Assignment #3 Backdoor

Mario Enriquez

British Columbia Institute of Technology

COMP 8505, 7D

Aman Abdulla

October 31, 2016

Contents

[Introduction 3](#_Toc465637844)

[Design Work 3](#_Toc465637845)

[Command.py 3](#_Toc465637846)

[Sniffer.py 4](#_Toc465637847)

[Test Cases 4](#_Toc465637848)

[1. Test 1 6](#_Toc465637849)

[2. Test 2 6](#_Toc465637850)

[3. Test 3 8](#_Toc465637851)

[4. Test 4 10](#_Toc465637852)

[5. Test 5 10](#_Toc465637853)

[6. Test 6 14](#_Toc465637854)

[7. Test 7 14](#_Toc465637855)

[8. Test 8 15](#_Toc465637856)

[9. Test 9 16](#_Toc465637857)

[Observations 20](#_Toc465637858)

[Pseudocode 20](#_Toc465637859)

[Command.py 20](#_Toc465637860)

[Sniffer.py 22](#_Toc465637861)

[Cyphermod.py 24](#_Toc465637862)

[Maskname.py 25](#_Toc465637863)

**Introduction**

For this assignment, I had to develop a backdoor program that would execute instructions that I sent to it and receive the results. To achieve this I made two programs, one which would function as the backdoor and the other one which would send instructions and receive the results.

**Design Work**

Command.py



Sniffer.py



**Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Scenario** | **Tools** | **Expected Results** | **Actual Results** |
| **1** | Check process mask | Python, Fedora | Success, New name is updated and shown in ps auxw | Success, New name is updated and shown in ps auxw |
| **2** | List and create a new directory | Fedora, UDP send packet | Success, new directory is created | Success, new directory is shown in psauxw |
| **3** | Run ifconfig to confirm the victim’s | Fedora, UDP send packet | Success, data of the victim machine is shown | Success, data of the victim machine is shown |
| **4** | Run errnoneous command and get answer | Fedora, UDP send packet | Success, Command is run and we get error message | Success, Command is run and we get error message |
| **5** | TCP test, try to read large files | Fedora, TCP send packet | Success, we can read big files in our console | Success, we can read big files in our console |
| **6** | Try reading a privileged file | Fedora, TCP send packet | Success, we can read the file | Success, we can read the file |
| **7** | Check Wireshark to see encryption works | Wireshark, UDP | Success, message is encrypted | Success, message is encrypted |
| **8** | Check Wireshark to see encryption works | Wireshark, TCP | Success, message is encrypted | Success, message is encrypted |
| **9** | Try sending to a port outside the filter | Wireshark, send TCP packet | Failure, since it’s the wrong port | Failure, since it’s the wrong port |

