a Passthrough Router with Demos:
  - Intercepting DNS
  - Intercepting HTTPS
  - Intercepting SSH
- Using Linux as a MiTM Bridge

Introduction

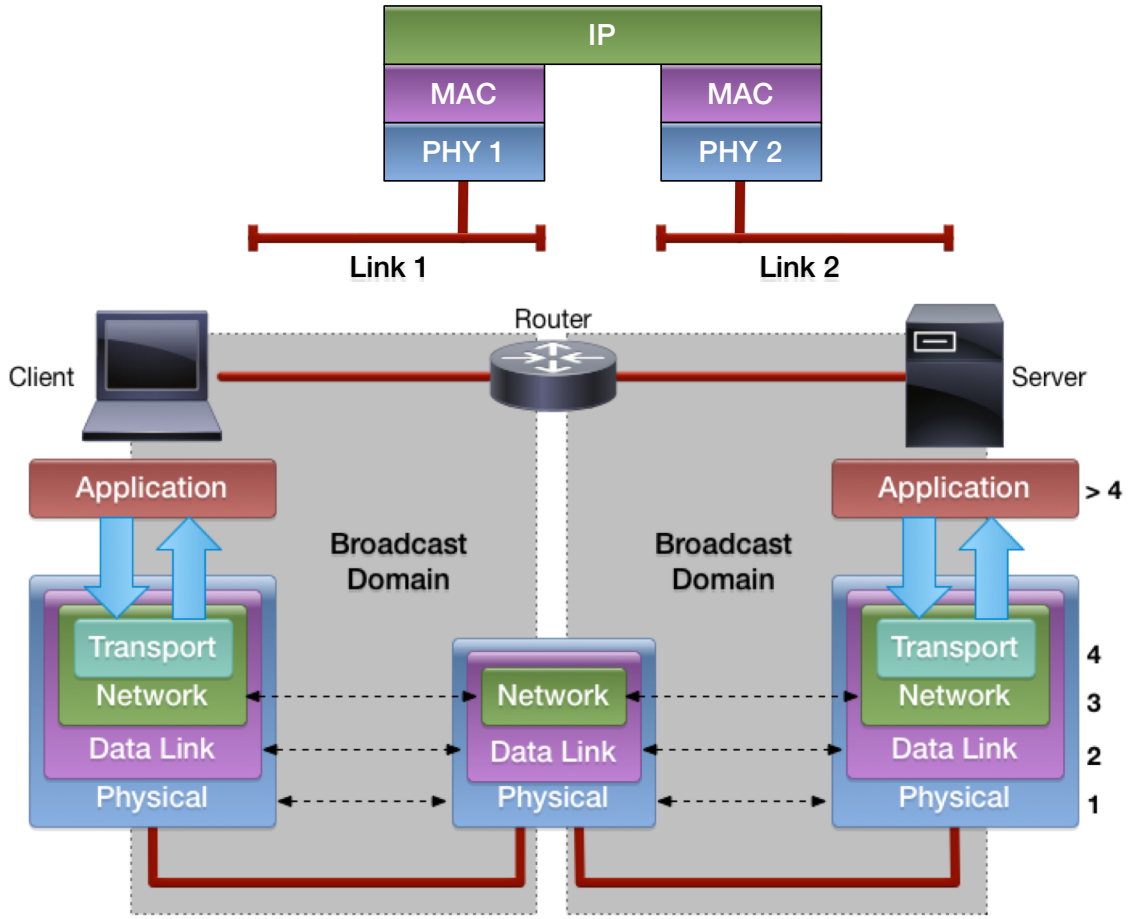
Understand What Bridges and Switches Are...





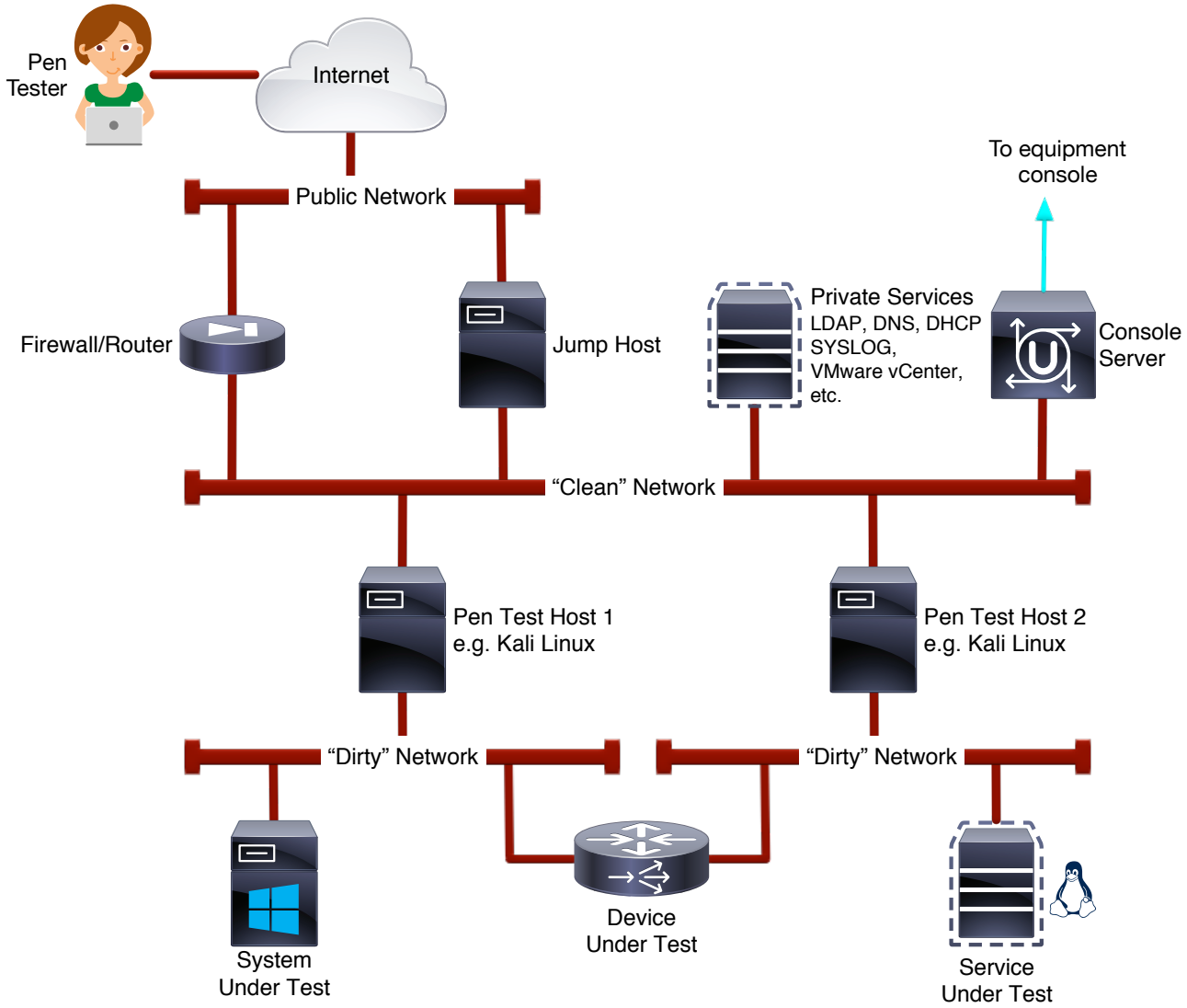
- Switches create a broadcast domain from a collection of individual LAN segments
- Switches learn, filter, and forward frames based on source and destination MAC addresses
- Switches can be internally partitioned using Virtual LANs (VLANs), which can be carried externally using 802.1Q tagged frames. Cisco calls these links "**Trunks**"

Understand What Routers Are…



- Routers create internetworks from groups of Broadcast Domains
  - These can be Links or VLANs
- Routers do not forward broadcast packets
- Hosts must be informed of the router's IP address on the link (Default Gateway) to forward packets
- **The most specific route and lowest cost metric wins for routing traffic to a destination**

Structure Your Lab As Shown...



Use PCI forwarding (VT-d) to place physical Interfaces in VMs!

Layer 3
Using Kali Linux as a Router

Learn to use iproute2 (`ip`) commands...

