CYBER SECURITY RESEARCH & PRACTICE PLATFORM FOR FINANCIAL SERVICES

USER MANUAL-A2

Version 2.3

Feb/15/2017

Table of Contents

VERSION HISTORY

Version #	Implemented By	Revision Date	Reason
1.0	Simon, Ruoyi	08/29/2016	API specifications for A1 deliveries
1.1	Grace	08/31/2016	Revise, A1 deliveries
1.2	Simon, Ruoyi	09/07/2016	User instructions
1.3	Grace	09/09/2016	Revise
1.4	Simon	29/09/2016	User instruction update
1.5	Grace	29/09/2016	Revise
1.6	Harry	19/12/2016	API specifications for A2 deliveries
1.7	Harry	11/01/2017	Add Section 2.2
1.8	Harry	08/02/2017	Add Section 2.1, Section 4.4
1.9	Grace	10/02/2017	Revise Section 2
2.0	Hary	14/02/2017	Revise Section 2.3.4
2.1	Grace	15/02/2017	Revise
2.2	Simon, Grace	19/03/2017	A1 Manual (Update v 1.5, solve winpcap issue and thrift directory copy issue)
2.3	Grace	20/03/2017	A2 Manual

1 A2 DELIVERABLES: OPENVPN

We aim to generate simulated OpenVPN network traffic and build a traffic detector, which describes how likely a given piece of network traffic resembles an OpenVPN traffic.

1.1 OPENVPN GENERATOR

This generator is used for purposes of simulations, observations of OpenVPN key negotiation and OpenVPN packet detection. Our generator can generate the real traffic packets with OpenVPN protocols from one sender to one receiver. Furthermore, our generator can simulate multiple OpenVPN connections in one computer, where the user can set the number and type(s) of the traffic. The generated packets would not be sent out, but you can capture them through our OpenVPN detector or other packet sniffing tool like Wireshark. Because of the implementation details of OpenVPN, only the traffic following OpenVPN of 1-1 scenario (from one sender to one receiver) can be simulated, and the scenarios of multi-1 (from multiple senders to one receiver) and 1-multi (from one sender to multiple receivers) do not exist in real applications, which are not considered in our OpenVPN generator.

1.1.1 API specification

In A2 deliveries of the generator part, we develop one API, which is used in sender's side. And the receiver's side is not limited.

Method	Params	Remark	Position

`packet_generator`	`srcl, dstl,`
	`layerl,`
	`num_people`
		srcl[i], dstl[i] and layerl[i] contain information of ith OpenVPN connection.	*ts2/main/views/detetor.py*

Params	Values	Description
`srcl`	`list of string`	*Contains source IP address*
`dstl`	`list of string`	*Contains destination IP address which is OpenVPN server address*
`layerl`	`list of int`	*Contains type of OpenVPN traffic. 0 is udp and 1 is tcp.*
`num_people`	`int`	*Length of srcl, dstl and layerl*

1.1.2 Scenario

We consider the following four scenarios for generating the OpenVPN traffics:

1) Scenario 1: Generate the UDP OpenVPN packets from one sender to one receiver;
2) Scenario 2: Generate the UDP OpenVPN packets from multiple senders to multiple receivers;

3) Scenario 3: Generate the TCP OpenVPN packets from one sender to one receiver;
4) Scenario 4: Generate the TCP OpenVPN packets from multiple senders to multiple receivers;

Scenario 2 and 4 simulate the multiple OpenVPN connections.

1.2 OPENVPN DETECTOR

Our OpenVPN detector can judge if a packet is following OpenVPN protocol. The packet can be either monitored within a real-time internet traffic or is recorded by offline data.

Online detector can monitor a real-time internet traffic and respond the number of captured packet in the internet traffic. After monitor, system will display data and diagram to describe situation of OpenVPN traffic and give a link for captured OpenVPN packet. Online detector consists of two part: Packets capturer and packets detector.

We capture packet with help of scapy, which is a GNU licensed open-source python project from Github. For every captured packet, we use our own developed detector to analyse if it is an OpenVPN packet. Although accuracy can't reach 100%, it can detect OpenVPN packet which doesn't use an official 1194 port and reach relative high accuracy on OpenVPN packets with opcode 5, 7 and 8. Besides, OpenVPN detector will generate a log page which helps make a judgement.

