

ARP address resolution protocol

protocol layer2. 转换mac地址到ip地址。

为什么会有arp协议？

因为switch交换机，一个网络中可能有多个设备，共用一个外部ip地址，所以就需要有一个中转站收到包裹后，来进一步地分发信息。通过辨识mac地址，来唯一验证要传递包裹的目标。

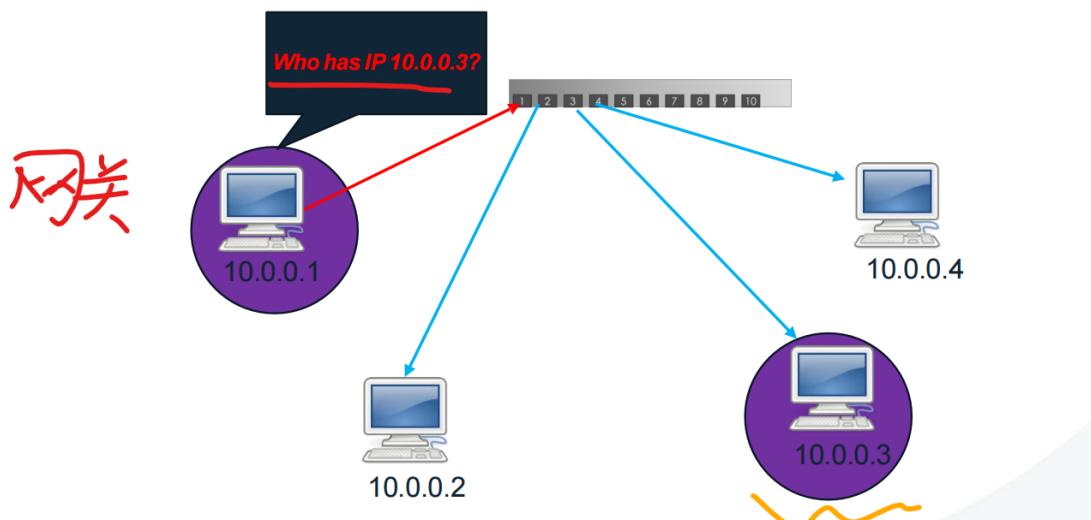
配对后，arp cache会保存ip地址与mac地址的配对信息。

arp request

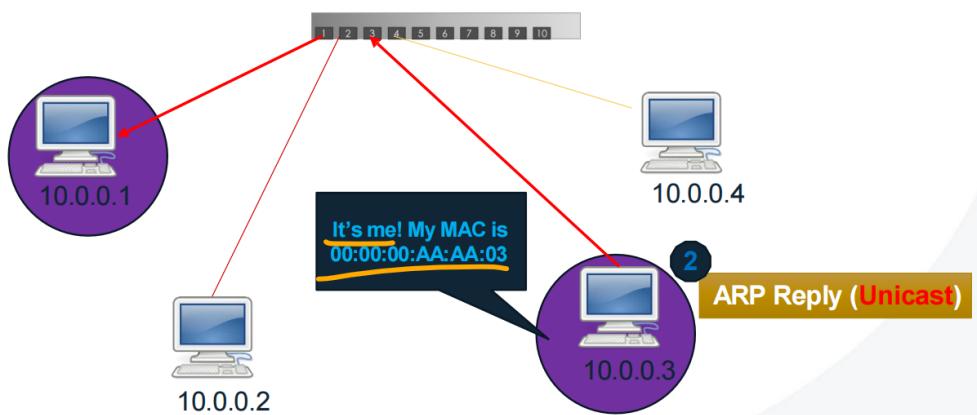
arp reply

IP Address → MAC address ARP (Address Resolution Protocol)

1 ARP Request (Broadcast – sent to all hosts in broadcast domain)



找到后，网关会保存这个ip地址与其mac地址的对应，这样下次收到它的包裹，就直接给对应mac地址的主机就好。



arp spoofing (也叫arp posioning)

其实也是一种flow control的攻击手段。改变了原有的通信流。

<https://www.youtube.com/watch?v=cVTUeEoJgEg> —— 使用ettercap工具，实现ARP spoofing攻击。

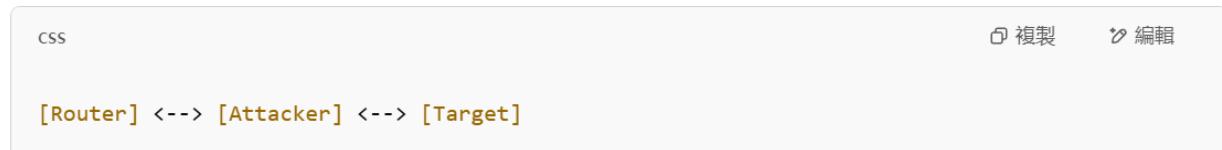
用人话来讲就是，attacker通过arp spoofing的方式，自己充当路由器或网关，于是乎目标target会将所有收发的信息传给attacker

对，就是这么回事！攻击者假装自己是路由器或网关，让目标设备（Target）把所有网络数据都发给他，而不是发给真正的网关。这样，攻击者就可以 **窃听、篡改**，甚至 **阻断** 目标设备的网络通信。

就像你去餐厅点餐，结果你的订单被别人截胡，餐厅以为那个人是你，而你啥都没吃到。 😊

所以拓扑结构是 路由器 <---> attacker <---> target

没错，就是这个结构！



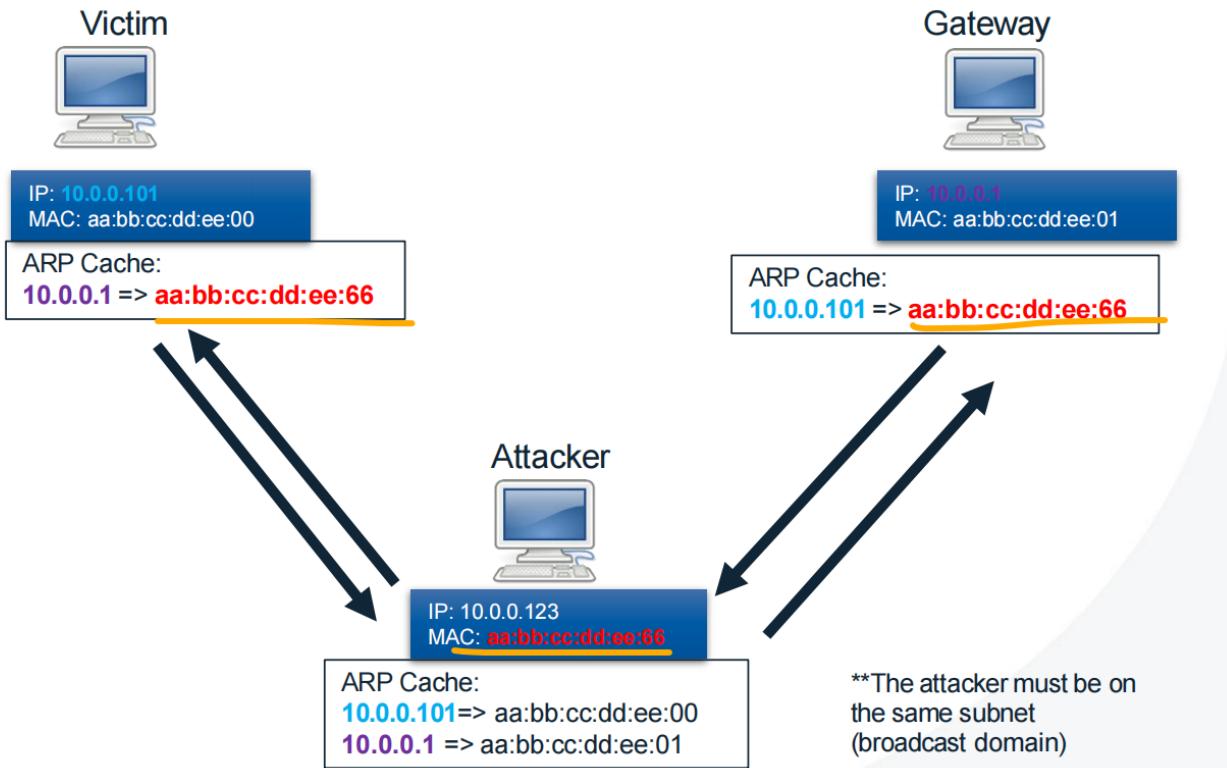
攻击者伪装成路由器骗目标，又伪装成目标骗路由器，这样 **所有数据都会经过攻击者**，相当于搭了个“中转站”，让他能 **偷看、修改** 或 **直接丢弃** 数据。

