

## پروژه دوم

مهسا امینی ۹۸۱۷۸۲۳

سوال یک

Wireshark یک ابزار تست نفوذ منبع باز است که به متخصصان و تیم های امنیتی کمک می کند تا آسیب پذیری های امنیتی در شبکه ها و سیستم ها را شناسایی کنند. از هر دو تست خودکار و دستی برای بررسی و ارزیابی نقاط ضعف استفاده می کند. Wireshark قابلیت های زیر را دارد:

آسیب پذیری ها و نقاط ضعف:

Wireshark می تواند به طور خودکار یا دستی سیستم ها را اسکن و تجزیه و تحلیل کند تا آسیب پذیری ها را شناسایی کند. این آسیب پذیری ها ممکن است شامل ضعف هایی در نرم افزار، سرویس ها یا حتی تنظیمات سیستم باشد.

تست امنیتی:

Wireshark امکان اجرای تست های امنیتی مختلف مانند حملات مخرب مانند تزریق SQL، حملات XSS، حملات CSRF و موارد دیگر را می دهد. این تست ها به شناسایی نقاط ضعف و رفتارهای نامناسب در برنامه ها کمک می کند.

شبیه سازی حملات:

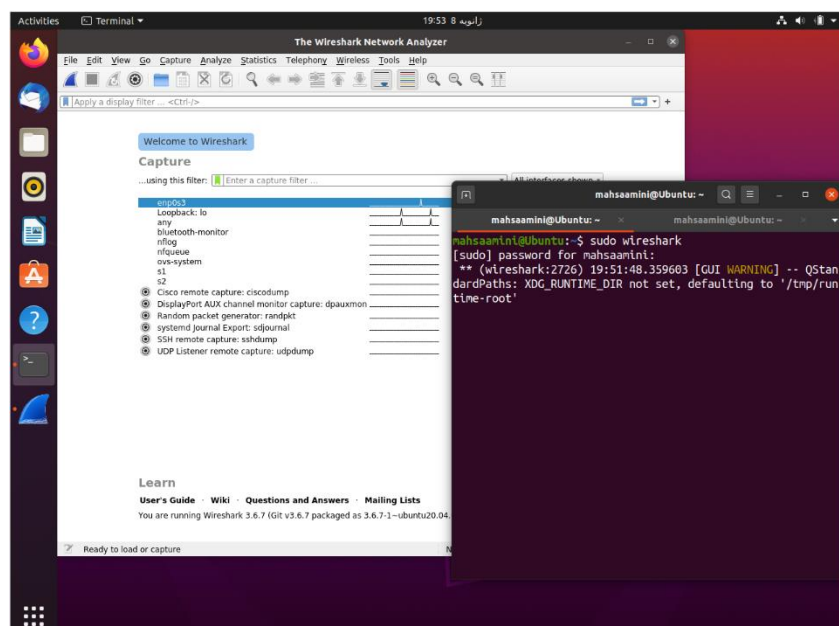
Wireshark تیم های امنیتی را قادر می سازد تا حملات شبیه سازی شده را برای ارزیابی عملکرد سیستم در برابر این حملات انجام دهند. این یک تجربه واقعی تر از نحوه رفتار سیستم در مواجهه با تهدیدات را ارائه می دهد.

گزارش نویسی:

Wireshark گزارش های جامعی در مورد آسیب پذیری ها، مسائل امنیتی و اقدامات اصلاحی توصیه شده تولید می کند. این گزارش ها تیم های امنیتی را قادر می سازد تا مسائل را به طور موثر اولویت بندی کرده و به آنها رسیدگی کنند.

سوال دو

2-



1-2

از پروتکل TCP و Openflow

The screenshot shows a terminal window with two main sections. The top section displays a table of network capture data from 'Loopback: lo'. The bottom section shows the output of Mininet CLI commands.

