Assignment #4 DNS Spoofing

Mario Enriquez

British Columbia Institute of Technology

COMP 8505, 7D

Aman Abdulla

November 14, 2016

Contents

[Introduction 3](#_Toc466846825)

[How to Run 3](#_Toc466846826)

[Design Work 3](#_Toc466846827)

[dns-spoof.rb 3](#_Toc466846828)

[Test Cases 4](#_Toc466846829)

[1. Test 1 5](#_Toc466846830)

[2. Test 2 8](#_Toc466846831)

[3. Test 3 11](#_Toc466846832)

[4. Test 4 13](#_Toc466846833)

[5. Test 5 14](#_Toc466846834)

[Observations 17](#_Toc466846835)

[Pseudocode 17](#_Toc466846836)

[Dns-spoof.rb 17](#_Toc466846837)

**Introduction**

For this assignment, I had to develop a DNS spoof program that: first, would perform ARP poisoning on the router and victim machines; and afterwards filters DNS queries from the victim and sends a spoofed packet to the victim to redirect the machine to another site.

Th main objective for this assignment is to understand the basics of ARP poisoning and DNS spoofing, as well as understanding how DNS queries and responses are crafted and the elements that they contain.

**How to Run**

Requisites:

* Ruby
* Packetfu gem
* Pcaprub gem

To Run:

* Ruby dns\_spoofv3.rb [router\_mac] [router\_ip] [victim\_mac] [victim\_ip] [redirect\_ip]

Note:

Uncomment ip tables to test program with ip tables.

**Design Work**

dns-spoof.rb



**Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Scenario** | **Tools** | **Expected Results** | **Actual Results** |
| **1** | Test the ARP poisoning | Ruby, Fedora, dns-spoof | Success we can see the arp poisoning working | Success the victim machine believes the router is the host |
| **2** | Test spoofing using ip tables to host httpd | Ruby, Fedora, dns-spoof | Success it redirects with no issue | Success redirects most websites to our hosted site |
| **3** | Test spoofing without ip tables to host httpd | Ruby, Fedora, dns-spoof | Success it redirects to some sites | Partly success, but ruby is not able to keep with most regular responses |
| **4** | Test spoofing using ip tables to an ip | Ruby, Fedora, dns-spoof | Success it redirects with no issue | Success redirects most websites to our hosted site |
| **5** | Test spoofing without ip tables to host httpd | Ruby, Fedora, dns-spoof | Success it redirects to some sites | Partly success, but ruby is not able to keep with most regular responses |

1. Test 1

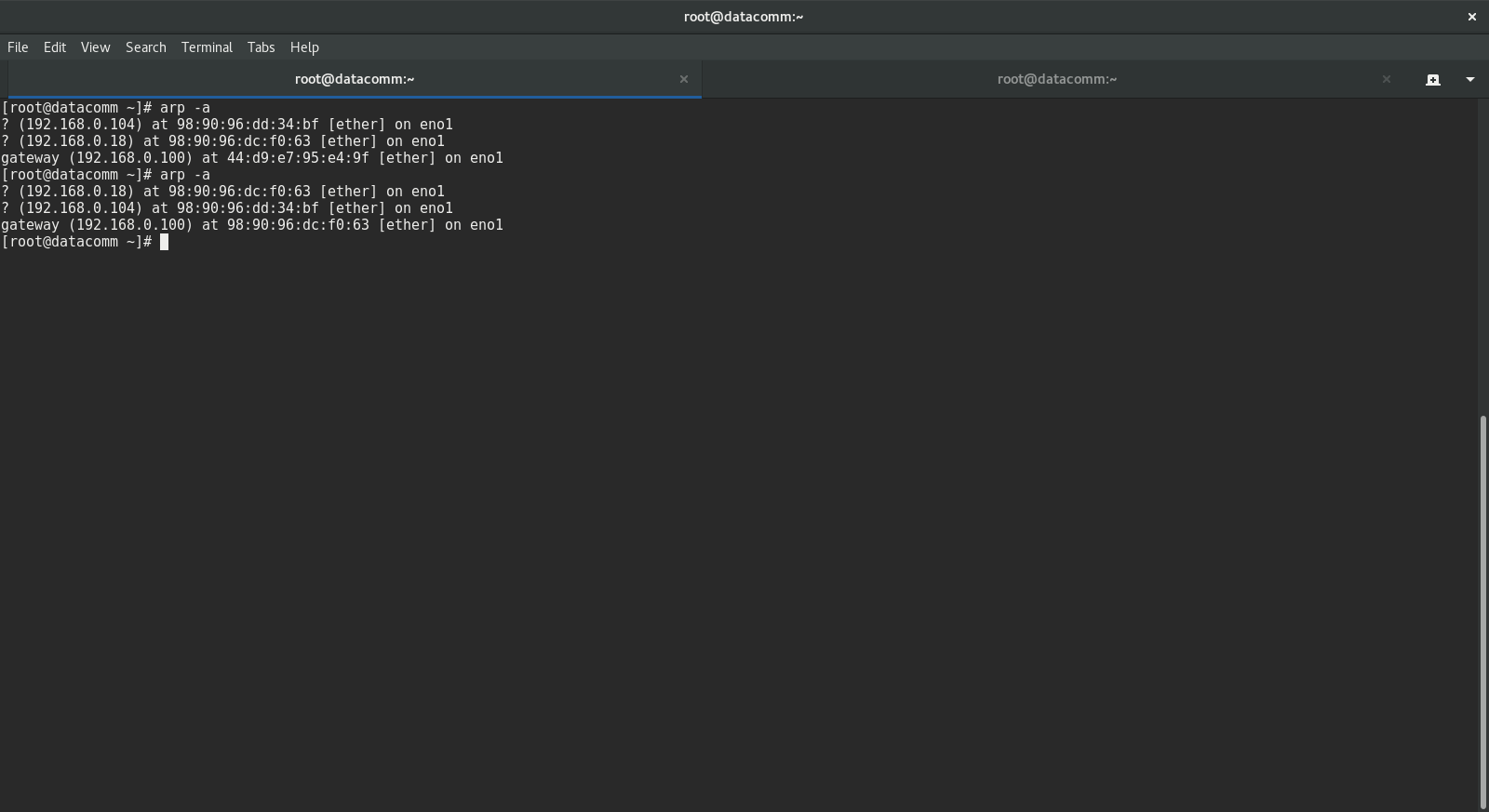
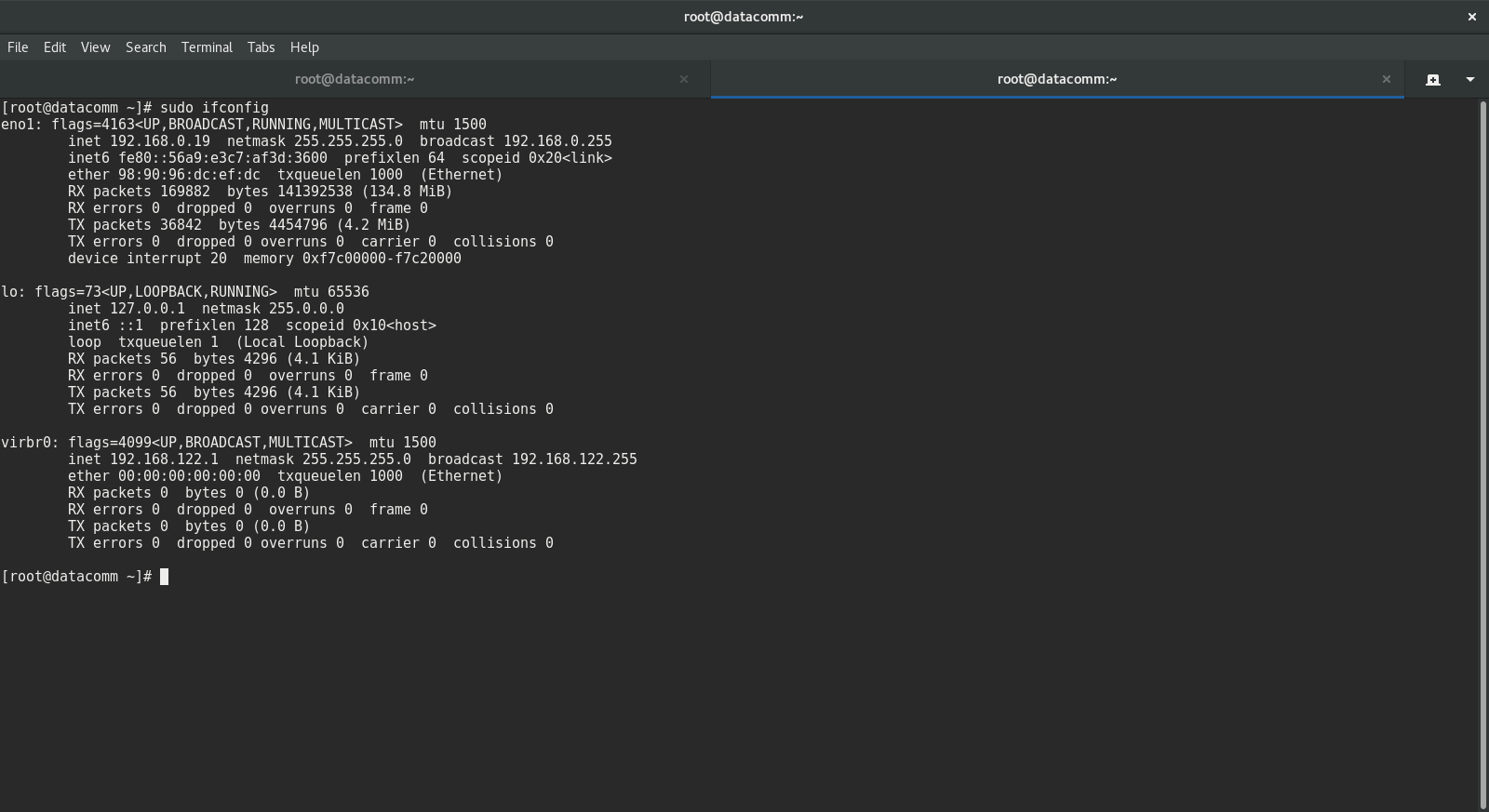
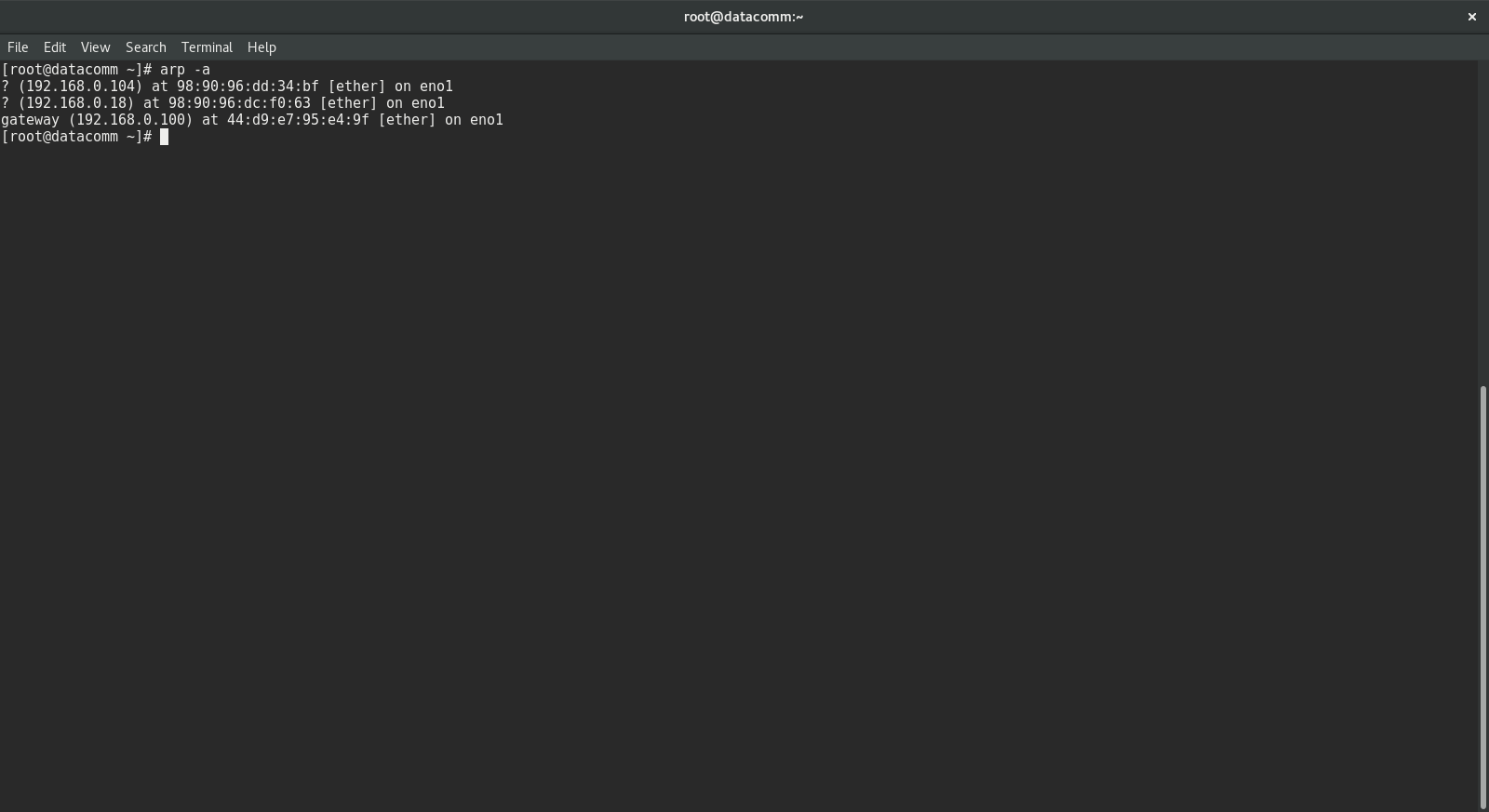
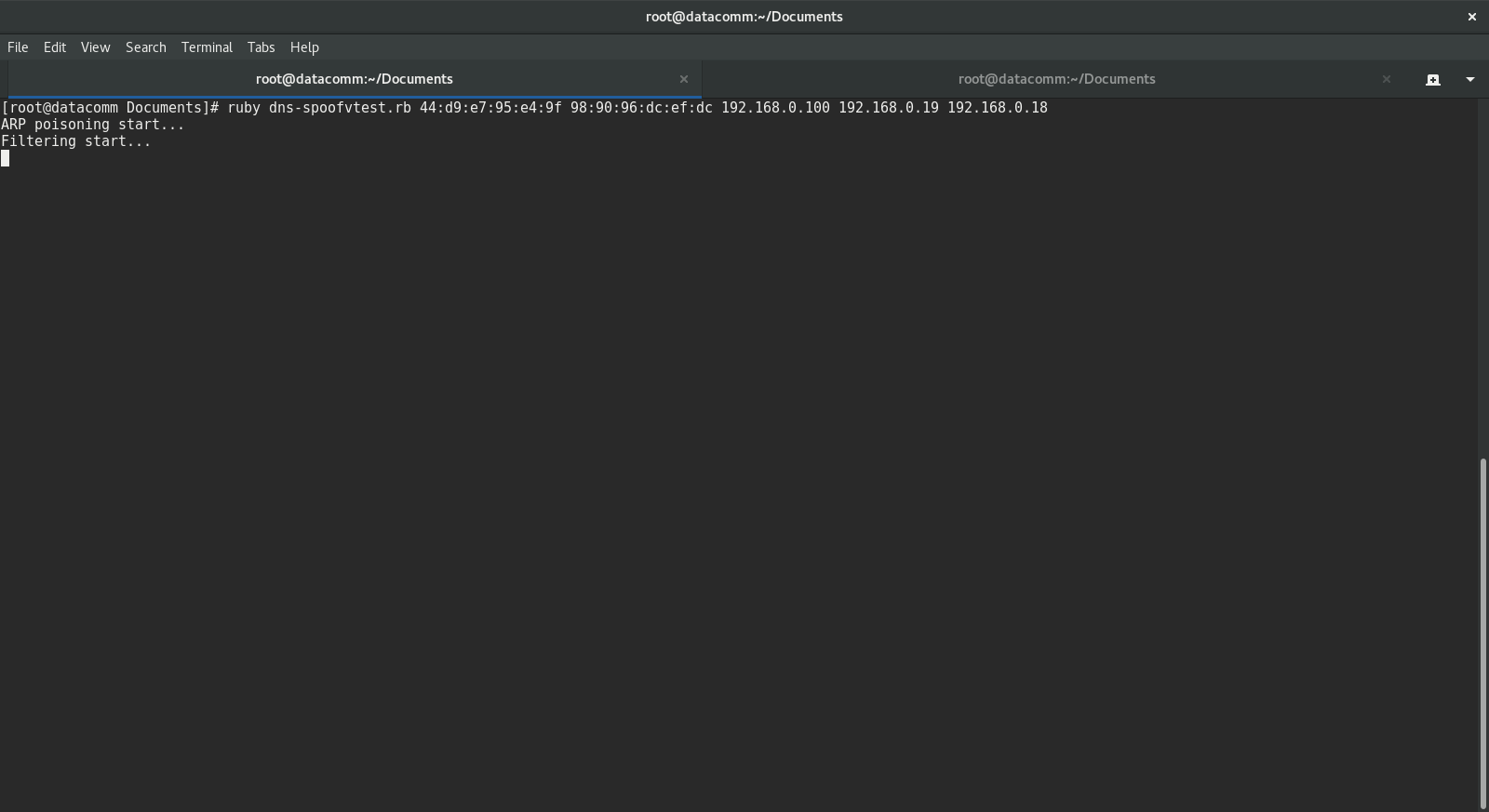