1. Test 1

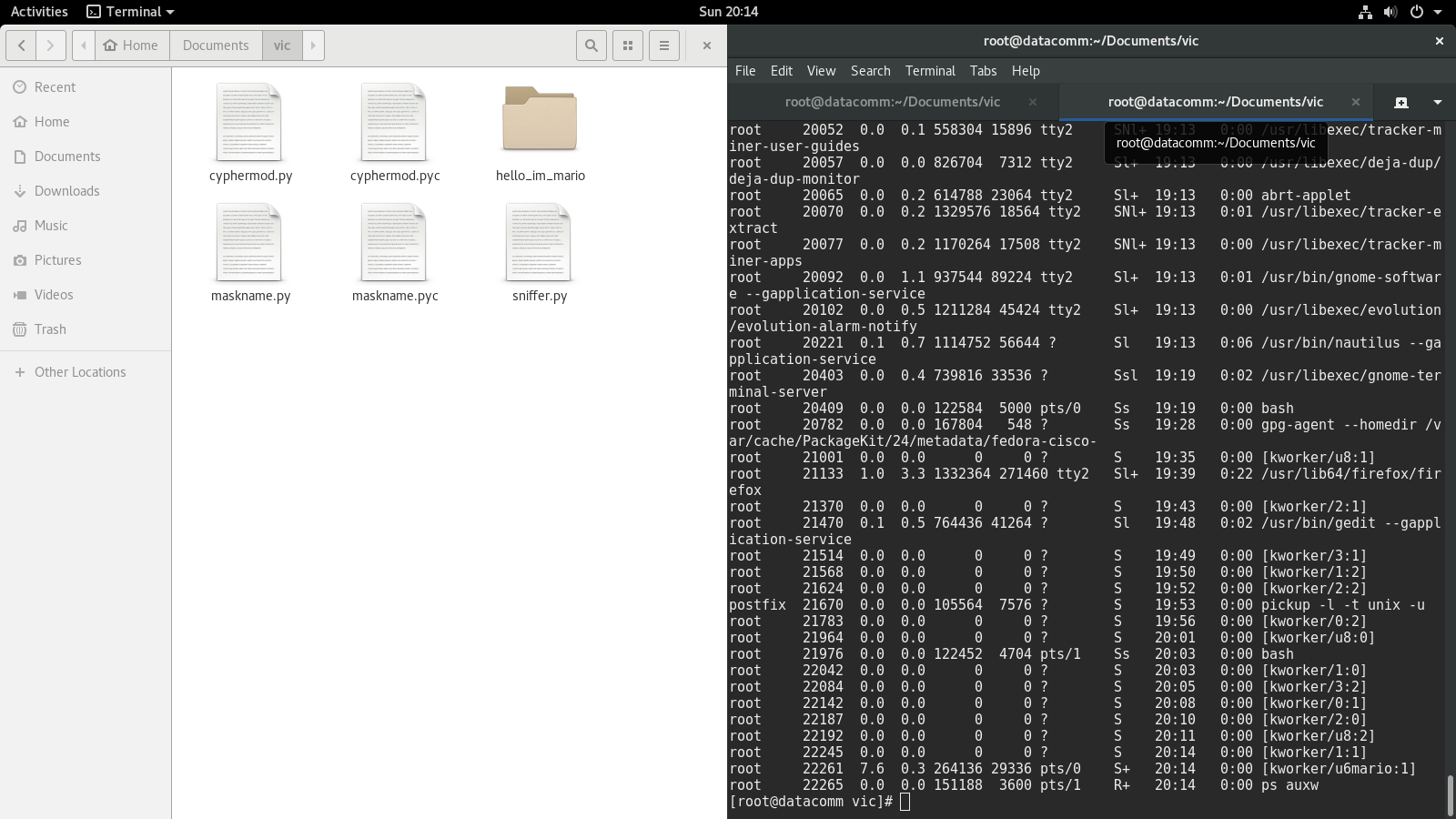


Figure 1 Process name change

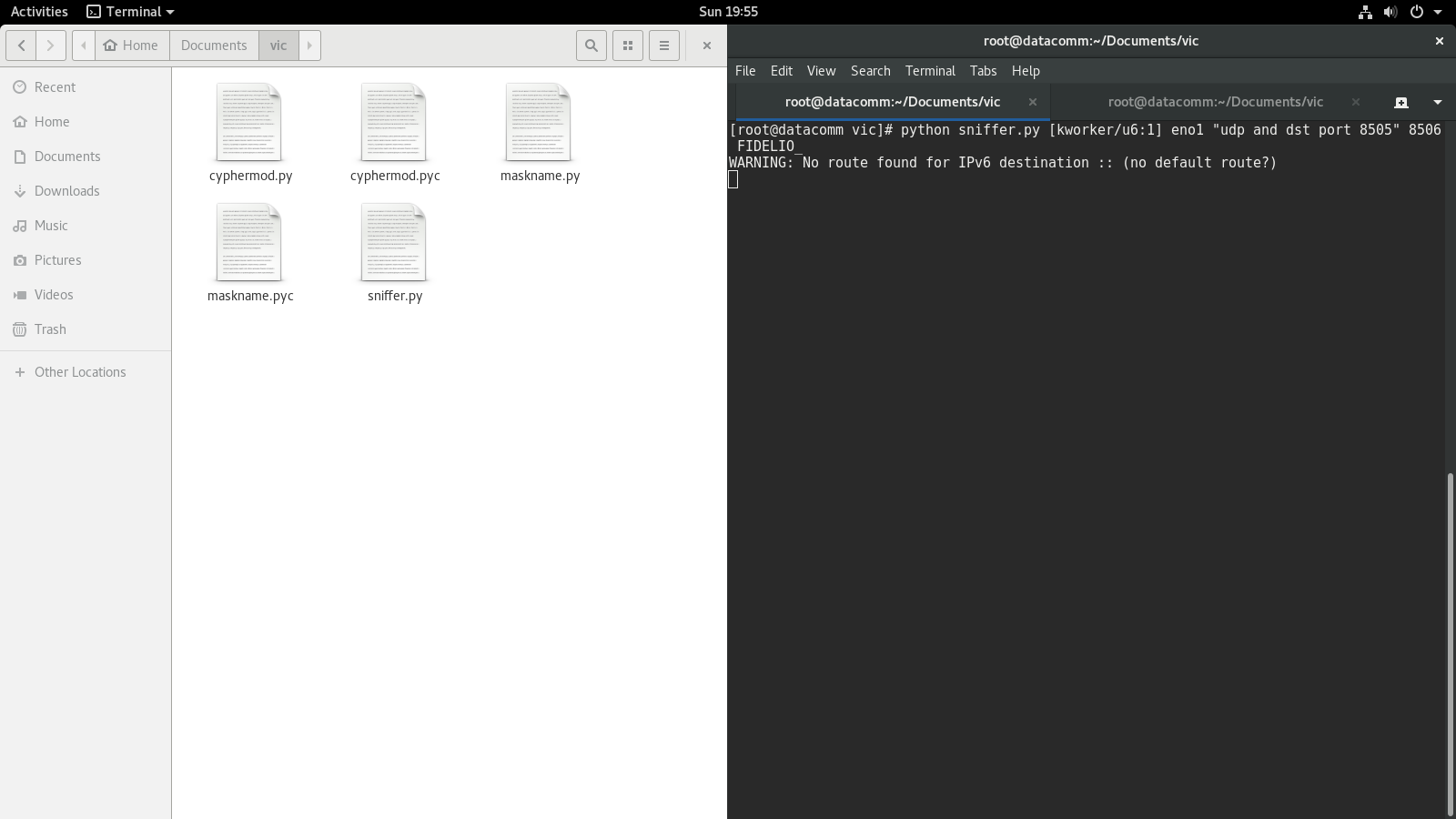
1. Test 2







Figure 2 Attacker Machine



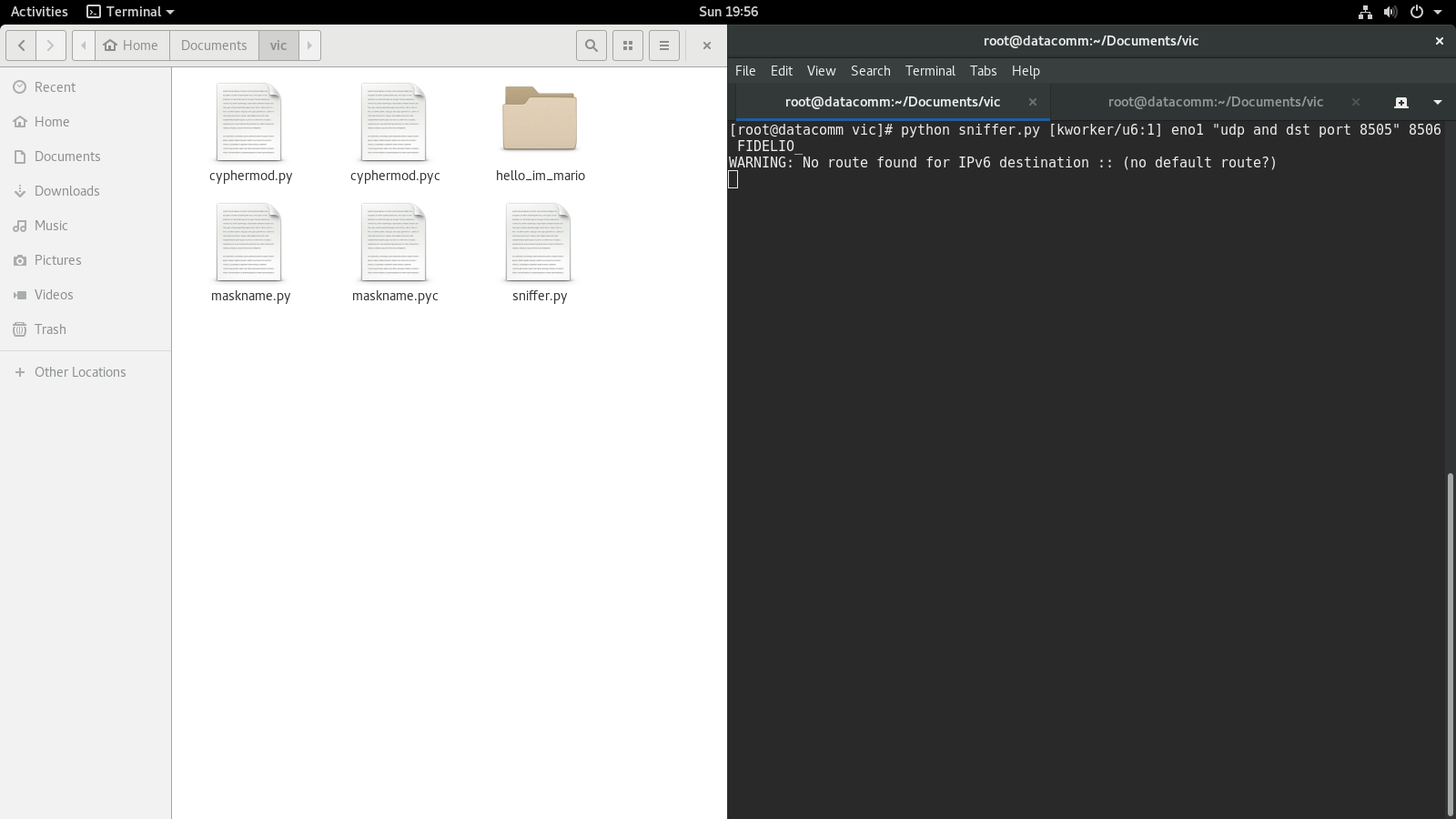
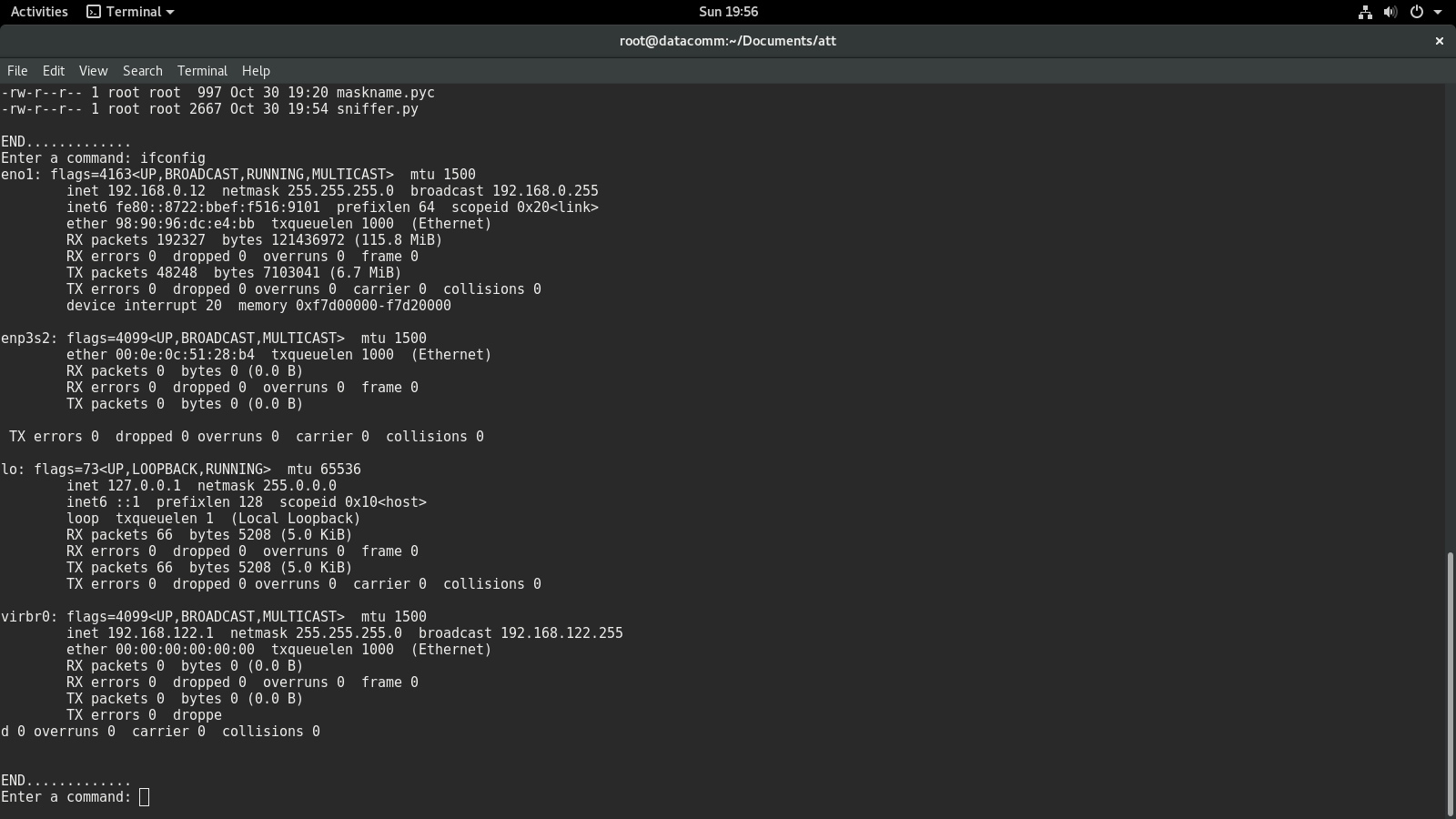