No.	Time	Source	Destination	Protocol	Length	Info
19	0.558031164	127.0.0.1	127.0.0.1	TCP	66	34458 → 6653 [ACK] Seq=9 Ack=29 Win=44032
20	0.574846134	127.0.0.1	127.0.0.1	OpenFlow	130	Type: OFPT_PORT_STATUS
21	0.574858025	127.0.0.1	127.0.0.1	TCP	66	6653 → 34458 [ACK] Seq=29 Ack=73 Win=44032
22	0.577945396	127.0.0.1	127.0.0.1	OpenFlow	290	Type: OFPT_FEATURES_REPLY
23	0.577954979	127.0.0.1	127.0.0.1	TCP	66	6653 → 34458 [ACK] Seq=29 Ack=297 Win=44032
24	0.602775848	127.0.0.1	127.0.0.1	TCP	74	38124 → 5937 [SYN] Seq=0 Win=43696 Len=0
25	0.602789497	127.0.0.1	127.0.0.1	TCP	54	5937 → 38124 [RST, ACK] Seq=1 Ack=1 Win=0
26	0.603099931	127.0.0.1	127.0.0.1	TCP	74	38124 → 5937 [SYN] Seq=0 Win=43696 Len=0
27	0.606811180	127.0.0.1	127.0.0.1	TCP	54	5937 → 38124 [RST, ACK] Seq=1 Ack=1 Win=0
28	0.606834858	127.0.0.1	127.0.0.1	TCP	74	38124 → 5937 [SYN] Seq=0 Win=43696 Len=0
29	0.608241111	127.0.0.1	127.0.0.1	TCP	54	5937 → 38124 [RST, ACK] Seq=1 Ack=1 Win=0
30	0.608690015	127.0.0.1	127.0.0.1	TCP	74	38124 → 5937 [SYN] Seq=0 Win=43696 Len=0
31	0.60906712	127.0.0.1	127.0.0.1	TCP	54	5937 → 38124 [RST, ACK] Seq=1 Ack=1 Win=0
32	0.609409003	127.0.0.1	127.0.0.1	TCP	74	38124 → 5937 [SYN] Seq=0 Win=43696 Len=0
33	0.609709040	127.0.0.1	127.0.0.1	TCP	54	5937 → 38124 [RST, ACK] Seq=1 Ack=1 Win=0
34	0.609844209	127.0.0.1	127.0.0.1	TCP	74	38124 → 5937 [SYN] Seq=0 Win=43696 Len=0
35	0.610005092	127.0.0.1	127.0.0.1	TCP	54	5937 → 38124 [RST, ACK] Seq=1 Ack=1 Win=0
36	0.610003708	127.0.0.1	127.0.0.1	TCP	74	38124 → 5937 [SYN] Seq=0 Win=43696 Len=0

```

mahsaamini@Ubuntu: ~/Desktop
mahsaamini@Ubuntu:~/Desktop$ sudo mn --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Setting CLI:
mininet>
  
```

2-2

The screenshot shows a terminal window with two main sections. The top section displays a table of network capture data from 'Loopback: lo'. The bottom section shows the output of Mininet CLI commands.

No.	Time	Source	Destination	Protocol	Length	Info
518	90.755559091	127.0.0.1	127.0.0.53	DNS	74	Standard query 0x4570 A ntp.ubuntu.com
519	90.755564820	127.0.0.1	127.0.0.53	DNS	74	Standard query 0x4377 AAAA ntp.ubuntu.com
520	90.755605657	127.0.0.1	127.0.0.53	DNS	74	Standard query response 0x4570 Server fail
521	90.755638622	127.0.0.1	127.0.0.53	DNS	74	Standard query response 0x4377 Server fail
522	90.755666456	127.0.0.1	127.0.0.53	DNS	74	Standard query 0x4570 A ntp.ubuntu.com
523	90.755678618	127.0.0.1	127.0.0.53	DNS	74	Standard query 0x4377 AAAA ntp.ubuntu.com
524	90.755702254	127.0.0.1	127.0.0.53	DNS	74	Standard query response 0x4570 Server fail
525	90.755732994	127.0.0.1	127.0.0.53	DNS	74	Standard query response 0x4377 Server fail
526	1.192029250	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_HELLO
527	1.192039709	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_HELLO
528	1.192082670	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_FEATURES_REQUEST
529	1.193849980	127.0.0.1	127.0.0.1	OpenFlow	78	Type: OFPT_SET_CONFIG
530	1.199213160	127.0.0.1	127.0.0.1	OpenFlow	130	Type: OFPT_PORT_STATUS
531	1.199403032	127.0.0.1	127.0.0.1	OpenFlow	130	Type: OFPT_PORT_STATUS
532	2.383709834	::	ff02::1:ff7a:4f82	OpenFlow	130	Type: OFPT_PORT_STATUS
533	2.383861367	::	ff02::1:ff7a:4f82	OpenFlow	130	Type: OFPT_PORT_STATUS
534	2.448848197	::	ff02::16	OpenFlow	130	Type: OFPT_PORT_STATUS
535	2.448154194	::	ff02::16	OpenFlow	130	Type: OFPT_PORT_STATUS

```

mahsaamini@Ubuntu: ~/Desktop
mahsaamini@Ubuntu:~/Desktop$ sudo mn --topo single,3
*** Stopping 1 switches
s1
*** Stopping 3 hosts
h1 h2 h3
*** Done
completed in 184.170 seconds
mahsaamini@Ubuntu:~/Desktop$ sudo mn --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Setting CLI:
mininet>
  