Figure 1 Victim’s machine



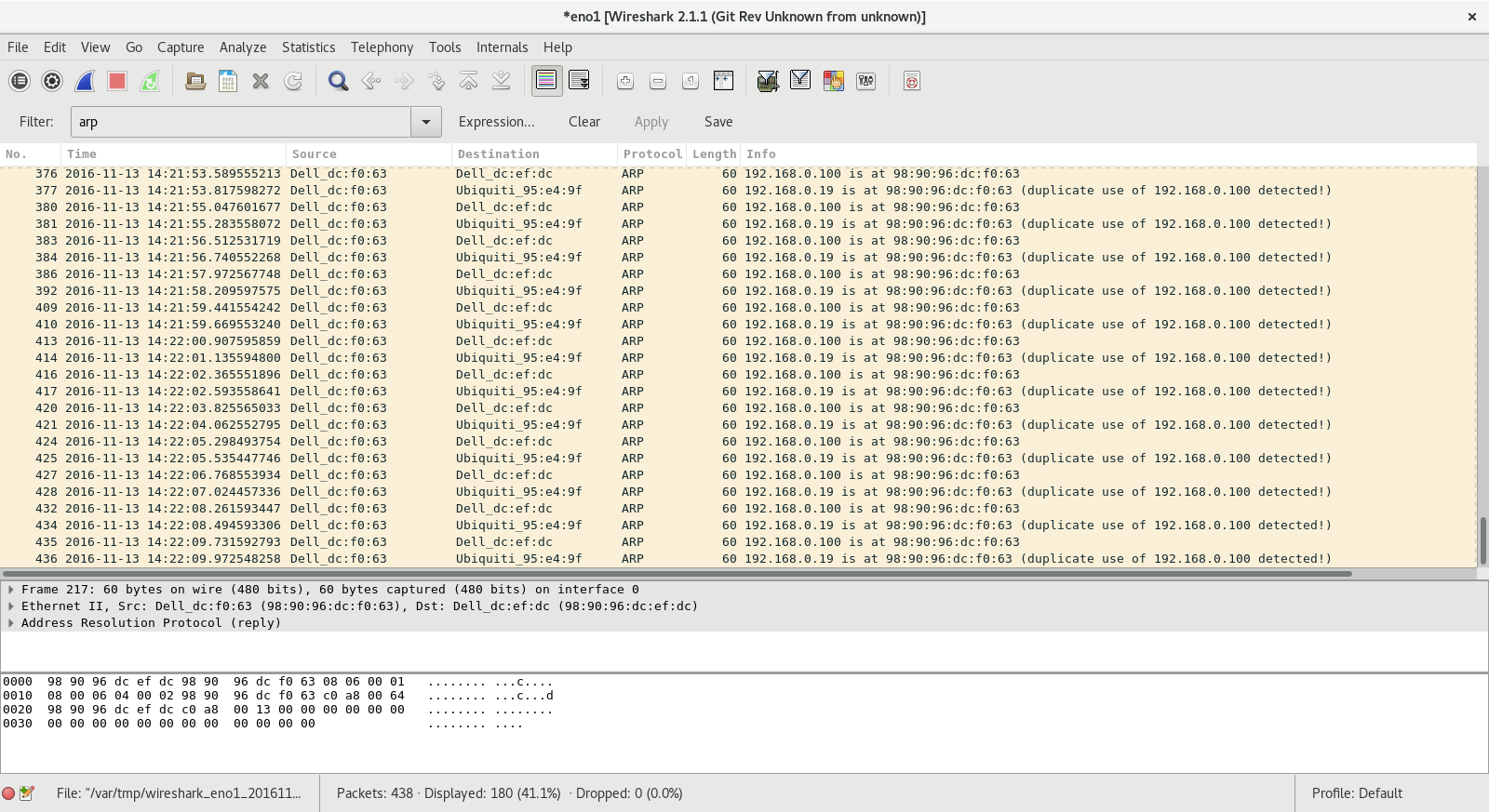


Figure 2 Host Machine

1. Test 2

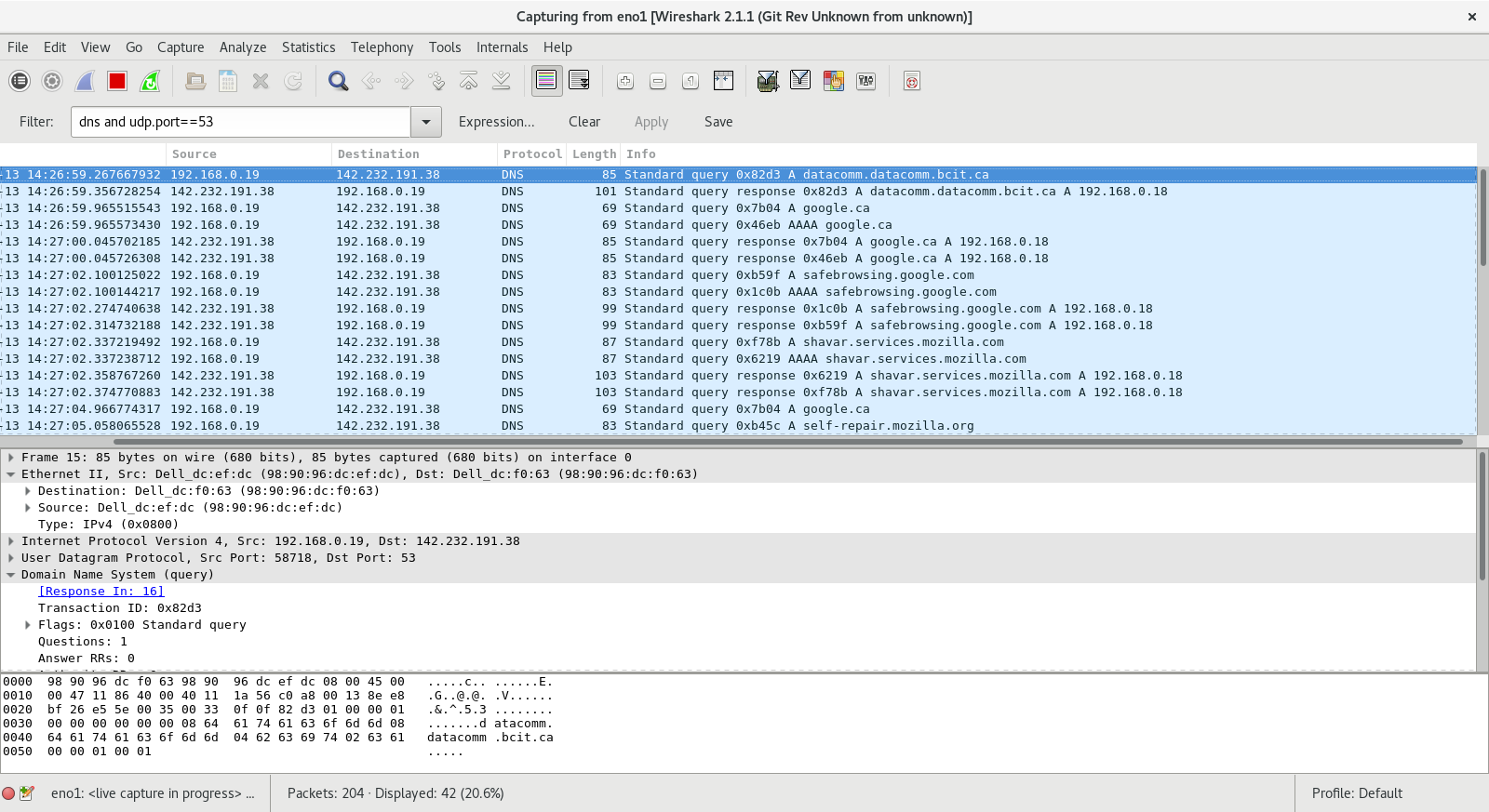
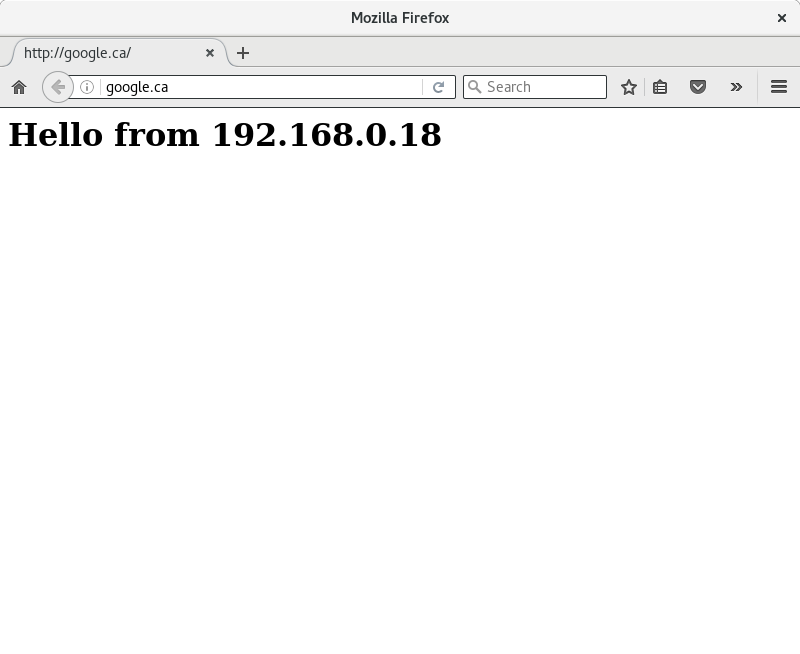
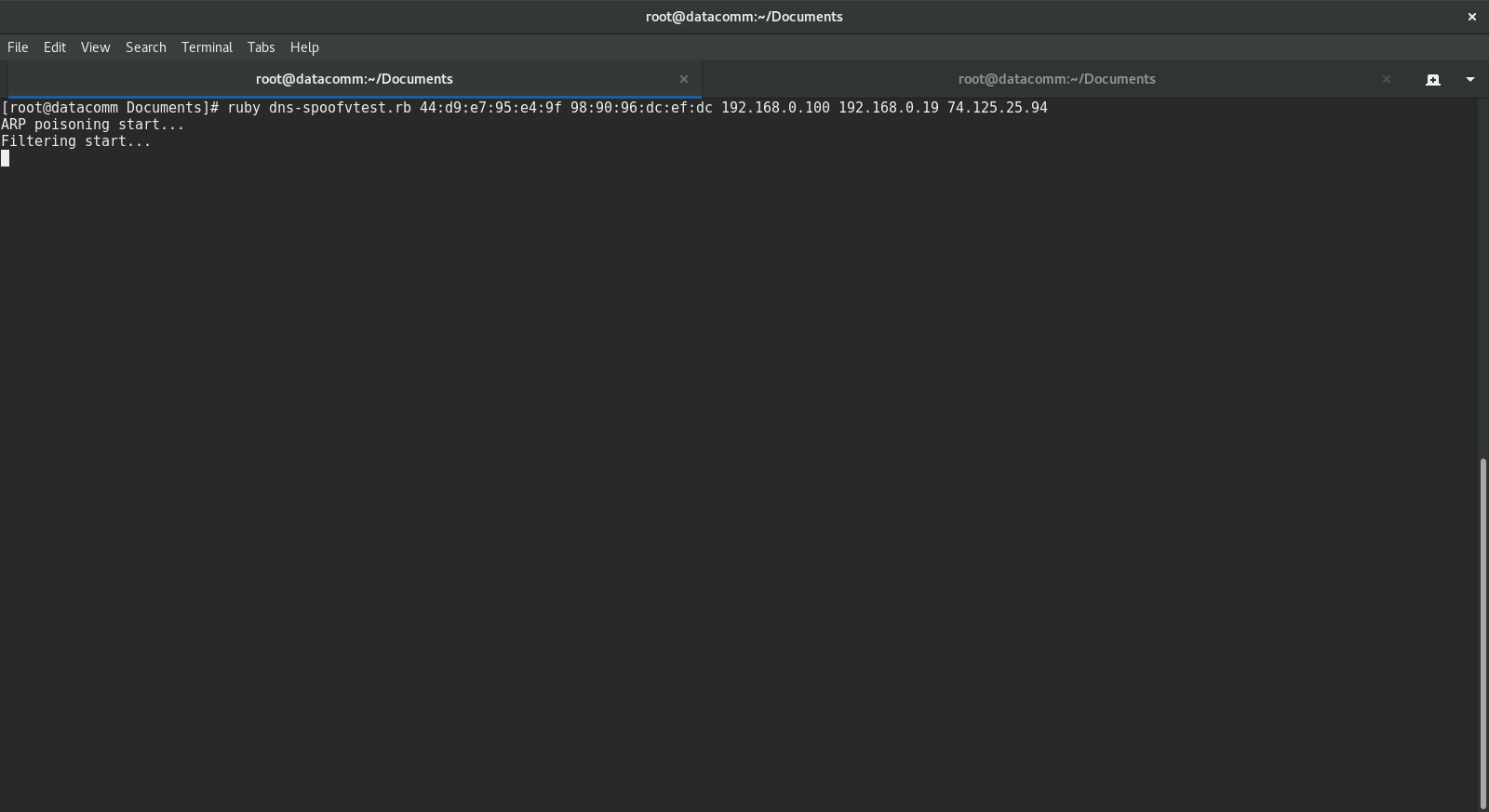
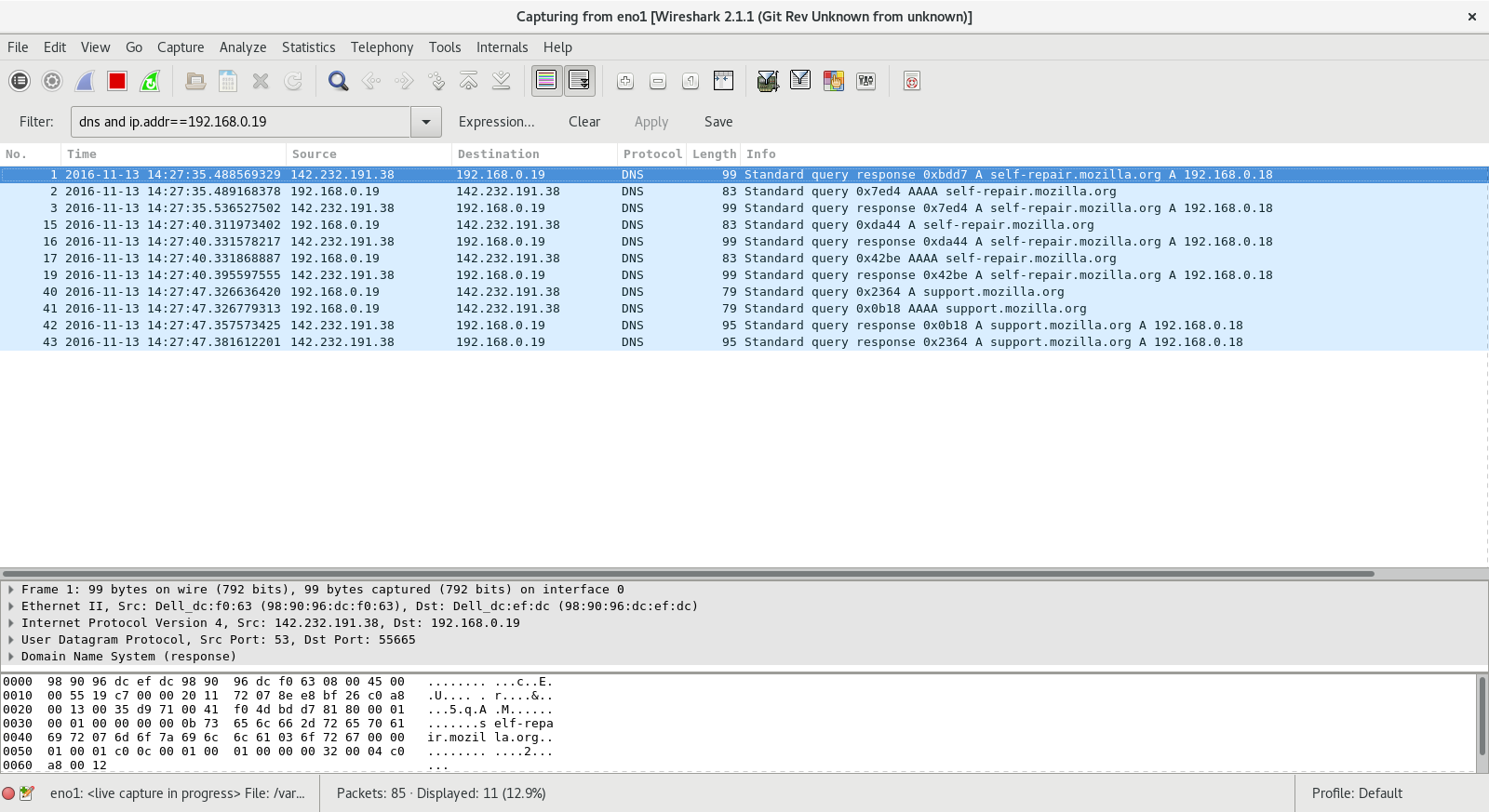


