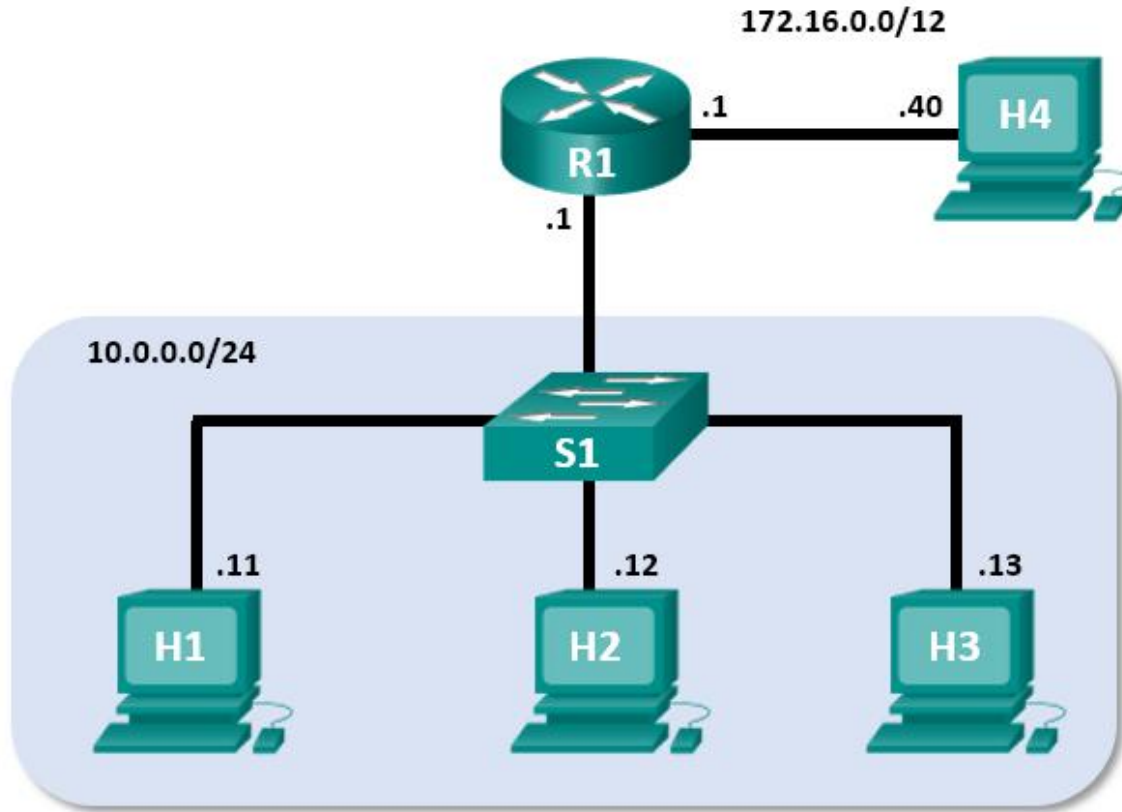


实验 - Wireshark 应用基础实验

Mininet 拓扑



目标

第 1 部分：安装和验证 Mininet 拓扑

第 2 部分：在 Wireshark 中捕获和分析 ICMP 数据

背景/场景

CyberOps 虚拟机包含一个 Python 脚本，运行该脚本时，它会设置并配置上图中所示的设备。然后，同学们便可以访问同学们的一台虚拟机内的四台主机、一台交换机和一个路由器。这样，同学们就可以模拟各种网络协议和服务，而无需配置设备的物理网络。例如，在此实验中，同学们将在 Mininet 拓扑中的两台主机之间使用 **ping** 命令，并使用 Wireshark 捕获这些 ping。

Wireshark 是一种协议分析器软件，即“数据包嗅探器”应用程序，适用于网络故障排除、分析、软件和协议开发以及教学。当数据流在网络中传输时，嗅探器可以“捕获”每个协议数据单元 (PDU)，并根据适当的 RFC 或其他规范对其内容进行解码和分析。

对于使用网络进行数据分析和排除故障的任何人来说，Wireshark 是一个有用的工具。同学们需要使用 Wireshark 来捕获 ICMP 数据包。

所需资源

- CyberOps 虚拟机
- 互联网接入

第 1 部分： 安装和验证 Mininet 拓扑

在此部分中，同学们将使用 Python 脚本在 CyberOps 虚拟机中设置 Mininet 拓扑。然后，同学们需要记录 H1 和 H2 的 IP 地址和 MAC 地址。