Figure 3 Victim Machine

1. Test 3



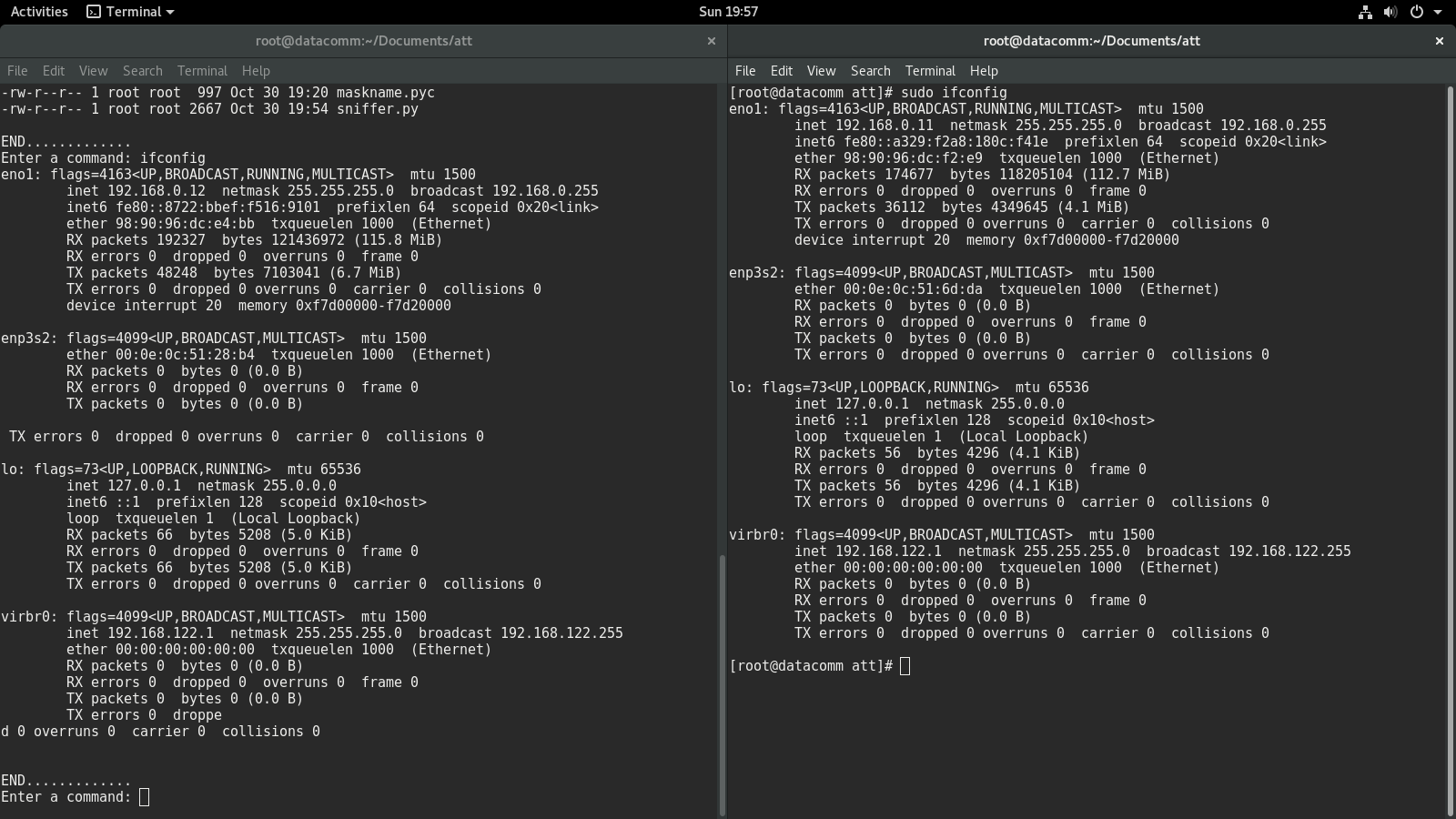


Figure 4Attacker's Machine

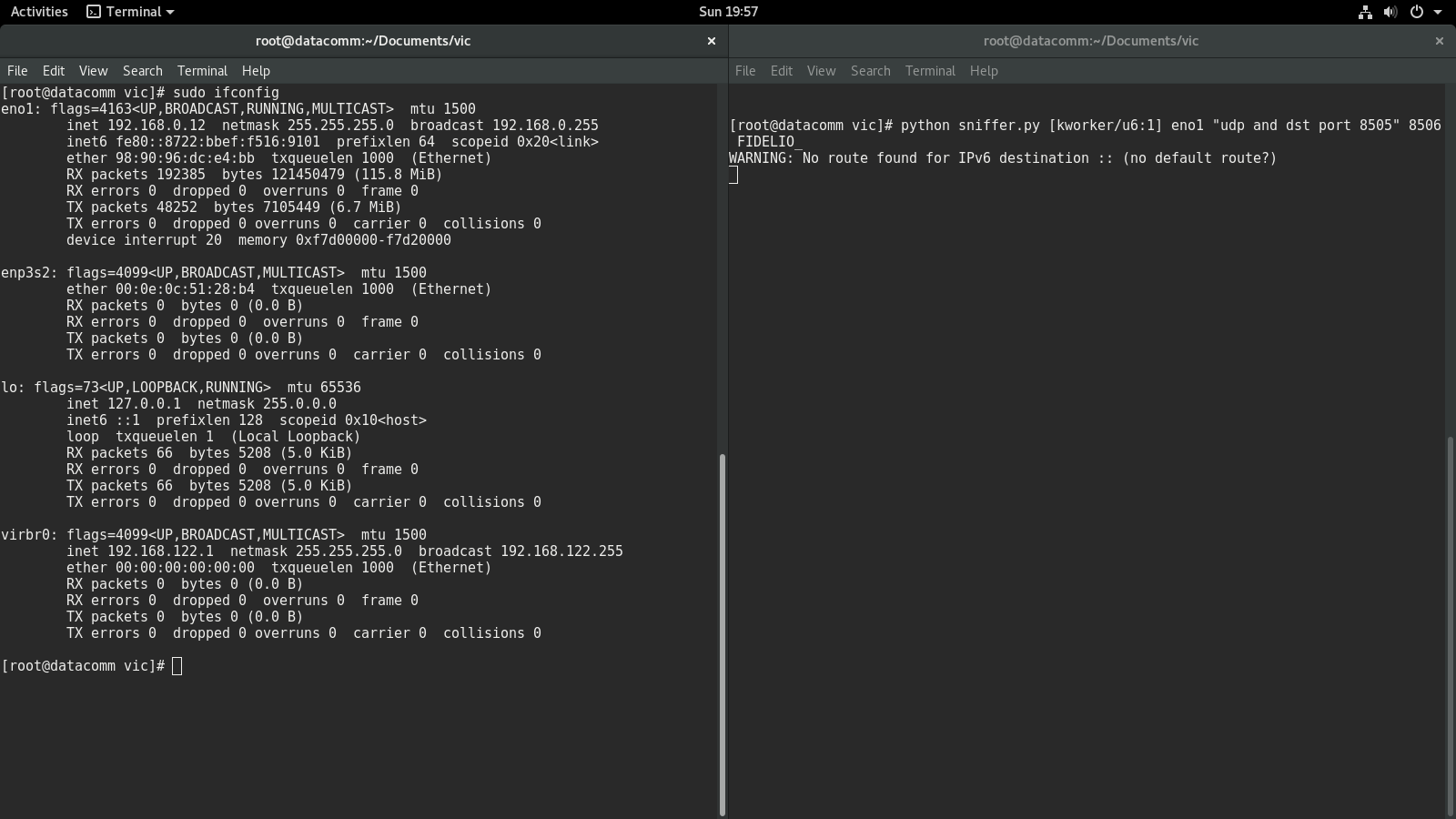


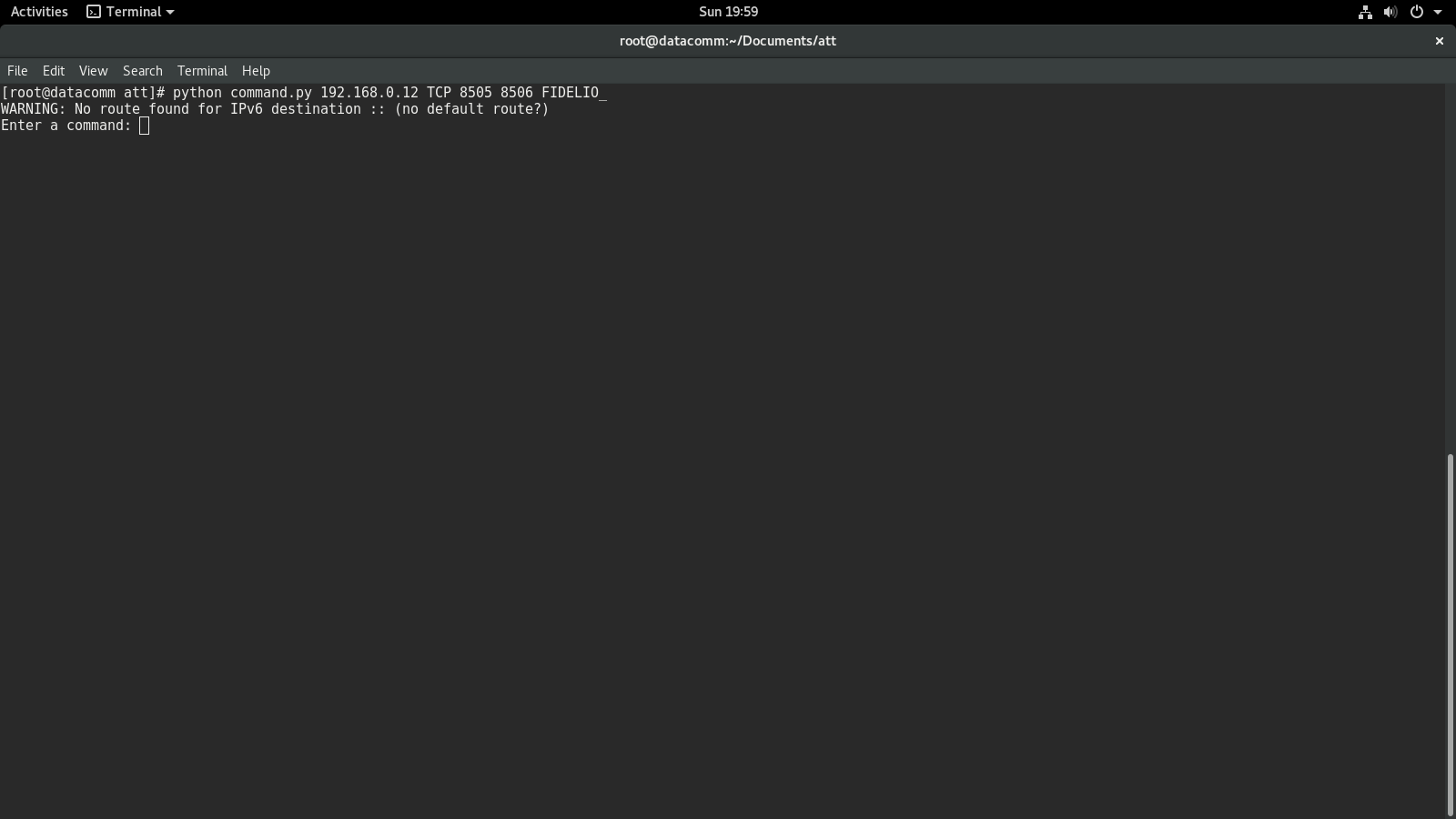