1.2.1 API specifications

In A2 deliveries of the detector part, we develop two APIs.

Method	**Params**	**Remark**	**Position**
`judge_valid`	`pkt`	*Return a tag number of packet. 0 is non-OpenVPN packet. 1 is OpenVPN packet. 2 is suspicious OpenVPN packet.*	*ts2/main/views/detector.py*
`test_type`	`pkt`	*Return opcode of packet. 0 is non-OpenVPN packet.*	*ts2/main/views/detector.py*

Params	**Values**	**Description**
`pkt`	`Ether`	*A scapy format packet. Scapy packet has format like* Ether()/IP()/UDP()/Payload

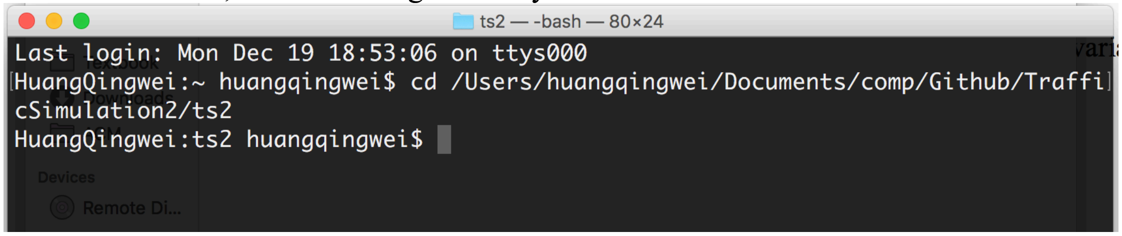
1.3 USER MANUAL (FOR MACOS&LINUX)

1.3.1 Configuring the environment

1. Install python2.7 from https://www.python.org or https://anaconda.org (recommended)
2. Install Django by pip (pip install Django==1.10.4)
3. Install netifaces-0.10.5 by pip (pip install netifaces)
4. Install scapy by pip (pip install scapy)
5. Install ecdsa by pip (pip install ecdsa)
6. Install libdnet-1.12 from https://github.com/todototry/scapyInstallDependents
7. Install PyX-0.10 from https://github.com/todototry/scapyInstallDependents
8. Install pypcap-1.1.4 from https://github.com/todototry/scapyInstallDependents
9. Install Django channel by pip (pip install -U channels)
10. Install redis server by pip (pip install asgi_redis)
11. Install Django-chartjs by pip (pip install django-chartjs)

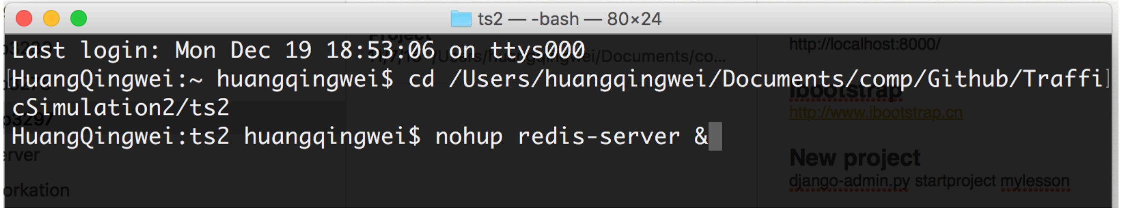
1.3.2 Using the system

1. In the terminal, make working directory to ts2.



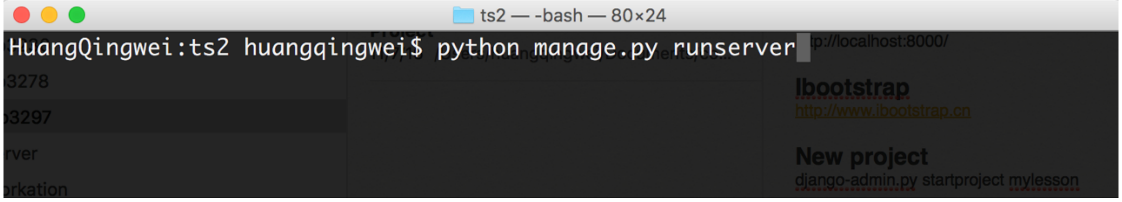
```
Last login: Mon Dec 19 18:53:06 on ttys000
[HuangQingwei:~ huangqingwei$ cd /Users/huangqingwei/Documents/comp/Github/Traffi]
cSimulation2/ts2
HuangQingwei:ts2 huangqingwei$ 
```

