Assignment 4

COMP 8506 – Arp poisoning & dns spoofing

Mao Sato - Shahin Nikvand

2016

Table of Contents

[Overview 2](#_Toc453596451)

[Design 2](#_Toc453596452)

[Constraints 2](#_Toc453596453)

[Pseudo Code 2](#_Toc453596454)

[Flow Chart 4](#_Toc453596455)

[Configuration 5](#_Toc453596456)

[How to Run 5](#_Toc453596457)

[Testing 6](#_Toc453596458)

[Summary 9](#_Toc453596459)

# Overview

This document details our understanding of the networking protocols, their weaknesses as well as their strengths. As a requirement of the assignment we’ve created an application that will craft packets to poison the Arp Cache of a target machine (and the firewall) then direct them to a fake website by capturing their DNS request packets and crafting fake responses.

Strengthening our understanding of all networking protocols such as TCP, UDP, ARP, DNS, and other general communication will help creating better defences against such attacks.

# Design

## Constraints

The application was developed with the following constraints in mind:

* Application must sense HTML DNS Query’s and respond with a crafted packet to redirect the target computer.
* Required to Handle any domain name and craft a spoofed response

We carried out this attack in the Data communications lab using Python 2.7. We used Arp poisoning to capture the DNS Query packet then scapy to respond with.

## Pseudo Code

Main()

Create process for Arpspoof(), Dnsspoof(), and Webserver()

Start processes and join processes

Arpspoof()

Create ARP packet for target machine

opcode = who has, src mac = sender’s MAC, src ip = router’s IP, dest ip = target’s ip

Create ARP packet for router

opcode = who has, src mac = sender’s MAC, src ip = target’s IP, dest ip = router’s ip

send both pakets every 1 second

Dnsspoof()

Capture DNS packet whose source IP is target’s IP and asking for a specific domain

Pkt = captured DNS packet

Create DNS packet to send

IP(Src ip = pkt.dist ip, dist ip = pkt.src ip)/

UDP(dest port = pkt.src port, src port=pkt.dest port)

DNS(id = pkt.id, pr = 1, qd = pkt.qd, an=

DNSRR(rrname=pkt.qd, ttl = 10, rdata = sender’s IP)

Send DNS packet

Webserver()

Run TCP server on port 80 with HTTP request handler

Show index.html on root directory

Handler()

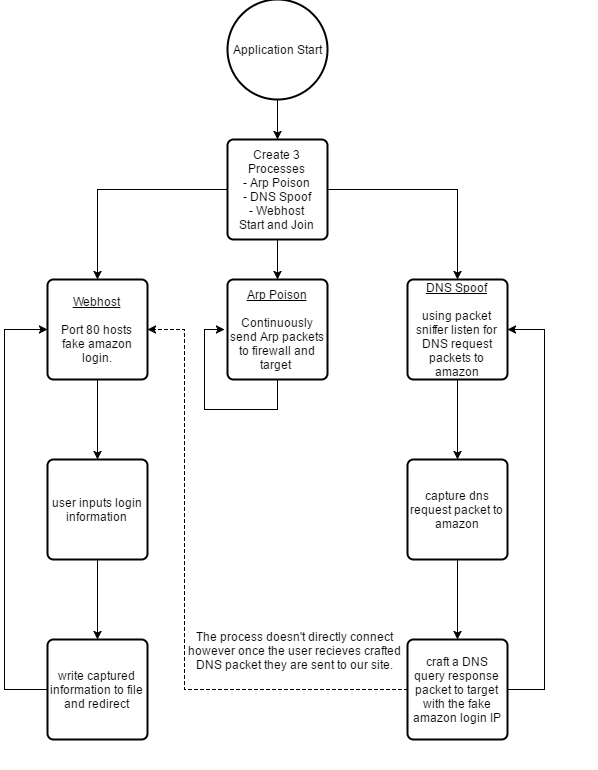
Get POST request

Open passwords.txt with option append

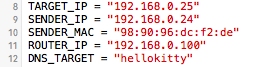
Write all variables with sent by POST request

## Flow Chart

The flow chart below shows how our application approaches the problem. While it’s not fast or efficient as a proof of concept it works quite well. One of the things we plan to do is down the road possibly attempt to implement something like this in a C and Assembly hybrid environment as speed is extremely important for delivering the response before the firewall.



# Configuration

1. Open the dns\_spoofer.py
2. Go to Line 8 – 12 and you should see the following lines:  
   
3. Fill in the variables as needed (Look below for the explanation)
   1. TARGET\_IP is the IP address of the machine you are targeting and would like to Arp poison and DNS spoof. You can either ping the machine and run the “arp –a” command or a tool like nmap.
   2. SENDER\_IP and \_MAC is your IP and MAC address. Can be retrieved through “/sbin/ipconfig” or “ip addr”
   3. ROUTER\_IP is IP address of the firewall to poison.
   4. DNS\_TARGET is a string which contains a portion of the DNS name you’re targeting. In the image above we’re targeting any DNS queries with “hellokitty” in it.
4. Create a web page called “index.html” with all the necessary files to (including CSS and javascript) for the web server to begin emulating the fake login page.
5. Save the file and execute the application.   
   **NOTE:** upon running the application the attack will begin. Only run it when you’re ready.