```

## OFPT\_FEATURES\_REQUEST از کنترلر به سویچ

این پیام برای دریافت شناسه ی مسیر داده (DPID) سوئیچ استفاده می‌شود. DPID به صورت منحصر به فرد یک مسیر داده را در Openflow شناسایی می کند و به صورت پویا با ترکیب MAC دستگاه در ۴۸ بیت پایین تر، همراه با رشته های ۱۶ بیتی که توسط پیاده کننده تعیین می‌شود ایجاد می‌شود. کنترلر به سوئیچ درخواست می‌دهد تا feature ها و قابلیت هایی که دارد را اعلام کند و در پاسخ هم سوئیچ به صورت OFPT\_FEATURES\_REPLY برای آن ارسال می‌کند.

The screenshot shows a Wireshark capture of network traffic on interface lo. The packet list shows several DNS queries and responses, followed by OpenFlow messages. The selected packet (No. 14) is an OFPT\_HELLO message. The packet details pane shows the OpenFlow message structure. The terminal window shows the mininet setup commands being executed in a shell.

```

mahsaamini@Ubuntu: ~/Desktop
mahsaamini@Ubuntu:~/Desktop$ sudo mn --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>

```

The screenshot shows a Wireshark capture of network traffic on interface lo. The packet list shows several TCP and UDP packets, followed by OpenFlow messages. The selected packet (No. 71) is an OFPT\_FEATURES\_REQUEST message. The packet details pane shows the OpenFlow message structure. The terminal window shows the mininet setup commands being executed in a shell.

```

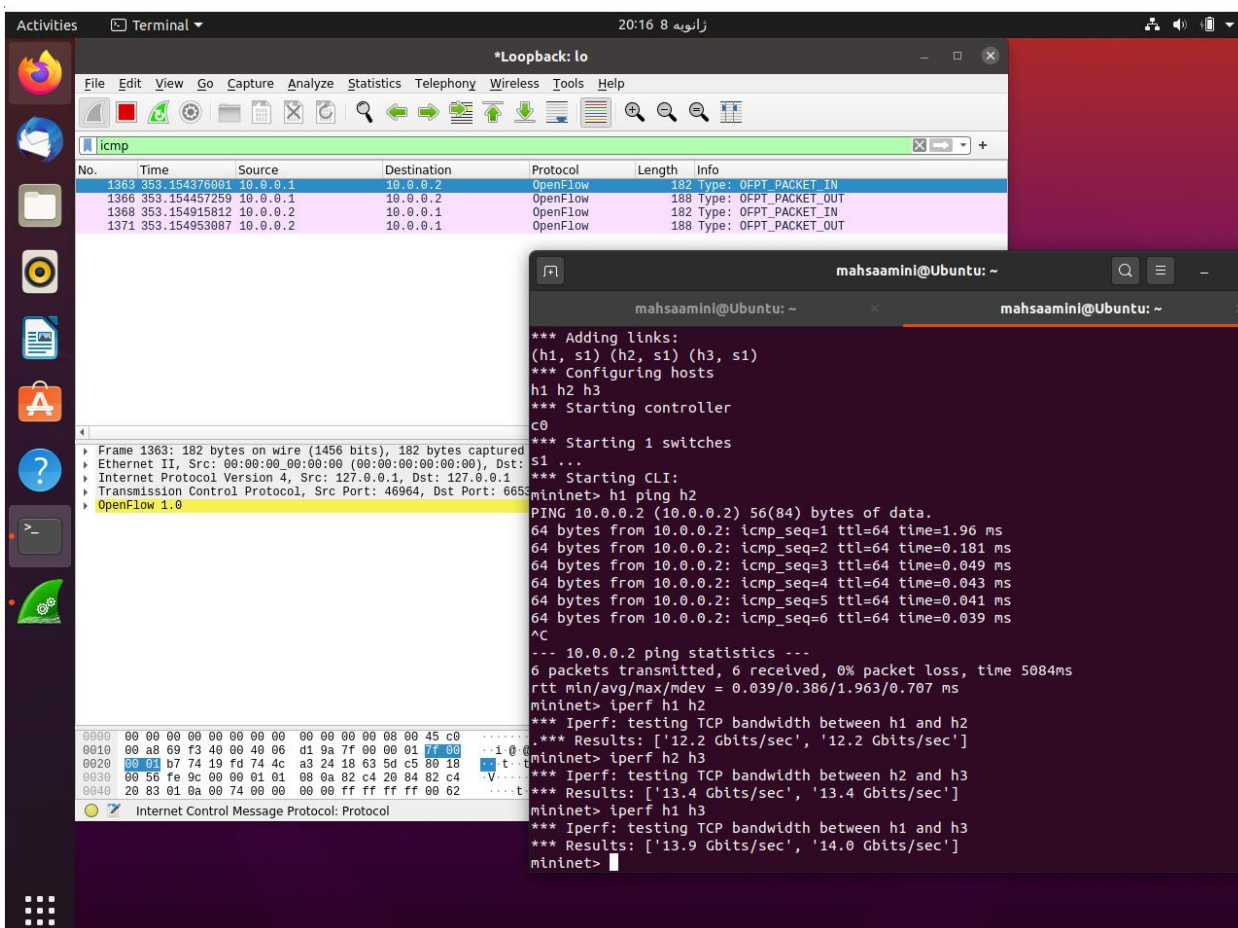
mahsaamini@Ubuntu: ~/Desktop
mahsaamini@Ubuntu:~/Desktop$ sudo mn --topo single,3
[sudo] password for mahsaamini:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>