第 1 步： 验证 PC 的接口地址。

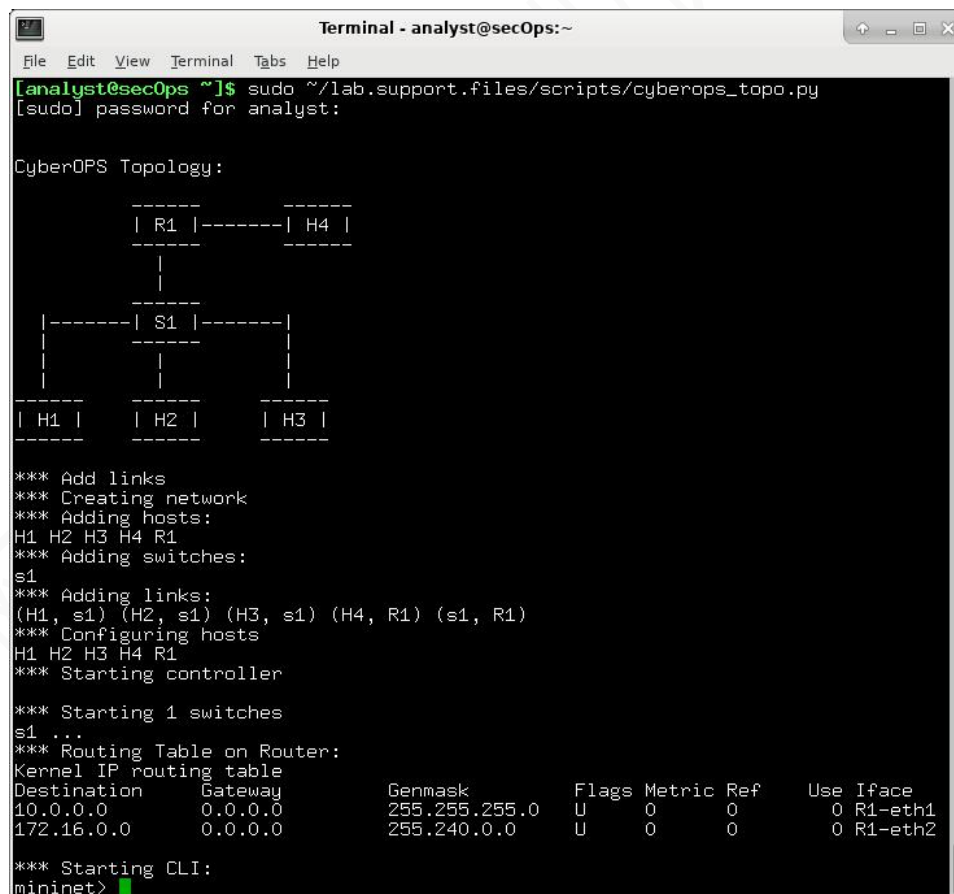
启动并登录同学们在之前的实验中使用以下凭证安装的 CyberOps Workstation：

用户名：analyst 密码：cyberops

第 2 步： 运行 Python 脚本以安装 Mininet 拓扑。

打开终端仿真程序以启动 Mininet，并在提示符后输入以下命令。系统提示时，输入 **cyberops** 作为密码。

```
[analyst@secOps ~]$ sudo ~/lab.support.files/scripts/cyberops_topo.py  
[sudo] password for analyst:
```



```
Terminal - analyst@secOps:~  
File Edit View Terminal Tabs Help  
[analyst@secOps ~]$ sudo ~/lab.support.files/scripts/cyberops_topo.py  
[sudo] password for analyst:  
  
CyberOPS Topology:  
  
      | R1 |-----| H4 |  
      |  
      |-----| S1 |-----|  
      |         |         |  
      |         |         |  
      | H1 |   | H2 |   | H3 |  
      |-----|-----|-----|  
  
*** Add links  
*** Creating network  
*** Adding hosts:  
H1 H2 H3 H4 R1  
*** Adding switches:  
s1  
*** Adding links:  
(H1, s1) (H2, s1) (H3, s1) (H4, R1) (s1, R1)  
*** Configuring hosts  
H1 H2 H3 H4 R1  
*** Starting controller  
  
*** Starting 1 switches  
s1 ...  
*** Routing Table on Router:  
Kernel IP routing table  
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface  
10.0.0.0          0.0.0.0          255.255.255.0    U        0      0      0 R1-eth1  
172.16.0.0        0.0.0.0          255.240.0.0      U        0      0      0 R1-eth2  
  
*** Starting CLI:  
mininet>
```