Figure 3 Victim Machine





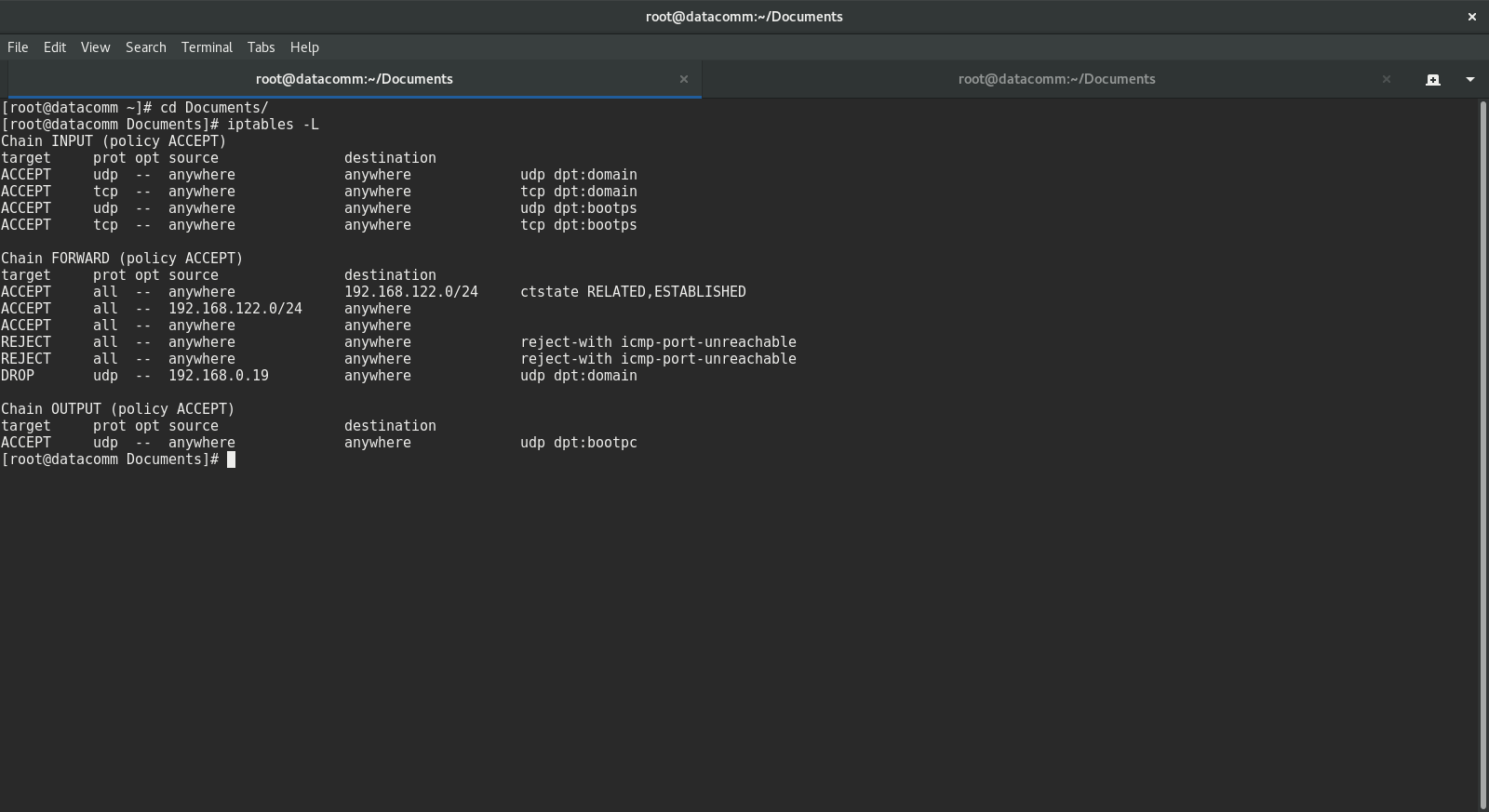


Figure 4 Host Machine

1. Test 3

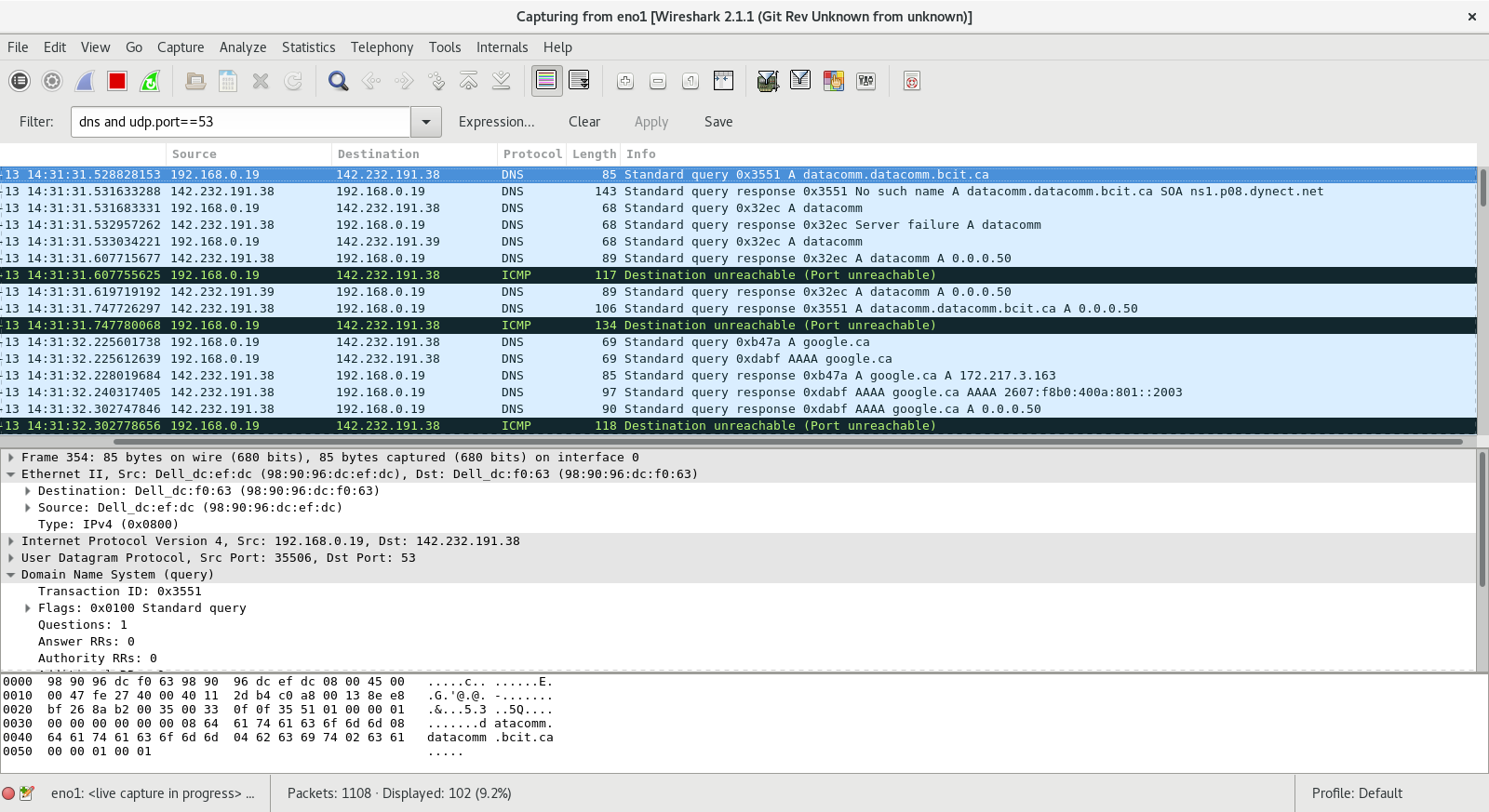
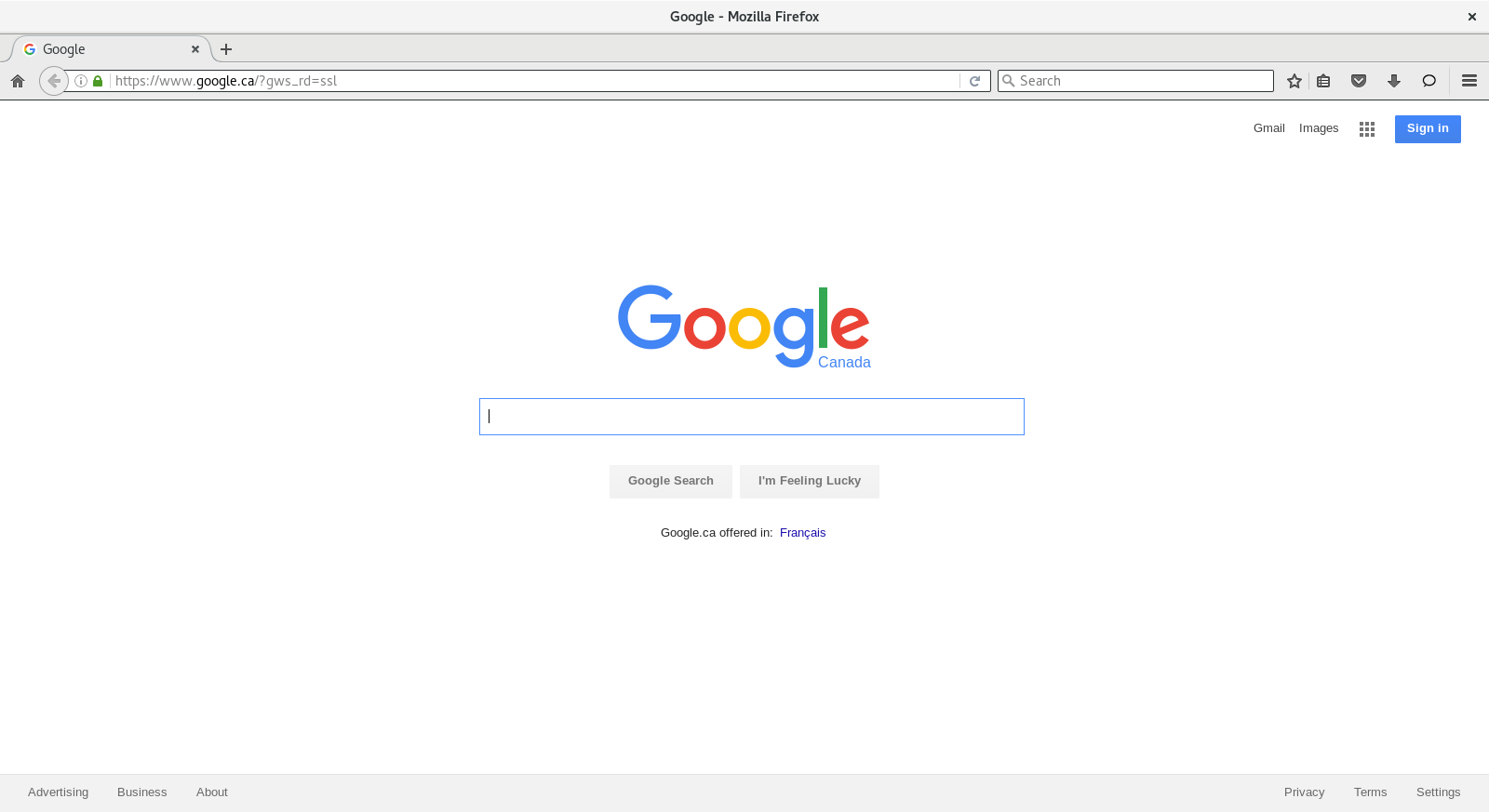
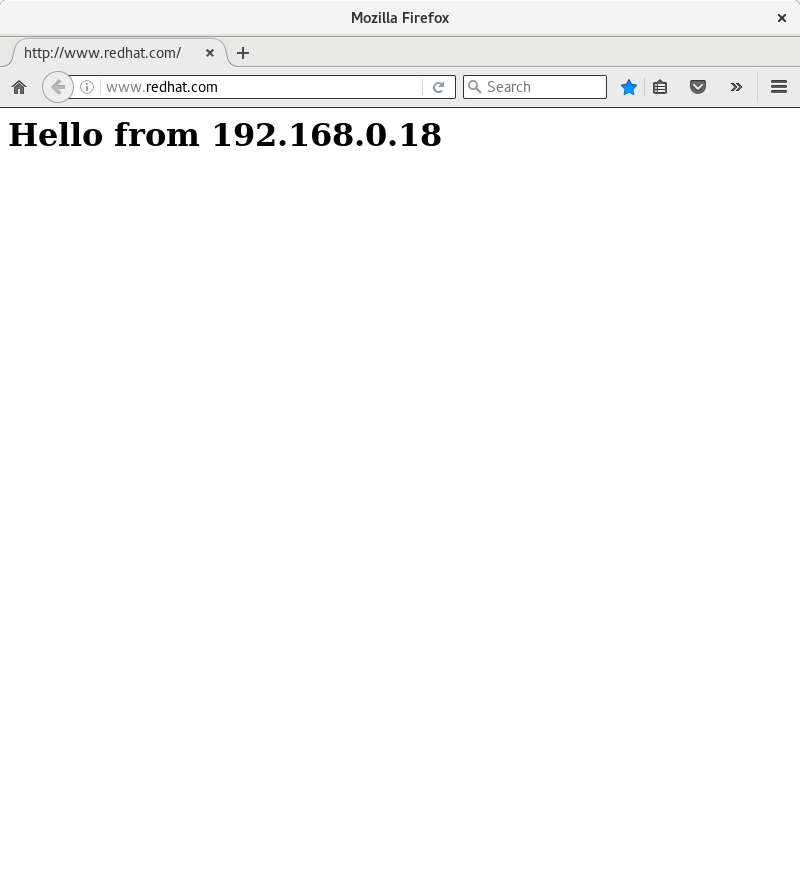