Figure 5 Victim's machine

1. Test 4



Figure 6 Erroneous command

1. Test 5







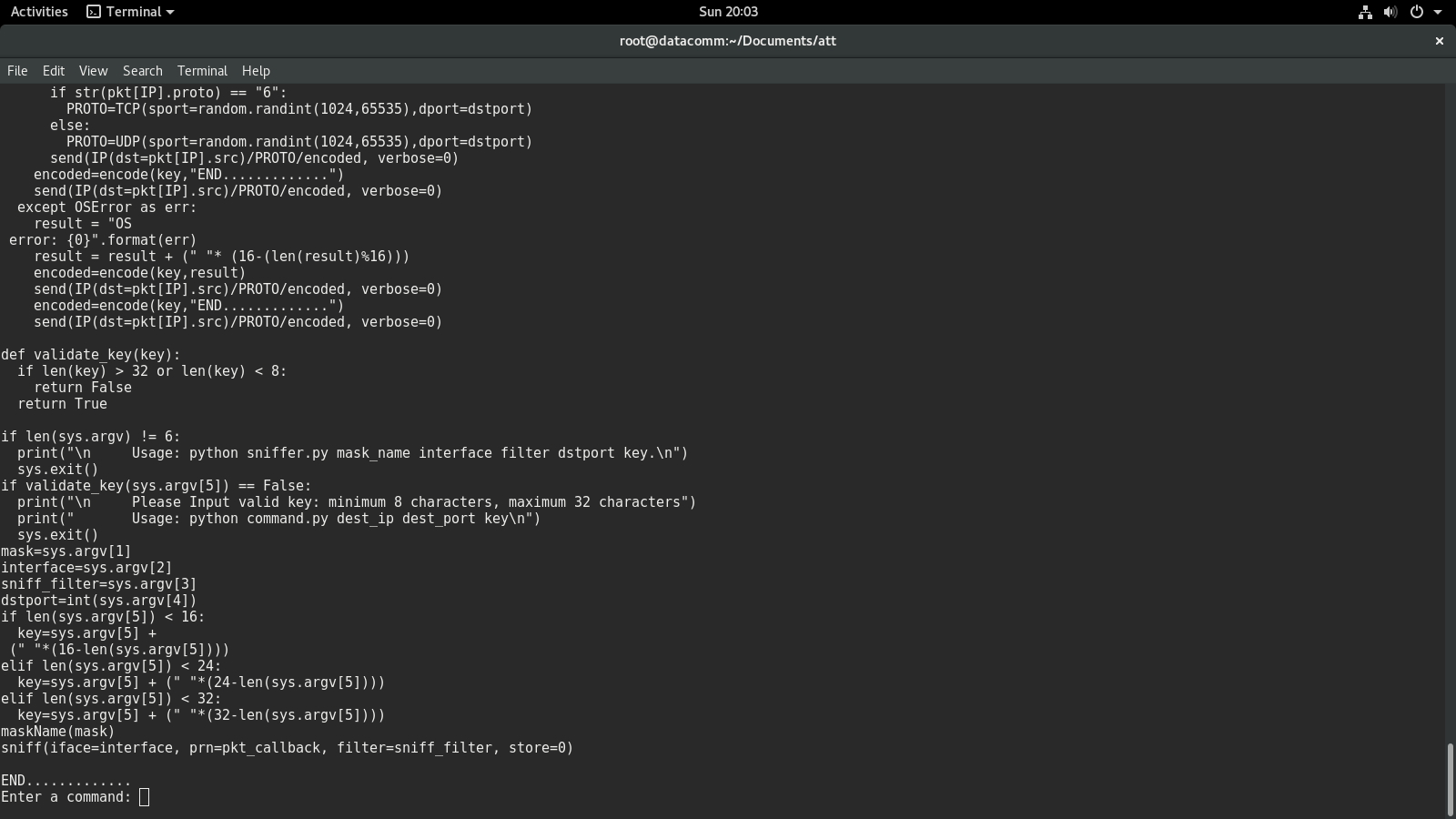
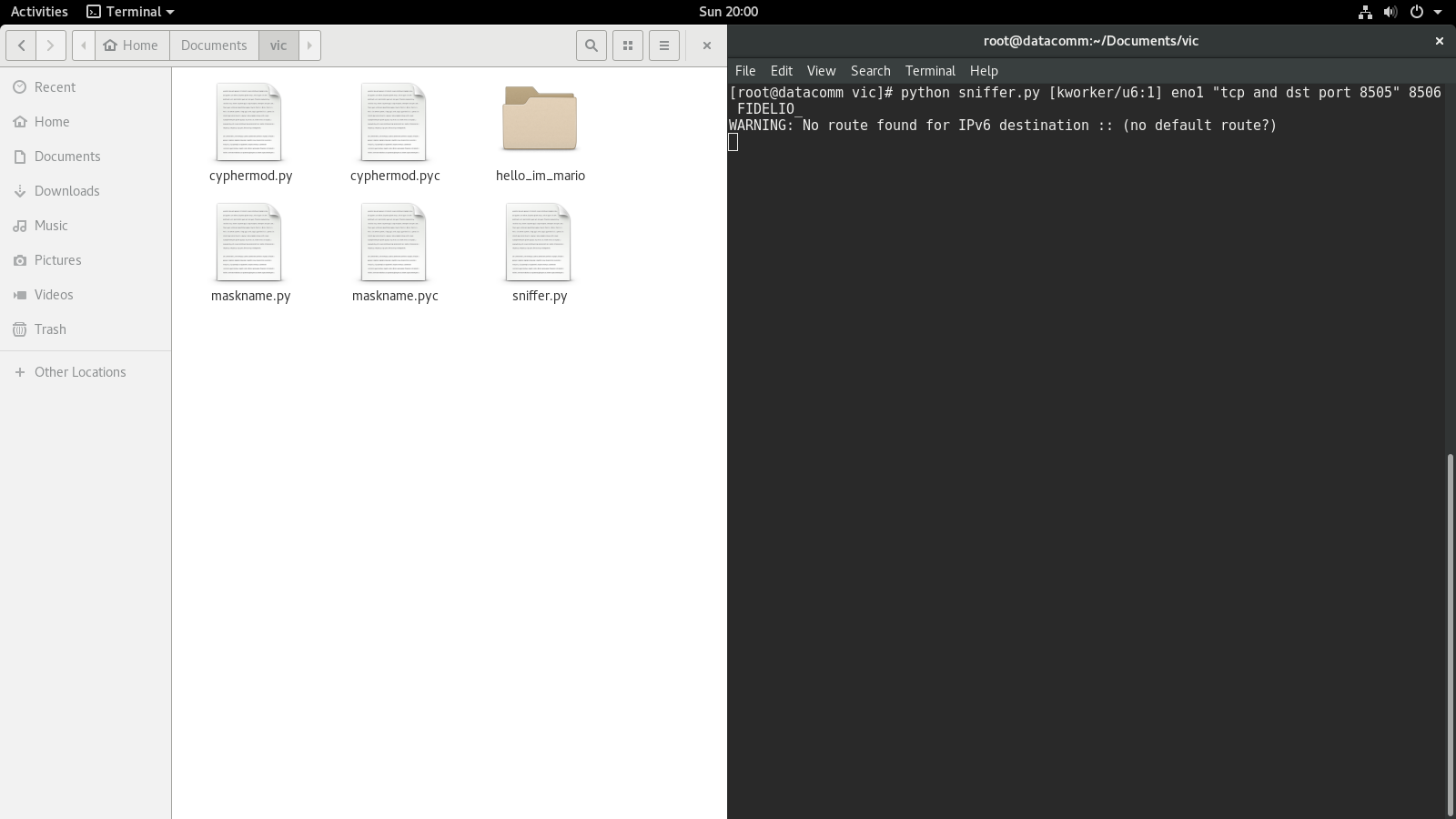
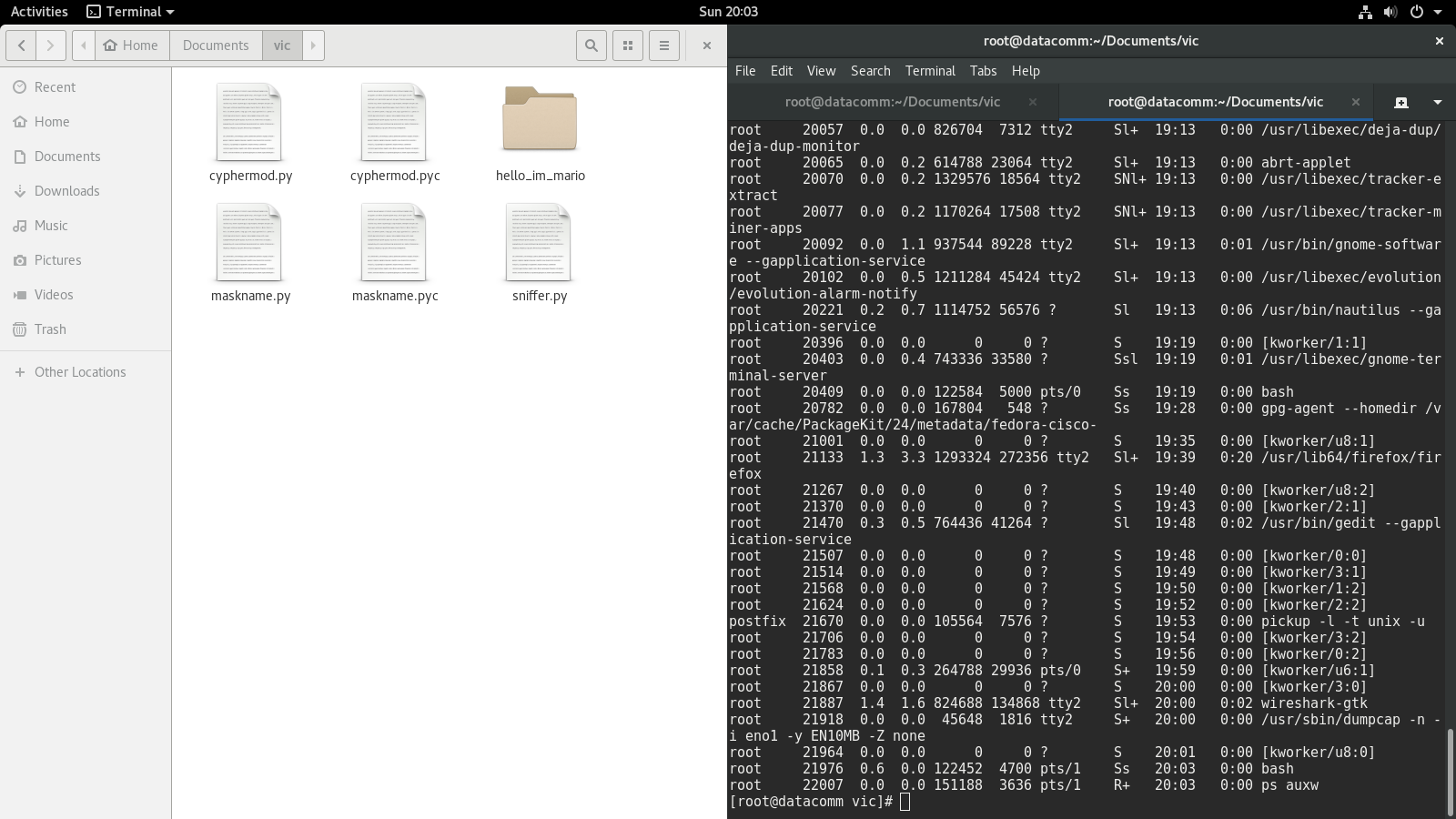