2. Open redis server.



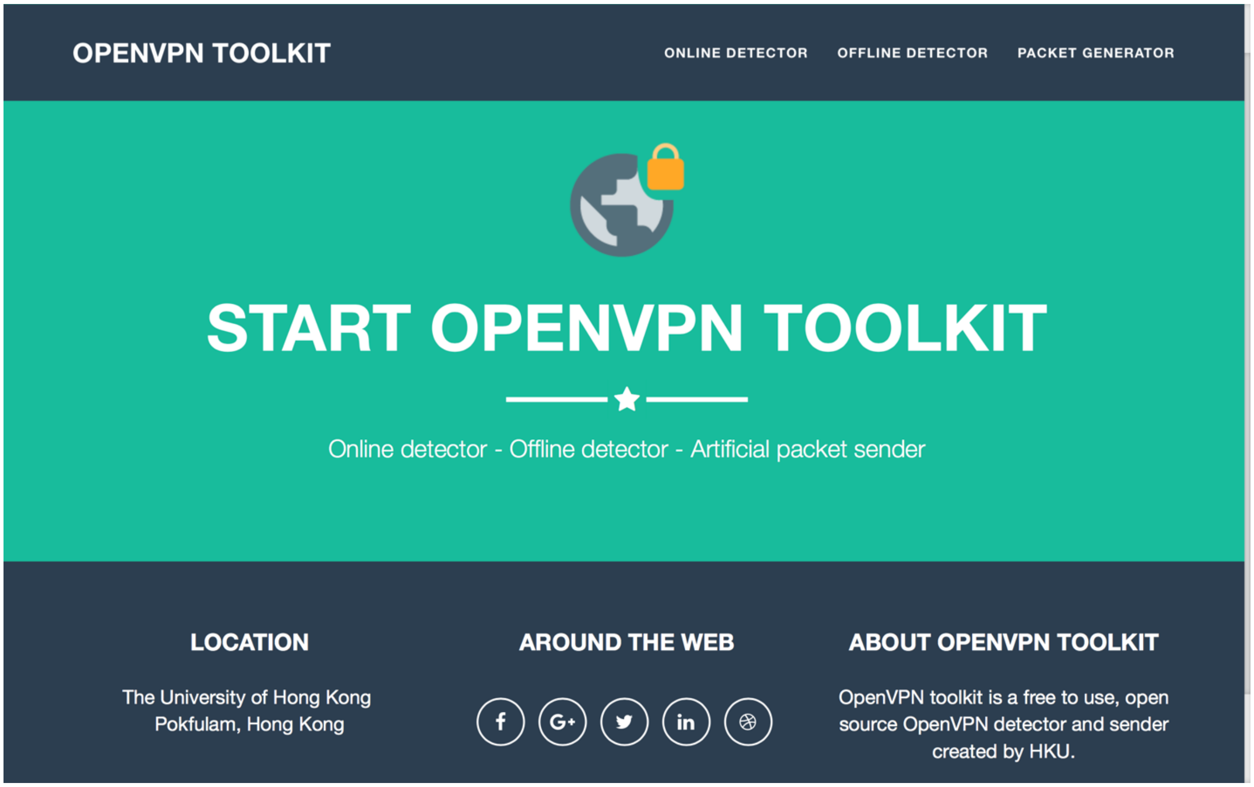
```
Last login: Mon Dec 19 18:53:06 on ttys000
[HuangQingwei:~ huangqingwei$ cd /Users/huangqingwei/Documents/comp/Github/Traffi]
cSimulation2/ts2
HuangQingwei:ts2 huangqingwei$ nohup redis-server &
```