Brief
Colors
All objects
Addresses (abbreviated)

```
$ ip -br -c -all addr
lo                UNKNOWN        127.0.0.1/8 ::1/128
enp9s0            UP             10.40.1.31/24 fe80::66f6:9dff:fe32:b7a/64
enp10s0           UP             10.41.1.31/24 fe80::66f6:9dff:fe32:b7b/64
enp19s0           DOWN
enp20s0           DOWN
enp1s0f0          DOWN           10.20.1.31/22
enp1s0f1          DOWN
docker0           DOWN           169.254.0.1/24
cali34e28bb8347@enp10s0 UP                fe80::585c:aaff:fed6:3337/64
calica54491c42e@enp10s0 UP                fe80::8466:d4ff:fe91:94a9/64
calicb033e13d28@enp10s0 UP                fe80::ecc6:41ff:feff:e76/64
tunl0@NONE        UNKNOWN        10.40.168.192/32
```

https://baturin.org/docs/iproute2/

Linux IP Stack Configuration

`sysfs - /sys/class/net/<intf-name>`

- Network configuration and status, more oriented towards layer 2 and below

`proc - /proc/net`

- Basic network configuration and status, more oriented towards Layer 3+

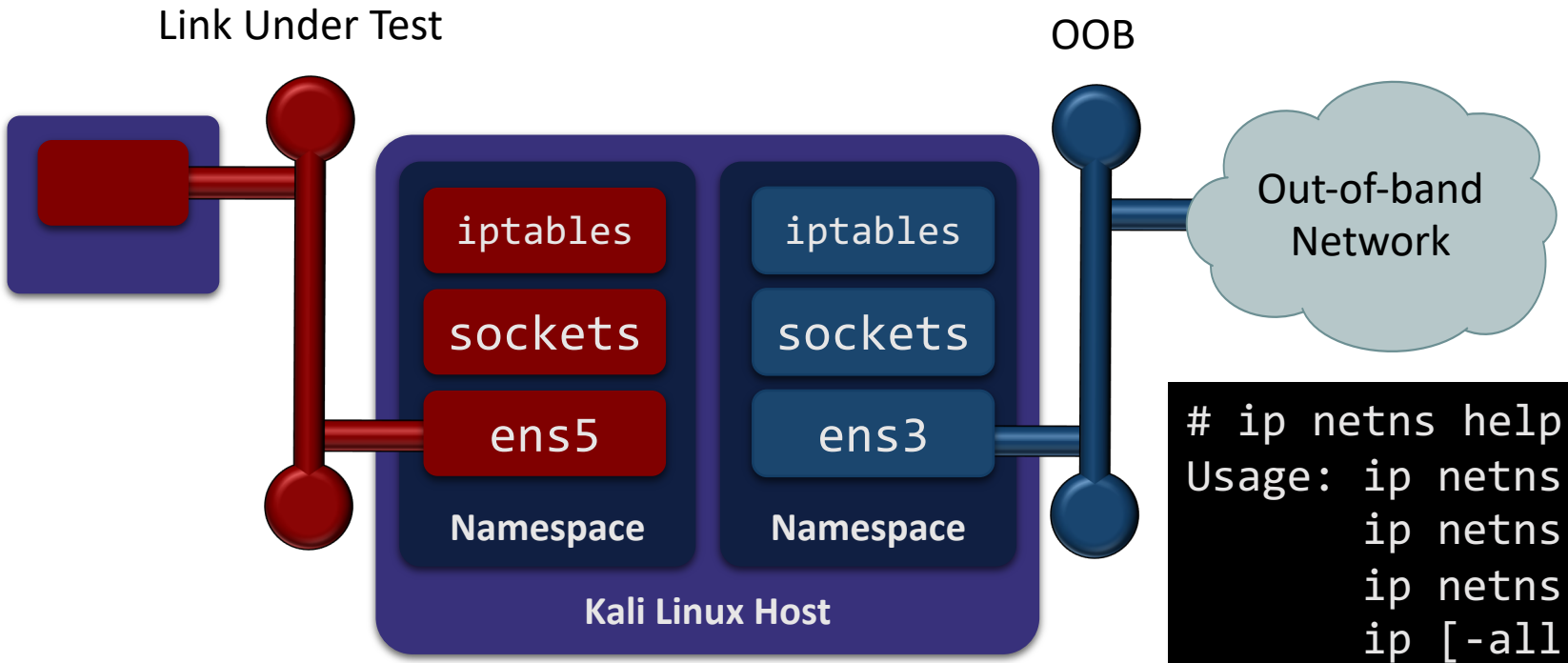
`proc - /proc/sys/net/<network-protocol>/conf/<intf-name>`

- **Probably the most useful**. Detailed network configuration and status

- These are all reflected by the Network Namespace currently in use

kernel.org Documentation

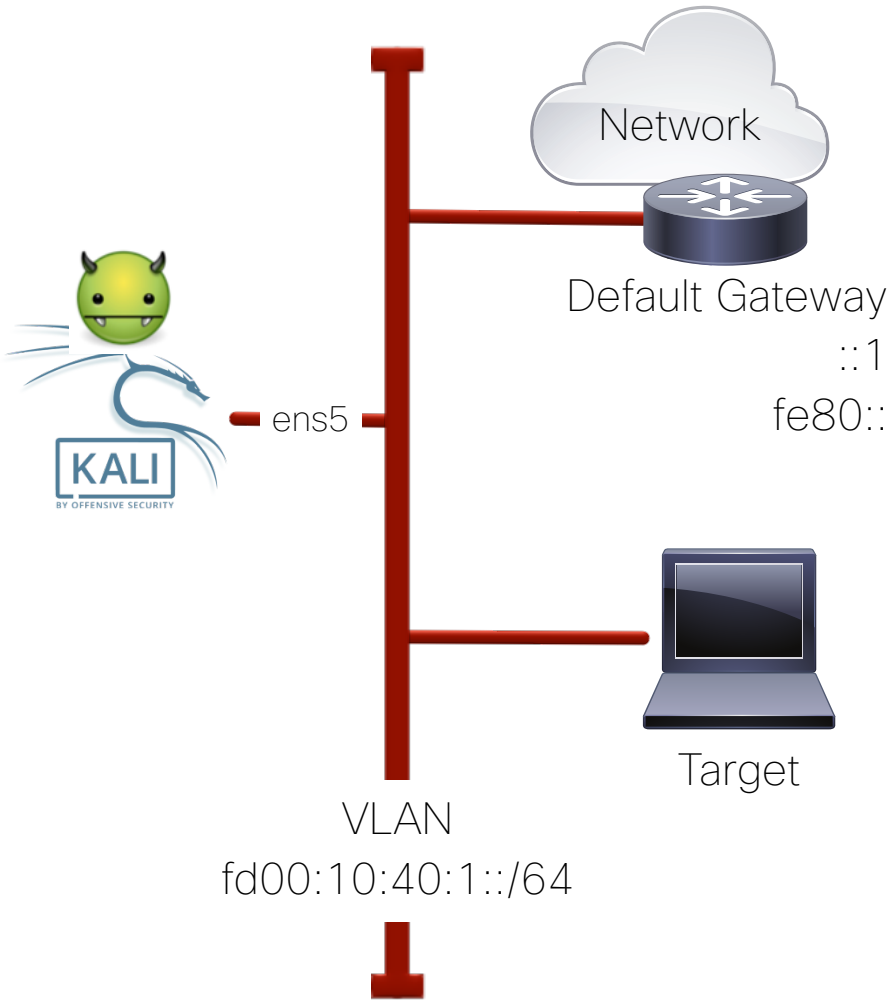
Isolate Yourself Using Network Namespaces




```
# ip netns help
Usage: ip netns list
       ip netns add NAME
       ip netns set NAME NETNSID
       ip [-all] netns delete [NAME]
       ip netns identify [PID]
       ip netns pids NAME
       ip [-all] netns exec [NAME] cmd ...
       ip netns monitor
       ip netns list-id
```

https://baturin.org/docs/iproute2/ - Network namespace management

IPv6 is Active by Default



To Disable IPv6 SLAAC

```
# echo 0 > \
   /proc/sys/net/ipv6/conf/<intf-name>/autoconf
```

To Disable Listening of Router Advertisements for Routes

```
# echo 0 > \
   /proc/sys/net/ipv6/conf/<intf-name>/accept_ra
```

To Disable IPv6 SLAAC by setting the address gen mode to None

```
# echo 1 > \
   /proc/sys/net/ipv6/conf/<intf-name>/addr_gen_mode

# ip link set <intf-name> down
# ip link set <intf-name> up
```