Figure 7 Attacker's Machine





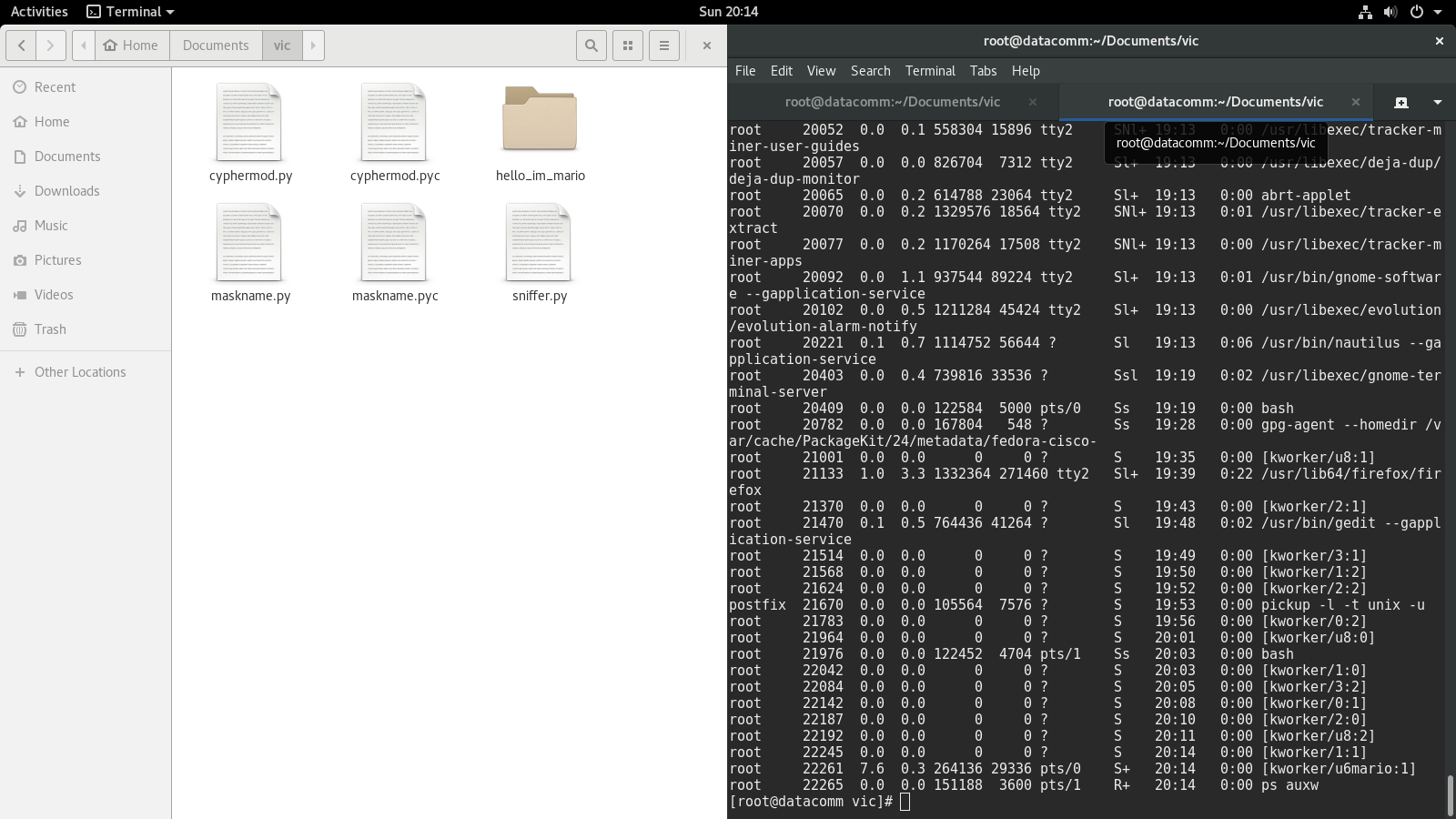


Figure 8 Victim's Machine

1. Test 6



Figure 9 Attacker's Machine

1. Test 7

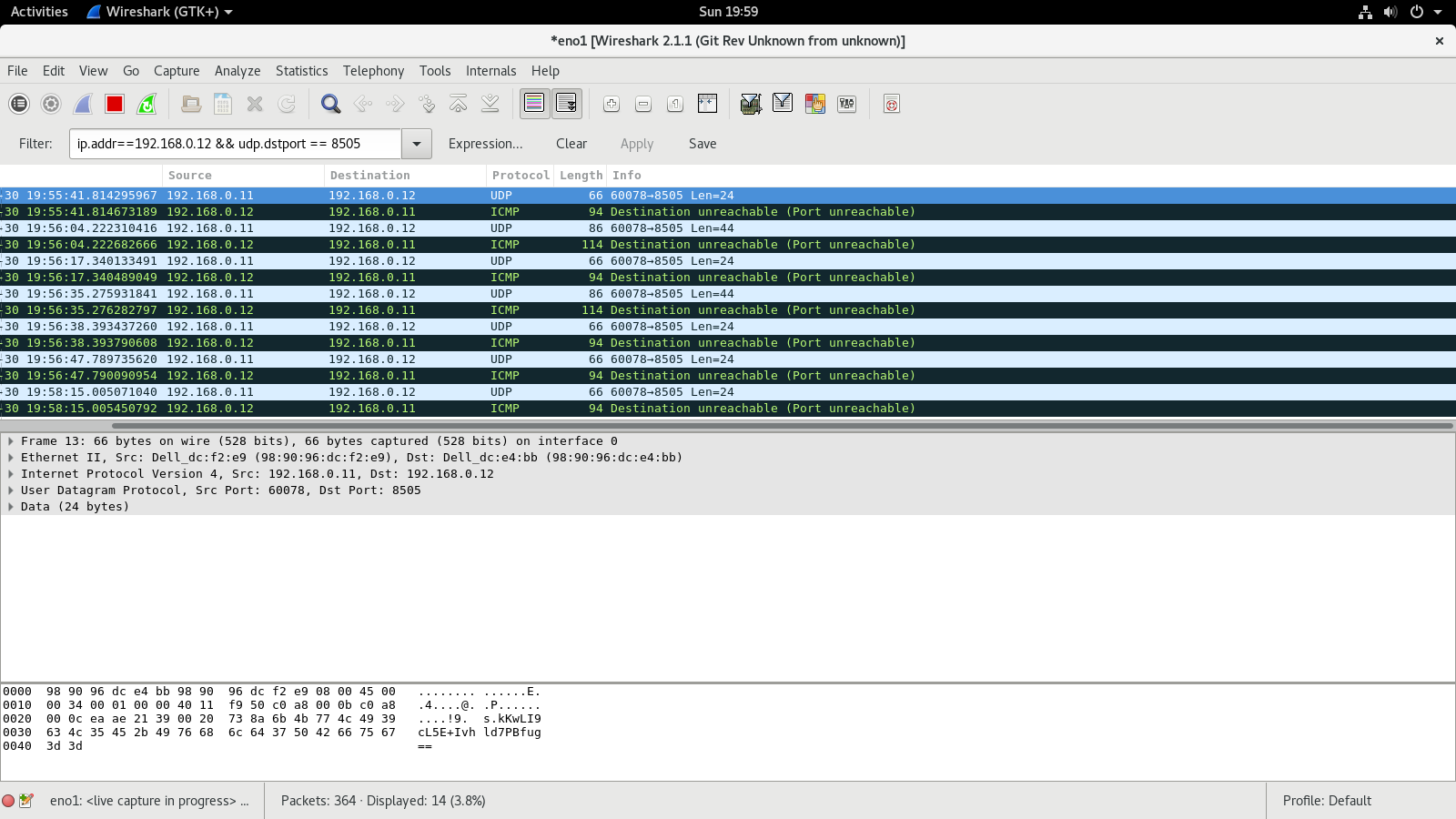
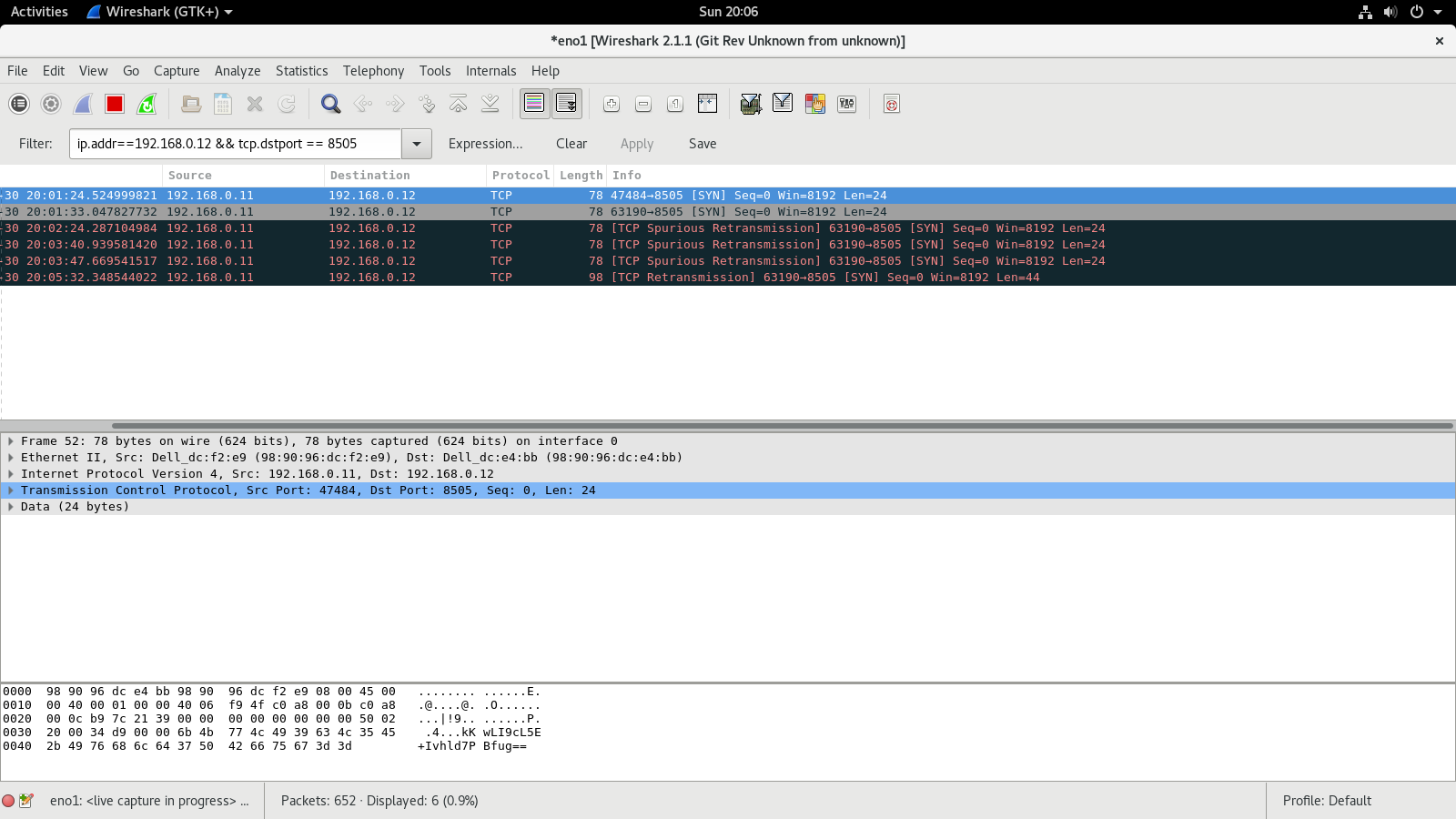