第 3 步： 记录 H1 和 H2 的 IP 地址和 MAC 地址。

- a. 在 mininet 提示符后，启动主机 H1 和 H2 上的终端窗口。这将为这些主机打开单独的窗口。每台主机都将有单独的网络配置，包括唯一的 IP 地址和 MAC 地址。

*** 启动 CLI:

```
mininet> xterm H1
```

```
mininet> xterm H2
```

- b. 在 **Node: H1** 上的提示符后，输入 **ifconfig** 以验证 IPv4 地址并记录 MAC 地址。对 **Node: H2** 执行相同的操作。下面突出显示了 IPv4 地址和 MAC 地址以供参考。

```
[root@secOps analyst]# ifconfig
```

```
H1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

```
inet 10.0.0.11 netmask 255.255.255.0 broadcast 10.0.0.255
```

```
inet6 fe80::2c69:4dff:febb:a219 prefixlen 64 scopeid 0x20<link>
```

```
ether 26:3a:45:65:75:23 txqueuelen 1000 (Ethernet)
```

```
RX packets 152 bytes 13036 (12.7 KiB)
```

```
RX errors 0 dropped 0 overruns 0 frame 0
```

```
TX packets 107 bytes 9658 (9.4 KiB)
```

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

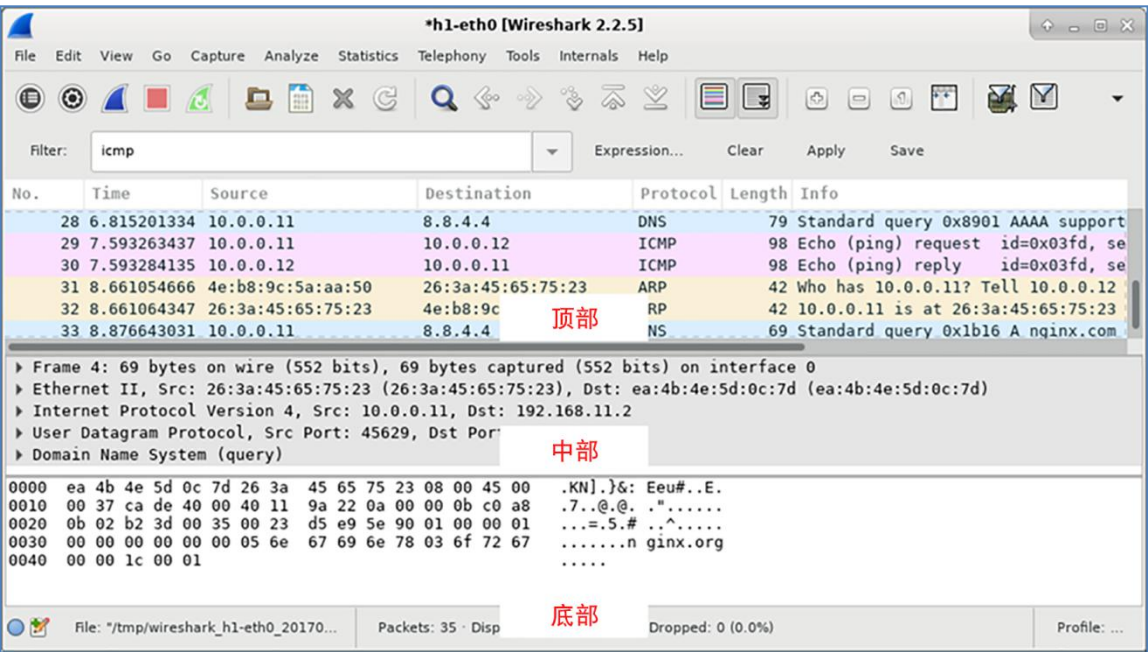
主机-接口	IP 地址	MAC 地址
H1-eth0		
H2-eth0		

第 2 部分： 在 Wireshark 中捕获和分析 ICMP 数据

在此部分，同学们需要在 Mininet 中的两台主机之间进行 ping 操作，并在 Wireshark 中捕获 ICMP 请求和应答。同学们还将查看已捕获的 PDU 内是否存在特定信息。这种分析应有助于阐明数据包报头如何用于将数据传输到目的地。

第 1 步： 检查同一 LAN 上捕获的数据。

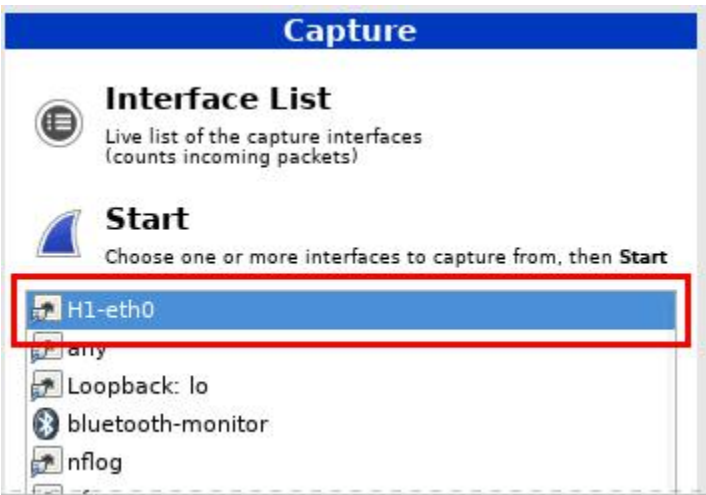
在此步骤中，同学们需要研究团队成员 PC 的 ping 请求生成的数据。Wireshark 数据分三个部分显示：1) 顶部部分显示捕获的 PDU 帧列表，其中列出 IP 数据包信息总结，2) 中间部分列出屏幕顶部部分中所选帧的 PDU 信息，并根据协议层分隔捕获的 PDU 帧，以及 3) 底部部分显示每层的原始数据。原始数据同时以十六进制和十进制形式显示。



- a. 在 **Node: H1** 上，输入 **wireshark-gtk &** 以启动 Wireshark（在此实验中，弹出警告无关紧要）。点击**确定**继续。