To Disable IPv6 completely:

```
# echo 1 > \
  /proc/sys/net/ipv6/conf/<intf-name>/disable_ipv6
```

kernel.org Documentation

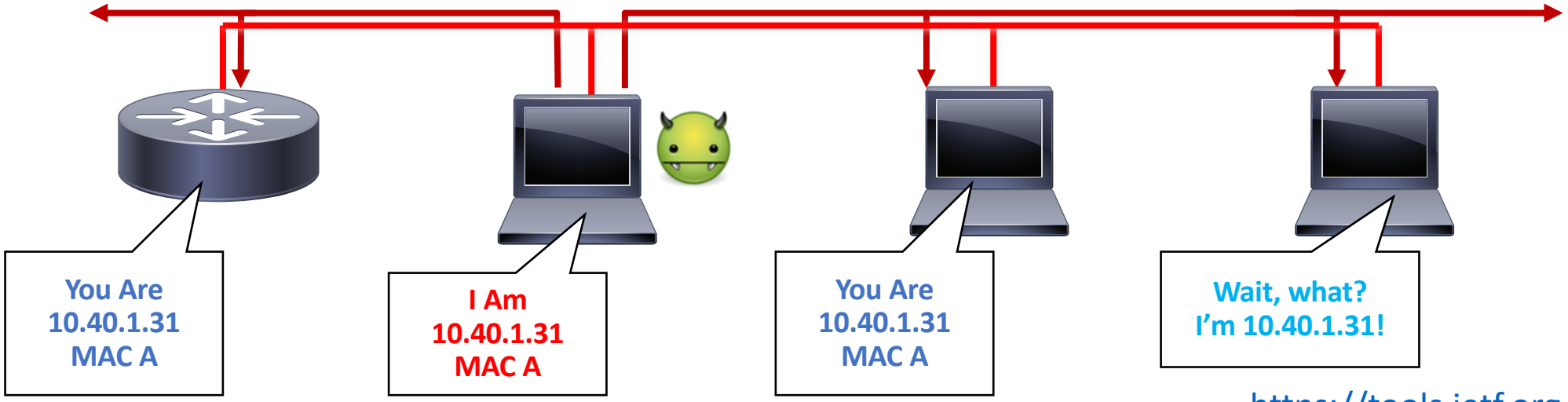
```
root@cmm-40g:~#
```

Attacker

Layer 3 Man in the Middle: ARP Cache Poisoning and the "One-armed Router"

Gratuitous ARP & Cache Poisoning

- According to the ARP RFC, a client is allowed to send an unsolicited ARP reply; this is called a gratuitous ARP; other hosts on the same subnet can store this information in their ARP tables
- Anyone can claim to be the owner of any IP/MAC address they like
- ARP cache poisoning attacks use this to redirect traffic

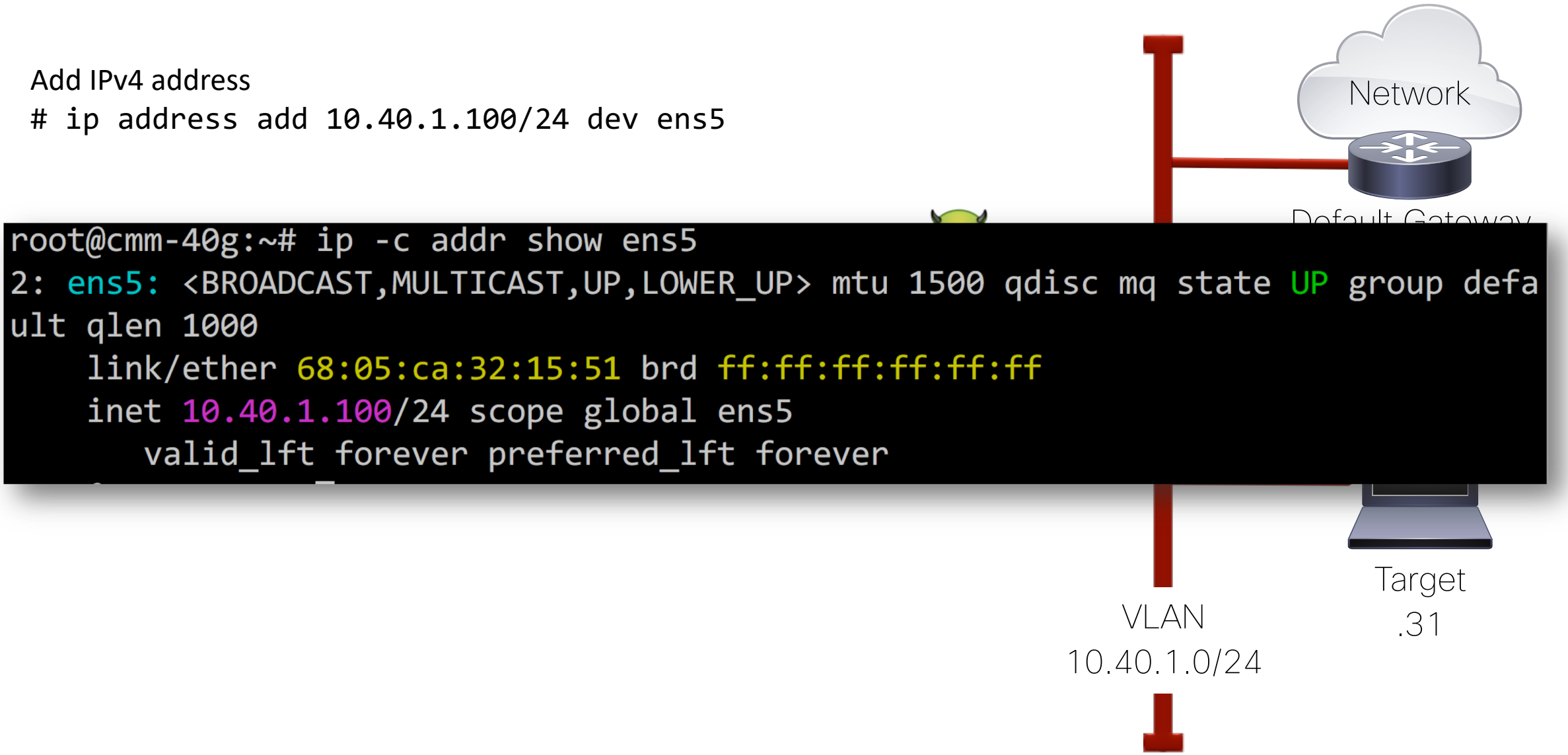


https://tools.ietf.org/html/rfc826

The “One-armed” Router

Add IPv4 address

```
# ip address add 10.40.1.100/24 dev ens5
```



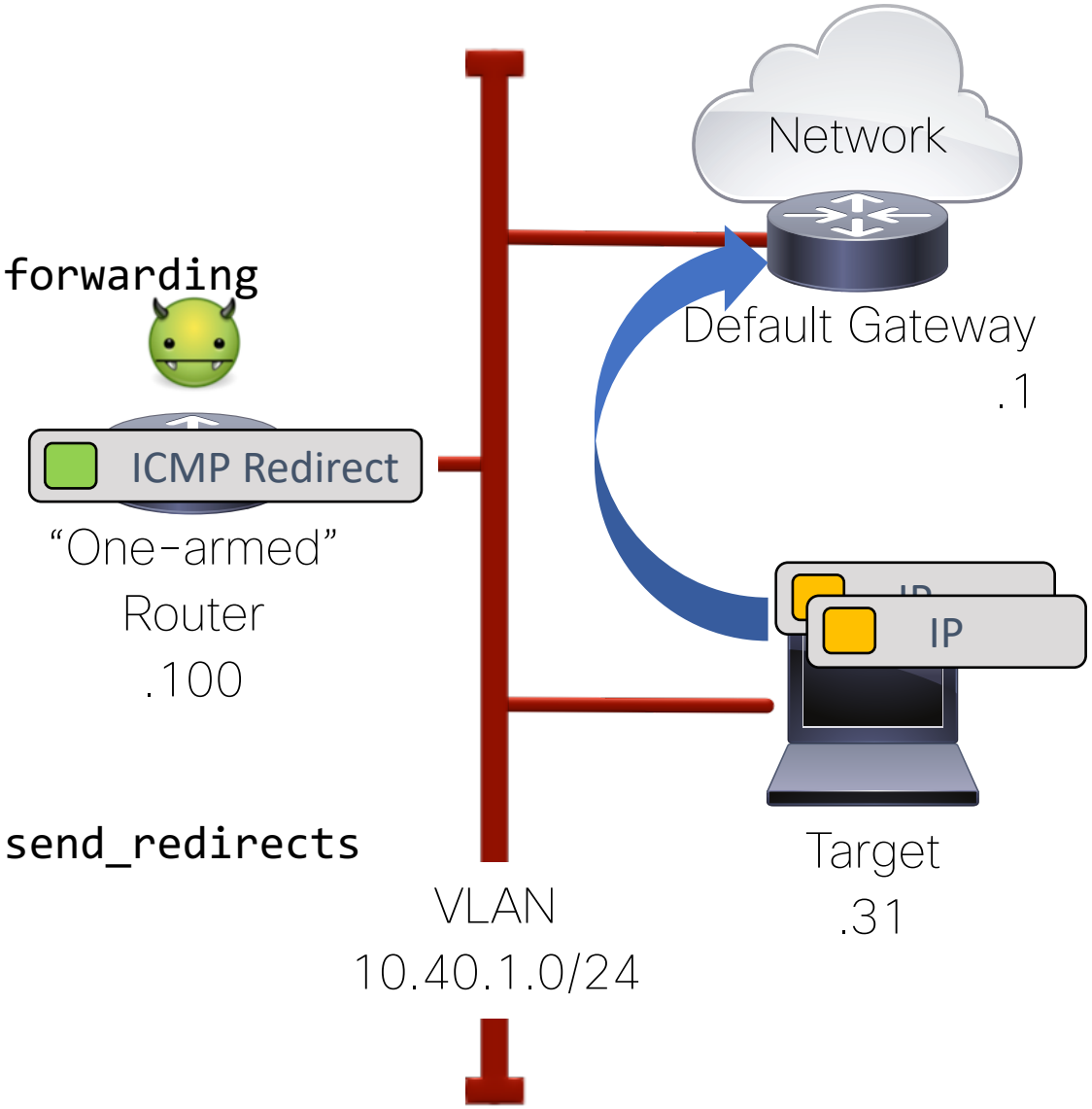
“One-armed” Router Behavior

To Enable IPv4 Forwarding:

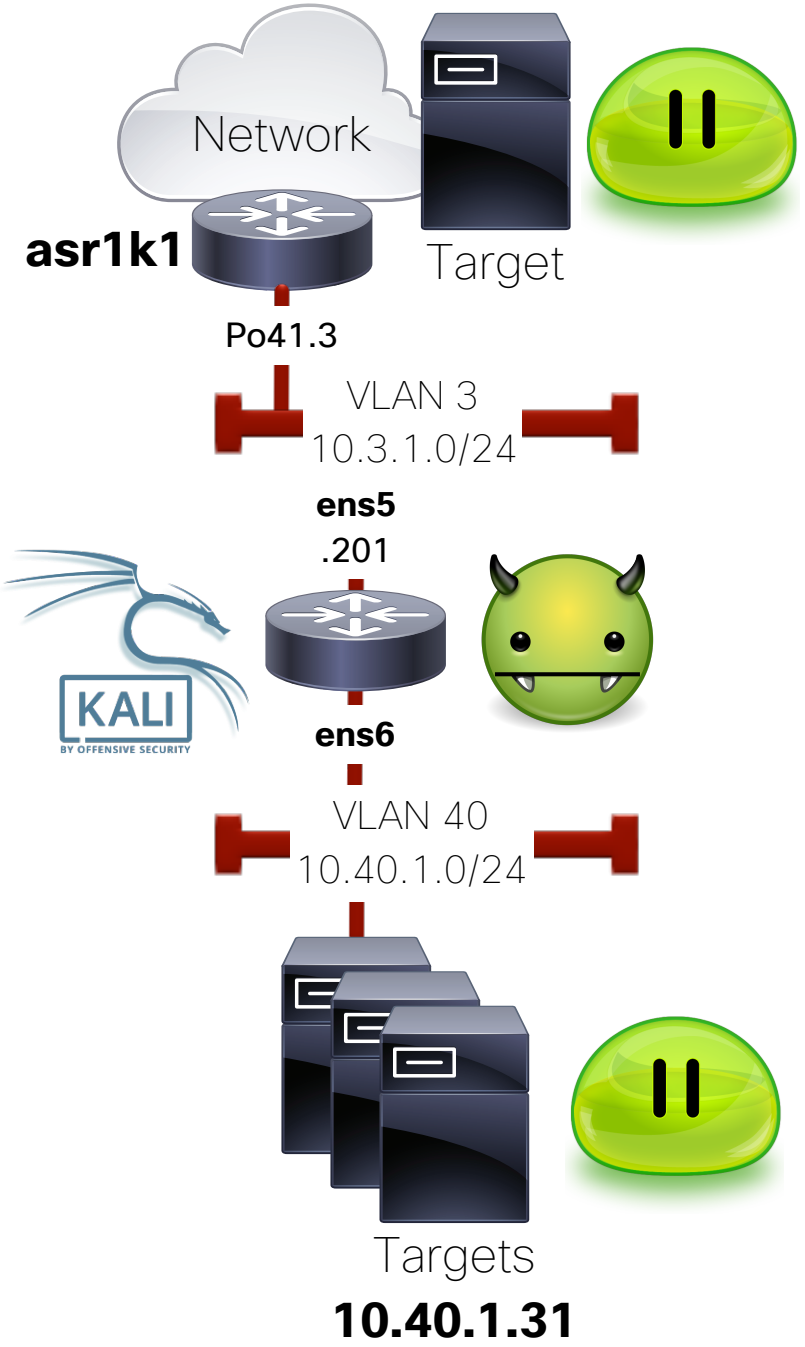
```
# echo 1 > \
  /proc/sys/net/ipv4/conf/<intf-name>/forwarding
```

To Disable ICMP Redirects:

```
# echo 0 > \
  /proc/sys/net/ipv4/conf/<intf-name>/send_redirects
```

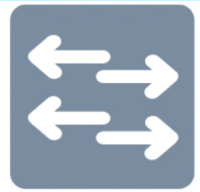


Layer 3 Man in the Middle : Target Network & Demo

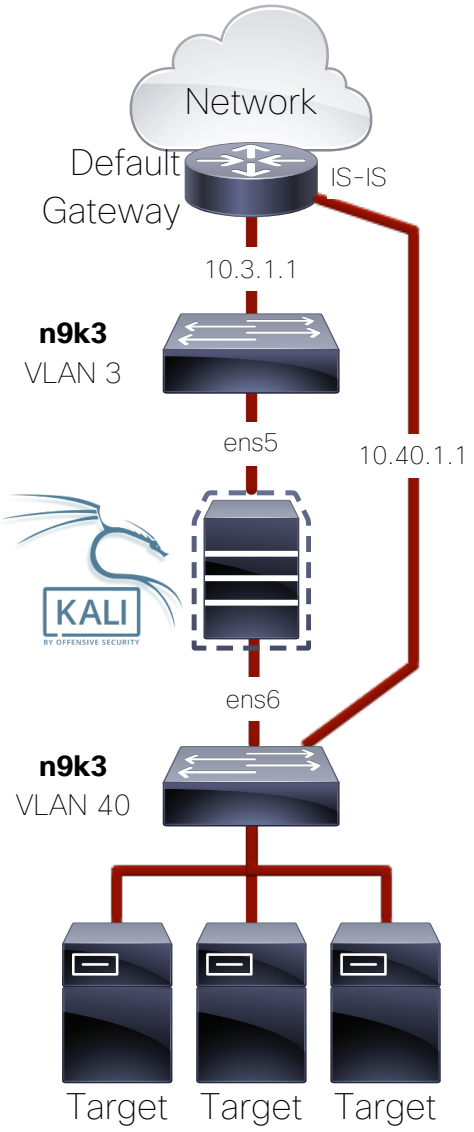

Network
asr1k1
Target
Po41.3
VLAN 3
10.3.1.0/24
ens5
.201
KALI
BY OFFENSIVE SECURITY
ens6
VLAN 40
10.40.1.0/24
Targets
10.40.1.31