Figure 10 UDP exchange

1. Test 8



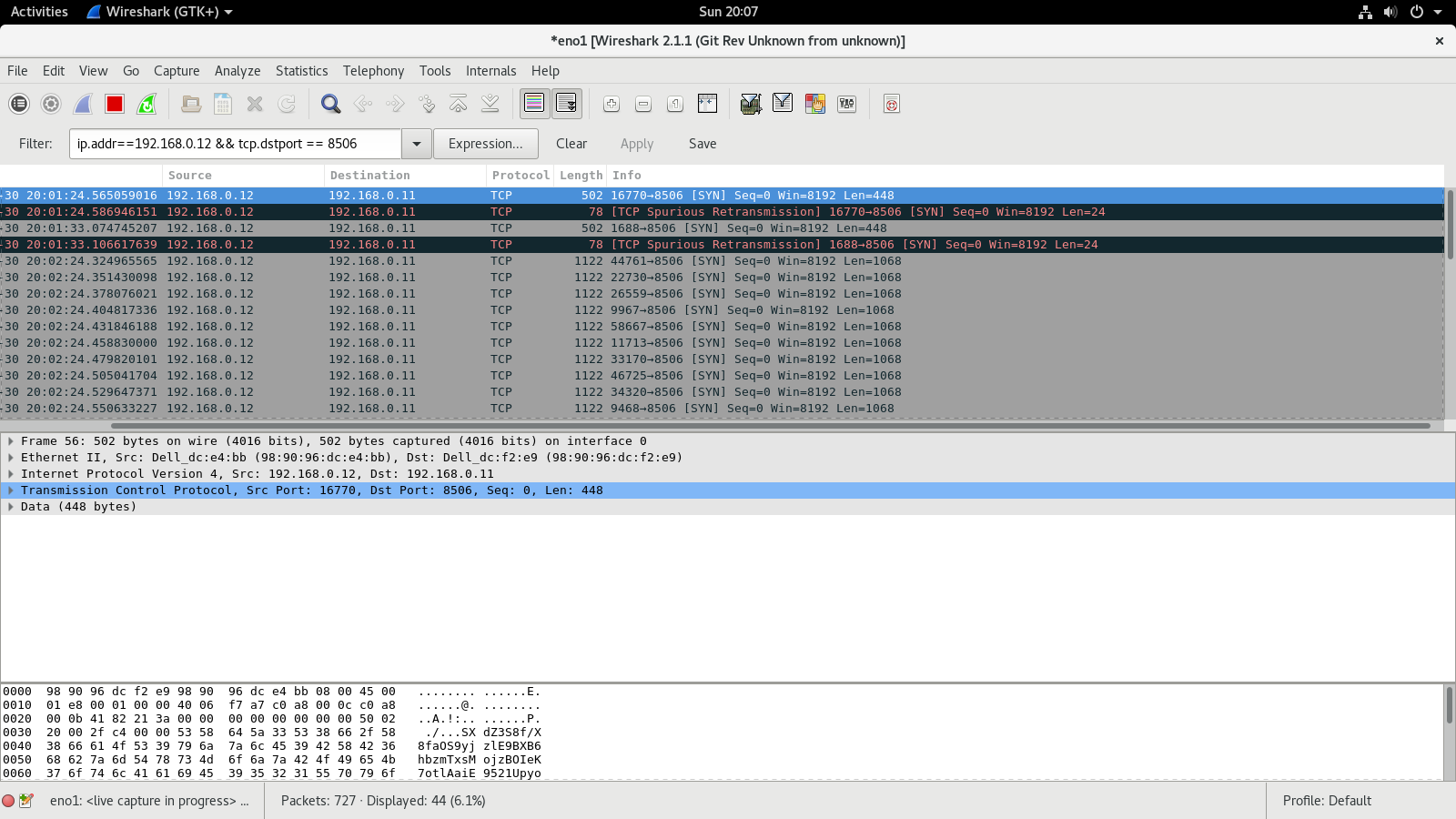


Figure 11 TCP sent packets

1. Test 9