Figure 5 Victim Machine



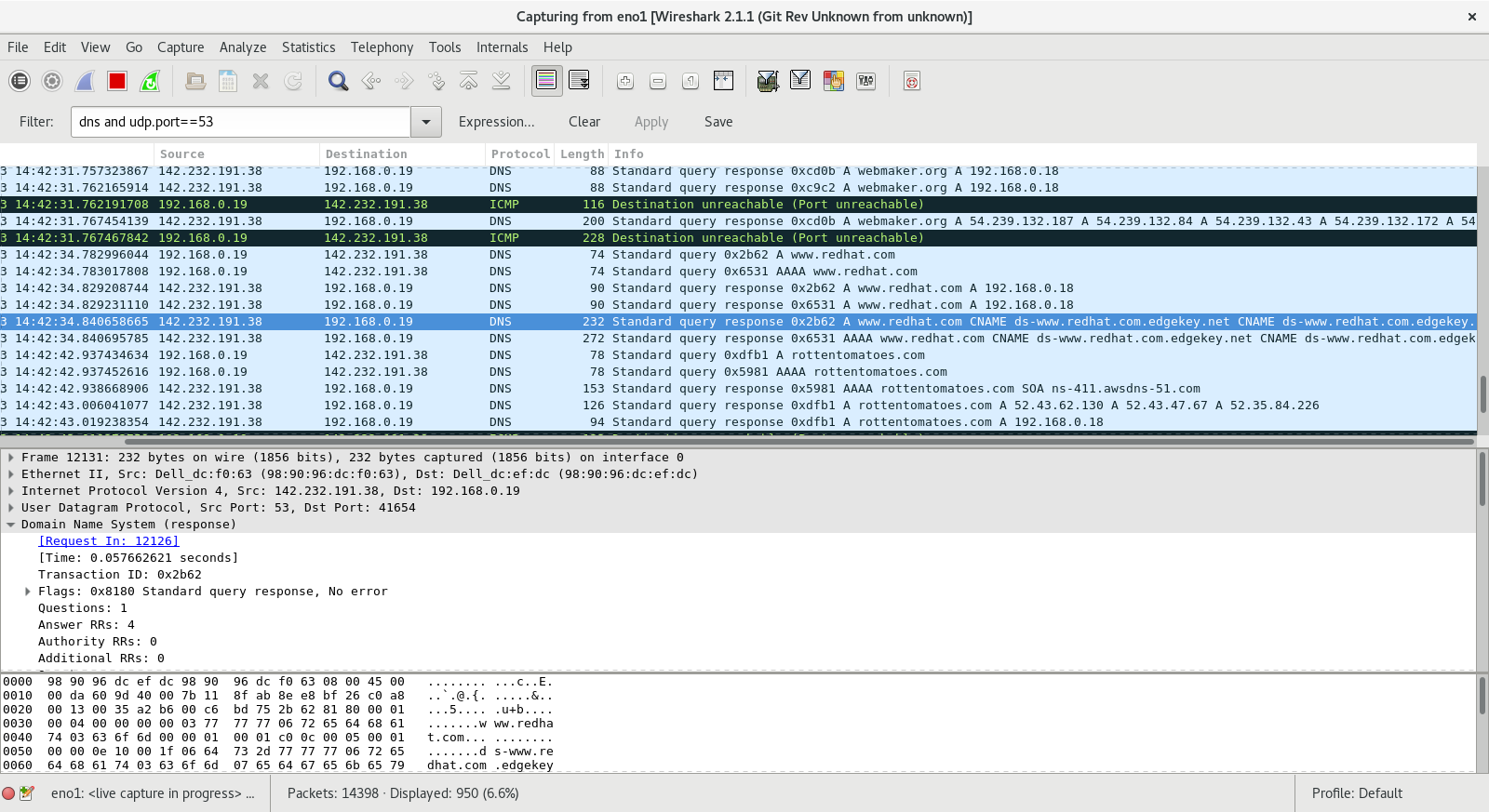
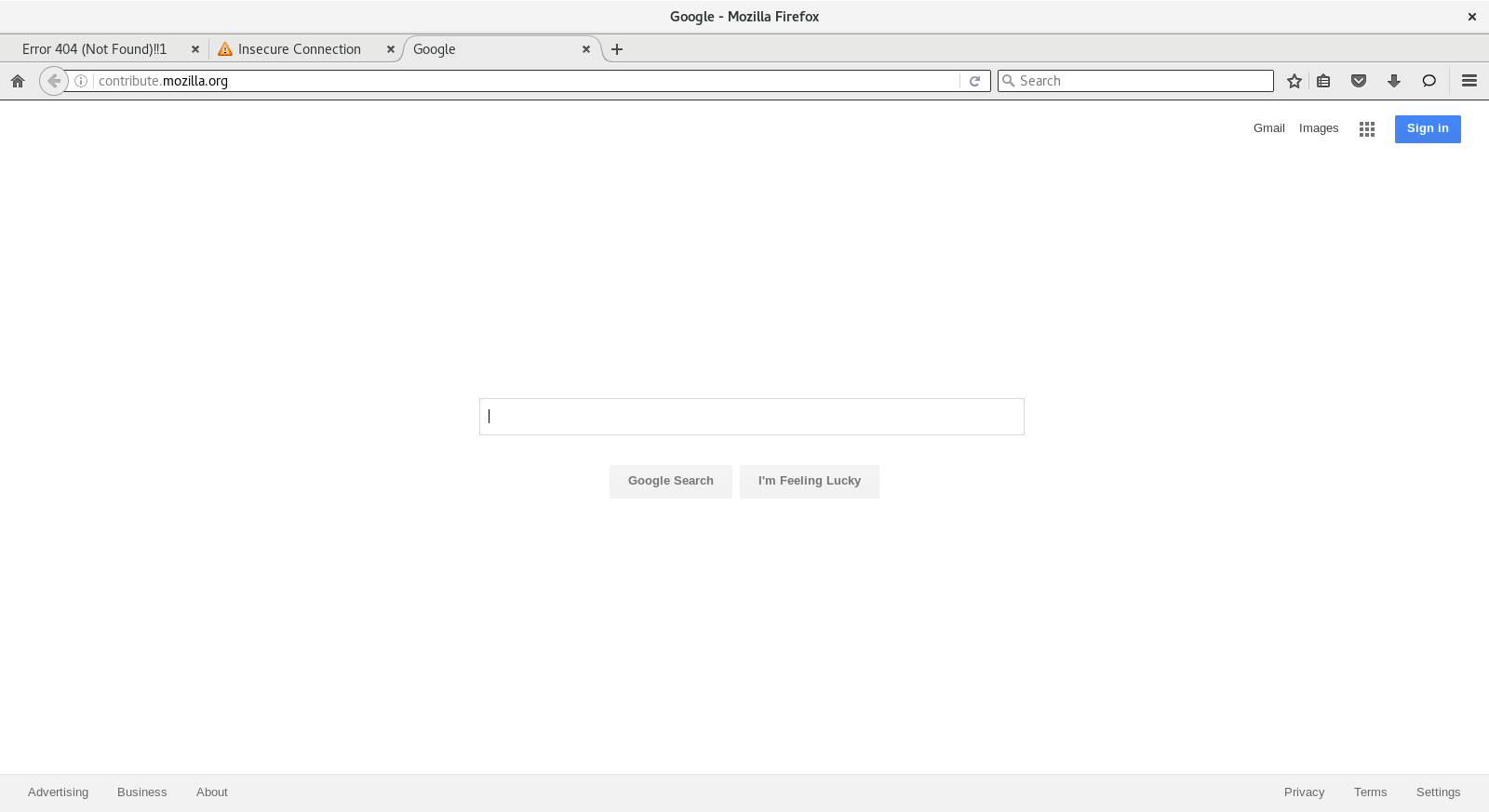