Layer 3 Man in the Middle : Isolate Targets with VLANs

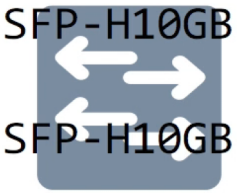
```
knv5801-n9k3#
```

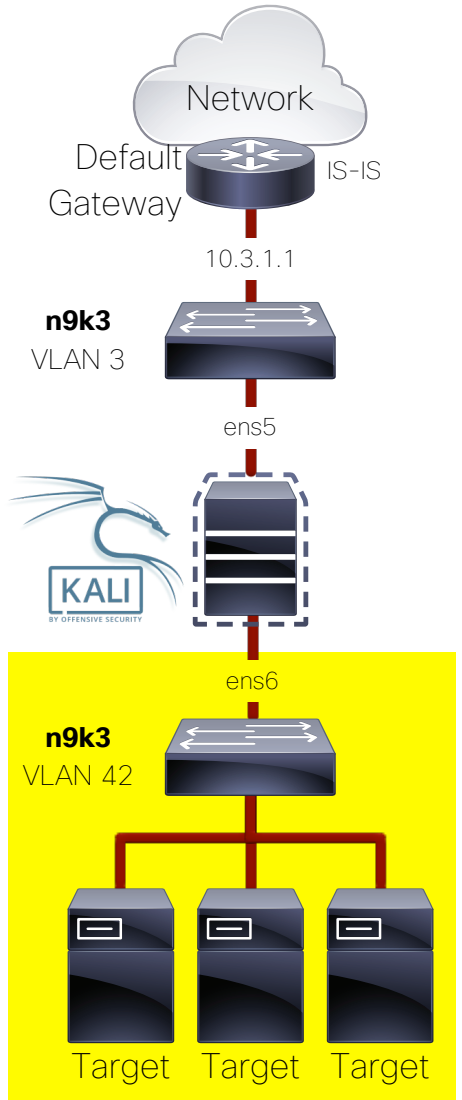


Switch
Console



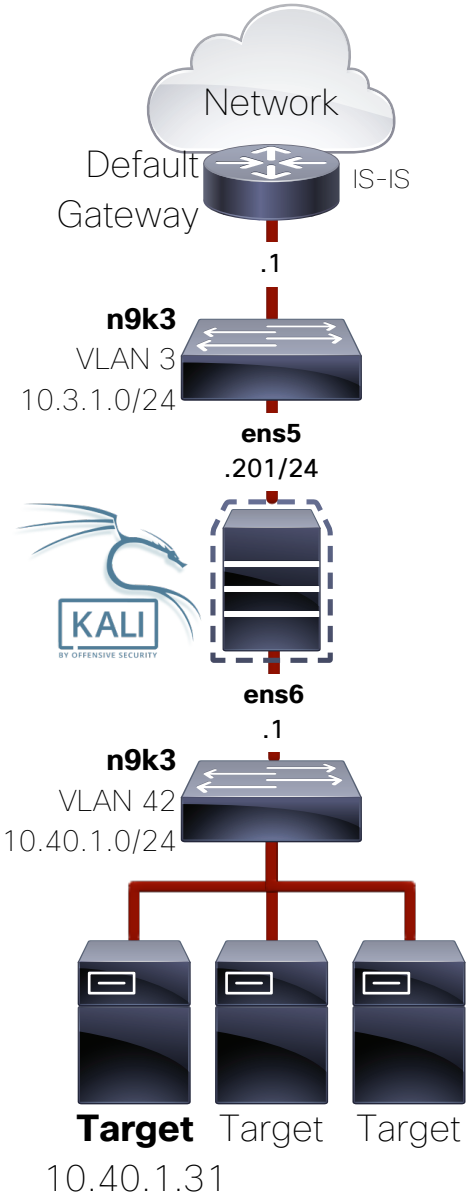
```
knv5801-n9k3# show interface status | i mitm
Eth1/17       mitm knv5795 enp9s connected 40         full    10G     SFP-H10GB-C
U1M
Eth1/19       mitm knv5499 enp9s connected 40         full    10G     SFP-H10GB-C
U1M
Eth1/21       mitm knv5046 enp9s connected 40         full    10G     SFP-H10GB-C
U1M
Eth1/51       mitm knv5570 68:05 connected 40         full    40G     QSFP-H40G-A
OC1M
Eth1/52       mitm knv5570 68:05 connected 3          full    40G     QSFP-H40G-A
OC1M
knv5801-n9k3#
```



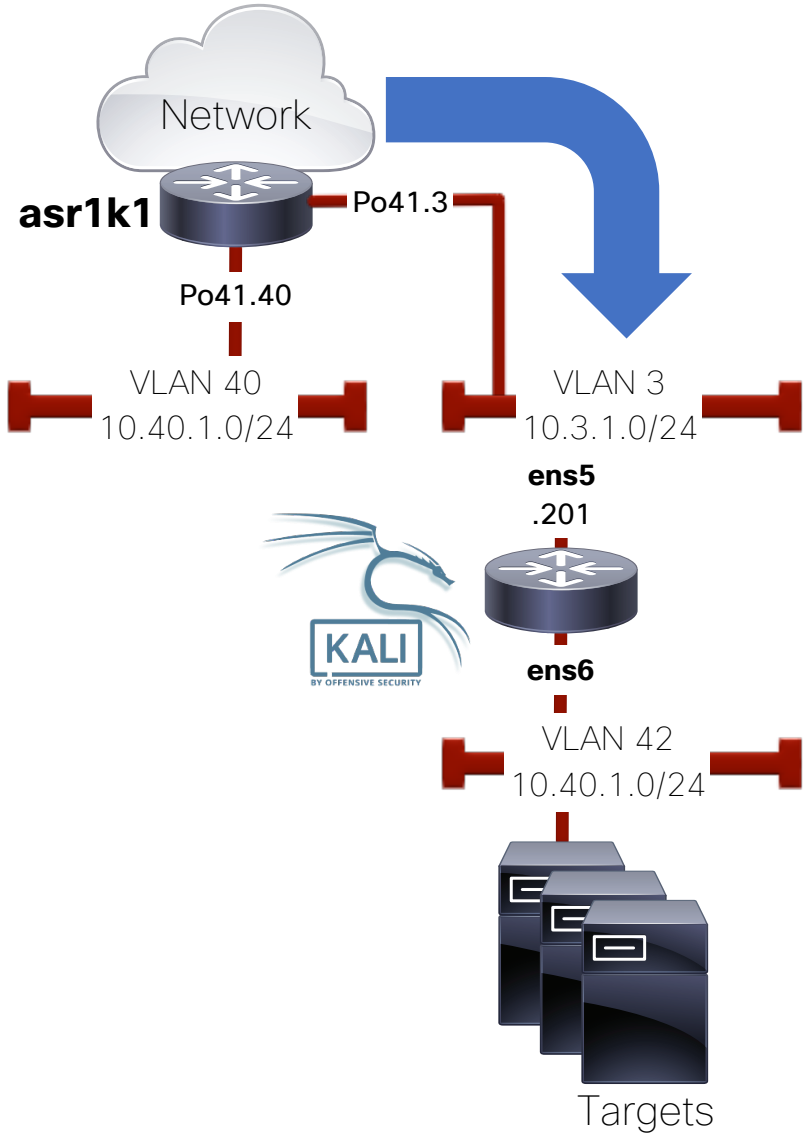


Layer 3 Man in the Middle: Replace Victim's Default Gateway

```
root@cmm-40g:~# ip -br -c addr
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens5             UP
ens6             UP
root@cmm-40g:~#
```



State of the Network

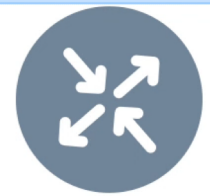


Layer 3 Man in the Middle: Override Routing by Subverting IGPs:

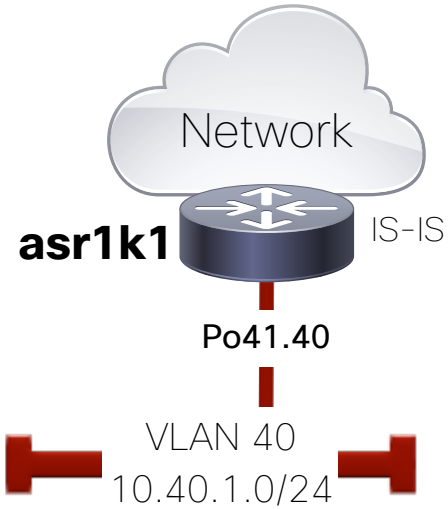
*Introduction to **routem***

https://developer.cisco.com/modeling-labs/

```
knv5569-asr1k1#
```



Router
Console

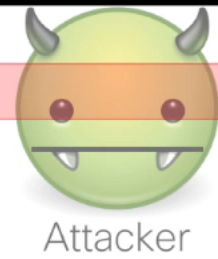


```
root@cmm-40g:~#
```

Attacker

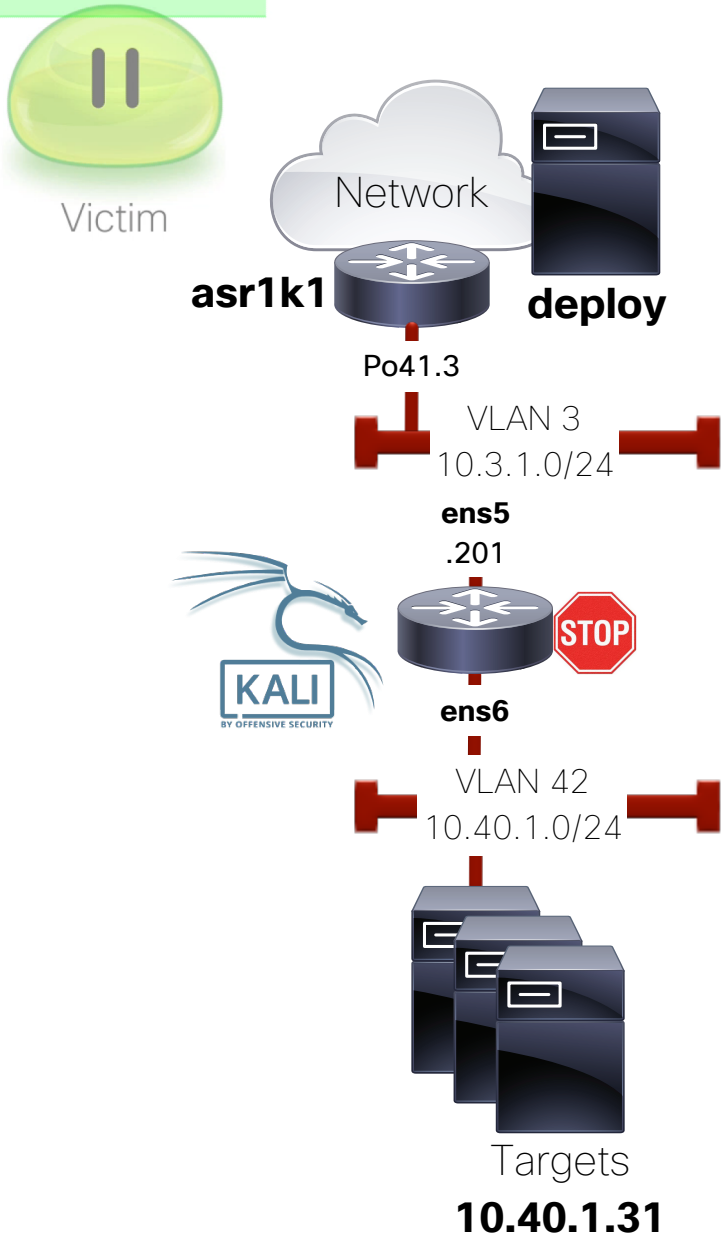
Layer 3 Man in the Middle: The Most-specific Prefix Wins

```
 GNU nano 2.9.8                    isis.cfg

router isis
 net 49.0003.0100.0300.1202.00
 interface ens5
 mac-address 6805.ca32.1551
 neighbor 10.3.1.41 update-source 10.3.1.201
 is-type level-1
 metric-style wide
 ip-internal 1 10.40.2.1/24 10
 ip-internal-metric 1 100




                  [ Read 10 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell ^_ Go To Line
```



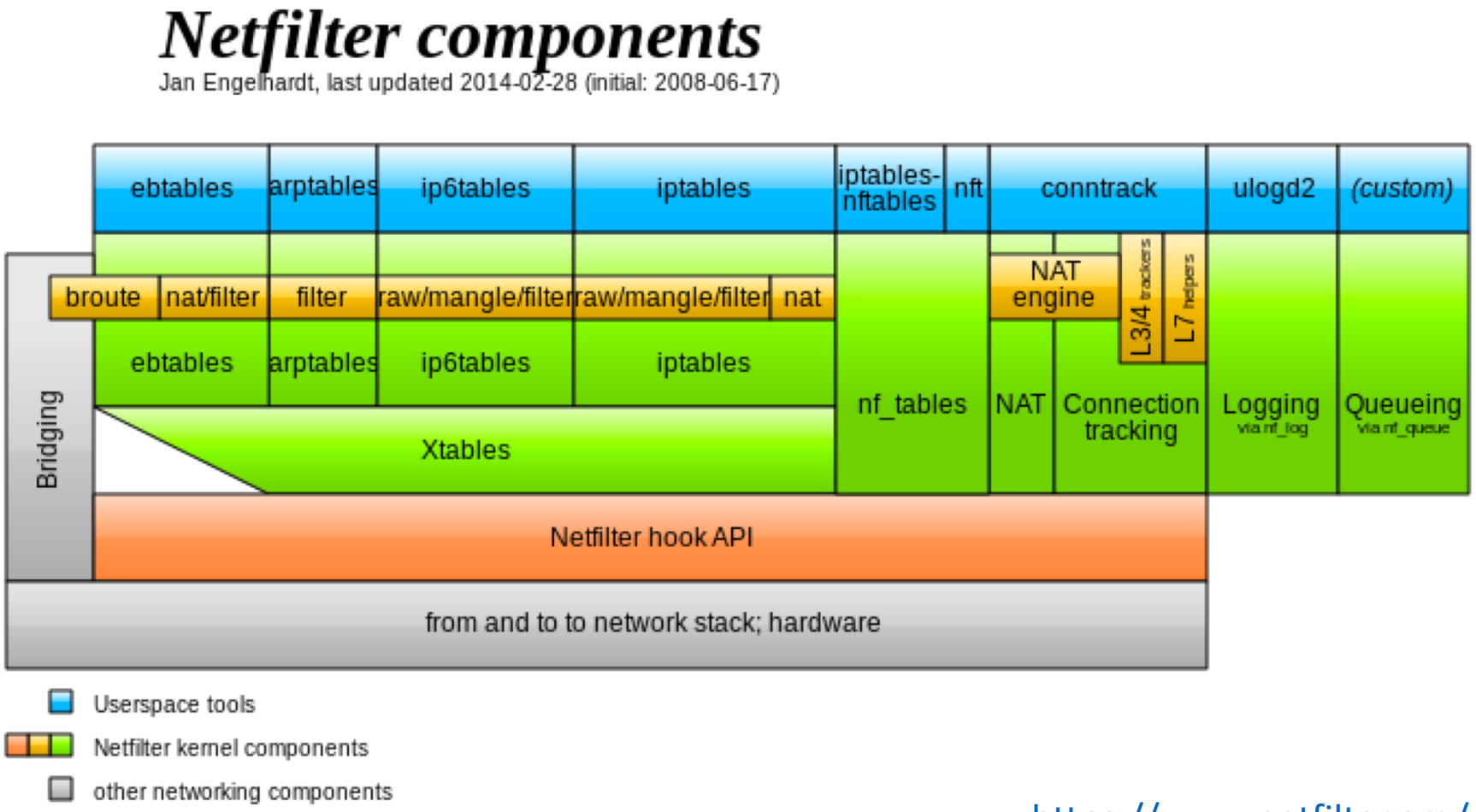
Layer 3 Man in the Middle: Don’t Forget to Enable IP Forwarding!

```
cmm@knv3762-deploy:~$
```



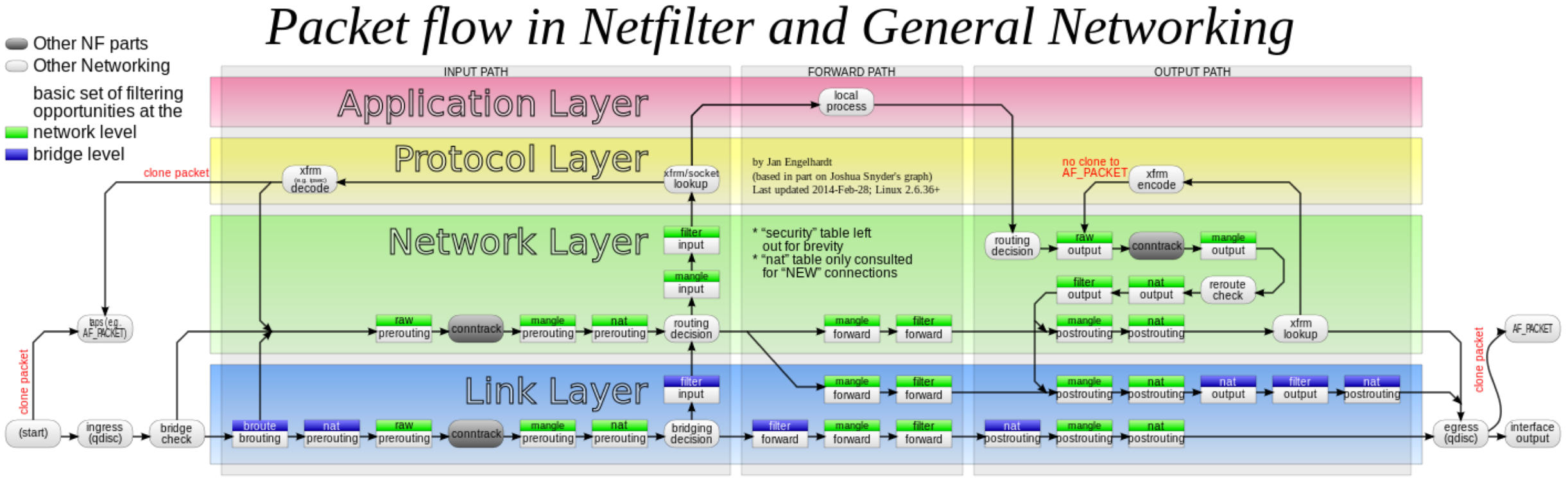
Layer 3 Man in the Middle: Using Linux's Stateful Firewall (conntrack)

Netfilters/iptables



https://www.netfilter.org/projects/iptables/index.html
https://en.wikipedia.org/wiki/Netfilter