如果攻击者愿意，他可以：

- **偷偷监听**（嗅探登录信息、聊天记录等） 
- **篡改数据**（比如改你访问的网站内容） 
- **断网**（直接不转发，目标设备就上不了网） 

详细图

MITM using ARP Cache Poisoning



attacker对victim说，我就是gateway，然后对gateway说，我就是victim。这样两边都一起骗。

DHCP dynamic host configuration protocol

DHCP的作用，告诉一个新加入网络的设备，它的ip是什么，dns 服务器的ip地址和网关的地址。

所以DHCP协议，会告诉一个新加入网络的设备，它的ip地址是什么，DNS的地址和这个网络的网关的地址

没错！DHCP 服务器的任务就是 告诉新设备：

- 1 你的 IP 地址 (例如 192.168.1.100)
- 2 你的子网掩码 (例如 255.255.255.0)
- 3 你的默认网关 (即路由器的 IP, 例如 192.168.1.1)
- 4 你的 DNS 服务器 (例如 8.8.8.8 或 192.168.1.1)

这样，新设备才能正确地连接到网络并访问互联网！

DHCP: Initial Network Configuration

加入了网络的必要条件

- To connect to a network, a user needs:
 - An IP address so that other people can contact the user
 - The IP address of the DNS server
 - The IP address of the router (gateway) so that the user can contact machines outside of the LAN
- The first time a user connects, they don't have this information yet
 - The user also doesn't know who to ask for this information
- DHCP gives the user a configuration when they first join the network

Why use DHCP?

Every device on a TCP/IP-based network must have a unique unicast IP address to access the network and its resources. Without DHCP, IP addresses for new computers or computers that are moved from one subnet to another must be configured manually; IP addresses for computers that are removed from the network must be manually reclaimed.

With DHCP, this entire process is automated and managed centrally. The DHCP server maintains a pool of IP addresses and leases an address to any DHCP-enabled client when it starts up on the network. Because the IP addresses are dynamic (leased) rather than static (permanently assigned), addresses no longer in use are automatically returned to the pool for reallocation.

DHCP handshake

首先是在网络中，**主机广播自己需要配置**，接着网络中的dhcp server就会回应给予**DHCP OFFER**，主机会广播一个**client request**，它选了哪一个DHCP OFFER (因为有可能一个网络中有多个DHCP SERVER)，接着对应的DHCP server就会发送**DHCP ACK**. 完成配置。

Steps of the DHCP Handshake

1. Client Discover: The client broadcasts a request for a configuration
2. DHCP Offer: Any DHCP server can respond with a configuration offer
 - Usually only one DHCP server responds
 - The offer includes an IP address for the client, the DNS server's IP address, and the (gateway) router's IP address
 - The offer also has an expiration time (how long the user can use this configuration)
3. Client Request: The client broadcasts which configuration it has chosen
 - If multiple DHCP servers made offers, the ones that were not chosen discard their offer
 - The chosen DHCP server gives the offer to the client
4. DHCP Acknowledgement: The chosen server confirms that its configuration has been given to the client

DHCP Attacks and Defence

- The attacks and defence on ARP and DHCP are very similar
 - Spoofing: The attacker claims to have an answer
 - Race condition: The requester accepts the first response. As long as the attacker's response arrives first, it is accepted
- Main vulnerabilities
 - Broadcast protocols: Requests are sent to everyone on the LAN, so the attacker can see every request
 - No trust anchor: There is no way to verify that responses are legitimate

DHCP实战

因为virtual box自带dhcp server，所以我们直接请求dhcp服务即可。

在linux vm中， dhclient -r 去去除当前ip地址，然后sudo dhclient enp0s8来请求一个新的ip地址。

```
caseylao@caseylao-VirtualBox:~/Desktop$ sudo dhclient -r
Killed old client process
caseylao@caseylao-VirtualBox:~/Desktop$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
  inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 08:00:27:7d:34:2d brd ff:ff:ff:ff:ff:ff
  inet6 fe80::e3d9:98ef:5d47:e10a/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
3: enp0s17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 08:00:27:41:24:1d brd ff:ff:ff:ff:ff:ff
  inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s17
    valid_lft 86183sec preferred_lft 86183sec
  inet6 fd00::76bf:b5ba:4c58:6485/64 scope global temporary dynamic
    valid_lft 86307sec preferred_lft 14307sec
  inet6 fd00::911f:b015:2ce6:80e9/64 scope global dynamic mngtmpaddr noprefixroute
    valid_lft 86307sec preferred_lft 14307sec
  inet6 fe80::3f64:40a9:83f4:141a/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
caseylao@caseylao-VirtualBox:~/Desktop$ sudo dhclient enp0s8
caseylao@caseylao-VirtualBox:~/Desktop$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
  inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 08:00:27:7d:34:2d brd ff:ff:ff:ff:ff:ff
  inet 192.168.116.5/24 brd 192.168.116.255 scope global dynamic enp0s8
    valid_lft 575sec preferred_lft 575sec
  inet6 fe80::e3d9:98ef:5d47:e10a/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
3: enp0s17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 08:00:27:41:24:1d brd ff:ff:ff:ff:ff:ff
  inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s17
    valid_lft 86141sec preferred_lft 86141sec
  inet6 fd00::76bf:b5ba:4c58:6485/64 scope global temporary dynamic
    valid_lft 86265sec preferred_lft 14265sec
```

27	66.083518017	0.0.0.0	255.255.255.255	DHCP	342 DHCP Discover	- Transaction ID 0x37ad1e54
28	66.084638142	192.168.116.2	255.255.255.255	DHCP	590 DHCP Offer	- Transaction ID 0x37ad1e54
29	66.084638523	0.0.0.0	255.255.255.255	DHCP	342 DHCP Request	- Transaction ID 0x37ad1e54
30	66.087710847	192.168.116.2	255.255.255.255	DHCP	590 DHCP ACK	- Transaction ID 0x37ad1e54

-DHCp

Port 67 is the DHCP service port.

In the DHCP OFFER, your (client) IP address : 192.168.116.5, is the IP address allocated by the DHCP server.

27	66.083518017	0.0.0.0	255.255.255.255	DHCP	342 DHCP Discover	- Transaction ID 0x37ad1e54
28	66.084638142	192.168.116.2	255.255.255.255	DHCP	590 DHCP Offer	- Transaction ID 0x37ad1e54
29	66.084638523	0.0.0.0	255.255.255.255	DHCP	342 DHCP Request	- Transaction ID 0x37ad1e54
30	66.087710847	192.168.116.2	255.255.255.255	DHCP	590 DHCP ACK	- Transaction ID 0x37ad1e54

Your (client) IP address: 192.168.116.5

28	66.084638142	192.168.116.2	255.255.255.255	DHCP	590	DHCP Offer	- Transaction ID 0x3/ad1e54
29	66.084638523	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request	- Transaction ID 0x37ad1e54
30	66.087710847	192.168.116.2	255.255.255.255	DHCP	590	DHCP ACK	- Transaction ID 0x37ad1e54

```

Frame 29: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_7d:34:2d (08:00:27:7d:34:2d), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x37ad1e54
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: PCSSystemtec_7d:34:2d (08:00:27:7d:34:2d)
    Client hardware address padding: 000000000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    Option: (53) DHCP Message Type (Request)
    Option: (54) DHCP Server Identifier (192.168.116.2)
    Option: (50) Requested IP Address (192.168.116.5)
        Length: 4
        Requested IP Address: 192.168.116.5
    Option: (12) Host Name

```

requested ip address, 就是主机确认了自己要这个ip地址。最后就是server 回复ack .

DHCP SPOOFING

<https://www.twingate.com/blog/glossary/dhcp%20spoofing>

机制：攻击者设置自己的DHCP server，与client处于同一网络中。当client send a DHCP discover时，恶意DHCP server会截取这个message并回应自己的DHCP offer. 内容一般是将自己设置为default gateway, 以及配置一个想要的DNS ip地址。

DHCP Starvation attack

攻击者，还能发送巨量的DHCP request来消耗掉这个网络中，合法的DHCP server的ip pool，让它没有办法给予合法用户以ip地址使用，这就是所谓的**DHCP Starvation attack**.