Figure 6 Host machine

1. Test 4



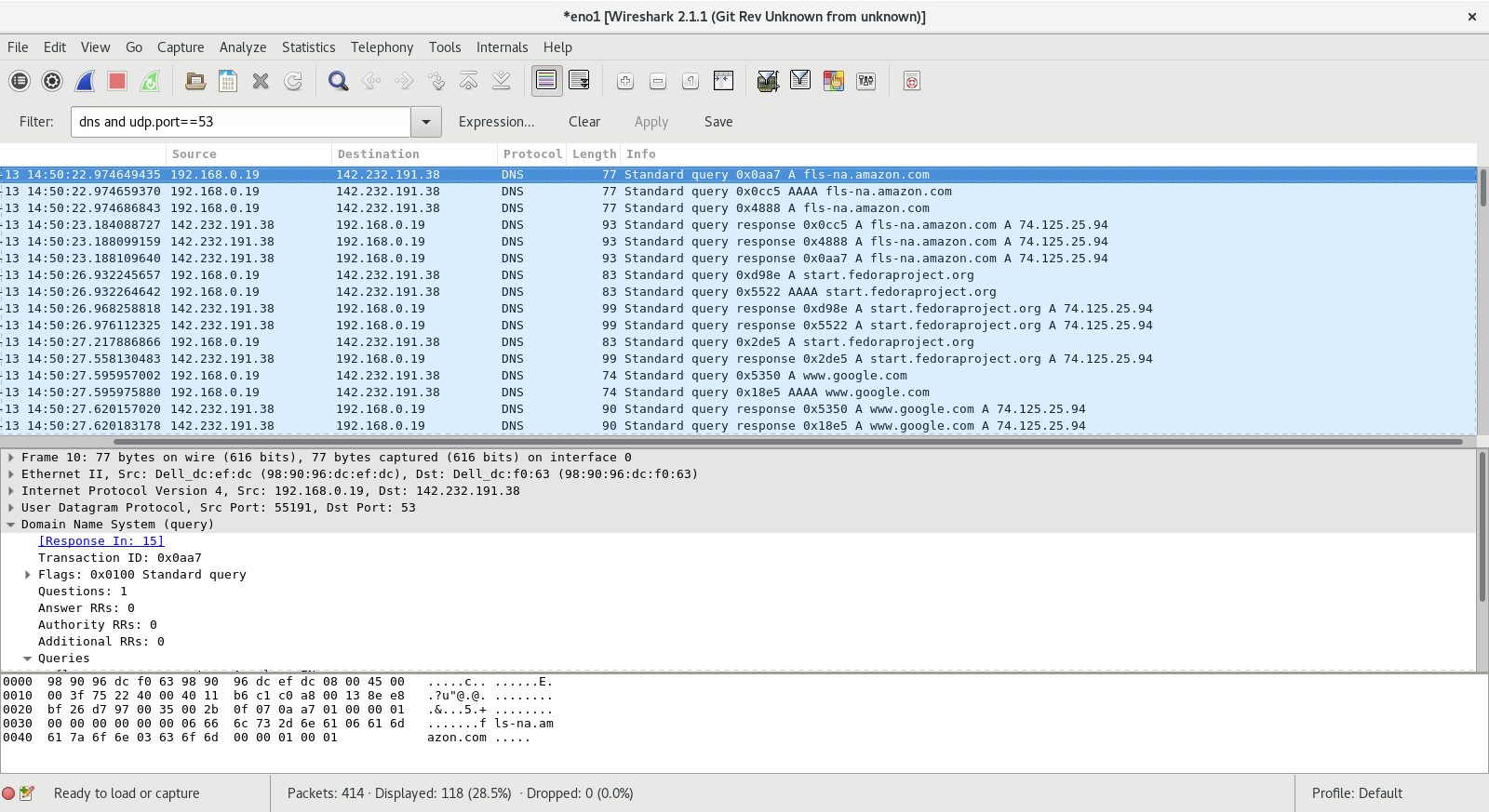


Figure 7 Victim machine

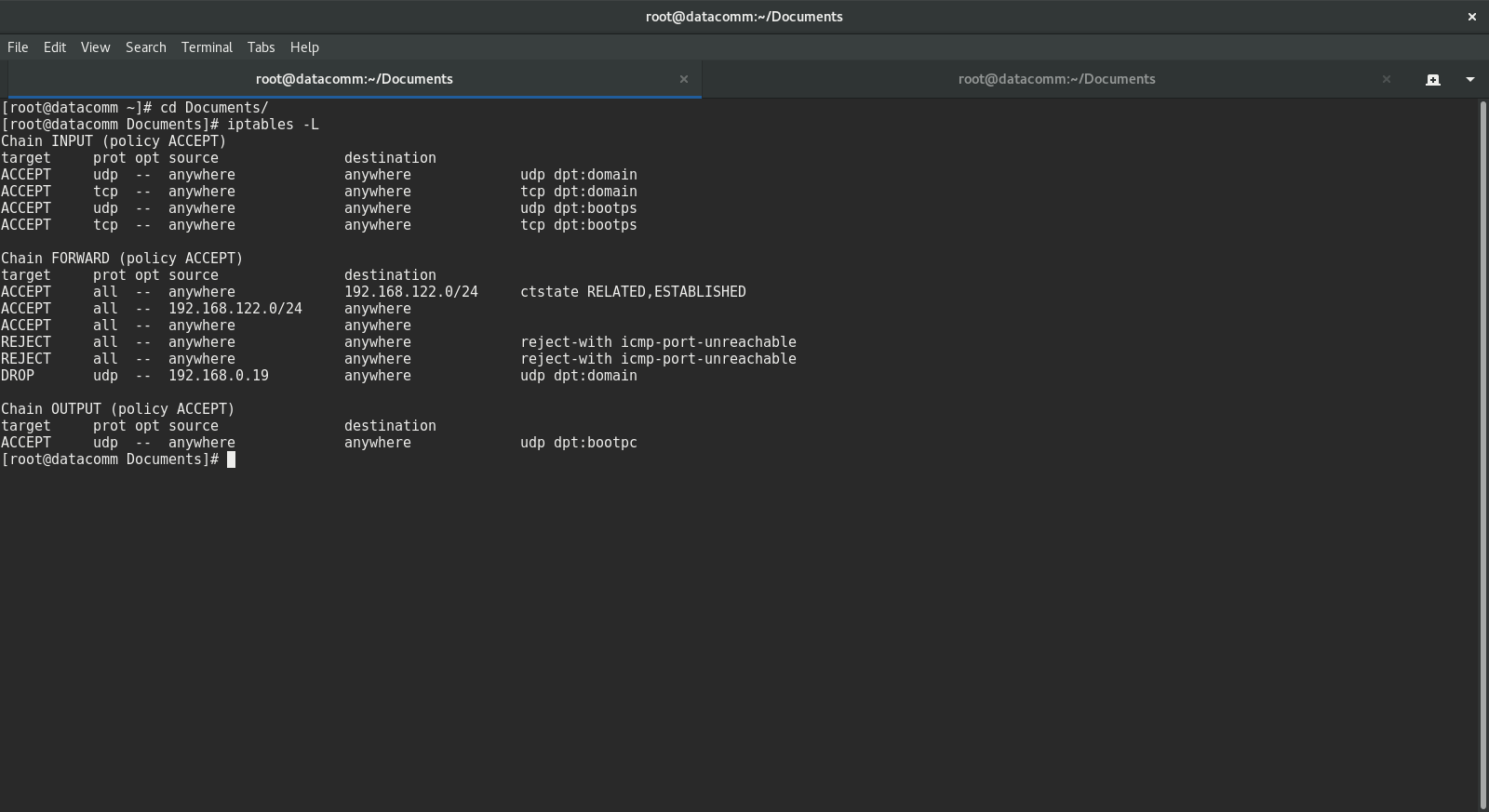
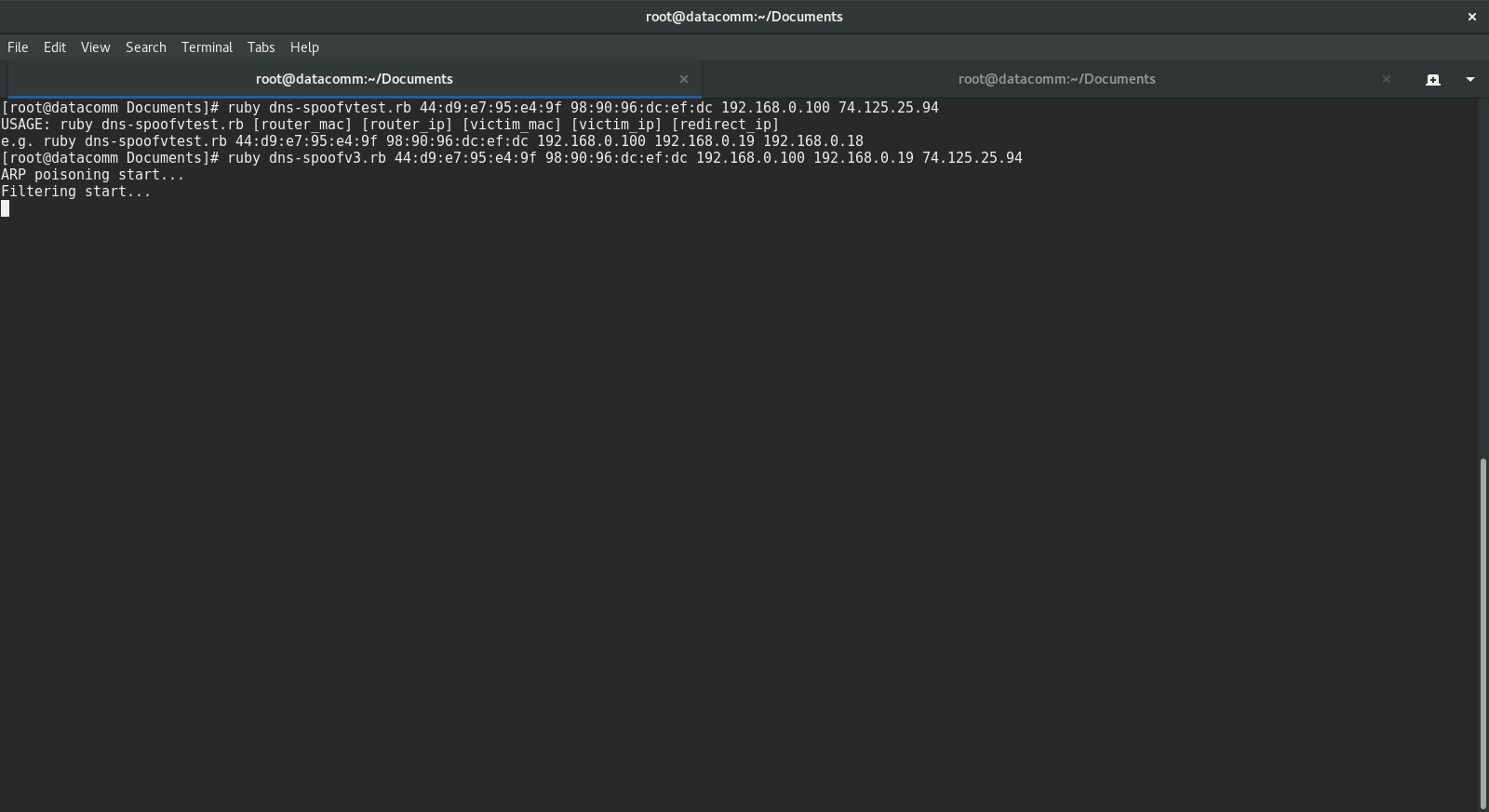