3. Open Django server at local host.



```
HuangQingwei:ts2 huangqingwei$ python manage.py runserver
```

4. Visit localhost:8000 through your browser.



START OPENVPN TOOLKIT

Online detector - Offline detector - Artificial packet sender

LOCATION

The University of Hong Kong
Pokfulam, Hong Kong

AROUND THE WEB

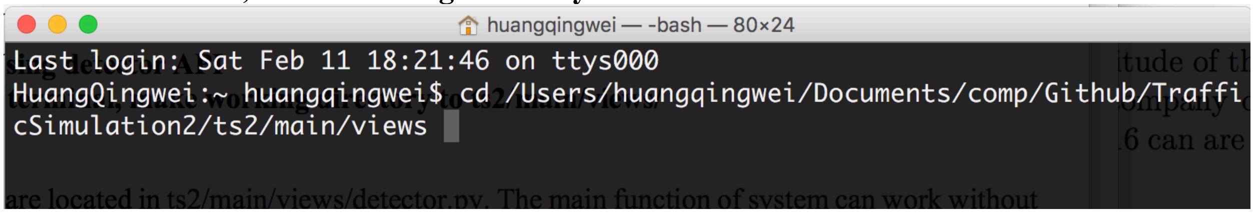
ABOUT OPENVPN TOOLKIT

OpenVPN toolkit is a free to use, open source OpenVPN detector and sender created by HKU.

1.3.3 Using the system in API

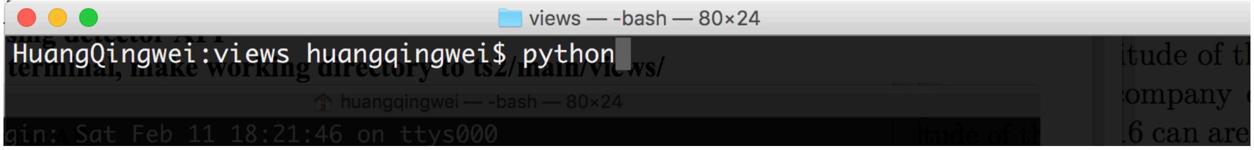
1.3.2.1 Using detector API

1. In the terminal, make working directory to ts2/main/views/



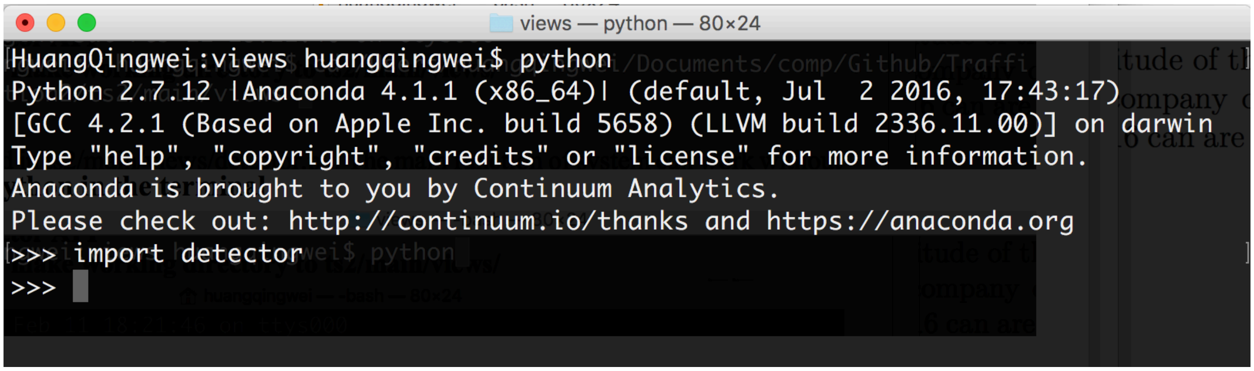
```
Last login: Sat Feb 11 18:21:46 on ttys000
HuangQingwei:~ huangqingwei$ cd /Users/huangqingwei/Documents/comp/Github/Traffi
cSimulation2/ts2/main/views
```