Netfilters/iptables



https://www.netfilter.org/projects/iptables/index.html
https://en.wikipedia.org/wiki/Netfilter

conntrack

- Conntrack is Linux's stateful packet tracking firewall
- Conntrack is intimately connected with NAT
- Connection state can be categorized into the following:
  - **NEW** – new connections
  - **ESTABLISHED** – connections that have completed a TCP three-way handshake, UDP flows that have seen a reply, or ICMP echo requests that have seen an echo reply
  - **RELATED** – connections related to an already permitted connection, such as the data connection of FTP or ICMP error messages relating to a flow such as unreachables
  - **INVALID** – connections that do not adhere to a strict TCP state
  - **UNTRACKED** – connections that are explicitly targeted by NOTRACK in raw table
- Once a packet flow is managed through conntrack and established, it will **bypass** other netfilters rules **bidirectionally**

http://conntrack-tools.netfilter.org/support.html

```
root@cmm-40g:~# i
```

Attacker

Layer 3 Man in the Middle: Introduction to the DNAT Target & DNS interception using dnsmasq

The DNAT target

- Unlike Cisco IOS, Linux doesn't have a concept of "inside" vs. "outside" interfaces with NAT. NAT can even be done while entering and leaving the same interface (as used with ARP cache poisoning)
- **DNAT is a key MiTM tool that can be used with TCP or UDP and can even be used when Layer 2 bridging**

```
iptables –t nat –A PREROUTING –s 10.40.1.31 –d 10.3.1.15 \
  –p tcp –m tcp --dport 80 \
  -j DNAT --to-destination 10.40.1.100:8880
```

http://linux-ip.net/html/nat-dnat.html

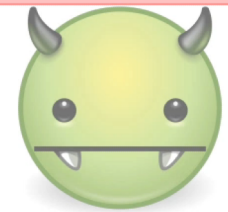
dnsmasq

- dnsmasq provides a nice lightweight DNS server that works great for performing simple MiTM
- If it doesn't have a local entry, it'll forward the request using `/etc/resolv.conf`

```
# dnsmasq -d
dnsmasq: started, version 2.79 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt
DBus i18n IDN DHCP DHCPv6 no-Lua TFTP conntrack
ipset auth DNSSEC loop-detect inotify
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 10.3.1.15#53
dnsmasq: read /etc/hosts - 4 addresses
```

http://www.thekelleys.org.uk/dnsmasq/doc.html

```
root@cmm-40g:~#
```

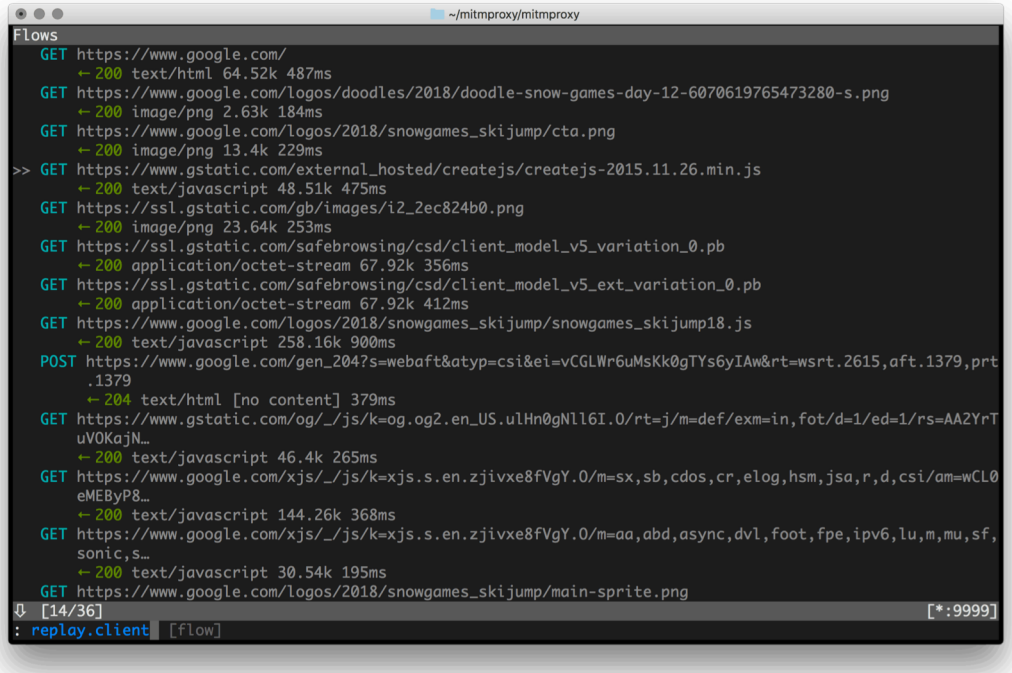


Attacker

Layer 3 Man in the Middle: HTTPS interception using mitmproxy

mitmproxy

- mitmproxy is an amazing project that provides an interactive command-line oriented proxy for HTTPS
- Decodes a variety of payloads including WebSockets and protobufs



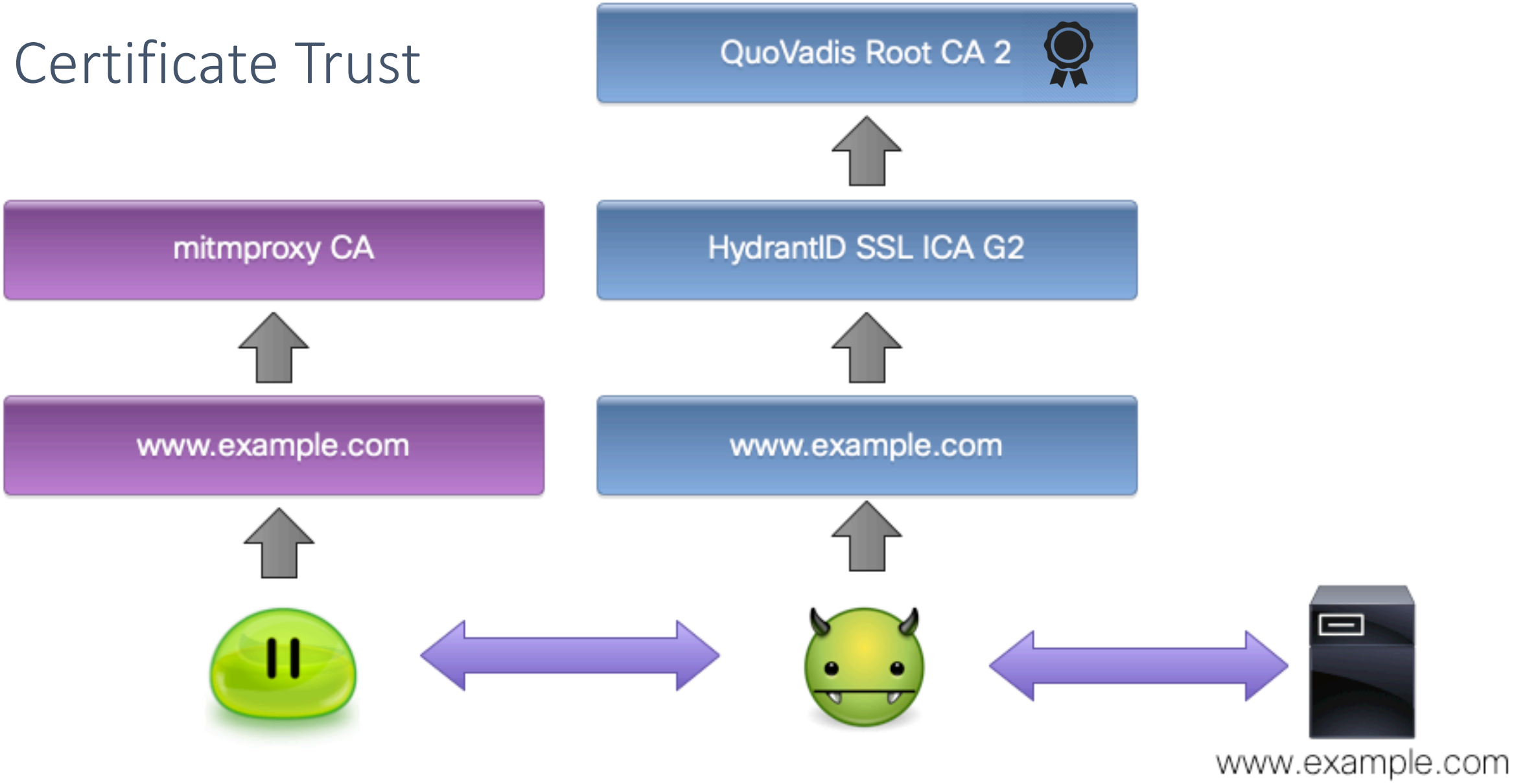
https://mitmproxy.org/

```
root@cmm-40g:~#
```