Figure 8 Host Machine

1. Test 5

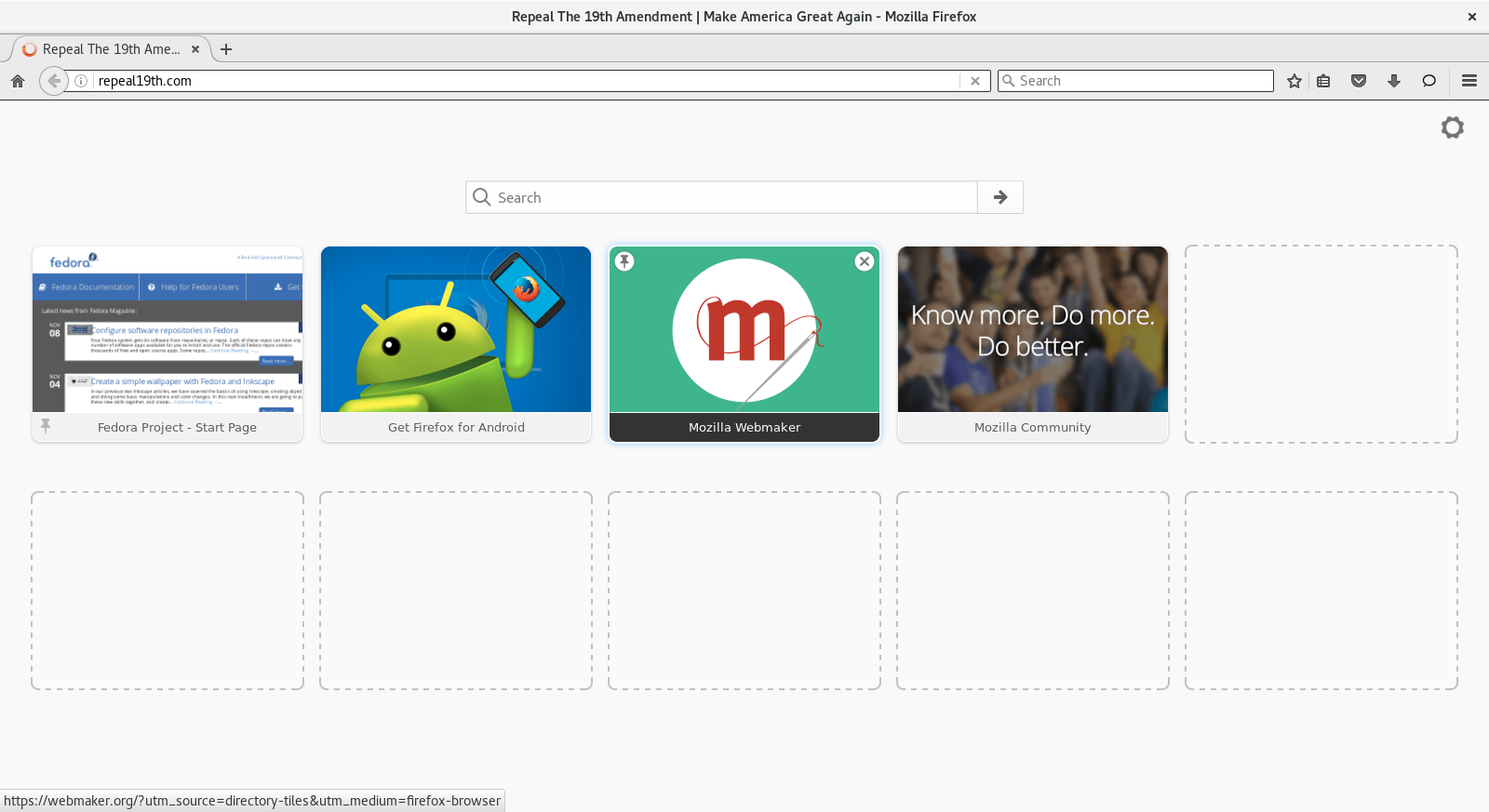
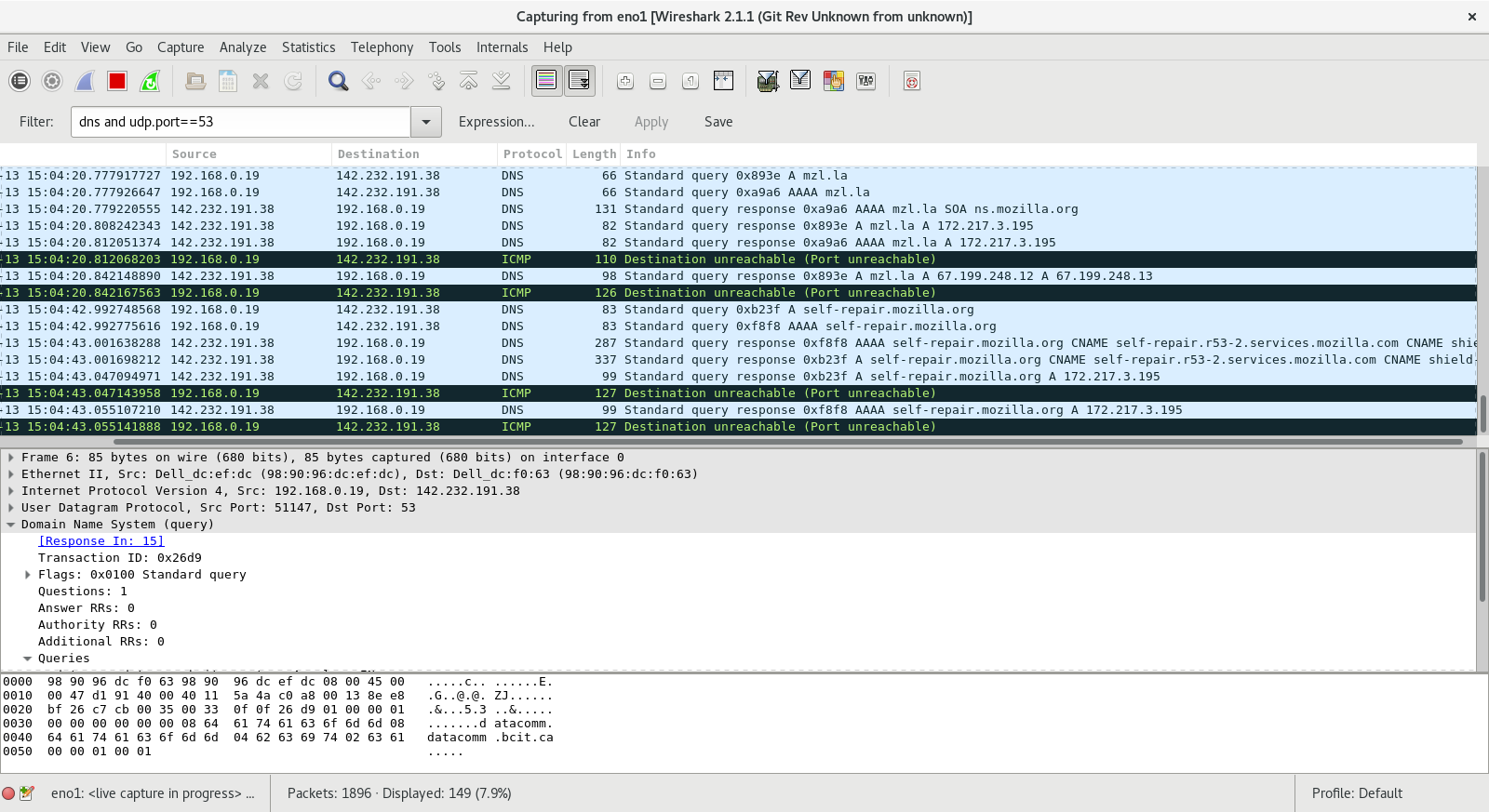
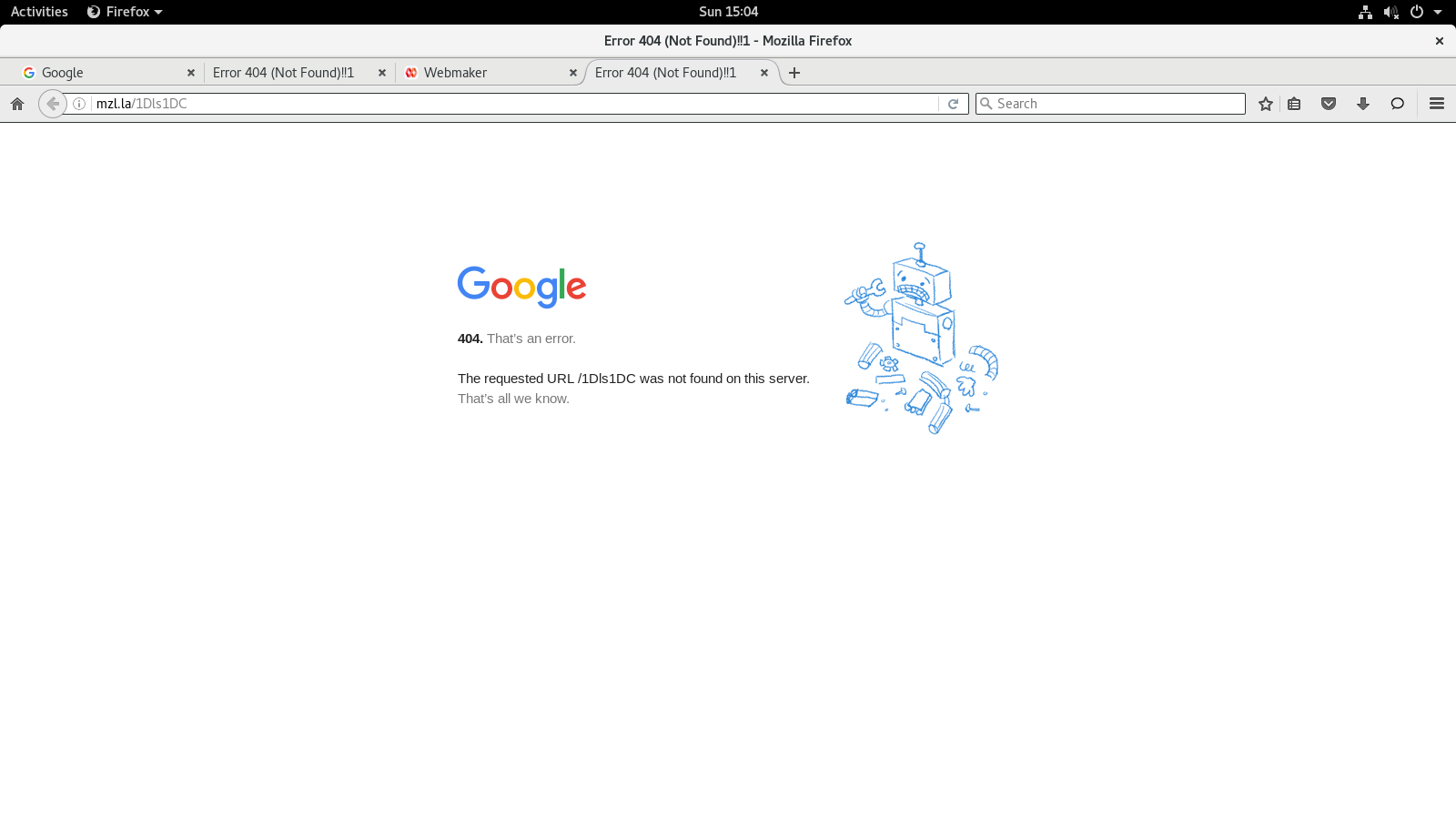