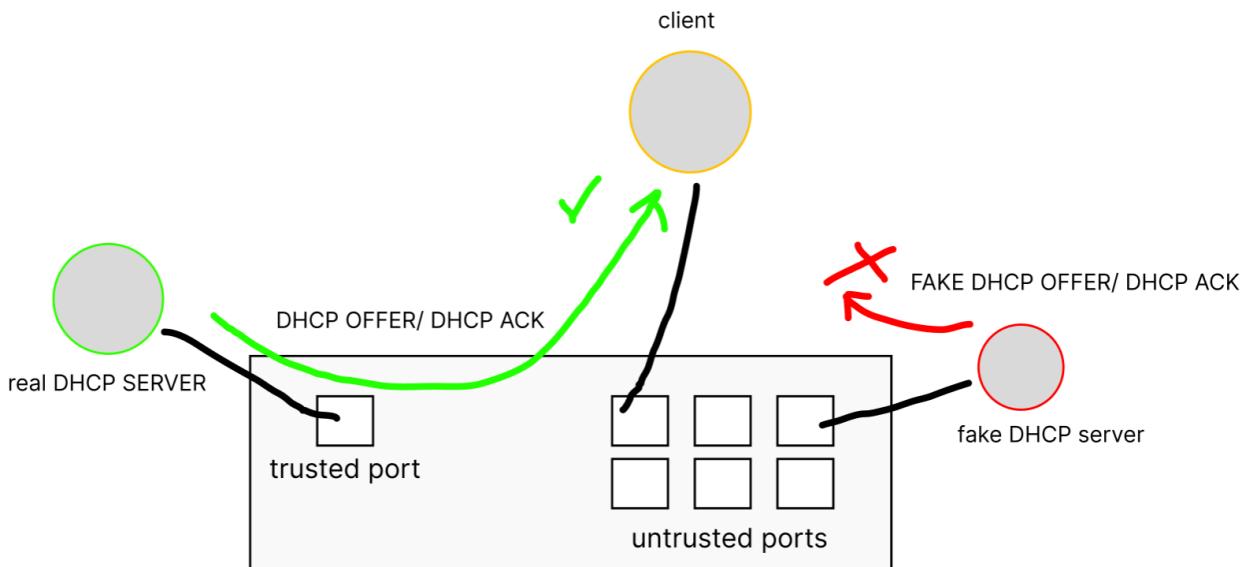
DHCP MITM

Another example is the **Man-In-The-Middle (MITM) Attack**. Here, the attacker combines DHCP Starvation and Rogue DHCP attacks to intercept and manipulate network traffic. By setting themselves as the DNS server and default gateway, the attacker can redirect DNS requests, leading to credential harvesting and forced authentication scenarios. These examples highlight the diverse tactics attackers use to exploit DHCP vulnerabilities.

DHCP Snooping

用于防止网络中，有主机充当Fake DHCP Server.

真DHCP Server连接trusted port, 通常其他主机都连接至untrusted port。只有从trusted port传出来的DHCP Responses(DHCP OFFER, DHCP ACK)才可信。否则通通discard掉。

