

Assignment 3 - DNS Spoofing

Contents

Introduction.....	2
Design	3
Testing	5
Test 1	6
Test 2	8

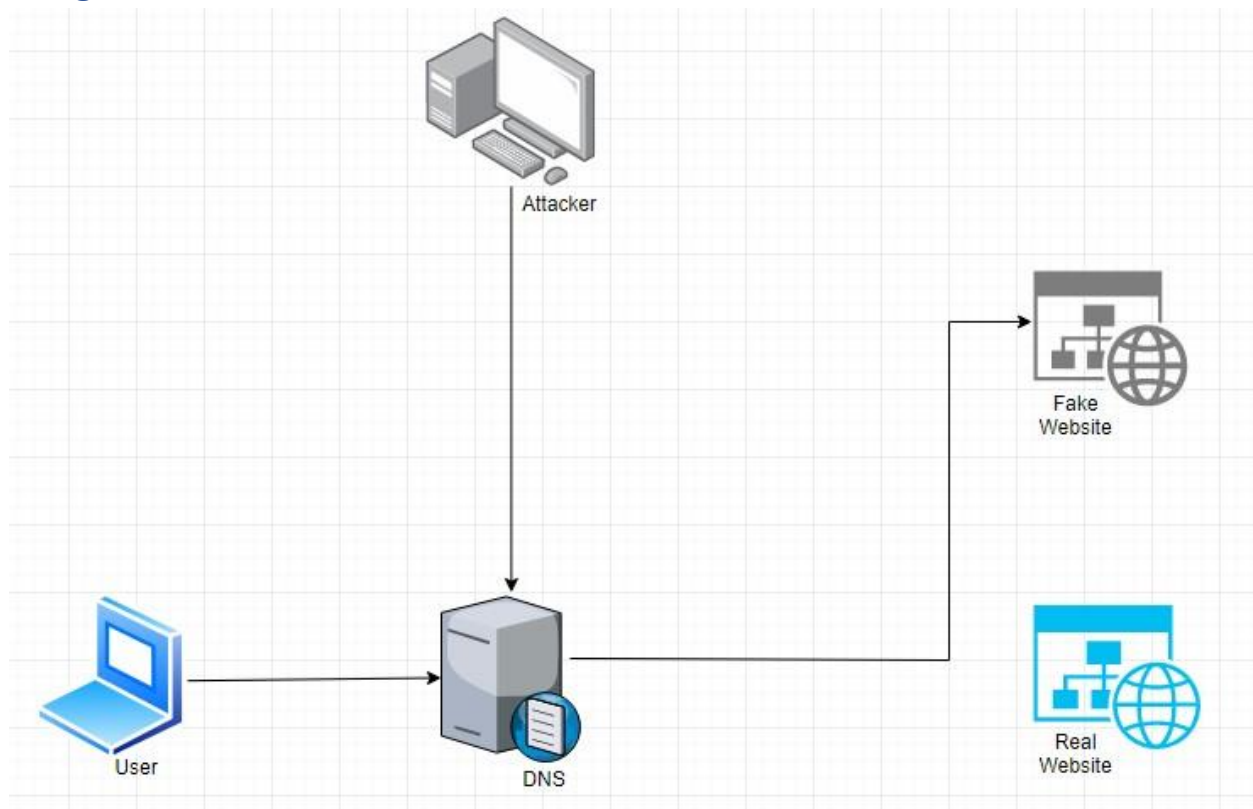
Yoshiaki Ryuzaki
Sukh Atwal

Introduction

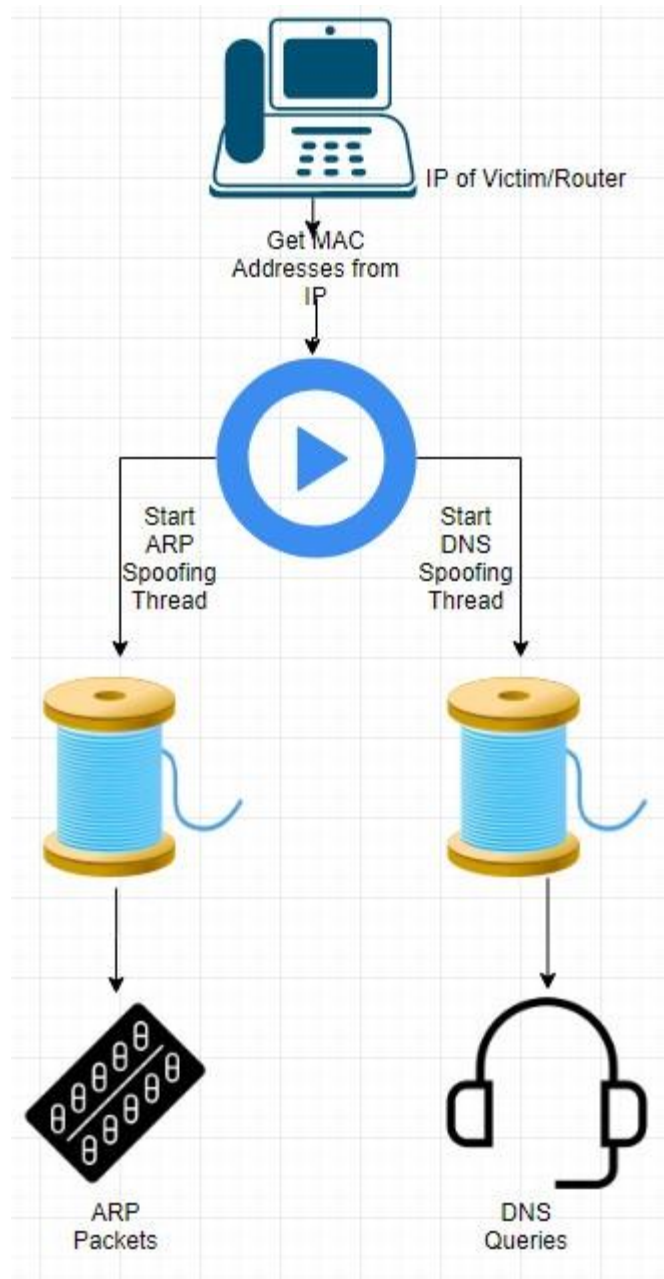
The objective is to learn how DNS spoofing works directly by writing the code. We created a simple DNS and ARP spoof application in Python2 with the goal to redirect the victims web traffic to an IP address that we've specified. All code as well as a how-to quick reference guide are located in the same zip file.