Figure 9 Victim Machine

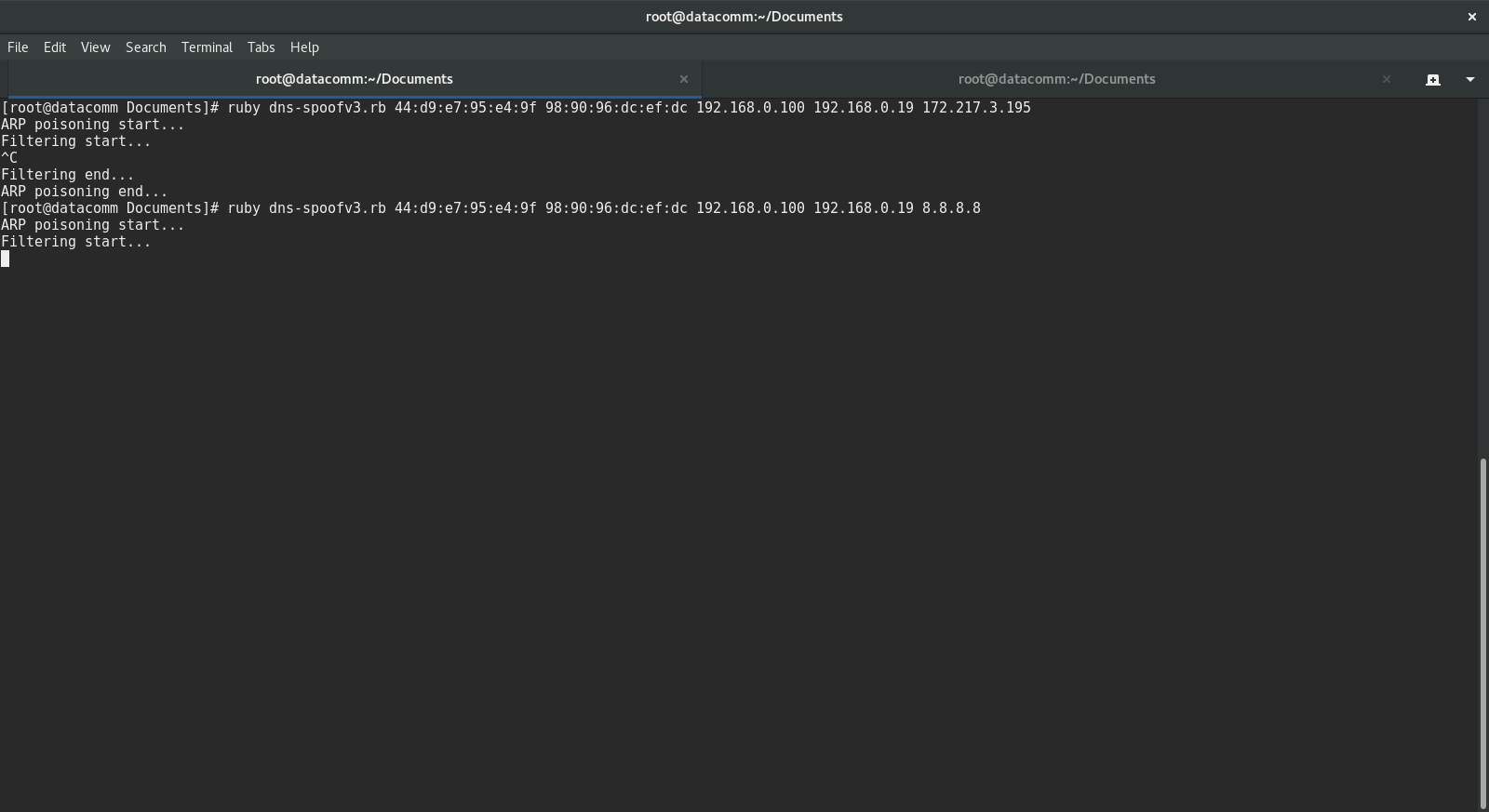
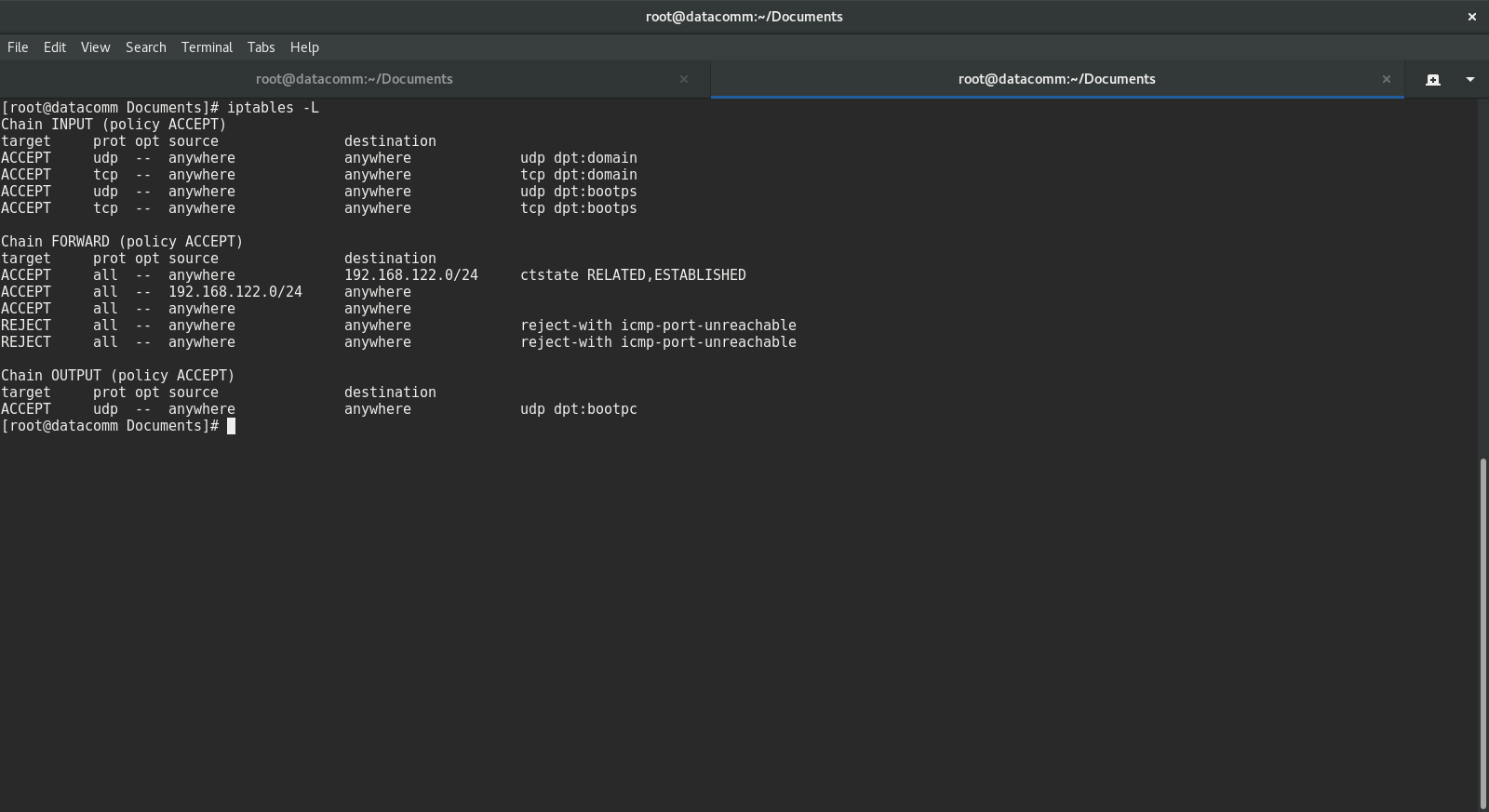


Figure 10 Host Machine

**Observations**

* Ruby without ip tables is not the best option for making the DNS spoofing. While it’s easy to craft packets with packetfu, it’s not as fast as C++.

**Pseudocode**

**Dns-spoof.rb**

|  |
| --- |
| Poison function receives packets to victim and router  While program isn’t interrupted  Send packets to respective targets  End while  End function  function getdomain receives payload  Get the domain name from the query payload  End function  function sniff receives interface  Start filtering DNS packets from victim directed to port 53  while there isn’t any interruption  for each packet, we receive  get packet  fork and create a child process  if dns packet is a query  craft dns response  send dns response  end if  exit child process  end fork  end for  end while  end function  function to build arp packet receives target’s Ethernet daddr, arp saddr, arp\_daddr  Create new ARP packet  Assign Values to it  Return packet  End of function  function help  display instructions for the code  end of function  Main function  Set parameters  Call function to build victim arp packet with the victim’s mac, router ip and victim ip  Call function to build router arp packet with the router’s mac, victim ip and router ip  Enable port forwarding  Optional set ip tables to drop dns packets  Start thread with poison function with victim and router arp packets  Start thread with sniff function with interface  Join threads  Catch interruption  Remove port forwarding  Optional remove ip tables rules  Kill threads  End catch  End of main |