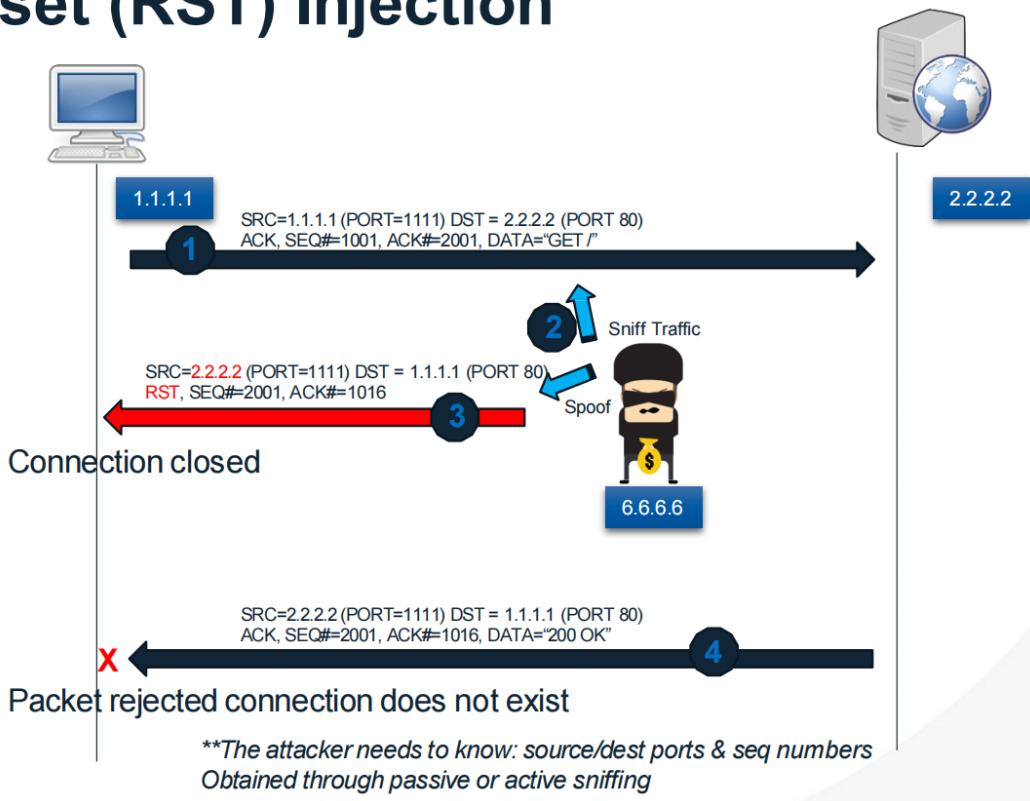


TCP reset injection

⊕ 协议的标志位flags

reset，通常意味着强制终止当前的连接或状态

TCP Reset (RST) Injection

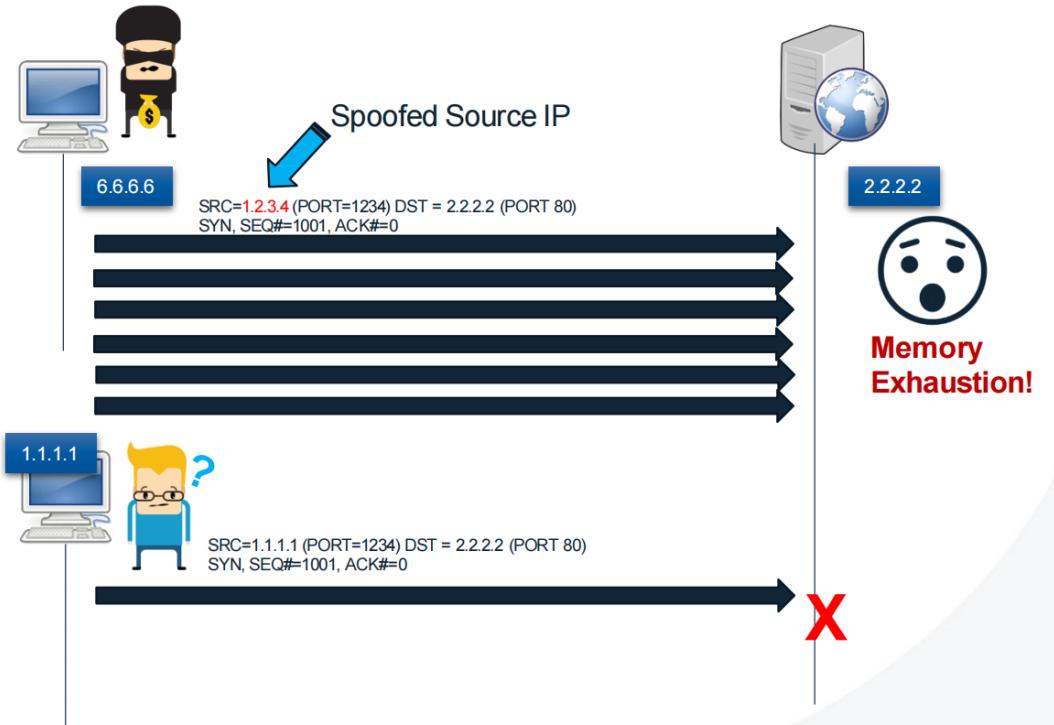


DOS attack

⊕ ping

本质上，是发送过量的请求，使得目标服务器无法承受，以至于不回应合法用户的请求。

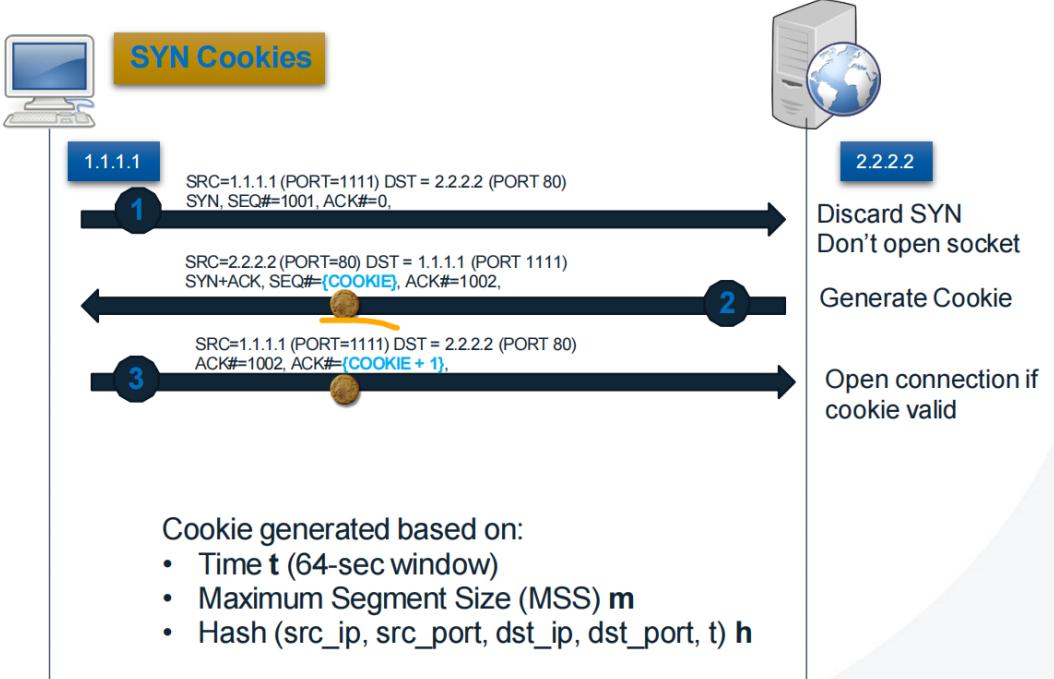
SYN Flooding DoS



应对syn flooding dos的方法

服务器先不open syn连接，而是返回一个cookie，如果对方回复以valid的cookie，则正式开展连接。

SYN Flood Countermeasure



smurf attack

ping

Smurf 攻击是一种利用 ICMP 回显请求 (ping) + IP 地址欺骗 + 网络广播地址 来对目标发起洪水攻击的方式。

attacker, 构造ICMP echo request, 写入victim的ip地址, 将这个request发送到广播地址, 告诉所有主机, 请向该victim发送ICMP echo reply

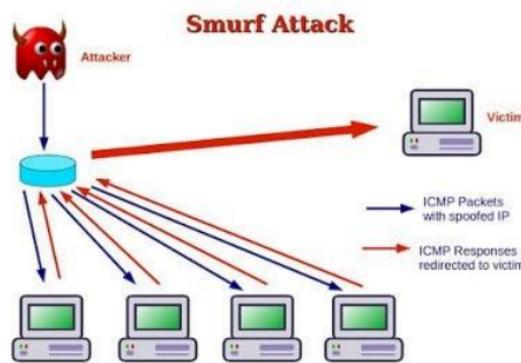
DDoS attacks

Smurf

**ICMP Echo (PING) request using forged IP of the Victim
Sent to broadcast address**

Amplification factor ~ number of participating PCs on the same broadcast network

Can be prevented by the router configuration (disallow broadcast PING)



<https://computersecuritypgp.blogspot.jp/2015/09/what-is-smurf-attack.html>

Amplified denial of service

简单来说, 通过使用某些服务器服务, 伪造victim的ip, 替victim去request该服务, 而服务器则将response (通常比request package大) 都发给victim。如果attacker做了大量request, 那么victim将接受大量的response, 这就是放大DOS.

* 放大攻击的工作原理:

1. 攻击者伪造一个请求, 把源IP地址伪装成受害者的IP地址。
2. 攻击者向某些“可被利用”的服务发送体积很小的请求。
3. 这些服务返回一个远远大于请求的数据包 (这个“放大倍数”可以是几十倍、上百倍甚至上千倍)。
4. 因为请求中的IP是伪造的, 所有响应都会被发回给受害者, 造成流量洪水。

这些是常用的协议, 因为是UDP, 所以可以伪造ip地址。

NTP network time protocol.

* 常见的“放大器”服务包括：

放大器类型	协议	放大倍数 (近似)
DNS	UDP	高达 50 倍以上
NTP	UDP	高达 500 倍
Memcached	UDP	可高达 50000 倍
SSDP	UDP	约 30 倍
CLDAP	UDP	超过 50 倍

这些服务通常运行在UDP协议之上，**UDP的“无连接”特性**使得IP伪造成为可能。

如何防御Amplified denial of service？

- 1运营商检验package是否存在伪造ip的package，使其无法进入网络。
- 2对提供服务的server，自己做好封锁相应的接口或服务，避免被attacker当作放大器。
- 3使用CDN或DDOS抗服务，增加自己接受大量请求的能力
- 4限制单个ip地址短时间内的多个请求。

✓ 1. 防止 IP 被“假冒”——启用源地址验证

- 技术名：BCP 38 / uRPF (Unicast Reverse Path Forwarding)
- 谁做？ISP/网络运营商。
- 它能防止“伪造IP地址”的包离开网络，阻止攻击者冒用受害者IP发请求。
- 缺点：不是你作为受害者自己能控制的，需要整个网络链路中各级运营商配合。

如果你自己运营有DNS、NTP、Memcached等服务，以下措施非常关键：

服务	防御措施
DNS	禁止递归查询给外部用户，只对内网提供递归服务。配置防止 ANY 查询。
NTP	禁止“monlist”命令，使用 noquery 设置或升级版本。
Memcached	禁止UDP，或配置认证、仅允许内网访问。
SSDP/CLDAP	阻止向公网暴露服务，限制端口访问。

3. 作为“受害者”该怎么防御？

A. 启用抗DDoS服务（强烈推荐）

- 使用 **CDN + 云抗DDoS服务**（如 Cloudflare、AWS Shield、腾讯云 DDoS 高防等）。
- 这些服务可以：
 - 过滤流量；
 - 缓解突发洪水；
 - 使用 Anycast 分布式节点吸收攻击。

B. 部署 Rate Limiting / Connection Tracking

- **防火墙或负载均衡器上做速率限制**，防止单个IP或短时间内的请求过多。
- 使用工具如 `iptables` + `conntrack` 来控制并发连接和包速率。

C. IP黑名单 + 自动响应系统

- 部署基于日志分析的自动封锁策略（比如用 fail2ban 检测异常流量并暂时封禁IP）。
- 不过对 Amplified DoS 效果有限，因为攻击包是伪造的源IP，封锁那些IP并不能阻止真实流量。

DNS Domain name system

通过域名找ip地址。

DNS attack

host file posioning

比如输入的是adelaide.edu.au域名，但是去了一个malicious ip地址。

Hosts File Poisoning

If user computer is compromised, the hosts file can be poisoned (/etc/hosts in UNIX or c:\windows\system32\drivers\etc\hosts in Windows)