The application starts with obtaining the IP address of the victim, your router and the spoofed website. Due to the nature of the victim PC often receiving a legitimate DNS response before our spoofing application can, we have to add iptables rules. These rules would make it so that any legitimate DNS responses that are being forwarded would be blocked. It would then move forward to crafting ARP packets, sending out spoofed ARP packets and sniffing DNS requests where we then send back our spoofed response.

Design



This is a general overview of how spoofing happens. Here, the user sends a request to a website they want to visit. However, the Attack injects a fake DNS entry. This causes the User to reach the Fake Website instead of the Real Website that they intended on visiting.



This is a brief overview of how the Python application will work. You start by inputting the Victim and Router IP in addition to changing the web server IP if needed in the main.py file. It'll search for the MAC Addresses using those IP addresses. Afterwards two threads will start, one for ARP and the other for DNS. The ARP would be responsible for sending packets to the victim and router. Meanwhile, the DNS would listen for queries and send spoofed responses back. During all this, the attacking machine will sniff all the traffic on port 53.

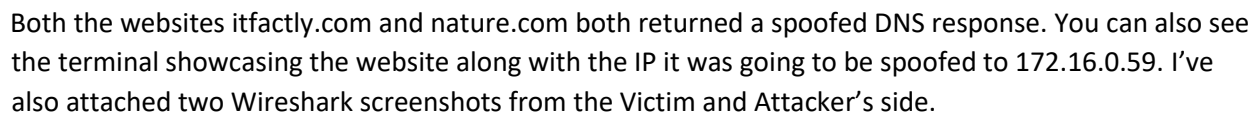
Yoshiaki Ryuzaki
Sukh Atwal

Testing

The victim IP is 172.16.0.60, our router IP is 172.16.0.1 and the website is hosted on our IP 172.16.0.59.
This was all done on Fedora KDE Plasma using two Oracle Virtual Machines.

#	Description	Tools Used	Expectations	Actuality	Pass/Fail
1	Check if Victim is redirected to our locally hosted website	Chrome, Python, Wireshark	The Victim is redirected to our spoofed website when trying to access any website	The Victim is redirected to our spoofed website	Pass
2	Check if spoofing works for arbitrary domains	Wireshark	Spoofed DNS response is sent to any random string that the Victim requests	If the website doesn't exist, it'll be spoofed to our locally hosted website	Pass