```
[root@secOps]# wireshark-gtk &
[1] 1552
[root@secOps ~]#
** (wireshark-gtk:1552): WARNING **: Couldn't connect to accessibility bus: Failed
to connect to socket /tmp/dbus-f0dFz9baYA: Connection refused
Gtk-Message: GtkDialog mapped without a transient parent.This is discouraged.
```

- b. 在 Wireshark 窗口的**捕获**标题下，选择 **H1-eth0** 接口。点击**开始**捕获数据流量。



- c. 在 **Node: H1** 上，如有必要，按 Enter 键以获取提示。然后键入 **ping-c 5 10.0.0.12**，对 H2 执行五次 ping 操作。命令选项 **-c** 指定 ping 操作的次数或数量。**5** 指定应发送五次 ping。所有 ping 操作均会成功。

```
[root@secOps analyst]# ping -c 5 10.0.0.12
```

- d. 导航到 Wireshark 窗口，点击**停止**以停止数据包捕获。

- e. 可以应用过滤器，只显示关注的流量。

在**过滤器**字段中键入 **icmp**，然后点击**应用**。

- f. 如有必要，点击 Wireshark 顶部部分的第一个 ICMP 请求 PDU 帧。请注意，“源”列有 H1 的 IP 地址，“目的地”列有 H2 的 IP 地址。

No.	Time	Source	Destination	Protocol	Length	Info
19	6.791692257	10.0.0.11	10.0.0.12	ICMP	98	Echo (ping) request id=0x064e, seq=1/256, ttl=64 (reply
20	6.791712977	10.0.0.12	10.0.0.11	ICMP	98	Echo (ping) reply id=0x064e, seq=1/256, ttl=64 (reque
21	7.813333879	10.0.0.11	10.0.0.12	ICMP	98	Echo (ping) request id=0x064e, seq=2/512, ttl=64 (reply
22	7.813352185	10.0.0.12	10.0.0.11	ICMP	98	Echo (ping) reply id=0x064e, seq=2/512, ttl=64 (reque
23	8.826749959	10.0.0.11	10.0.0.12	ICMP	98	Echo (ping) request id=0x064e, seq=3/768, ttl=64 (reply
24	8.826773579	10.0.0.12	10.0.0.11	ICMP	98	Echo (ping) reply id=0x064e, seq=3/768, ttl=64 (reque
25	9.839970864	10.0.0.11	10.0.0.12	ICMP	98	Echo (ping) request id=0x064e, seq=4/1024, ttl=64 (repl
26	9.839991646	10.0.0.12	10.0.0.11	ICMP	98	Echo (ping) reply id=0x064e, seq=4/1024, ttl=64 (requ

- g. 仍然在顶部部分选中此 PDU 帧，导航至中间部分。点击“Ethernet II”行左侧的箭头，查看目的 MAC 地址和源 MAC 地址。

▶ Frame 19: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▼ Ethernet II, Src: 26:3a:45:65:75:23 (26:3a:45:65:75:23), Dst: 4e:b8:9c:5a:aa:50 (4e:b8:9c:5a:aa:50)
▼ Destination: 4e:b8:9c:5a:aa:50 (4e:b8:9c:5a:aa:50)
Address: 4e:b8:9c:5a:aa:50 (4e:b8:9c:5a:aa:50)