2. Run python in the terminal



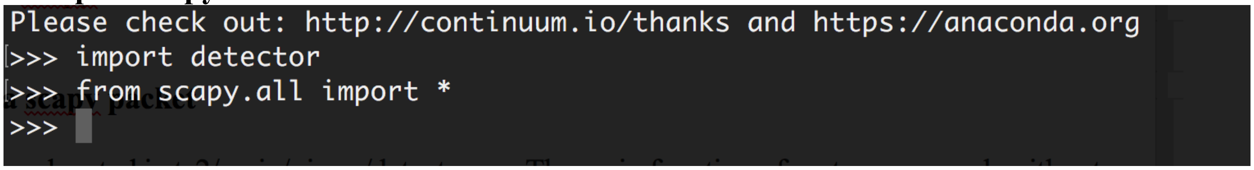
```
HuangQingwei:views huangqingwei$ python
```

3. Import detector



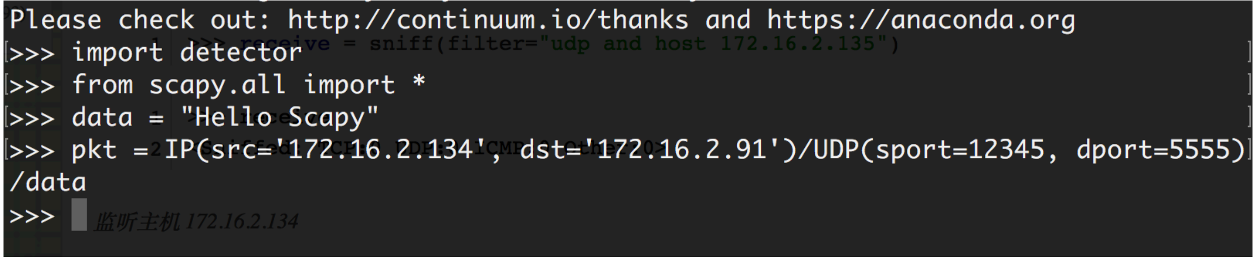
```
[HuangQingwei:views huangqingwei$ python
Python 2.7.12 |Anaconda 4.1.1 (x86_64)| (default, Jul  2 2016, 17:43:17)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
[>>> import detector
>>> 
```

4. Import scapy



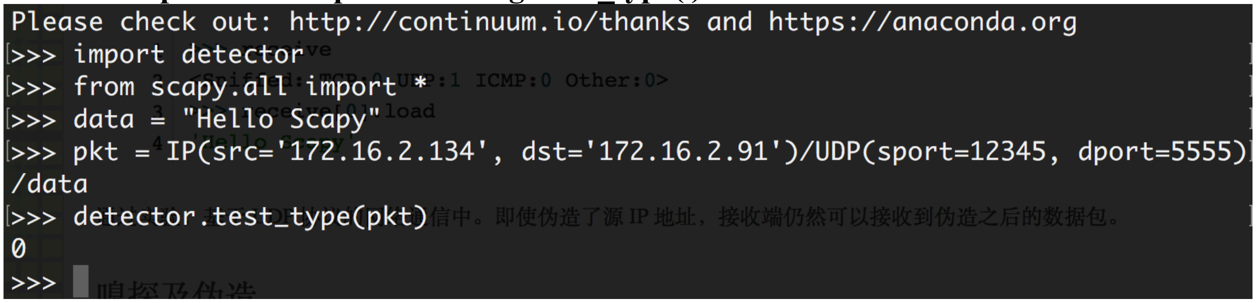
```
Please check out: http://continuum.io/thanks and https://anaconda.org
[>>> import detector
[>>> from scapy.all import *
>>> 
```