Test 1



```

(judcp port == 53 || kcp port == 53)

No.    Time    Source          Destination      Protocol Length Info
-----
5788   12.12144429 172.16.0.1       172.16.0.50     DNS        180 Standard query response 0x67ed A 1 [information] A 172.16.0.50
5789   12.12238428 172.16.0.60       172.16.0.1       DNS        72 Standard query 0xaaf3 AAAA iffracty.com
5790   12.12254723 172.16.0.60       172.16.0.1       DNS        72 Standard query 0xaaf3 AAAA iffracty.com
5791   12.12348430 172.16.0.1       172.16.0.60     DNS        180 Standard query response 0x67ed A iffracty.com A 172.16.0.50
5792   12.12445435 172.16.0.1       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5793   12.12454540 172.16.0.60       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5794   12.12465646 172.16.0.1       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5795   12.124701251 172.16.0.60       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5796   12.12479956 172.16.0.60       172.16.0.1       ICMP       116 Destination unreachable (port unreachable)
5797   12.12483430 172.16.0.60       172.16.0.50     DNS        156 Standard query response 0xaaf3 AAAA iffracty.com SOA ns=820.suondra-59.net
5805   12.138928637 172.16.0.1       172.16.0.60     DNS        180 Standard query response 0xa7a A iffracty.com A 172.16.0.50
5806   12.13900000 172.16.0.1       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5807   12.139277715 172.16.0.60       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5808   12.13934444 172.16.0.1       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5809   12.139449277 172.16.0.60       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5810   12.13950144 172.16.0.60       172.16.0.1       ICMP       128 Destination unreachable (port unreachable)
5811   12.13954243 172.16.0.60       172.16.0.1       ICMP       116 Standard query response 0xaaf3 A iffracty.com A 172.16.0.50

+ Frame 5792: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface epnbl3, id 0
+ Ethernet II, Src: Netgear d8:c7c2 (8d:1e:5e:d5:c7c2), Dst: PcsCompu-dc:7b:ec (08:00:27:dc:7b:ec)
+ Internet Protocol Version 4, Src: 172.16.0.1, Dst: 172.16.0.60
+ User Datagram Protocol, Src Port: 53, Dst Port: 42227
  Source Port: 53
  Destination Port: 42227
  Length: 54
  Checksum: 0xa402 [Unverified]
  Checksum Status: Unverified
  Stream index: 228
  + [Timestamps]

+ Domain Name System (response)
+ [Length: 180]
+ [Export info (Warning/Protocol): DNS response retransmission, original response in frame 5788]
  [DNS response retransmission, original response in frame 5788]
  Severity level: Warning!
  (Group: Protocol)

+ Flags: 0x0100 Standard query response, No error
0000  00 00 27 dc 7b ec 8d 15 5e d5 c7 c2 08 00 45 00  ...f...A...E
0010  00 4a 00 00 00 40 00 41 12 e5 ac 10 00 01 ac 10  ..J..0..0..
0020  00 3c 00 25 a4 f2 00 30 a4 52 0000 00 00 00 00 02  ...5..0..0...
0030  00 01 00 00 00 00 00 00 69 74 66 53 03 74 6c 79 63  ....1fracty...
0040  00 6f 6d 00 00 01 00 02 c0 9c 98 01 00 95 00 00  com.....
0050  00 3c 00 00 00 00 00 00


```

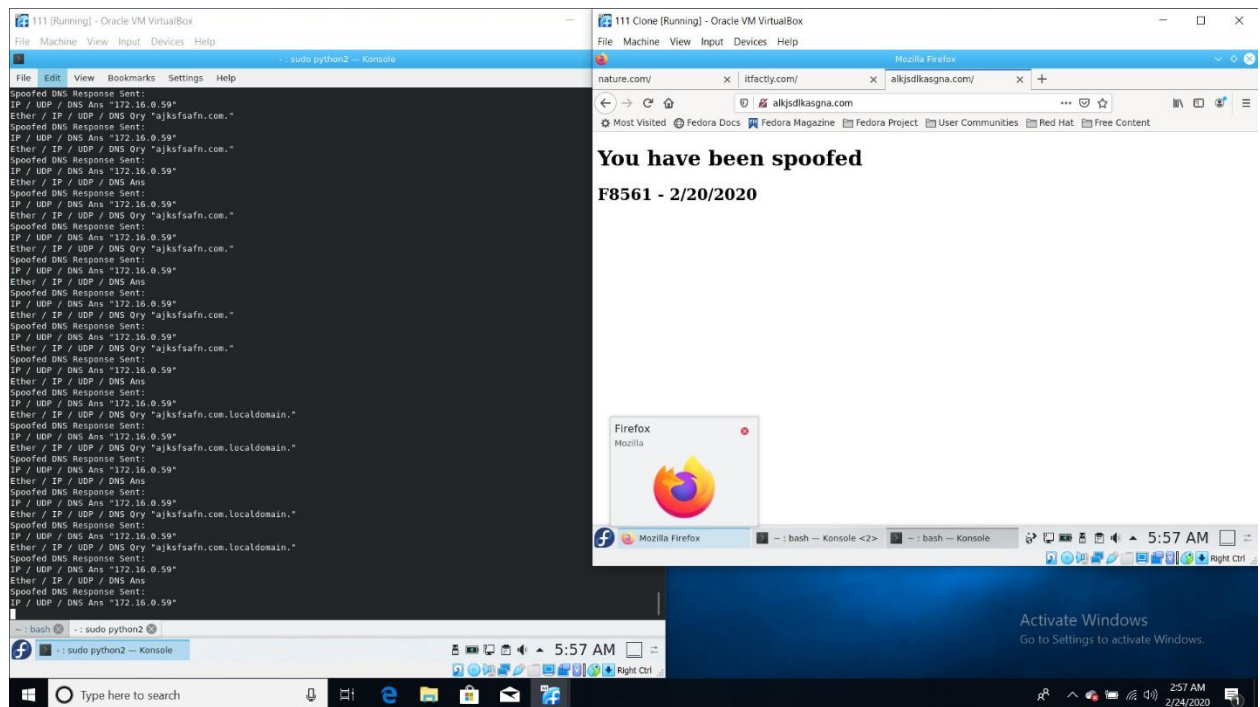
Victim:

Yoshiaki Ryuzaki
Sukh Atwal

[illegible]

Yoshiaki Ryuzaki
Sukh Atwal

Test 2



This showcases it's able to handle any arbitrary(random) domain strings.