This can direct user to a malicious website

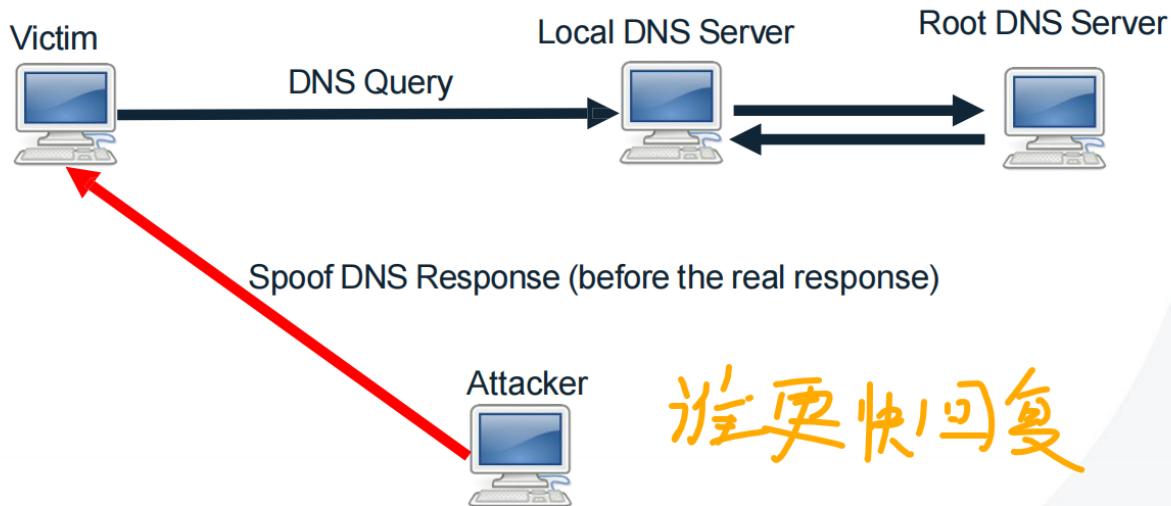
```
File Edit View Search Terminal Tabs Help
root@kali: ~/Desktop x      hosts + (/etc) - VIM x      root@kali: ~/
127.0.0.1      localhost
127.0.1.1      kali
43.241.200.112 myuni.adelaide.edu.au
# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

DNS Spoofing

只要攻击者能够监听到，并且回复地比DNS server还要快，那就容易做到spoofing.

DNS Poisoning (dnsspoof)

DNS Poisoning via Spoofed UDP Response

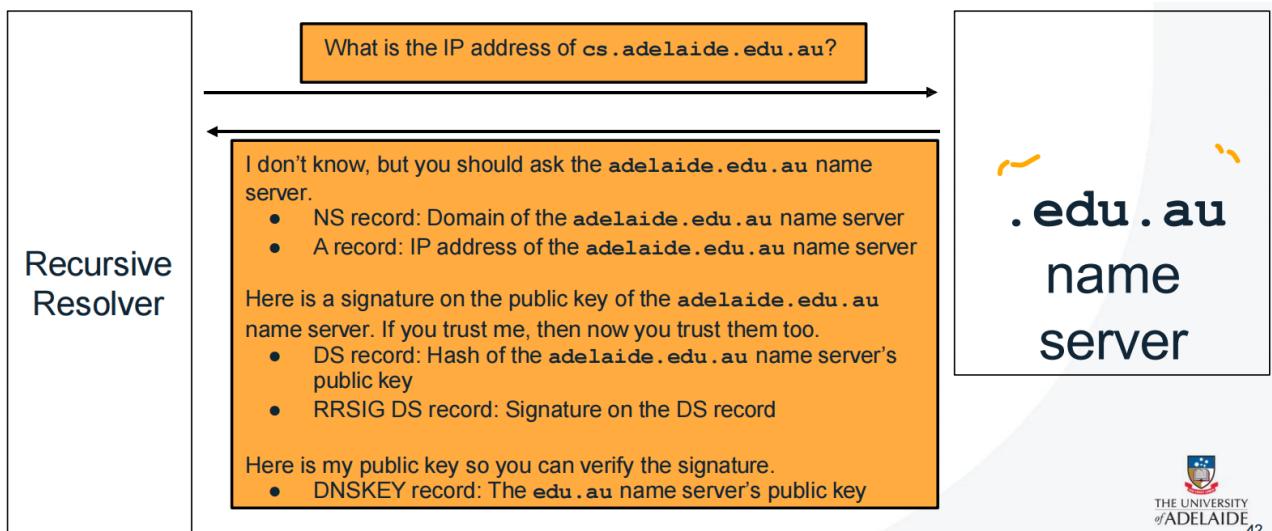
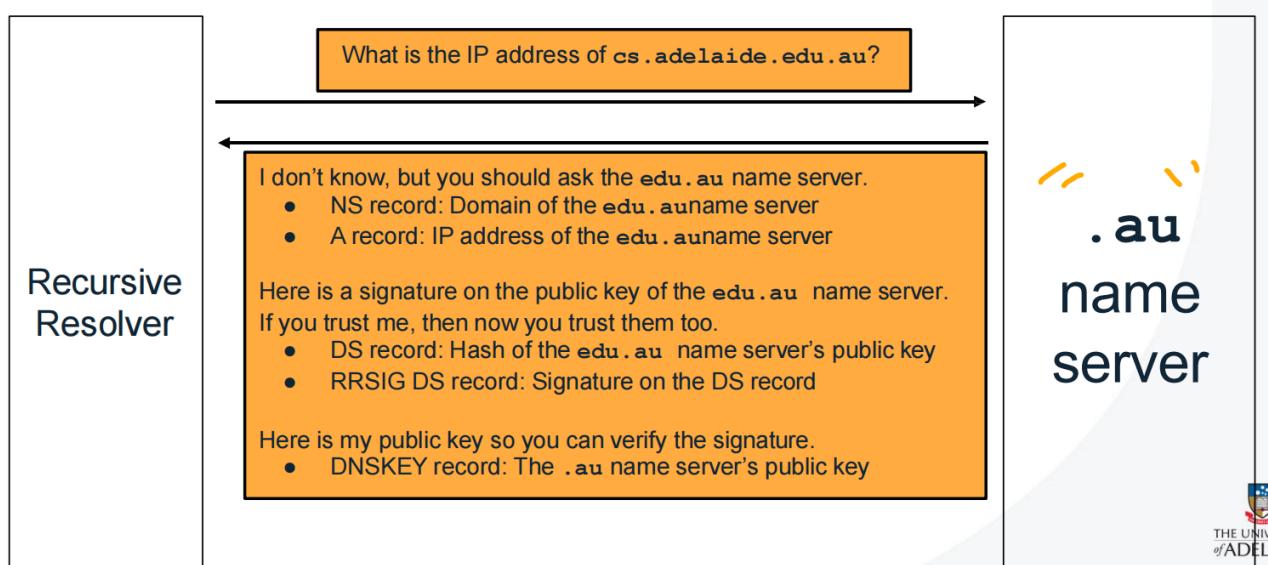
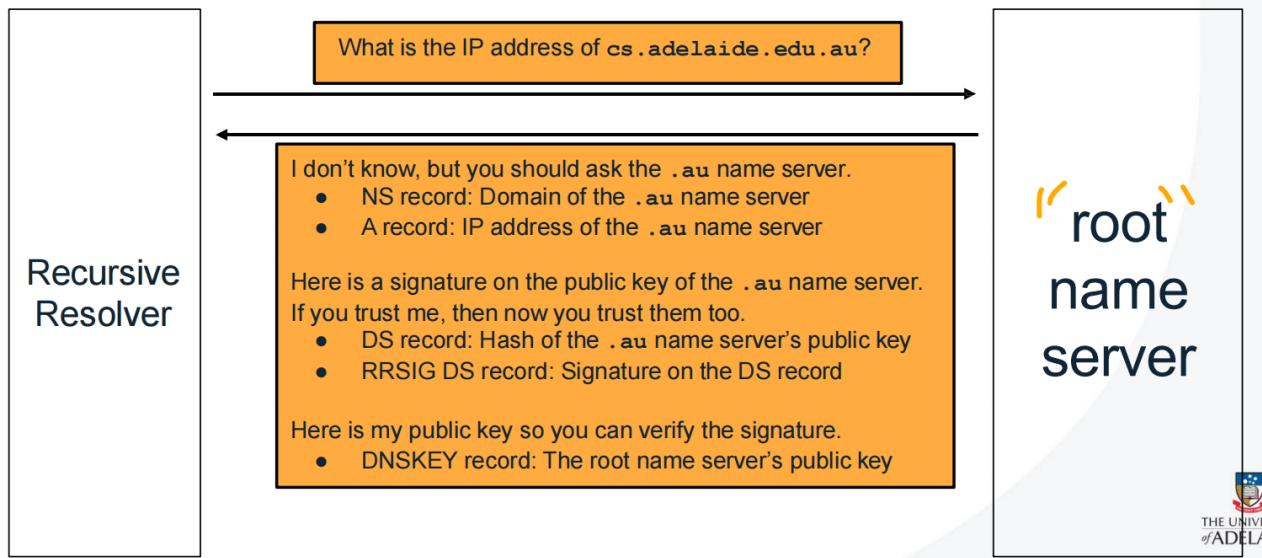


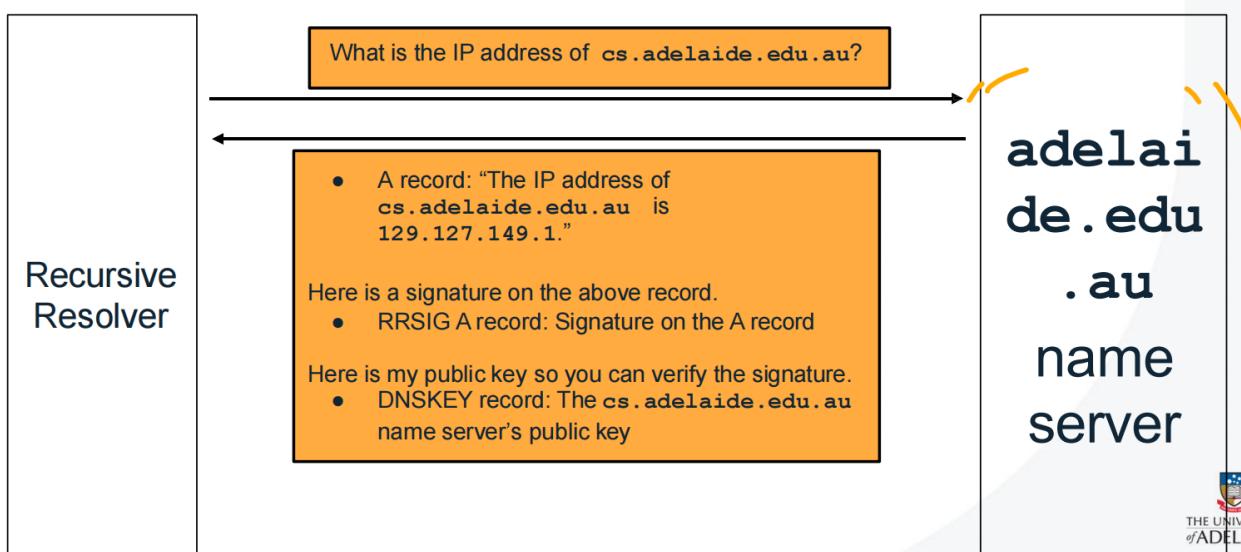
**The attacker must be on the same subnet and able to sniff all traffic (open Wi-Fi or via MITM using ARP cache poisoning)

这种攻击方式，可以用DNSSEC来防御。也就是多个认证，比如数字签名。

DNSSEC 递归lookup 一个域名

先从主domain, 比如.au, .cn, .com等大域名开始递归lookup.





也就是逐级验证，从root开始获得其DS (delegation signer) 记录以及public key，用作验证身份。

锁 它是怎么运作的？（逐级验证思路）

当你要查询某个域名的 IP (比如 `login.example.com`)，DNSSEC 会做类似下面的验证流程：

1. 从根域开始 Root

- 解析器先请求 `.` (根) 的签名公钥 (DS 记录 + RRSIG 签名)。
- 根域的公钥是“预装”的 (叫做 **trust anchor**, 信任锚)。

2. 验证顶级域 (TLD)

- 比如 `.com`，它的 DNS 响应会带有数字签名 (RRSIG)。
- 根域提供的 DS 记录可以用来验证 `.com` 提供的公钥 (DNSKEY)。

3. 验证主域名

- `.com` 会提供 `example.com` 的 DS 记录。
- 然后递归解析器会继续向 `example.com` 服务器请求 IP，同时验证它的 DNSKEY 和签名。

4. 最终验证目标子域名 (如 `login.example.com`)

- 如果 `login.example.com` 有 DNSSEC 支持，它也会带签名，继续验证。
- 如果中间某一级没有开启 DNSSEC，那么链条就中断，无法验证。

🔑 DS = Delegation Signer

这是 DNSSEC 中的一个关键记录类型，它的作用是用来建立**“从父域到子域”的信任链**。

✳ 简单解释：

- 当父域（比如 .com）要告诉解析器：
 “我信任 example.com 的公钥是这个指纹”
- 它就会在自己的区域文件中发布一个 **DS 记录**，这个记录里包含了：
 - 子域的 **公钥的哈希值（摘要）**
 - 用的哈希算法
 - 公钥的关键ID（Key Tag）
 - 签名算法

这样一来，解析器就可以用 .com 提供的 **DS 记录**，去验证 example.com 提供的公钥（DNSKEY 记录）是不是“真的”。

WIFI security

WIFI有多种连接协议。WEP, WPA, WPA2

WEP已被淘汰，因为可以用ARP去生成key stream，去破解。

WiFi Security - Introduction



1. Open
2. WEP (Wired Equivalent Privacy)
3. WPA (Wi-Fi Protected Access)
4. WPA2 (Wi-Fi Protected Access Version2)

WiFi: wireless local area networks. It is based on the IEEE 802.11 standard.

THE
of A

WPA2协议

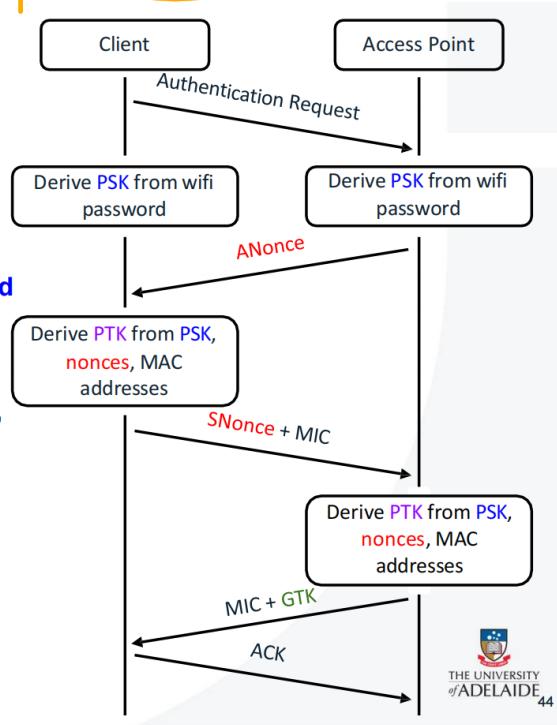
这个协议的目的是 知道密码的人，可以加入到该网络。所有信息，都会由Keys加密传输。只有知道密码的人，才能知道keys.

- **Wi-Fi Protected Access 2 (WPA2): A protocol for securing Wi-Fi network communications with cryptography**
- **Design goals**
 - Everyone with the Wi-Fi password can join the network
 - Messages sent over the network are encrypted with keys
 - An attacker who does not know the Wi-Fi password cannot learn the keys

DH 加密协议

WPA2 4-Way Handshake

1. The client sends an authentication request to the access point
2. Both use the password to derive the **PSK (pre-shared key)**
3. The AP sends **ANonce** to the client
4. The client generates **SNonce**, uses the **PSK**, **nonces**, and MAC addresses to derive the **PTK (pairwise transport keys)**
5. The client sends **SNonce** and its **MIC** to the AP
6. The AP uses the **PSK**, **nonces**, and MAC addresses to derive the **PTK (pairwise transport keys)**
7. The AP sends its **MIC** and **GTK** to the client
8. The client acknowledges receiving the **GTK**



握手四步具体流程如下（简化版）：

步骤	发送方	内容	说明
①	AP → STA	Anonce	AP 发起握手，发送随机数 Anonce
②	STA → AP	Snonce + MIC	STA 回复自己的 Snonce，并计算 MIC
③	AP → STA	GTK + MIC	AP 发送 GTK (加密的) 和校验 MIC