```

در دو حالت این پیام ها ارسال می‌شوند:

- ۱- Reserve connection
- ۲- Missing fellow control

نگاهی که یک سوئیچ بسته ای را دریافت می کند که با هیچ ورودی در جدول جریان خود مطابقت ندارد، با خطای جدول جریان مواجه می شود. در OpenFlow، این وضعیت سوئیچ را برای ایجاد یک رویداد بسته در و ارسال بسته به کنترلر فعال می کند. سپس کنترلر کننده می تواند بسته را تجزیه و تحلیل کند و تصمیم بگیرد که چگونه آن را مدیریت کند. این مکانیسم برای مدیریت بسته‌هایی که با ورودی‌های جریان موجود مطابقت ندارند، بسیار مهم است و به کنترلرکننده اجازه می‌دهد تا تصمیمات پویا در مورد ارسال بسته بگیرد. همچنین هنگامی که مقدار TTL در یک بسته به صفر می رسد، سوئیچ می تواند بسته را به کنترلرکننده ارسال کند.



با استفاده از پروتکل openflow بسته‌هایی از h1 به h2 و همچنین از h2 به h1 ارسال شده است. آی پی h1 برابر 10.0.0.1 و آی پی h2 برابر 10.0.0.2 است.

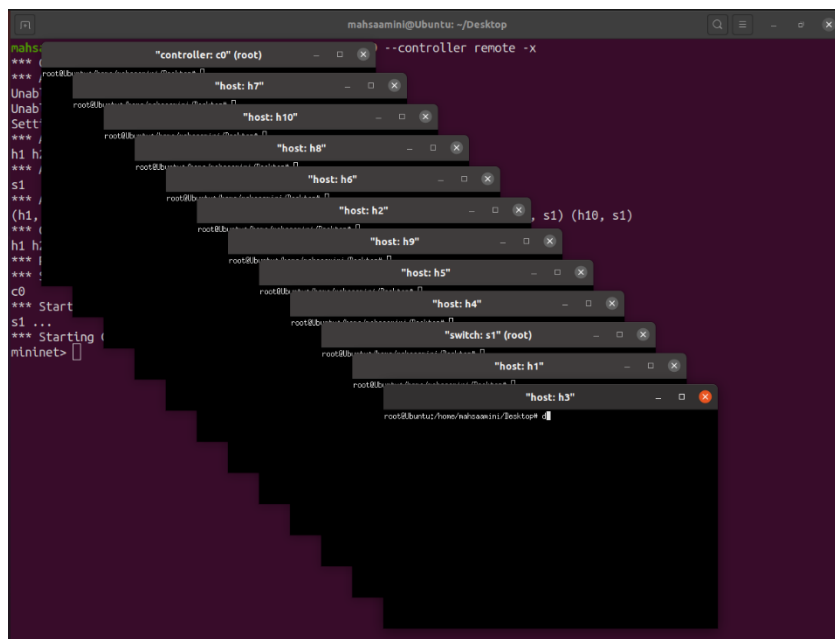
سوال سه

9817823

8+2=10

```
mahsaamini@Ubuntu: ~/Desktop