Attacker

Layer 3 Man in the Middle: Adding Certificate to Host Truststore

Certificate Trust



```
root@maglev-master-1:~$
```

Victim

Layer 3 Man in the Middle: SSH Interception using 'Lil SSHniffer

Decept Proxy and 'lil SSHniffer

- Decept Proxy provides an simple Python proxy with no dependencies
- Supports TLS, IPv6, Unix Domain Sockets and packet captures
- Comes with ‘lil SSHniffer (needs Paramiko) which can be used to MiTM SSH and capture passwords

```
<(^.^)>Inbound connection from 192.0.0.33:62924
Posing as: 10.20.0.11:22
__________________
[^.^] Good login | admin:au1piThoi5vuosie
_____________________________
<<<<<<<<<<<<

Router>
```

https://github.com/Cisco-Talos/Decept

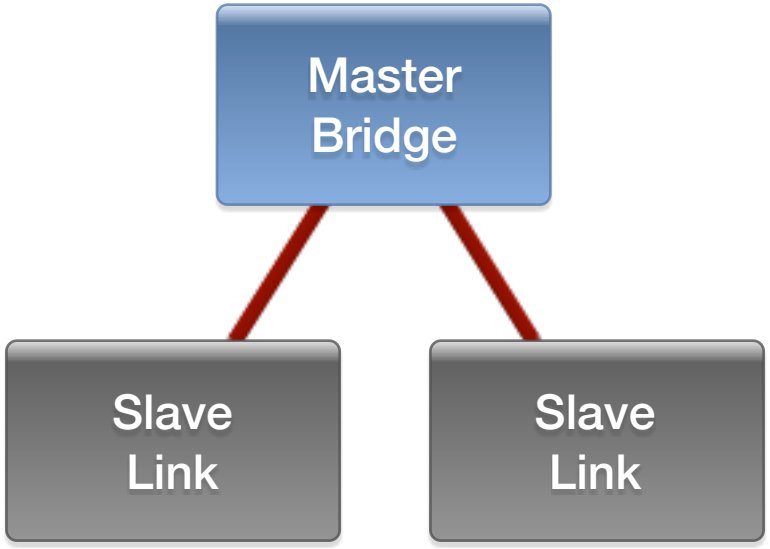
root@cmm-40g:~#

Attacker

Layer 2
Using Linux as a Bridge

The Linux master-slave interface concept

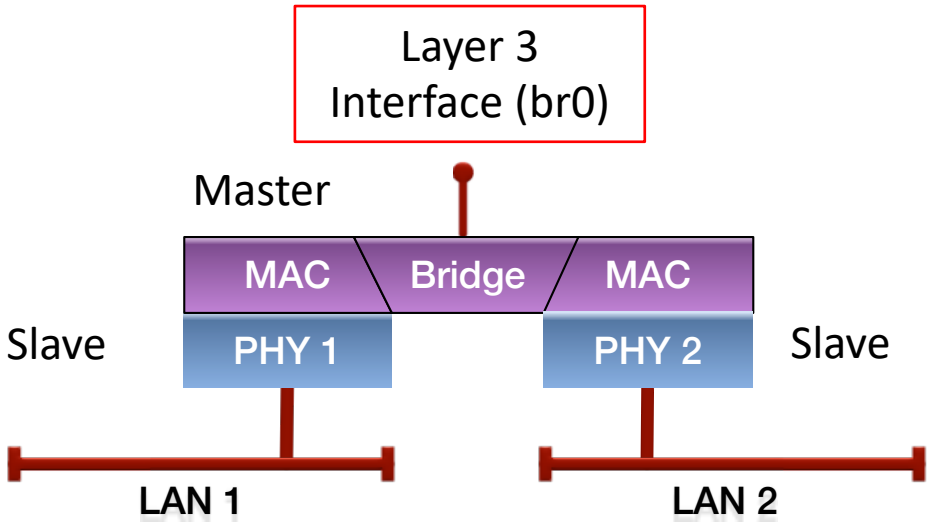
- Linux links can be “stacked” hierarchically into parent/child relationships
- This is used in multiple scenarios:
  - Bonding multiple physical links into a single virtual link (802.3ad)
  - VRFs to place interfaces in a separate routing table in a netns
  - **Bridging**



https://www.linux.org/docs/man8/ip-link.html

Creating a bridge

- First create the bridge link
- Set the physical links use the bridge as a master
- This will create a layer 2 broadcast domain with a **stub layer 3** interface




```
ip link add br0 type bridge
ip link set br0 up
ip link set ens5 master br0
ip link set ens6 master br0
```

https://wiki.archlinux.org/index.php/Network_bridge

Netfilters with bridging

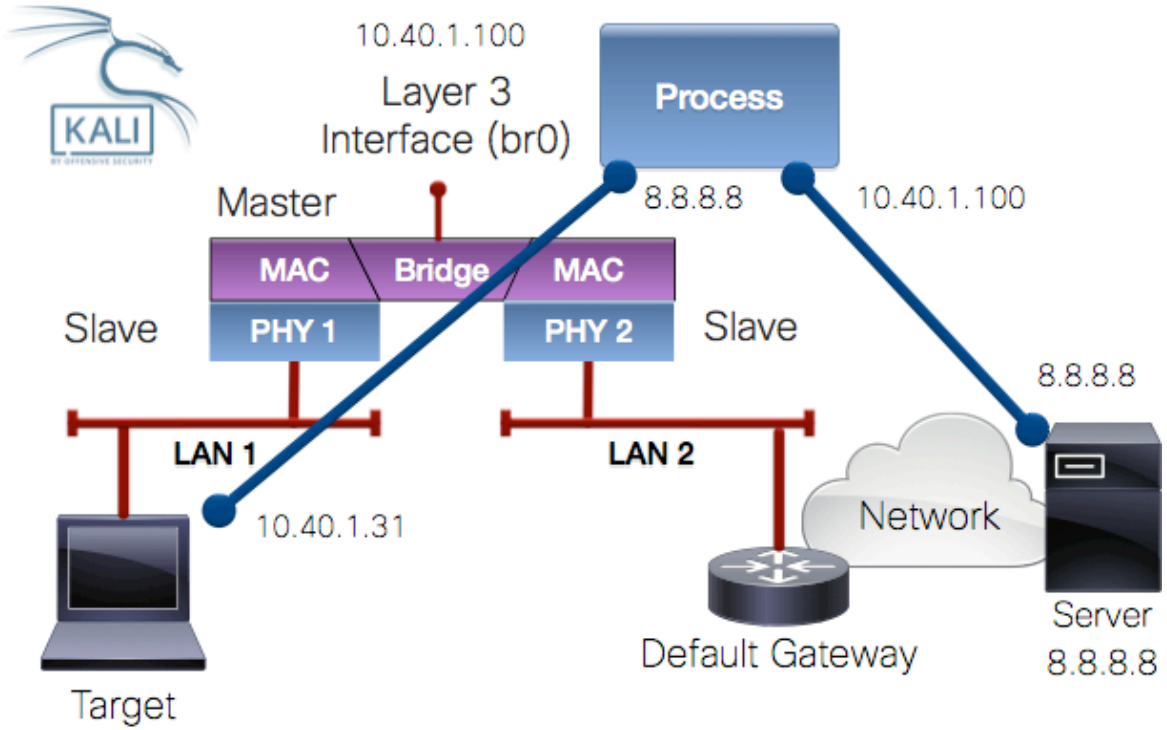
- Linux bridges can also support L3 features such as the **DNAT target**
- Flows will virtually terminate in the bridge and a new connection can be made by the mitm through the **stub L3** interface (so make sure the subnet matches)
- You'll need to insert the `br_netfilter` kernel module to support this

```
root@cmm-40g:~# modprobe br_netfilter
root@cmm-40g:/sys/class/net/br0/bridge# ls nf_call*
nf_call_arptables  nf_call_ip6tables  nf_call_iptables
```

http://thornton.info/tools/mibr.htm

Netfilters with bridging and DNAT

```
iptables –t nat \
   –A PREROUTING –i br0 \
   –s 10.40.1.31 –d 8.8.8.8 \
   –p udp –m udp --dport 53 \
   -j DNAT --to-destination 10.40.1.100:5053
```



http://thornton.info/tools/mibr.htm

What we learned…

- Set up your lab so packets can't "escape" without you wanting them to
- How to isolate the MiTM machine using network namespaces so local processes don't interfere and create a clean iptables/routing table
- Man-in-the-Middle techniques using Linux as a Router: DNAT is key
  - Intercepting DNS with dnsmasq
  - Intercepting HTTPS with mitmproxy, how to add certs to target truststore
  - Intercepting SSH with 'lil SSHniffer
- Using Linux as a MiTM bridge with the `br_netfilter` kernel module and `nf_call_iptables=1`
  - Use this when you can't/don't want to manipulate interior gateway protocols

Thank you! Questions?

*Please Join the **#red-team-talks** Channel*

https://redteamvillage.io/discord

Slides:

https://github.com/chris-mccoy/mayhem-2020