④	STA → AP	MIC	STA 确认无误，发回最终确认

此后，STA 和 AP 都拥有了：

- 共享的 **PTK** (用于单播加密)
- **GTK** (用于组播/广播加密)

Anonce, Snonce, MIC GTK

nonce都是随机数，相当于DH密钥交换过程中，自己选择初始颜色一样。

它们各自选择的颜色，会共同生成一对PTK密钥。

💡 四个关键字段解释 (A nonce、S nonce、MIC、GTK)

1. 🔒 A nonce (Authenticator Nonce)

- 是由 AP (路由器) 生成的随机数，用于防止重放攻击。
- 它在握手第1步中发送给客户端。

2. 🔒 S nonce (Suplicant Nonce)

- 是由 STA (客户端) 生成的随机数，同样用于防止重放攻击。
- 它在握手第2步中发送给AP。

👉 A nonce 和 S nonce 是共同用于生成密钥 (PTK) 的原材料。

3. ✅ MIC (Message Integrity Code)

- 是一种“校验值”，作用类似于 HMAC。
- 用于确保握手信息没被篡改，防止中间人攻击。
- 计算 MIC 时用的是 PTK (Pairwise Transient Key) 的一部分。

4. 💙 GTK (Group Temporal Key)

- 是用来加密 广播和组播数据 的密钥。
- 由 AP生成，并在 handshake 的 第3步发送给客户端 (是加密发送的)。
- 与 PTK 不同，GTK 是“组密钥”，多个设备共享。

WPA2 - PSK attack

Casey, 你抓住了重点! 是的, **WPA2-PSK 攻击的本质就是:**

 **试图破解用户设置的 Wi-Fi 密码 (预共享密钥, Pre-Shared Key)。**

但它并不是“直接连接Wi-Fi然后猜密码”那么简单, 而是利用 **握手包 (4-Way Handshake)** 来“离线爆破密码”。

攻击的机制

获得设备在4次握手过程中的信息, 得知其nonce数据和mic信息, 因为mac地址和nonce是公开信息, 所以重点是mic信息, 用来验证生成出来的PTK对不对。

字典呢, 则是一堆PSK(Pre_shared key), 一个个生成PTK (pairwise transport key), 由于mic是由PTK的一部分, 所以可以用捕捉到的mic去验证我们生成ptk对不对。

1. 捕获握手包 (Handshake)

攻击者会:

- 等待目标设备连接 Wi-Fi;
- 或者主动发送“Deauth (断开连接) ”帧, 把设备踢下线;
- 然后在设备重新连接时, 抓住 WPA2 的 4 次握手 (尤其第2步和第3步)。

 因为握手过程中包含了 MIC (由密码间接计算得来), 所以可以用来验证密码是否正确。

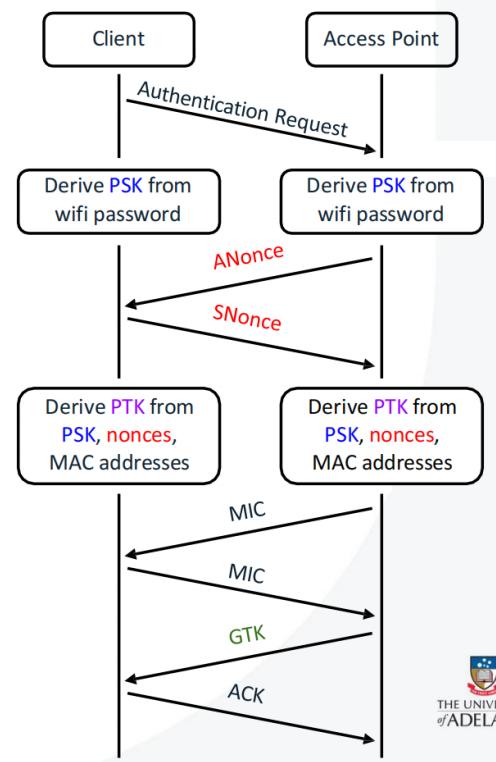
2. 离线爆破密码

- 抓到握手包后, 就可以在自己的电脑上慢慢尝试密码 (不用再连接 Wi-Fi)。
- 用字典或暴力方式一个个尝试:
 - 把猜测的密码 + 捕获的 Anonce、Snonce、MAC 等信息一起用算法生成 PTK;
 - 再用这个 PTK 计算 MIC, 看看是否与握手包中的一致。
 - 如果匹配成功, 说明密码被猜中了。

 爆破速度完全取决于攻击者的计算能力和字典质量。

WPA2-PSK Attacks

- Offline brute-force attack: People tend to choose bad passwords, and you have enough information to know if you guessed the password correctly
 - Nonces are sent unencrypted, and client and AP MAC addresses are public
 - Eavesdropper guesses a password and derives:
 - Wi-Fi password → PSK
 - PSK + nonces + MAC addresses → PTK
 - Eavesdropper checks that the MIC from the guess matches the MIC that was sent

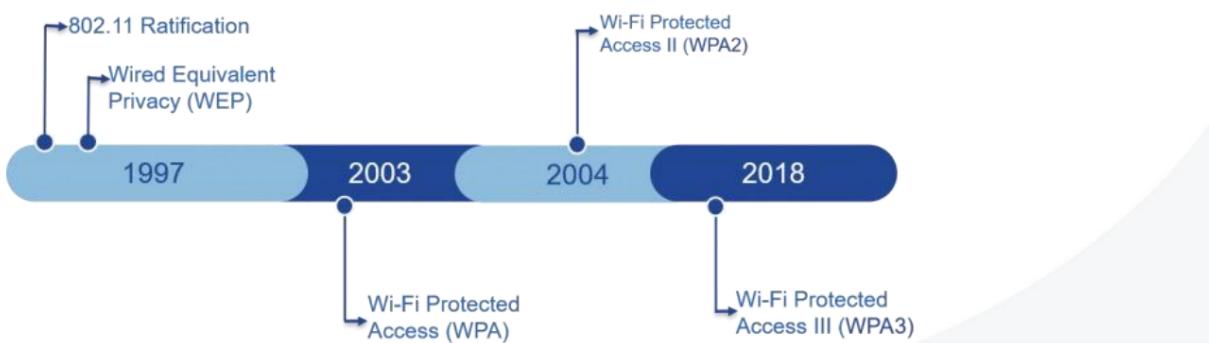


现在最新的wifi连接协议是WPA3，但是已有Dragonblood 攻击，将其攻破。

WiFi Security - Latest

- WPA2 has many more vulnerabilities
 - New offline attack on WPA2-PSK found in 2018 by Hashcat author - Jens “Atom” Steube
 - Key Reinstallation Attacks (CRACK -2017)
- WPA3 is supposed to be much harder to attack (?)
 - Dragonblood attack (2019)

Wi-Fi Security Standards Timeline



dragonblood attack on wpa3

Dragonblood Attack 是针对 **WPA3 (Wi-Fi Protected Access 3)** 安全协议的一组漏洞攻击，主要利用了 **WPA3-SAE (Simultaneous Authentication of Equals, 亦称为 Dragonfly Handshake)** 认证机制中的设计缺陷。该攻击由安全研究员 **Mathy Vanhoef** 和 **Eyal Ronen** 在 2019 年披露。

🔥 Dragonblood Attack 主要包含哪些漏洞？

Dragonblood 主要包含两类漏洞：

1 侧信道攻击 (Side-Channel Attack)

WPA3-SAE 采用了一种抗字典攻击的 PAKE (密码认证密钥交换协议)，但其密码派生过程中的某些计算泄露了信息，使得攻击者可以利用侧信道进行密码猜测攻击。

- **Cache-based Attack (缓存攻击)**：攻击者可以通过观察 CPU 缓存行为来推测密钥的某些部分。
- **Timing Attack (时间分析攻击)**：攻击者可以通过测量计算时间的变化来猜测密钥。

2 Downgrade Attack (降级攻击)

许多 WPA3 设备出于向后兼容的考虑，会同时支持 **WPA2-PSK** 和 **WPA3-SAE**。攻击者可以利用中间人 (Man-in-the-Middle, MitM) 攻击，将客户端强制降级到 WPA2，进而对 WPA2 进行已知的 **KRACK 攻击** 或其他密码破解手段。

WORKSHOP7 --- arp spoofing

attacker的ip地址234.130与mac地址8a:e9

1. Before proceeding further, re-confirm the IP address and the MAC address of your Kali machine and the IP address of your Linux VM. See Figure 3.28.