[sudo] password for mahsaamini:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2950>
<Host h2: h2-eth0:10.0.0.2 pid=2954>
<Host h3: h3-eth0:10.0.0.3 pid=2956>
<Host h4: h4-eth0:10.0.0.4 pid=2958>
<Host h5: h5-eth0:10.0.0.5 pid=2960>
<Host h6: h6-eth0:10.0.0.6 pid=2962>
<Host h7: h7-eth0:10.0.0.7 pid=2964>
<Host h8: h8-eth0:10.0.0.8 pid=2966>
<Host h9: h9-eth0:10.0.0.9 pid=2968>
<Host h10: h10-eth0:10.0.0.10 pid=2970>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-eth6:None,s1-eth7:None,
s1-eth8:None,s1-eth9:None,s1-eth10:None pid=2975>
<Controller c0: 127.0.0.1:6653 pid=2943>
mininet>
```

طور خاص استفاده از Open vSwitch را به عنوان نوع سوئیچ نشان می دهد که معمولاً در محیط های مجازی استفاده می شود. این به کاربران اجازه می دهد تا تنظیمات Open vSwitch را آزمایش و آزمایش کنند



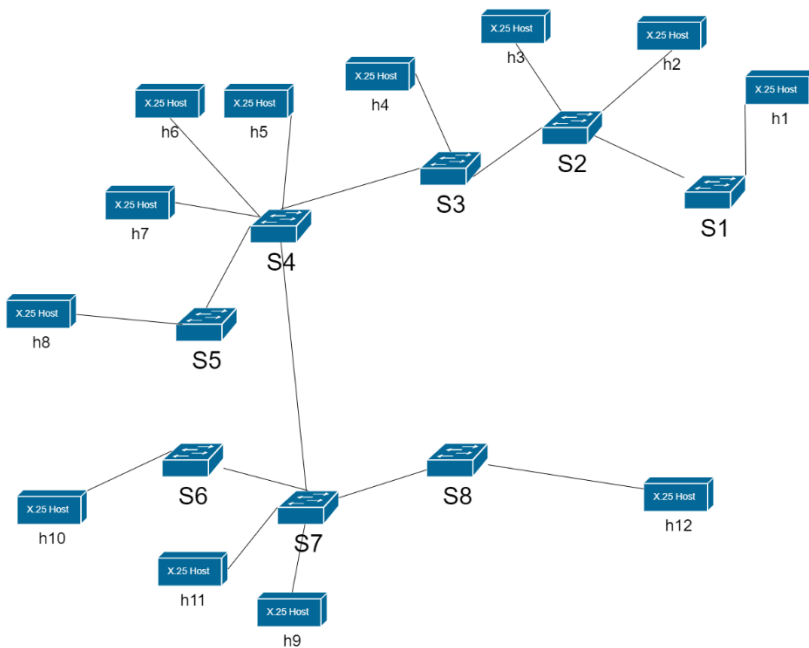
این بدان معناست که Mininet به یک کنترلر SDN خارجی متصل می شود. ممکن است لازم باشد آدرس IP یا نام میزبان کنترلر SDN را ارائه داد.