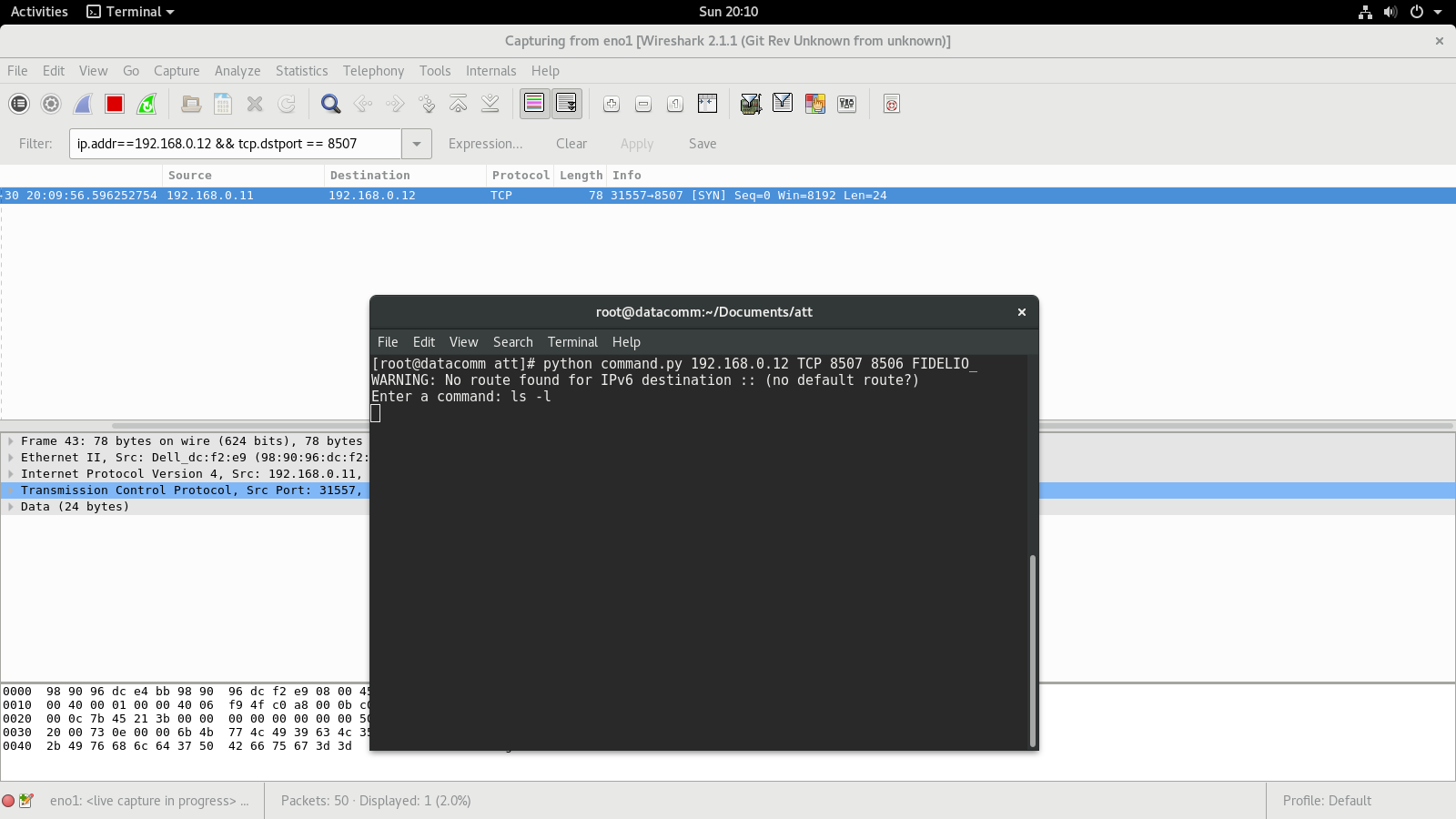
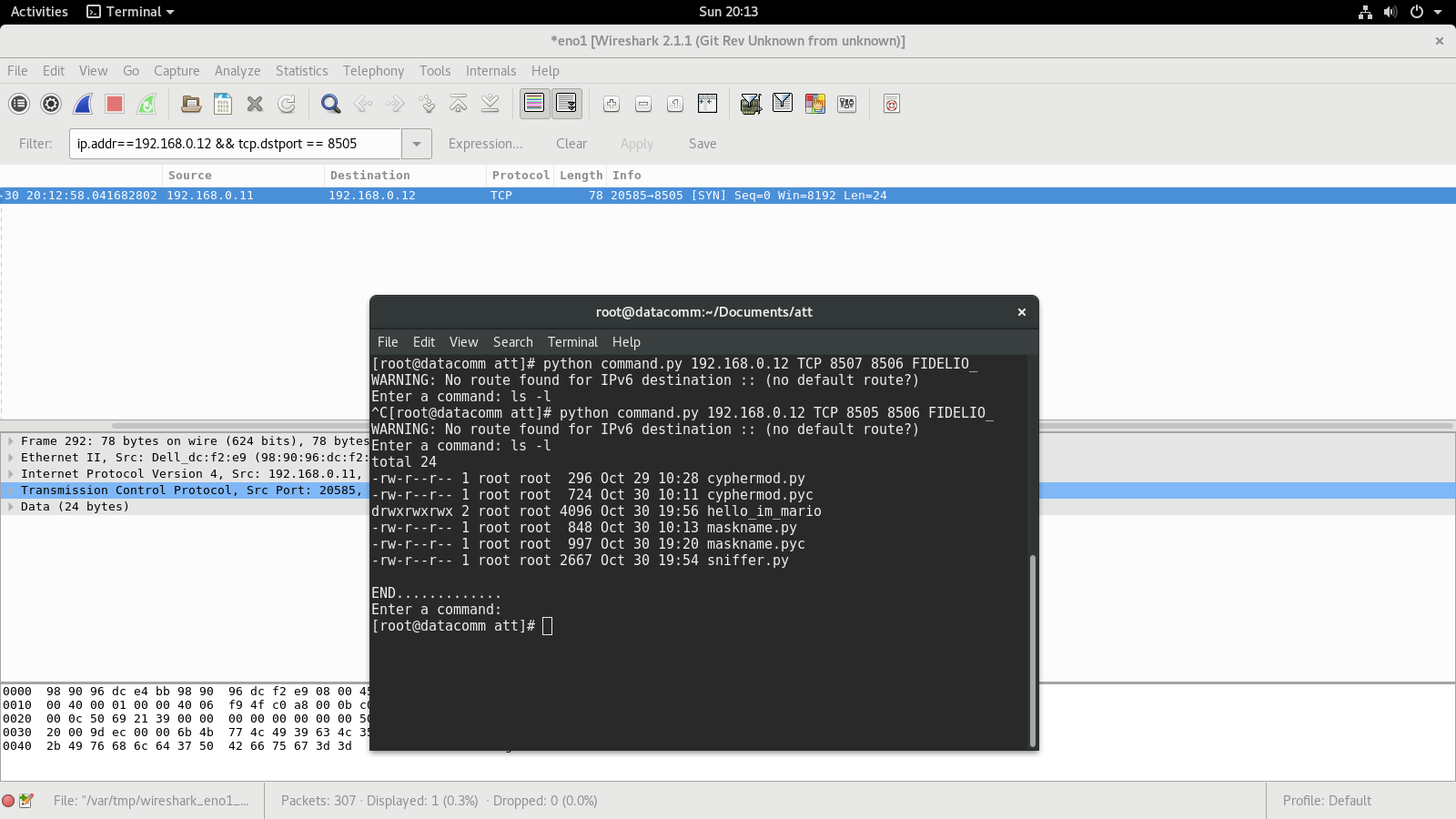
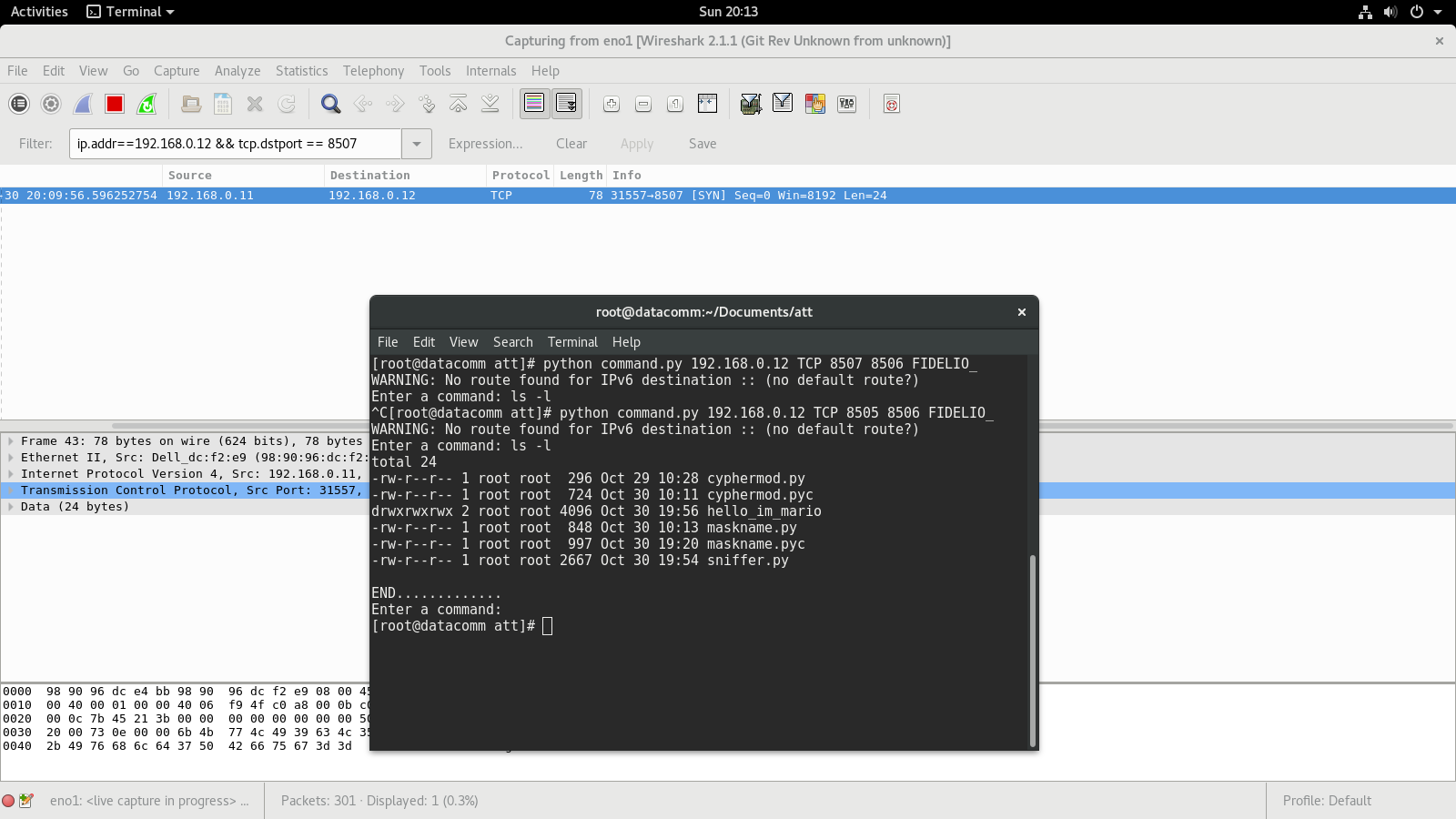