5. Make a scapy packet



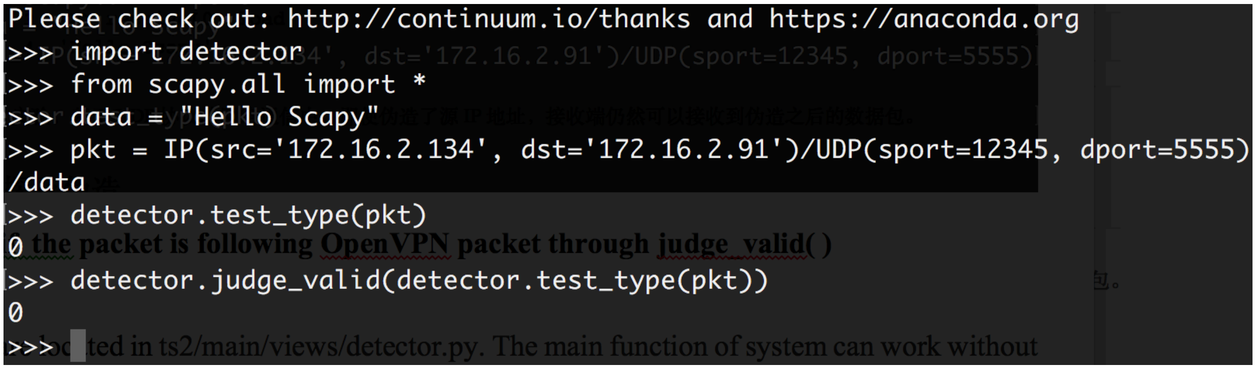
```
Please check out: http://continuum.io/thanks and https://anaconda.org
[>>> import detector
[>>> from scapy.all import *
[>>> data = "Hello Scapy"
[>>> pkt = IP(src='172.16.2.134', dst='172.16.2.91')/UDP(sport=12345, dport=5555)]
/data
>>> 
```

6. Get the opcode of the packet through test_type( )



```
Please check out: http://continuum.io/thanks and https://anaconda.org
[>>> import detector
[>>> from scapy.all import *
[>>> data = "Hello Scapy"
[>>> pkt = IP(src='172.16.2.134', dst='172.16.2.91')/UDP(sport=12345, dport=5555)]
/data
[>>> detector.test_type(pkt)
0
>>> 
```

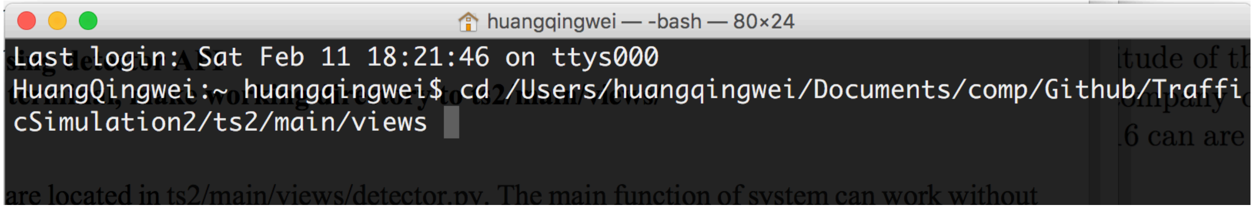
7. Judge if the packet is following OpenVPN packet through judge_valid( )



```
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import detector
>>> from scapy.all import *
>>> data = "Hello Scapy"
>>> pkt = IP(src='172.16.2.134', dst='172.16.2.91')/UDP(sport=12345, dport=5555)
/data
>>> detector.test_type(pkt)
0
>>> detector.judge_valid(detector.test_type(pkt))
0
>>> 
```