سوال چهار

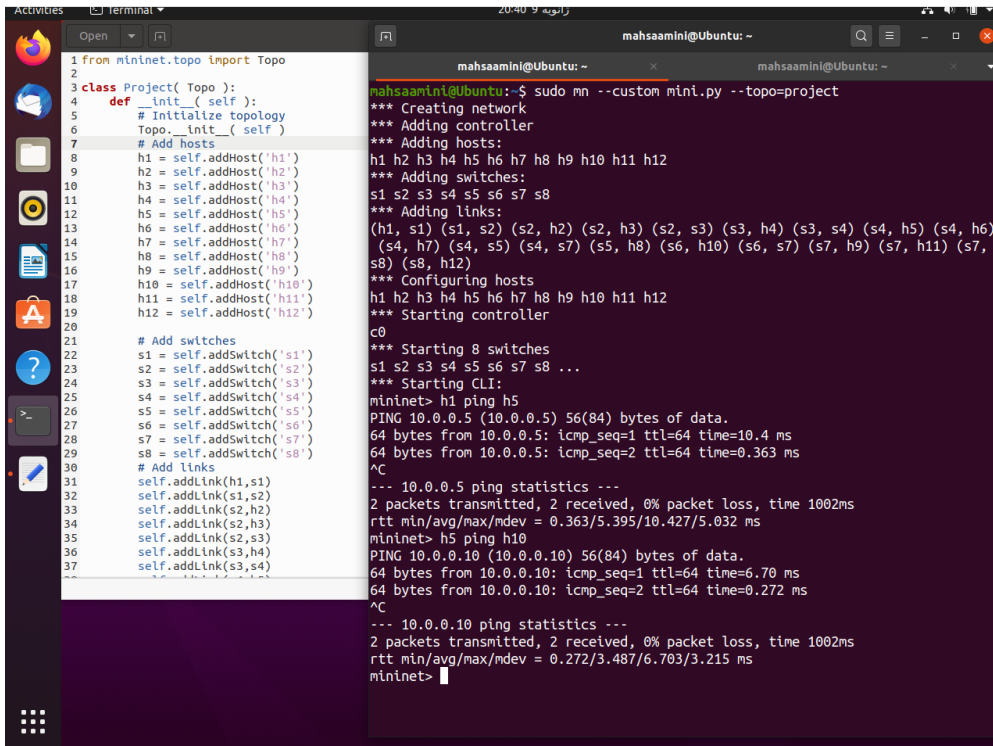
9817823



```
1 from mininet.topo import Topo
2
3 class Project( Topo ):
4     def _init_( self ):
5         # Initialize topology
6         Topo._init_( self )
7         # Add hosts
8         h1 = self.addHost('h1')
9         h2 = self.addHost('h2')
10        h3 = self.addHost('h3')
11        h4 = self.addHost('h4')
12        h5 = self.addHost('h5')
13        h6 = self.addHost('h6')
14        h7 = self.addHost('h7')
15        h8 = self.addHost('h8')
16        h9 = self.addHost('h9')
17        h10 = self.addHost('h10')
18        h11 = self.addHost('h11')
19        h12 = self.addHost('h12')
20
21        # Add switches
22        s1 = self.addSwitch('s1')
23        s2 = self.addSwitch('s2')
24        s3 = self.addSwitch('s3')
25        s4 = self.addSwitch('s4')
26        s5 = self.addSwitch('s5')
27        s6 = self.addSwitch('s6')
28        s7 = self.addSwitch('s7')
29        s8 = self.addSwitch('s8')
30
31        # Add links
32        self.addLink(h1,s1)
33        self.addLink(s1,s2)
34        self.addLink(s2,h2)
35        self.addLink(s2,s3)
36        self.addLink(s3,h3)
37        self.addLink(s3,s4)
38        self.addLink(s4,h4)
39        self.addLink(s4,s5)
40        self.addLink(s5,h5)
41        self.addLink(s5,s6)
42        self.addLink(s6,h6)
43        self.addLink(s6,s7)
44        self.addLink(s7,h7)
45        self.addLink(s7,s8)
46        self.addLink(s8,h8)
47        self.addLink(s8,s1)
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
mininet> sudo mn --custom mini.py --topo=project
[sudo] password for mahsaamini:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8
*** Adding links:
(h1, s1) (s1, s2) (s2, h2) (s2, s3) (s3, h3) (s3, s4) (s4, h4)
(s4, s5) (s5, h5) (s5, s6) (s6, h6) (s6, s7) (s7, h7) (s7, s8)
(s8, h8) (s8, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
mininet>
```