# How to Run

Before running the application make sure to have scapy installed onto the machine as well as python 2.7.x. To run the application just type “python dns\_spoofer.py”. the application will immediately begin arp poisoning and DNS spoofing.

Any captures passwords and usernames will be written to a file.

# Testing

Testing environment

Target’s IP = 192.168.0.25

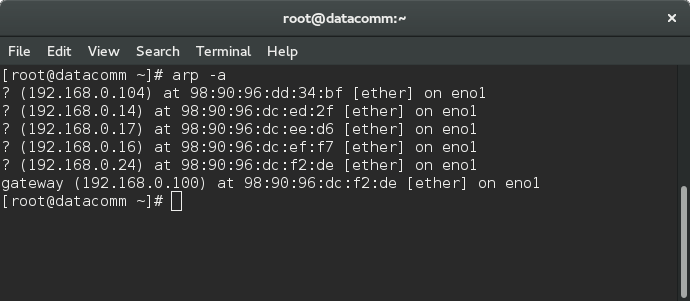
Sender’s IP = 192.168.0.24

Sender’s Mac address = 98:90:96:dc:f2:de

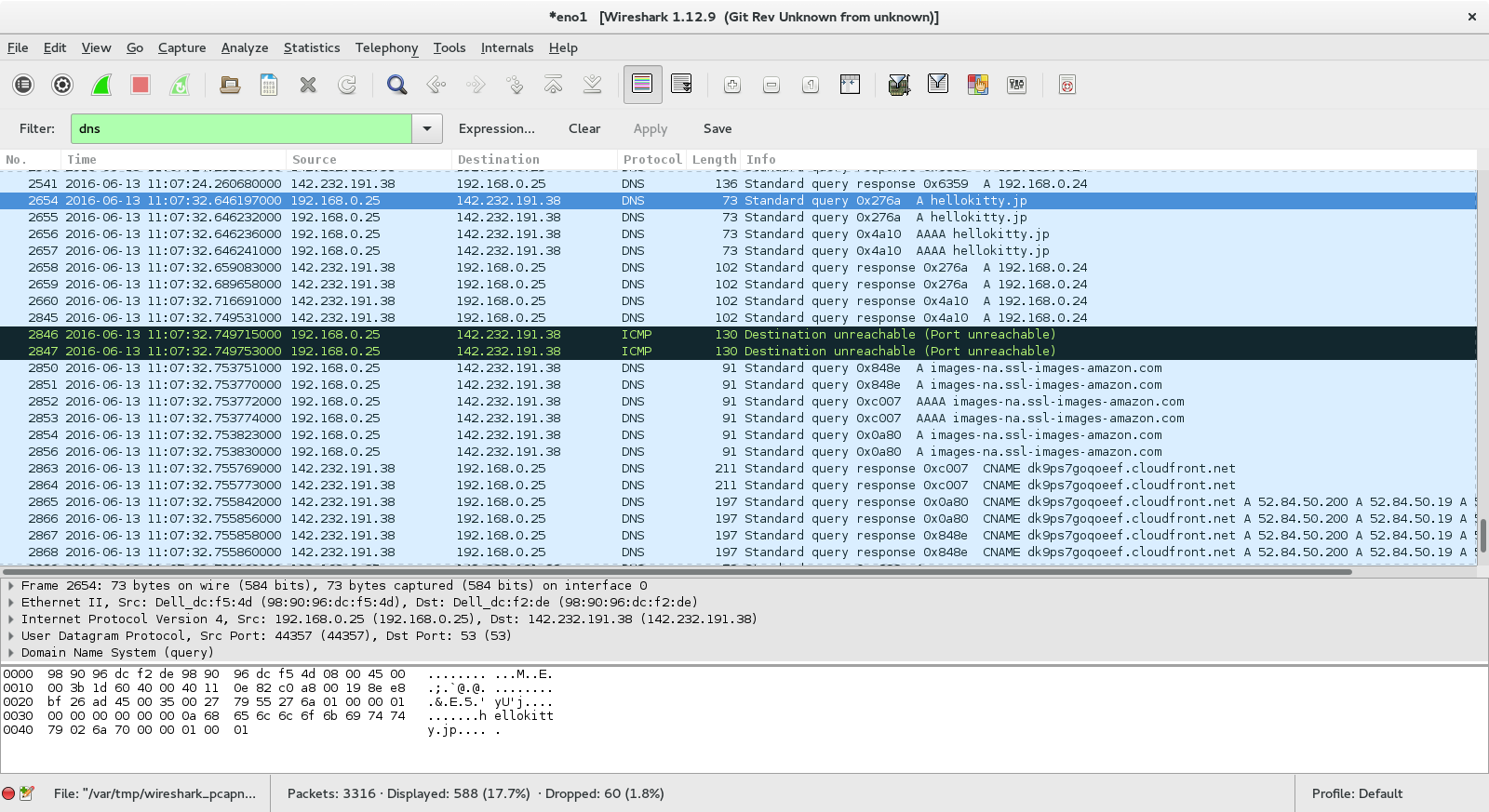
|  |  |  |  |
| --- | --- | --- | --- |
| ID | Purpose | Result | Pass / Fail |
| 1 | The sender changes the target’s ARP table after program starts | ARP table is modified and on gateway’s MAC address is changed to Sender’s MAC address on target. | Pass |
| 2 | For DNS query sent from the target, the sender sends back a fake DNS response packet. | DNS response packet is crafted and sent | Pass |
| 3 | The target accesses to sender’s web server when it receives a fake DNS packet | When target machine tries to access a website contains “hellokitty” in the URL, a fake sign in page shows up | Pass |
| 4 | The sender can get some information from the target when the target input something in input fields | Email and password are saved to passwords.txt when target enters them and click sign in | Pass |
| 5 | The target can access to any website normally except websites contain “hellokitty” in the URL | The target can access “amazon.ca”normally | Pass |