... ..1. = LG bit: Locally administered address (this is NOT the factory default)
... ..0 = IG bit: Individual address (unicast)
▼ Source: 26:3a:45:65:75:23 (26:3a:45:65:75:23)
Address: 26:3a:45:65:75:23 (26:3a:45:65:75:23)
... ..1. = LG bit: Locally administered address (this is NOT the factory default)
... ..0 = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 10.0.0.11, Dst: 10.0.0.12
▶ Internet Control Message Protocol

源 MAC 地址是否与 H1 的接口匹配? _____

Wireshark 中的目的 MAC 地址是否与 H2 的 MAC 地址匹配? _____

注意: 在之前的已捕获 ICMP 请求示例中, ICMP 数据封装在 IPv4 数据包 PDU (IPv4 报头) 内, 然后该 PDU 会被封装到以太网 II 帧的 PDU (以太网 II 报头) 中, 以便在 LAN 上传输。

第 2 步: 检查远程 LAN 上捕获的数据。

同学们将对远程主机 (不在 LAN 中的主机) 执行 ping 操作, 并研究这些 ping 操作生成的数据。然后同学们将确定该数据与第 1 部分中研究的数据有何不同。

- a. 在 mininet 提示符后, 启动主机 H4 和 R1 上的终端窗口。

```
mininet> xterm H4
mininet> xterm R1
```

- b. 在 **Node: H4** 上的提示符后, 输入 **ifconfig** 以验证 ipv4 地址并记录 MAC 地址。对 **Node: R1** 执行相同的操作。

```
[root@secOps analyst]# ifconfig
```

主机-接口	IP 地址	MAC 地址
H4-eth0		
R1-eth1		
R1-eth2		

- c. 依次选择**捕获 > 开始**, 在 H1 上开始新的 Wireshark 捕获。也可以点击**开始**按钮, 或键入 **Ctrl-E**。点击**继续而不保存**以开始新的捕获。

- d. H4 是模拟的远程服务器。从 H1 对 H4 执行 ping 操作。该 ping 操作应该能够成功。

```
[root@secOps analyst]# ping -c 5 172.16.0.40
```

- e. 查看 Wireshark 中捕获的数据。检查同学们 ping 过的 IP 地址和 MAC 地址。请注意, MAC 地址用于 R1-eth1 接口。列出目的 IP 地址和 MAC 地址。

IP: _____ MAC: _____

- f. 在主 CyberOps 虚拟机窗口中, 输入 **quit** 以停止 Mininet。

```
mininet> quit
*** Stopping 0 controllers

*** Stopping 4 terms
*** Stopping 5 links
.....
*** Stopping 1 switches
s1
*** Stopping 5 hosts
H1 H2 H3 H4 R1
*** Done
```

- g. 要清除 Mininet 使用的所有进程, 请在提示符后输入 **sudo mn -c** 命令。

```
analyst@secOps ~]$ sudo mn -c
[sudo] password for analyst:
```

```
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd
ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd
ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([_-[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```