|  |
| --- |
|  |
| COMP8505 Assignment 4 |
| DNS Spoofer |

|  |
| --- |
| Nishan Vivekanandan & Adam Harrison  11-9-2017 |

# Objective

# To design a basic DNS spoofing application using any language of your choice.

# Design

### Diagrams



Figure 1 - Conceptual MITM Diagram



Figure 2 - System Diagram



Figure 3 - State Diagram

## Pseudocode

**Program: poison**

READ routerIp, targetIP from user

routerMac = CALL GetMacFromIp(routerIp)

targetMac = CALL GetMacFromIp(targetMac)

DO

//Poison the targets and router ARP cache

CALL sendPoisonArp(targetIp, routerIp, targetMac)

CALL sendPoisonArp(routerIp, targetIp, routerMac)

UNITL program terminated

**Program: spoofer**

READ attackerIp from user

CREATE NetFilterQueue to filter DNS packets as pkt

DO

IF pkt is a DNS query

pkt CraftSpoofedDnsResponse(pkt)

END IF

ACCEPT pkt

UNTIL program terminated

## Implementation

The project was implemented in Python using the Scapy and NetFilterQueue libraries. It is comprised of two main components one for ARP poisoning and the other to reply with spoofed DNS packets. There is also a utility class to house common functions.

­­­­poison.py performs the ARP cache poisoning of the intended target system allowing the host system to intercept all requests going to the gateway as a MITM. It takes the target IP and gateway IP as arguments and then proceeds poison the targets ARP table such that the host systems MAC is associated with the gateway IP and the gateway ARP table such that the host systems MAC is associated with the target IP. These ARP packets are sent continually to ensure the ARP table is poisoned for the duration of the DNS spoofing.

spoof.py performs the DNS spoofing of the targets system DNS requests which are now directed to our host via the ARP cache poisoning. It modifies the Iptables of the host system such that UDP packets sent to port 53 are sent to a custom NetFilterQueue which in turn calls back to our python script allowing us to work with the packet. If the packet is a DNS request it is modified with a DNS response returning the IP of the host system. For this proof of concept all DNS requests are grabbed.

# Program Instructions

This project was written and tested on Fedora 25 64bit X86 workstations. Using Python 2.7.13, Scapy 2.3.3 and NetfilterQueue 0.8.1. Please ensure these requirements are met before attempting to run.

Extract the submitted zip file on the attacker machine and navigate to the “App” folder. on the command line. Launch the python scripts with the commands given below each in its own terminal session. Start the poison.py script first then the spoof.py script. Ensure you run as root. The utils.py file must also be present in the working directory as it holds shared functions.

“python poison.py -rip GATEWAY\_IP -tip TARGET\_IP”

“python spoof.py -aip ATTACKER\_IP ”

For example:

Target IP = 192.168.0.9

Attacker IP = 192.168.0.10

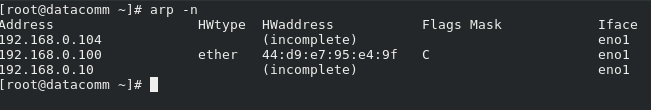
Gateway IP = 192.168.0.100

“python poison.py –rip 192.168.0.100 –tip 192.168.0.9”

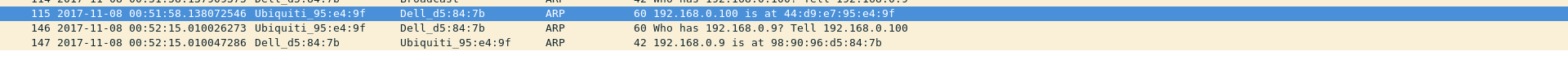
“python spoof.py -aip 192.168.0.10”

# Tests & Screen shots

Original ARP cache and packets:

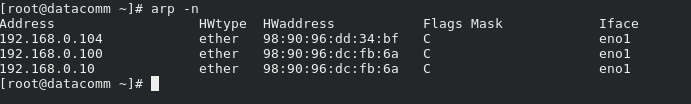


ScreenCap 1 - Original ARP Cache

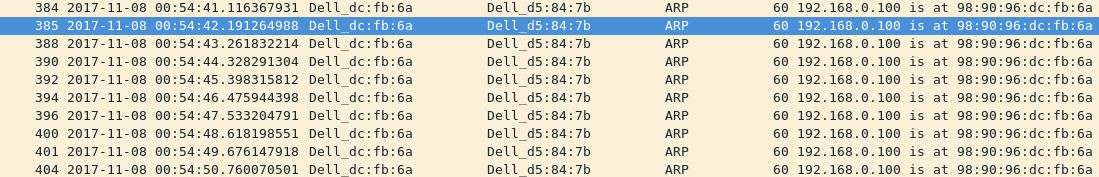


ScreenCap 2- Original ARP Packet Capture

Poisoned ARP cache and packets:

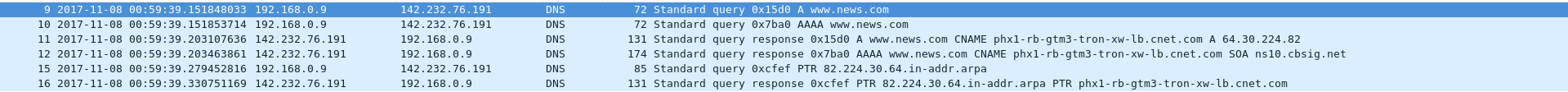


ScreenCap 3 - Poisoned ARP Cache

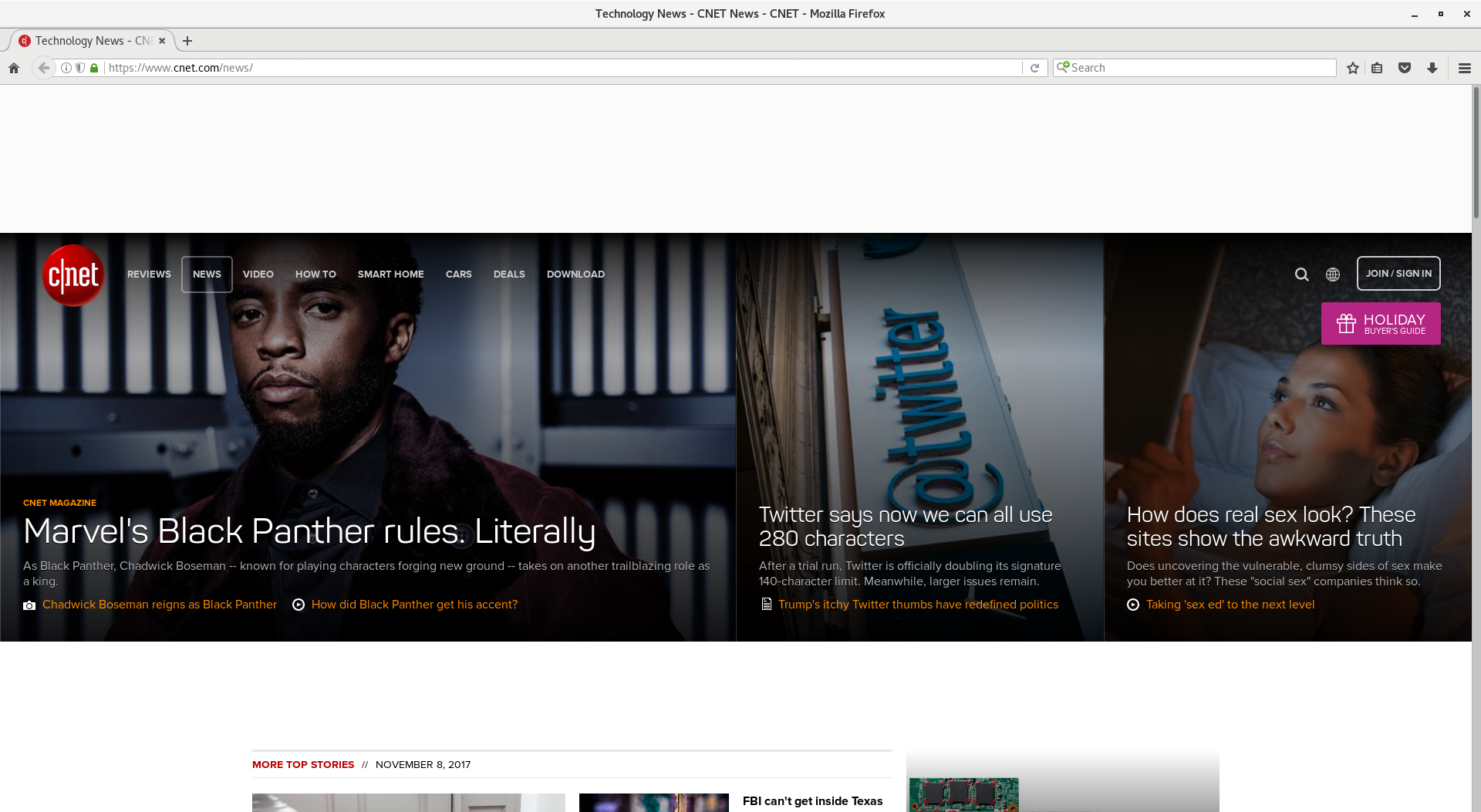


ScreenCap 4 - Poisoned ARP Packet Capture

Original DNS request, response and webpage:

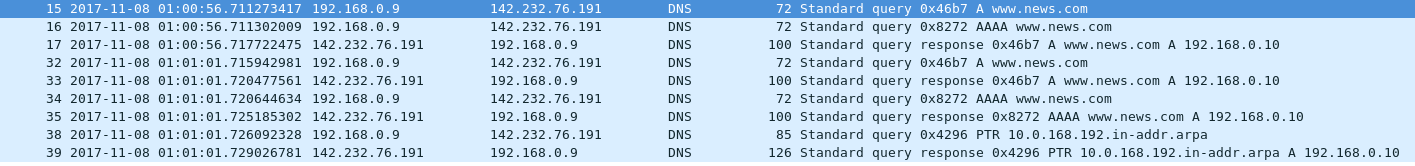


ScreenCap 5 - Original DNS Packet Captures

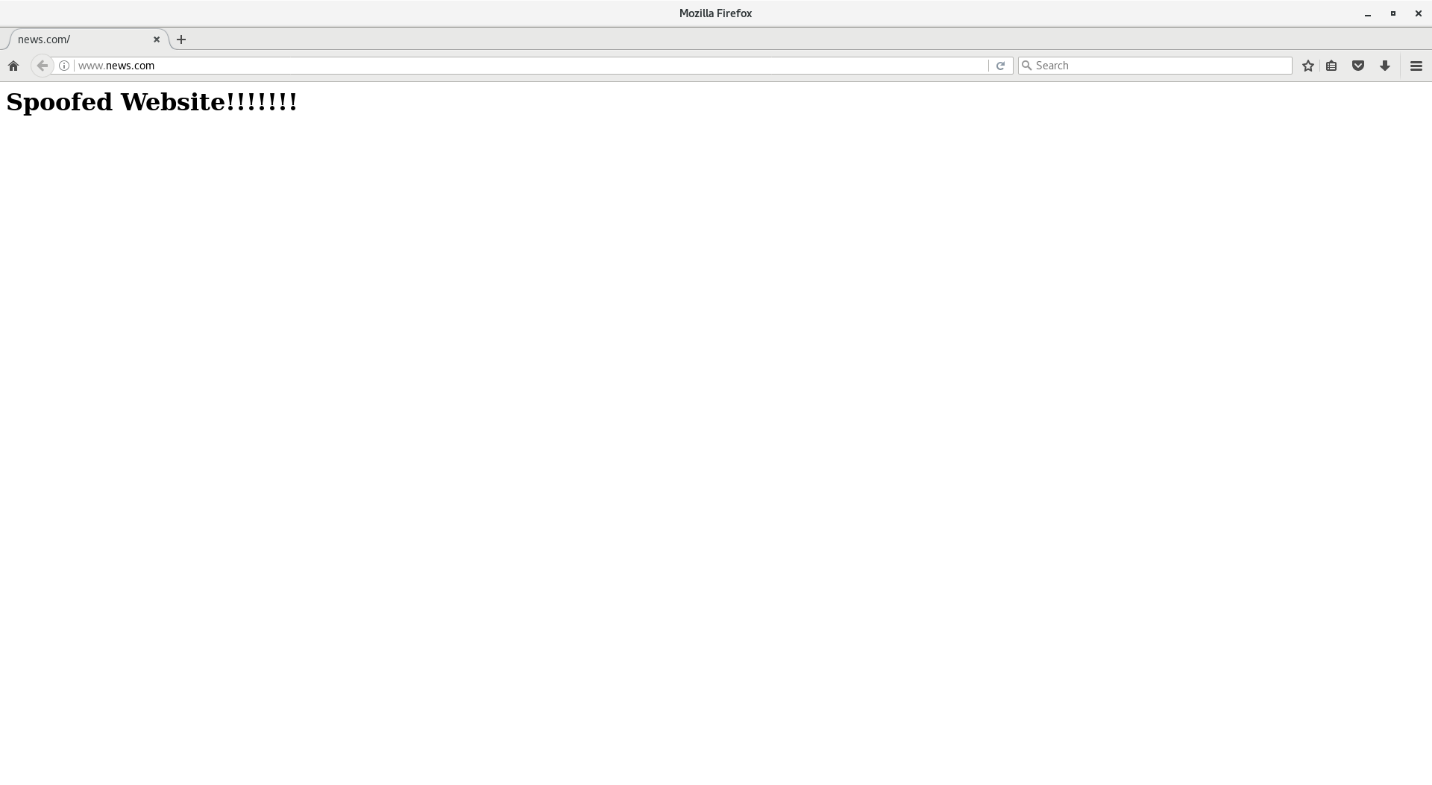


ScreenCap 6- Original Webpage

Spoofed DNS request, response and webpage:



ScreenCap 7- Spoofed DNS Packet Capture



ScreenCap 8 - Spoofed Webpage