Test 1

arp –a command on the target

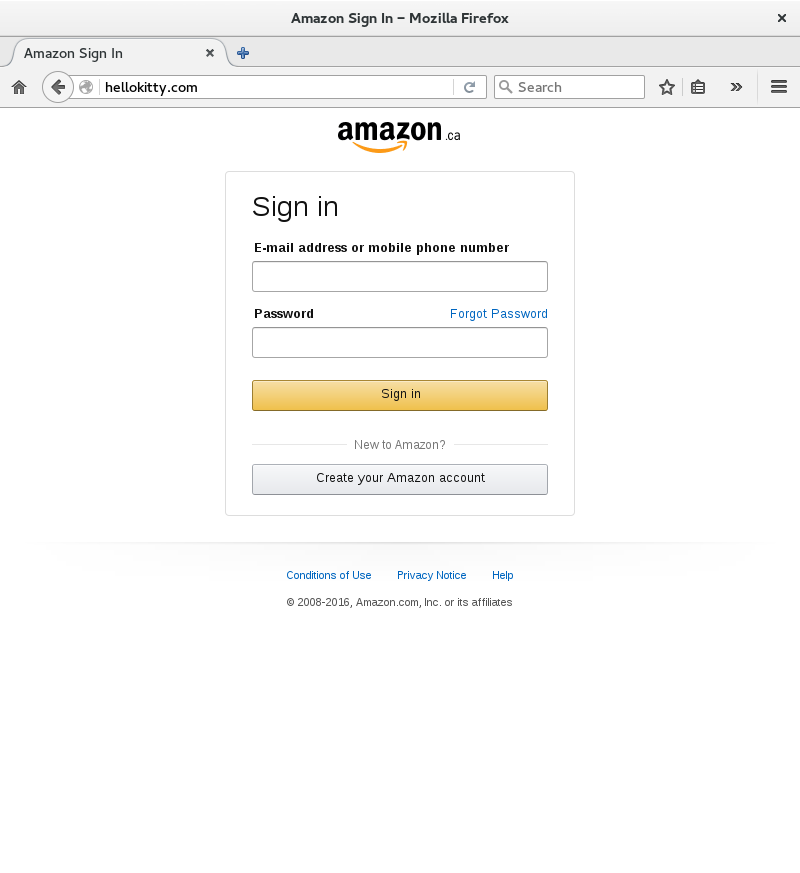


Test 2

Wireshark capture on the sender

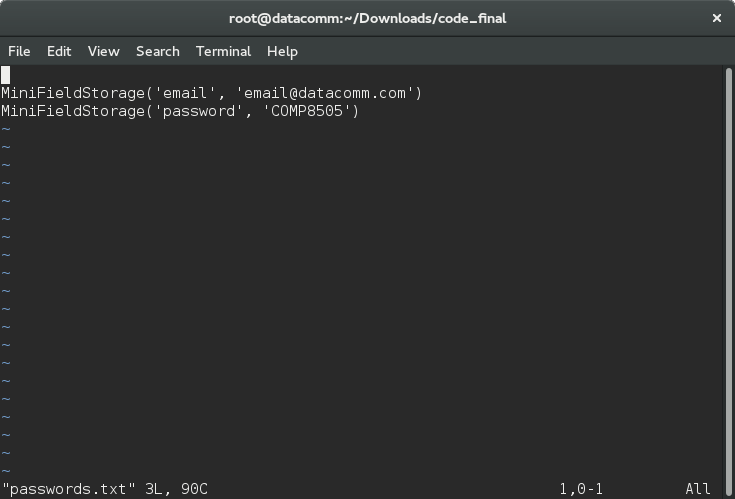
DNS query is sent from the client on No.2654 and the response is sent from the sender on No.2658

Test 3



Test 4

Password.txt file on sender



Test 5



# Summary

While our dns spoofer is quite bare bone this implementation was designed to be implemented in our backdoor. Doing this assignment helped us solidify our understanding of modern day protocols that are used for caching and serving web pages; in addition, it also showed us how easily almost anything can be exploited with a little bit of creativity and recon.