

DNS Spoofing Attacks

Without attack:

```
dnsquery h5 www.foo.local nohup.out tcpdump2.pcap topo.py
ubuntu@ip-172-31-25-108:~/18731$ sudo python topo.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3
*** Adding links:
(100.00Mbit 0ms delay) (100.00Mbit 0ms delay) (s1, h1) (100.00Mbit 0ms delay)
(100.00Mbit 0ms delay) (s1, h2) (100.00Mbit 0ms delay) (100.00Mbit 0ms delay)
(s1, s2) (100.00Mbit 1500ms delay) (100.00Mbit 1500ms delay) (s2, h3) (100.
00Mbit 0ms delay) (100.00Mbit 0ms delay) (s2, s3) (100.00Mbit 0ms delay) (100
.00Mbit 0ms delay) (s3, h4) (100.00Mbit 0ms delay) (100.00Mbit 0ms delay) (s3
, h5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ... (100.00Mbit 0ms delay) (100.00Mbit 0ms delay) (100.00Mbit 0ms del
ay) (100.00Mbit 0ms delay) (100.00Mbit 0ms delay) (100.00Mbit 1500ms delay) (
100.00Mbit 0ms delay) (100.00Mbit 0ms delay) (100.00Mbit 0ms delay)
*** Starting CLI:
mininet> h1 ./init
h1
mininet> h2 ./init
h2
mininet> h3 ./init
h3
mininet> h4 ./init
h4
mininet> h5 ./init
h5
mininet> h1 wget -q -O - http://foo.local:8080
I'm h4, hosting the REAL foo.local website.
mininet> h1 nslookup foo.local
Server:      10.2.3.1
Address:     10.2.3.1#53

Name:   foo.local
Address: 10.3.2.1

mininet>
```

```
Flow nexus
WARNING:openflow.of_01:<class 'pox.openflow.PortStatus'> raised on dummy Open
Flow nexus
WARNING:openflow.of_01:<class 'pox.openflow.PortStatus'> raised on dummy Open
Flow nexus
WARNING:openflow.of_01:<class 'pox.openflow.PacketIn'> raised on dummy OpenFl
ow nexus
INFO:openflow.of_01:[00-00-00-00-00-01 4] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-01
DEBUG:f.t_p.00-00-00-00-00-01:Connect [00-00-00-00-00-01 4]
DEBUG:f.t_p:Disabling flooding for 4 ports
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-02
DEBUG:f.t_p.00-00-00-00-00-02:Connect [00-00-00-00-00-02 3]
DEBUG:f.t_p:Disabling flooding for 4 ports
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-03
DEBUG:f.t_p.00-00-00-00-00-03:Connect [00-00-00-00-00-03 2]
DEBUG:f.t_p:Disabling flooding for 4 ports
INFO:openflow.discovery:link detected: 00-00-00-00-00-01.1 -> 00-00-00-00-00-
02.1
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.1 -> 00-00-00-00-00-
01.1
DEBUG:f.t_p.00-00-00-00-00-01:Learn 10.1.2.1 -> e6:a5:99:d5:be:bd by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.1.2.1 to e6:a5:99:d5:be:bd
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.2 -> 00-00-00-00-00-
03.1
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.1 -> 00-00-00-00-00-
02.2
DEBUG:f.t_p.00-00-00-00-00-01:Learn 10.1.3.1 -> 1e:06:50:3a:c0:8e by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.1.3.1 to 1e:06:50:3a:c0:8e
DEBUG:f.t_p.00-00-00-00-00-02:Learn 10.2.3.1 -> 86:b2:ff:b2:27:11 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.2.3.1 to 86:b2:ff:b2:27:11
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.1 -> 00-00-00-00-00-
01.1
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.2 -> 00-00-00-00-00-
03.1
DEBUG:f.t_p.00-00-00-00-00-01:Learn 10.1.2.1 -> 4e:93:0f:b1:54:33 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.1.2.1 to 4e:93:0f:b1:54:33
INFO:openflow.discovery:link detected: 00-00-00-00-00-01.1 -> 00-00-00-00-00-
02.1
DEBUG:f.t_p.00-00-00-00-00-01:Learn 10.1.3.1 -> 9a:5f:89:8d:29:b9 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.1.3.1 to 9a:5f:89:8d:29:b9
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.1 -> 00-00-00-00-00-
01.1
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.2 -> 00-00-00-00-00-
03.1
DEBUG:f.t_p.00-00-00-00-00-02:Learn 10.2.3.1 -> ae:68:7d:c4:6b:01 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.2.3.1 to ae:68:7d:c4:6b:01
DEBUG:f.t_p.00-00-00-00-00-03:Learn 10.3.2.1 -> f2:74:bf:cf:13:f7 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.3.2.1 to f2:74:bf:cf:13:f7
DEBUG:f.t_p.00-00-00-00-00-03:Learn 10.3.3.1 -> 06:e2:b9:5c:8d:2f by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.3.3.1 to 06:e2:b9:5c:8d:2f
```

`init` is a script setting up resolv.conf on h1 and h2, setting up dnsmasq on h3, setting up web servers on h4 and h5.

On h1, use `wget` to retrieve content from <http://foo.local:8080>. The page content is correct. Also, `nslookup` returned correct result.