## گرفتن ping



```
1 from mininet.topo import Topo
2
3 class Project( Topo ):
4     def __init__( self ):
5         # Initialize topology
6         Topo.__init__( self )
7
8         # Add hosts
9         h1 = self.addHost('h1')
10        h2 = self.addHost('h2')
11        h3 = self.addHost('h3')
12        h4 = self.addHost('h4')
13        h5 = self.addHost('h5')
14        h6 = self.addHost('h6')
15        h7 = self.addHost('h7')
16        h8 = self.addHost('h8')
17        h9 = self.addHost('h9')
18        h10 = self.addHost('h10')
19        h11 = self.addHost('h11')
20        h12 = self.addHost('h12')
21
22        # Add switches
23        s1 = self.addSwitch('s1')
24        s2 = self.addSwitch('s2')
25        s3 = self.addSwitch('s3')
26        s4 = self.addSwitch('s4')
27        s5 = self.addSwitch('s5')
28        s6 = self.addSwitch('s6')
29        s7 = self.addSwitch('s7')
30        s8 = self.addSwitch('s8')
31
32        # Add links
33        self.addLink(h1,s1)
34        self.addLink(s1,s2)
35        self.addLink(s2,h2)
36        self.addLink(s2,h3)
37        self.addLink(s2,s3)
38        self.addLink(s3,h4)
39        self.addLink(s3,s4)
40        self.addLink(s4,s5)
41        self.addLink(s5,h5)
42        self.addLink(s5,s6)
43        self.addLink(s6,h6)
44        self.addLink(s6,s7)
45        self.addLink(s7,h7)
46        self.addLink(s7,s8)
47        self.addLink(s8,h8)
48        self.addLink(s8,h9)
49        self.addLink(s8,h10)
50        self.addLink(s8,h11)
51        self.addLink(s8,h12)
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
mahsaamini@Ubuntu:~$ sudo mn --custom mini.py --topo=project
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8
*** Adding links:
(h1, s1) (s1, s2) (s2, h2) (s2, h3) (s2, s3) (s3, h4) (s3, s4) (s4, h5) (s4, h6)
(s4, h7) (s4, s5) (s4, s7) (s5, h8) (s5, h9) (s5, h10) (s6, h10) (s6, s7) (s7, h9) (s7, h11) (s7,
s8) (s8, h12)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=10.4 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.363 ms
^C
--- 10.0.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.363/5.395/10.427/5.032 ms
mininet> h5 ping h10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=6.70 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.272 ms
^C
--- 10.0.0.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.272/3.487/6.703/3.215 ms
mininet>
```

2

هنگامی که یک فرمان پینگ برای اولین بار صادر می شود، زیرسیستم شبکه و پروتکل های مرتبط ممکن است نیاز به مقداری اولیه داشته باشند. این شامل راه اندازی ساختارهای داده لازم، تخصیص منابع و آماده سازی سیستم برای ارتباطات شبکه است. همچنین سیستم ممکن است نیاز به تعیین مسیر بهینه برای ارسال بسته پینگ داشته باشد، و این فرآیند می تواند به افزایش تاخیر در اولین تلاش کمک کند.

تلاش های پینگ بعدی از ذخیره سازی و بهینه سازی سود می برند. اطلاعات مربوط به مقصد، مانند آدرس های و مسیرهای مسیریابی، ممکن است پس از اولین تلاش ذخیره شوند. این اطلاعات ذخیره شده در حافظه پنهان امکان پردازش سریعتر در پینگ های بعدی را فراهم می کند.