```
(kali㉿kali)-[~]
└─$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:5f:8a:e9 brd ff:ff:ff:ff:ff:ff
        inet 192.168.234.130/24 brd 192.168.234.255 scope global dynamic noprefixroute eth0
            valid_lft 1417sec preferred_lft 1417sec
        inet6 fe80::11e:152:300a:3d/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

2. In the Linux VM, display the current ARP cache table using the "arp -n" command (Note: If the arp command is not found, you can install it using `sudo apt install net-tools`). You can see a single entry for 192.168.234.2 (the default route, which is the VMWare NAT gateway) and the associated MAC address. If you can alter this cache table, then you can get the victim machine to send packets to a fake gateway machine.

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.234.2	ether	00:50:56:ee:6c:cd	C		ens33

Figure 3.29: Checking the ARP table of the Linux VM

3. In Kali Linux, enable IPv4 forwarding using this command to allow the forwarding of IP packets not intended for Kali. Make sure you are root before executing this command.

```
root@kali:/var/www/html# echo 1 > /proc/sys/net/ipv4/ip_forward
```

4. Next, execute arpspoof as follows.

```
kali@kali:/var/www/html# sudo arpspoof -t 192.168.234.131 192.168.234.2
```

target gateway

.234.2是网关的ip地址。

arpspoof具体的操作

这个指令，是攻击者在使用，所以就不需要填入攻击者自己的ip地址？

是的，`arpspoof` 不需要填入攻击者自己的 IP 地址，因为它的工作方式是伪造 ARP 响应，让目标设备错误地认为攻击者的 MAC 地址属于某个受信任的 IP（如网关或另一台设备）。攻击者的 IP 地址并不重要，关键是 攻击者的 MAC 地址 被错误地绑定到受害者的 IP 地址上。

回顾 arpspoof 的执行逻辑

1. 攻击者发送伪造的 ARP 响应：

- 目标设备 192.168.1.100 认为网关 192.168.1.1 的 MAC 地址是 攻击者的 MAC 地址。
- 网关 192.168.1.1 认为目标设备 192.168.1.100 的 MAC 地址是 攻击者的 MAC 地址。

第一个<目标IP>是我们要欺骗的target, 第二个参数，是我们要欺骗target所使用的ip地址。

因为指令是由attacker攻击者来实施的，所以命令执行后，第二个参数的ip地址，就会map成attacker的MAC地址。

arpspoof 命令使用

基本语法

bash

複製 編輯

```
arpspoof -i <网络接口> -t <目标IP> <欺骗目标的IP>
```

示例

1. 欺骗目标设备（192.168.1.100），让其认为攻击者是网关（192.168.1.1）

bash

複製 編輯

```
arpspoof -i eth0 -t 192.168.1.100 192.168.1.1
```

2. 欺骗网关 (192.168.1.1), 让其认为攻击者是目标设备 (192.168.1.100)

bash

 複製 編

```
arpspoof -i eth0 -t 192.168.1.1 192.168.1.100
```

- 网关 192.168.1.1 认为 192.168.1.100 (受害者) 对应的是攻击者的 MAC 地址。

实操 arpspoofing

kali机: 192.168.116.5 (...f2:62)

ubuntu: 192.168.116.3

gateway: 192.168.116.2

第一条指令是告诉ubuntu， 我们是gateway

第二条指令是告诉gateway，我们是ubuntu.