With attack:

```
mininet> h2 ./init
h2
mininet> h3 ./init
h3
mininet> h4 ./init
h4
mininet> h5 ./init
h5
mininet> s1 cat slinit
#!/bin/bash
sudo route add -host 10.1.2.1 s1-eth2
sudo arp -s 10.1.2.1 $1
sudo tcpdump -s 0 udp port 53 -w sl.pcap &
sudo python attack.py &
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr 4e:93:0f:b1:54:33
         inet addr:10.1.2.1  Bcast:10.255.255.255  Mask:255.0.0.0
         inet6 addr: fe80::4c93:fff:feb1:5433/64  Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:14 errors:0 dropped:0 overruns:0 frame:0
         TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:1158 (1.1 KB)  TX bytes:990 (990.0 B)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128  Scope:Host
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> s1 ./slinit 4e:93:0f:b1:54:33
mininet> h1 wget -q -O - http://foo.local:8080
I'm h5, hosting the FAKE foo.local website.
mininet> h1 nslookup foo.local
Server:      10.2.3.1
Address:     10.2.3.1#53

Name:   foo.local
Address: 10.3.3.1

mininet>
```

```
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Jul 22 2015 17:58:13)
DEBUG:core:Platform is Linux-3.13.0-48-generic-x86_64-with-Ubuntu-14.04-trust
y
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-03
DEBUG:f.t_p.00-00-00-00-00-03:Connect [00-00-00-00-00-03 2]
DEBUG:f.t_p:Disabling flooding for 4 ports
INFO:openflow.of_01:[00-00-00-00-00-01 4] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-01
DEBUG:f.t_p.00-00-00-00-00-01:Connect [00-00-00-00-00-01 4]
DEBUG:f.t_p:Disabling flooding for 4 ports
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-02
DEBUG:f.t_p.00-00-00-00-00-02:Connect [00-00-00-00-00-02 3]
DEBUG:f.t_p:Disabling flooding for 4 ports
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.1 -> 00-00-00-00-00-
02.2
DEBUG:f.t_p.00-00-00-00-00-01:Learn 10.1.2.1 -> 4e:93:0f:b1:54:33 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.1.2.1 to 4e:93:0f:b1:54:33
INFO:openflow.discovery:link detected: 00-00-00-00-00-01.1 -> 00-00-00-00-00-
02.1
DEBUG:f.t_p.00-00-00-00-00-01:Learn 10.1.3.1 -> 9a:5f:89:8d:29:b9 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.1.3.1 to 9a:5f:89:8d:29:b9
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.1 -> 00-00-00-00-00-
01.1
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.2 -> 00-00-00-00-00-
03.1
DEBUG:f.t_p.00-00-00-00-00-02:Learn 10.2.3.1 -> ae:68:7d:c4:6b:01 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.2.3.1 to ae:68:7d:c4:6b:01
DEBUG:f.t_p.00-00-00-00-00-03:Learn 10.3.2.1 -> f2:74:bf:cf:13:f7 by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.3.2.1 to f2:74:bf:cf:13:f7
DEBUG:f.t_p.00-00-00-00-00-03:Learn 10.3.3.1 -> 06:e2:b9:5c:8d:2f by DHCP Lea
se
INFO:proto.dhcpd:Leased 10.3.3.1 to 06:e2:b9:5c:8d:2f
```

This time, `s1init` is run on `s1` to do the DNS spoofing attack. Now, when h1 tries to retrieve content from <http://foo.local:8080>, he actually reaches h5, the fake foo.local website. And nslookup also shows the domain foo.local points to h5.

The log file of dnsmasq is shown here. The DNS query actually reached h3, but s1's spoofed DNS response reaches h1 first.

```
Apr 1 01:02:53 ip-172-31-25-108 dnsmasq[31933]: read /etc/hosts - 8 addresses
Apr 1 01:03:27 ip-172-31-25-108 dnsmasq[31933]: query[A] foo.local from 10.1.2.1
Apr 1 01:03:27 ip-172-31-25-108 dnsmasq[31933]: config foo.local is 10.3.2.1
Apr 1 01:03:27 ip-172-31-25-108 dnsmasq[31933]: query[AAAA] foo.local from 10.1.2.1
Apr 1 01:03:27 ip-172-31-25-108 dnsmasq[31933]: config foo.local is NODATA-IPv6
Apr 1 01:03:32 ip-172-31-25-108 dnsmasq[31933]: query[A] foo.local from 10.1.2.1
Apr 1 01:03:32 ip-172-31-25-108 dnsmasq[31933]: config foo.local is 10.3.2.1
Apr 1 01:03:34 ip-172-31-25-108 dnsmasq[31933]: query[AAAA] foo.local from 10.1.2.1
Apr 1 01:03:34 ip-172-31-25-108 dnsmasq[31933]: config foo.local is NODATA-IPv6
Apr 1 01:03:37 ip-172-31-25-108 dnsmasq[31933]: query[A] foo.local from 10.1.2.1
Apr 1 01:03:37 ip-172-31-25-108 dnsmasq[31933]: config foo.local is 10.3.2.1
Apr 1 01:03:39 ip-172-31-25-108 dnsmasq[31933]: query[AAAA] foo.local from 10.1.2.1
Apr 1 01:03:39 ip-172-31-25-108 dnsmasq[31933]: config foo.local is NODATA-IPv6
Apr 1 01:03:53 ip-172-31-25-108 dnsmasq[31933]: query[A] foo.local from 10.1.2.1
Apr 1 01:03:53 ip-172-31-25-108 dnsmasq[31933]: config foo.local is 10.3.2.1
```