دستور dump

```
1 from mininet.topo import Topo
2
3 class Project( Topo ):
4     def __init__( self ):
5         # Initialize topology
6         Topo.__init__( self )
7         # Add hosts
8         h1 = self.addHost('h1')
9         h2 = self.addHost('h2')
10        h3 = self.addHost('h3')
11        h4 = self.addHost('h4')
12        h5 = self.addHost('h5')
13        h6 = self.addHost('h6')
14        h7 = self.addHost('h7')
15        h8 = self.addHost('h8')
16        h9 = self.addHost('h9')
17        h10 = self.addHost('h10')
18        h11 = self.addHost('h11')
19        h12 = self.addHost('h12')
20
21        # Add switches
22        s1 = self.addSwitch('s1')
23        s2 = self.addSwitch('s2')
24        s3 = self.addSwitch('s3')
25        s4 = self.addSwitch('s4')
26        s5 = self.addSwitch('s5')
27        s6 = self.addSwitch('s6')
28        s7 = self.addSwitch('s7')
29        s8 = self.addSwitch('s8')
30        # Add links
31        self.addLink(h1,s1)
32        self.addLink(s1,s2)
33        self.addLink(s2,h2)
34        self.addLink(s2,h3)
35        self.addLink(s2,s3)
36        self.addLink(s3,h4)
37        self.addLink(s3,s4)
```

```
mahsaamini@Ubuntu: ~
*** Adding links:
(h1, s1) (s1, s2) (s2, h2) (s2, h3) (s2, s3) (s3, h4) (s3, s4) (s4, h5) (s4, h6)
(s4, h7) (s4, s5) (s4, s7) (s5, h8) (s6, h10) (s6, s7) (s7, h9) (s7, h11) (s7,
s8) (s8, h12)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2980>
<Host h2: h2-eth0:10.0.0.2 pid=2982>
<Host h3: h3-eth0:10.0.0.3 pid=2984>
<Host h4: h4-eth0:10.0.0.4 pid=2986>
<Host h5: h5-eth0:10.0.0.5 pid=2988>
<Host h6: h6-eth0:10.0.0.6 pid=2990>
<Host h7: h7-eth0:10.0.0.7 pid=2992>
<Host h8: h8-eth0:10.0.0.8 pid=2994>
<Host h9: h9-eth0:10.0.0.9 pid=2996>
<Host h10: h10-eth0:10.0.0.10 pid=2998>
<Host h11: h11-eth0:10.0.0.11 pid=3000>
<Host h12: h12-eth0:10.0.0.12 pid=3002>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=3007>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None
pid=3010>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=3013>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None,
s4-eth5:None,s4-eth6:None pid=3016>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None pid=3019>
<OVSSwitch s6: lo:127.0.0.1,s6-eth1:None,s6-eth2:None pid=3022>
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None,s7-eth4:None,
s7-eth5:None pid=3025>
<OVSSwitch s8: lo:127.0.0.1,s8-eth1:None,s8-eth2:None pid=3028>
<Controller c0: 127.0.0.1:6653 pid=2973>
mininet>
```

دستور nudes و دستور pingall

```
1 from mininet.topo import Topo
2
3 class Project( Topo ):
4     def __init__( self ):
5         # Initialize topology
6         Topo.__init__( self )
7         # Add hosts
8         h1 = self.addHost('h1')
9         h2 = self.addHost('h2')
10        h3 = self.addHost('h3')
11        h4 = self.addHost('h4')
12        h5 = self.addHost('h5')
13        h6 = self.addHost('h6')
14        h7 = self.addHost('h7')
15        h8 = self.addHost('h8')
16        h9 = self.addHost('h9')
17        h10 = self.addHost('h10')
18        h11 = self.addHost('h11')
19        h12 = self.addHost('h12')
20
21        # Add switches
22        s1 = self.addSwitch('s1')
23        s2 = self.addSwitch('s2')
24        s3 = self.addSwitch('s3')
25        s4 = self.addSwitch('s4')
26        s5 = self.addSwitch('s5')
27        s6 = self.addSwitch('s6')
28        s7 = self.addSwitch('s7')
29        s8 = self.addSwitch('s8')
30        # Add links
31        self.addLink(h1,s1)
32        self.addLink(s1,s2)
33        self.addLink(s2,h2)
34        self.addLink(s2,h3)
35        self.addLink(s2,s3)
36        self.addLink(s3,h4)
37        self.addLink(s3,s4)
```

```
mahsaamini@Ubuntu: ~
mahsaamini@Ubuntu:~$ sudo mn --custom mini.py --topo=project
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8
*** Adding links:
(h1, s1) (s1, s2) (s2, h2) (s2, h3) (s2, s3) (s3, h4) (s3, s4) (s4, h5) (s4, h6)
(s4, h7) (s4, s5) (s4, s7) (s5, h8) (s6, h10) (s6, s7) (s7, h9) (s7, h11) (s7,
s8) (s8, h12)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h10 h11 h12 h2 h3 h4 h5 h6 h7 h8 h9 s1 s2 s3 s4 s5 s6 s7 s8
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
mininet>
```