The screenshot shows a Wireshark interface with a terminal window integrated at the bottom. The terminal window displays a root shell on a Kali Linux system, performing an ARP spoofing attack. A yellow box highlights the terminal window, and several yellow arrows point from the terminal's command line to specific captured network frames in the main pane.

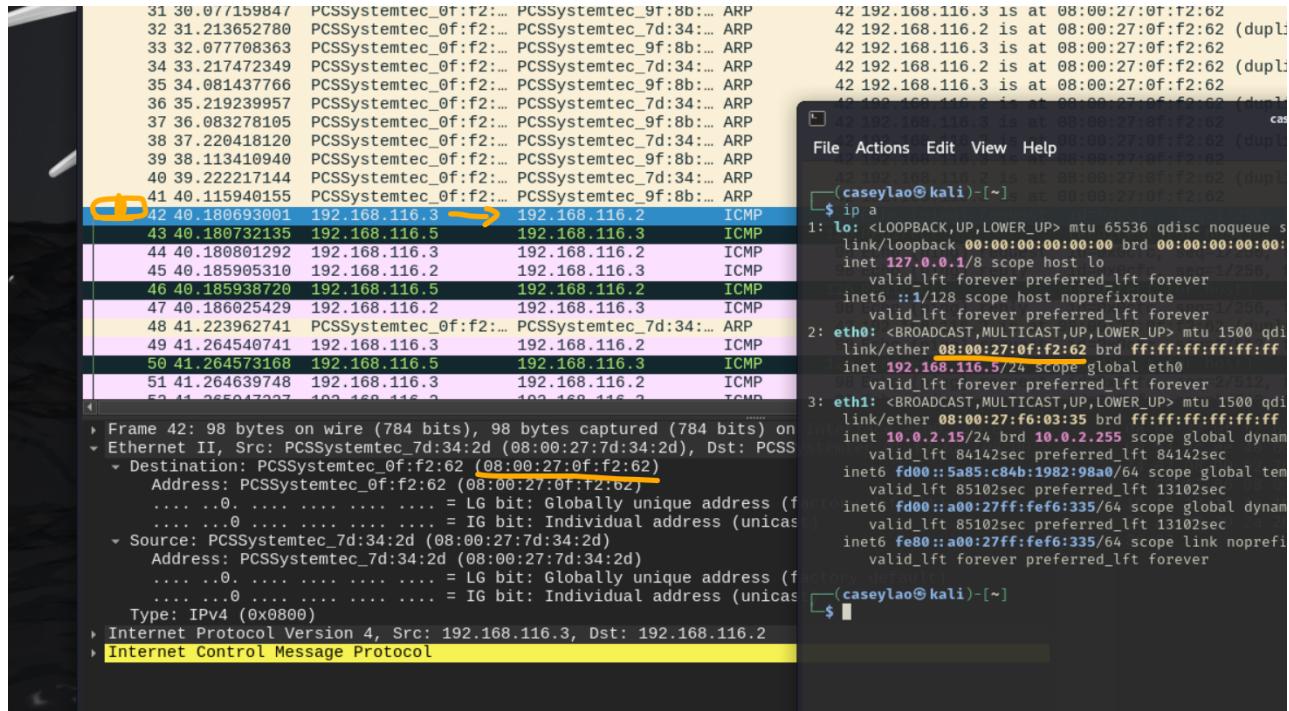
```
(caseylao㉿kali)-[~/advancedCyber/workshop7]
$ sudo arpspoof -t 192.168.116.3 192.168.116.2
8:0:27:f:f2:62 8:0:27:7d:34:2d 0806 42: arp reply 192.168.116.2 is-at 8
8:0:27:f:f2:62 8:0:27:7d:34:2d 0806 42: arp reply 192.168.116.2 is-at 8
(caseylao㉿kali)-[~]
$ sudo arpspoof -t 192.168.116.2 192.168.116.3
[sudo] password for caseylao:
```

现在测试一下，ubuntu去发消息给gateway，看看最终消息都给了谁

```
caseylao@caseylao-VirtualBox: ~/Desktop$ ping 192.168.116.2
PING 192.168.116.2 (192.168.116.2) 56(84) bytes of data.
From 192.168.116.5: icmp_seq=1 Redirect Host(New nexthop: 192.168.116.2)
64 bytes from 192.168.116.2: icmp_seq=1 ttl=254 time=6.05 ms
From 192.168.116.5: icmp_seq=2 Redirect Host(New nexthop: 192.168.116.2)
64 bytes from 192.168.116.2: icmp_seq=2 ttl=254 time=1.24 ms
From 192.168.116.5: icmp_seq=3 Redirect Host(New nexthop: 192.168.116.2)
64 bytes from 192.168.116.2: icmp_seq=3 ttl=254 time=1.52 ms
From 192.168.116.5: icmp_seq=4 Redirect Host(New nexthop: 192.168.116.2)
64 bytes from 192.168.116.2: icmp_seq=4 ttl=254 time=1.36 ms
^C
--- 192.168.116.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 4034ms
rtt min/avg/max/mdev = 1.238/2.542/6.047/2.025 ms
```

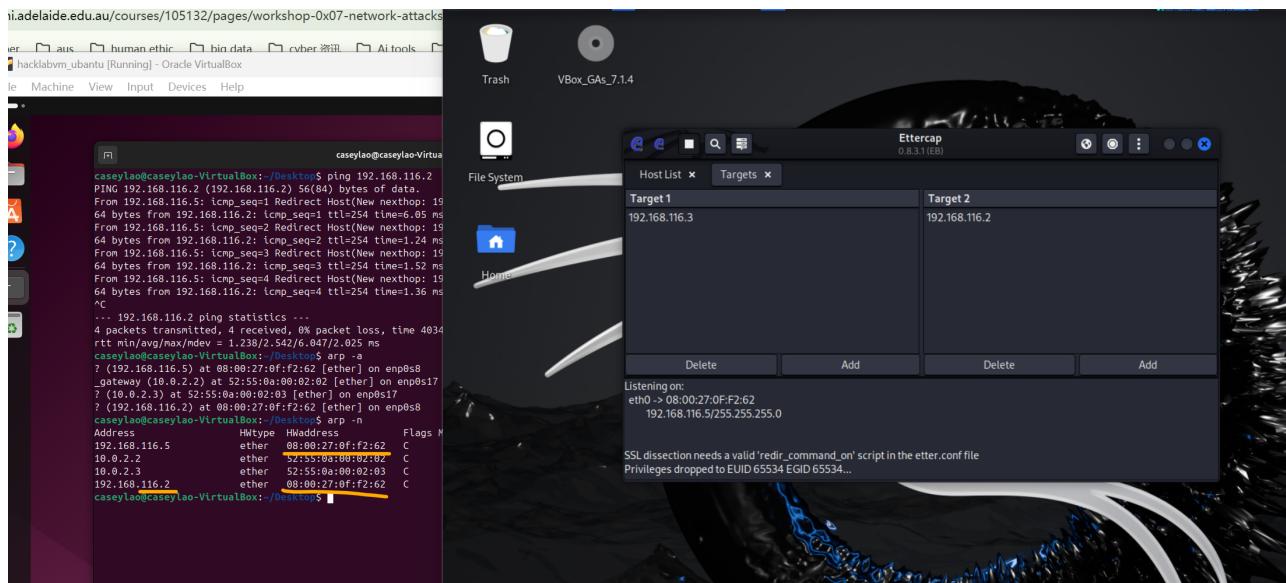
可以看到，虽然明面上是ubuntu发消息给gateway，然而看到其destination的mac地址，却是kali机的。还可以看到wireshark抓到了redirect信息，是从kali机转发给ubuntu。

简单来说，ubuntu，明明想发消息给gateway，但却发给了kali. kali返回消息给ubuntu.



使用ettercap工具，进行arpspoof

可以看到我们target 1是我们的目标ubuntu，target2，是我们的gateway地址，启动后在ubuntu中，可以看到192.168.116.2这个原本是gateway的地址，指向了kali attacker的地址。



dnsspoof

网络流量检测工具，捕获图片。

```
→ $ sudo dnsspoof
dnsspoof: eth0: no IPv4 address assigned
dnsspoof: couldn't initialize sniffing

(caseylao㉿kali)-[~/advancedCyber/workshop7]
$ dnsspoof -h
Version: 2.4
Usage: dnsspoof [-i interface] [-f hostsfile] [expression]

(caseylao㉿kali)-[~/advancedCyber/workshop7]
$ dnsspoof -i eth1
dnsspoof: libnet_open_link(): UID/EUID 0 or capability CAP_NET_RAW required

(caseylao㉿kali)-[~/advancedCyber/workshop7]
$ sudo dnsspoof -i eth1
dnsspoof: listening on eth1 [udp dst port 53 and not src 10.0.2.15]
^X@ss
```