Figure 12 Message sent to wrong port, we get nothing



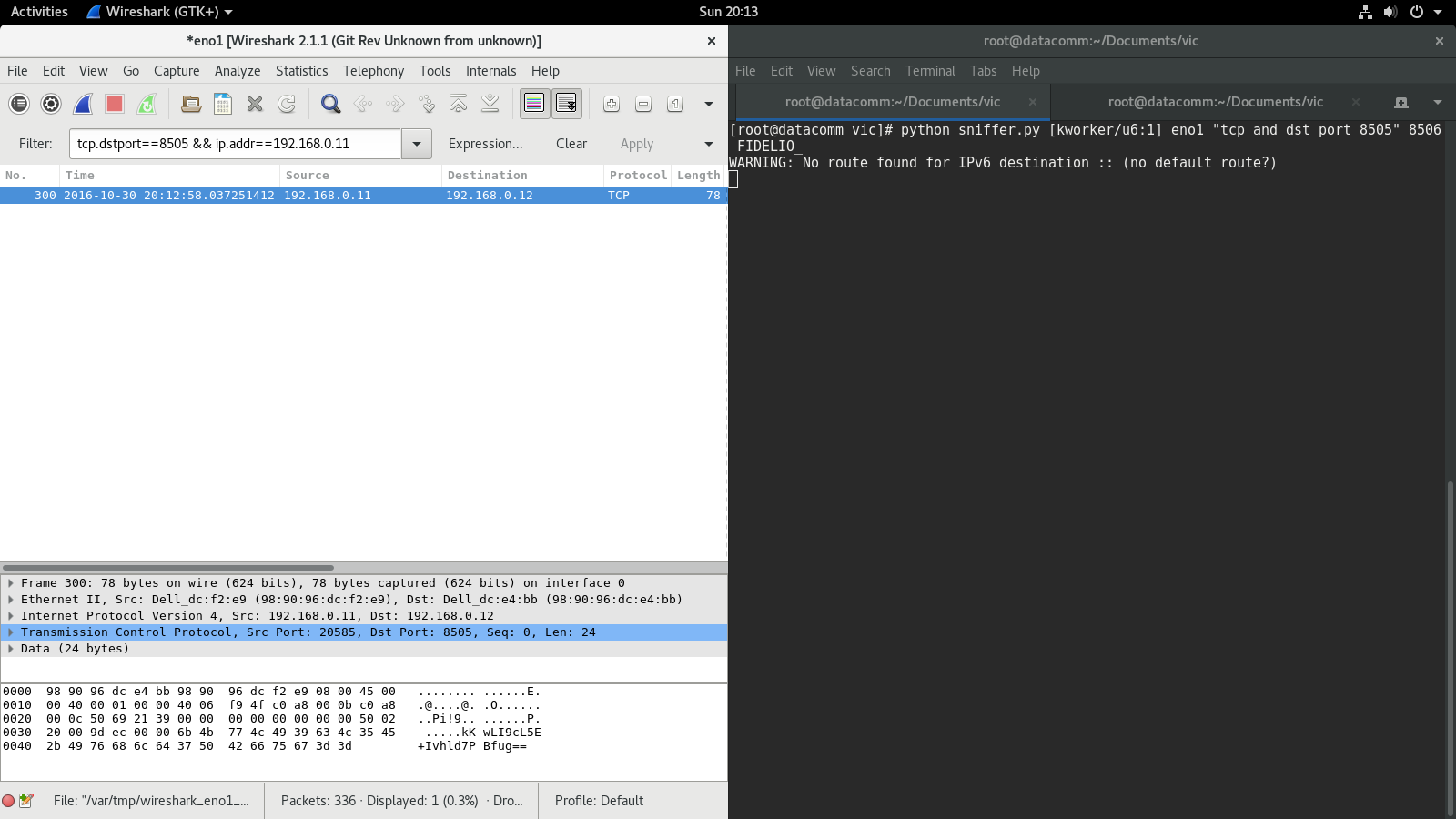
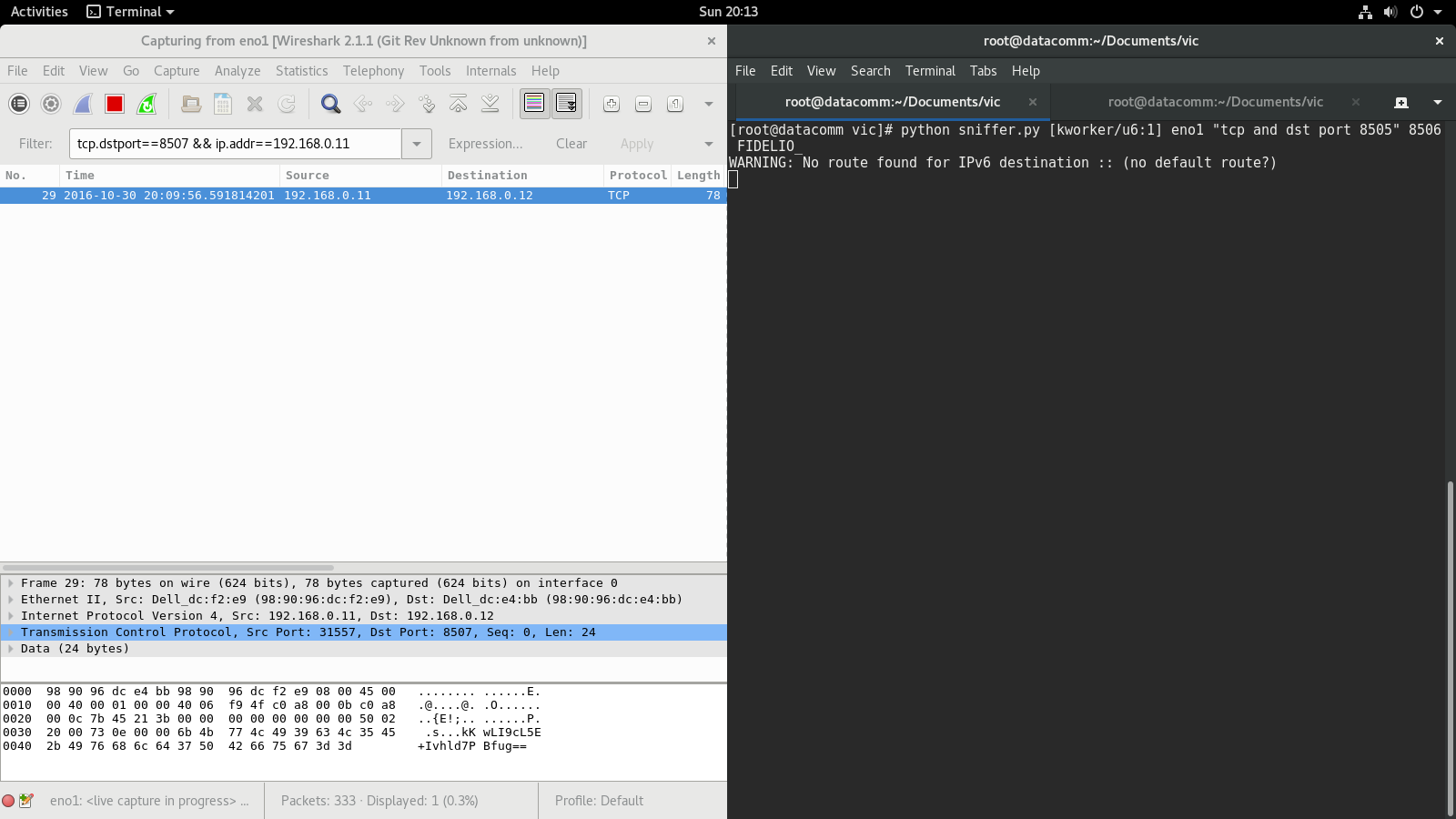
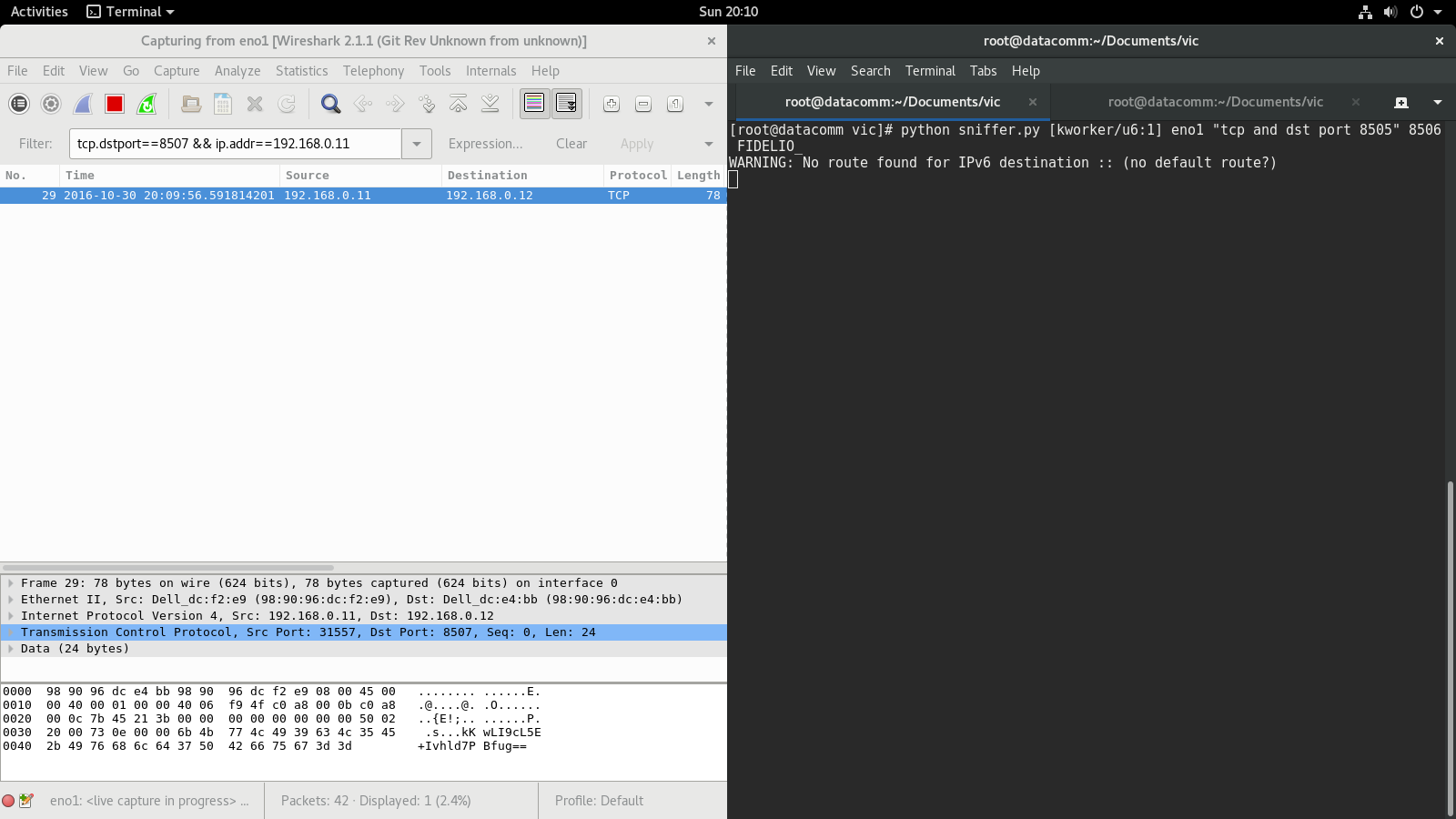


Figure 13 Message sent to the right port

**Observations**

* Most of the problems that I had in this program came with the filters, they need to be pretty specific

**Pseudocode**

**Command.py**

|  |
| --- |
| #! /usr/bin/env python  from scapy.all import \*  from random import randint  from cyphermod import \*  import sys  def pkt\_callback(pkt):  get current layers from pkt  if it doesn’t have valid layers, return  if packet is TCP packet:  If it’s not a syn packet, return  Get Payload from TCP payload  If packet is UDP packet:  Get Payload from Raw layer  If it’s neither TCP nor UDP:  Return  Decode payload with key  print decoded payload  if decoded payload equals end message  ask user input for command  if command length bigger than 0:  encode command  send command  else:  exit  Function validate\_ip receives ip:  If ip doesn’t contain 4 numbers separated by dots, and the numbers aren’t between 0 and 255 return False, otherwise return true  Function validate\_port receives port:  If port is not a digit nor between 0 and 65535 return false, otherwise return true  Function validate\_key receives key:  return false if key len is not between 8 and 32, if between return true  Main function:  Check if we have enough arguments, if not exit and error message  if any validations of arguments are false, return false and error message  assign arguments from command line to variables  convert key to 16, 24 or 32 characters  get a random port between 1024 and 65535  if packet is TCP packet:  set tcp protocol settings  if packet is UDP packet:  set udp protocol settings  ask for user input  if length of the command is bigger than 0:  encode user input  send user input  start sniffing packets |

**Sniffer.py**

|  |
| --- |
| Import Libraries  Function pkt\_callback receives pkt:  catch:  get current layers from pkt  if it doesn’t have valid layers, return  if packet is TCP packet:  If it’s not a syn packet, return  Get Payload from TCP payload  If packet is UDP packet:  Get Payload from Raw layer  If it’s neither TCP nor UDP:  return  decode and clean command  execute command and save result  divide result in blocks  for every block:  encode block  if packet is TCP packet:  set tcp protocol settings  if packet is UDP packet:  set udp protocol settings  send packet  Encode End of transmission message  Send End of transmission message  raise OSError:  Get Error message  Encode Error message  Send error message  Encode End of transmission message  Send End of transmission message  Function validate\_key receives key:  return false if key len is not between 8 and 32, if between return true  Main function  If there are not enough arguments, print error message  if validate\_key with the argument returns false, print error message an exit  assign arguments from command line to variables  convert key to 16, 24 or 32 characters  call function maskName with argument mask to change name  start sniffing for packets |

**Cyphermod.py**

|  |
| --- |
| Import libraries  function encode receives key and text:  create AES cypher based on key  encrypt text with cypher and encode in base64  return encoded text  function decode receives key and text:  create AES cypher based on key  decode text in base 64 and decrypt it with cypher  return decoded text |

**Maskname.py**

|  |
| --- |
| Import libraries  Function maskName receives name:  Mask name on console using prctl  Empty arguments so that they don’t appear in ps auxw  Replace arguments with sew name  Scale gid and uid privileges |