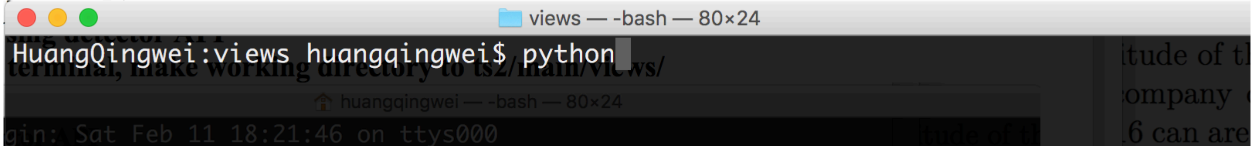
1.3.2.2 Using generator API

1. In the terminal, make working directory to ts2/main/views/



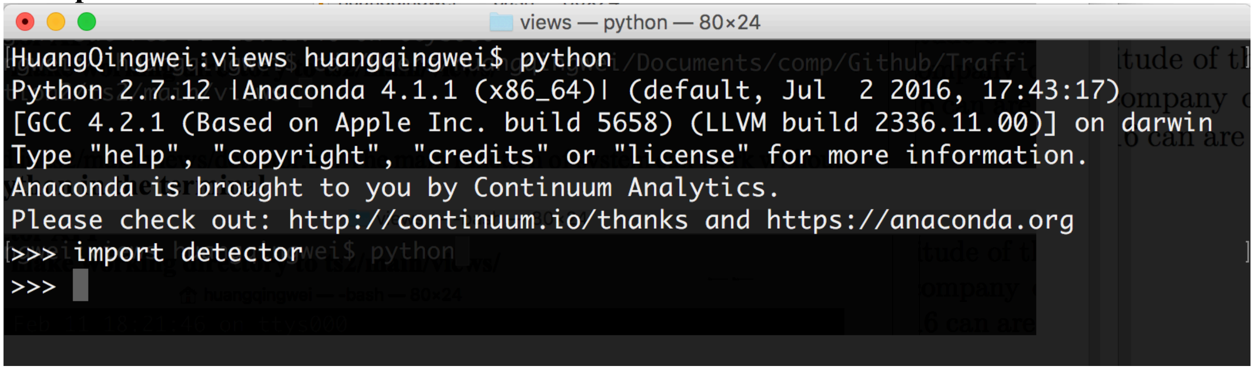
```
Last login: Sat Feb 11 18:21:46 on ttys000
HuangQingwei:~ huangqingwei$ cd /Users/huangqingwei/Documents/comp/Github/Traffi
cSimulation2/ts2/main/views 
```

2. Run python in the terminal



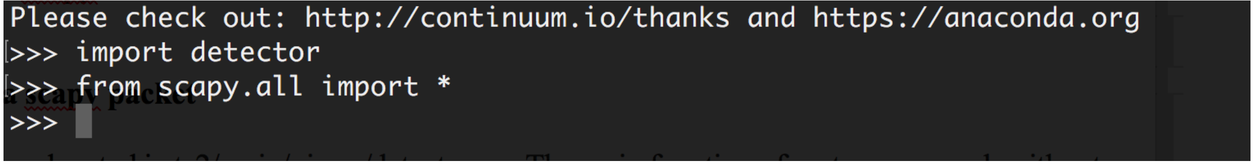
```
HuangQingwei:views huangqingwei$ python
```

3. Import detector



```
HuangQingwei:views huangqingwei$ python
Python 2.7.12 |Anaconda 4.1.1 (x86_64)| (default, Jul  2 2016, 17:43:17)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import detector
>>> 
```

4. Import scapy



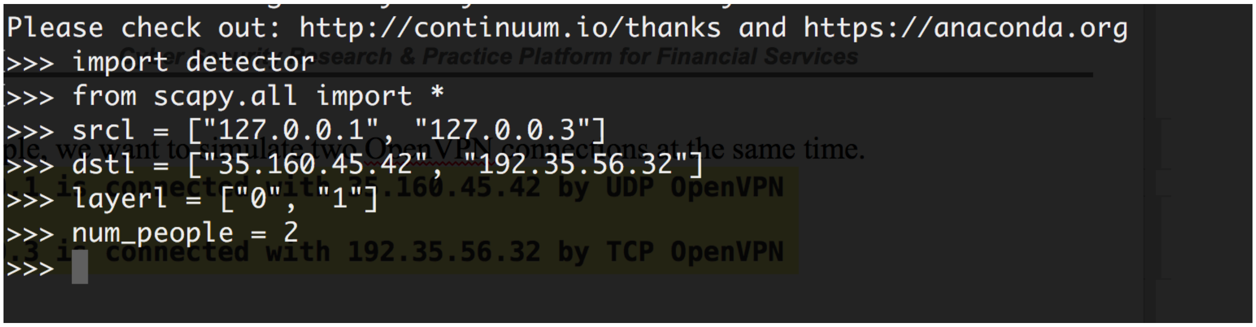
```
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import detector
>>> from scapy.all import *
>>> 
```

5. Prepare parameter

For example, we want to simulate two OpenVPN connections at the same time.

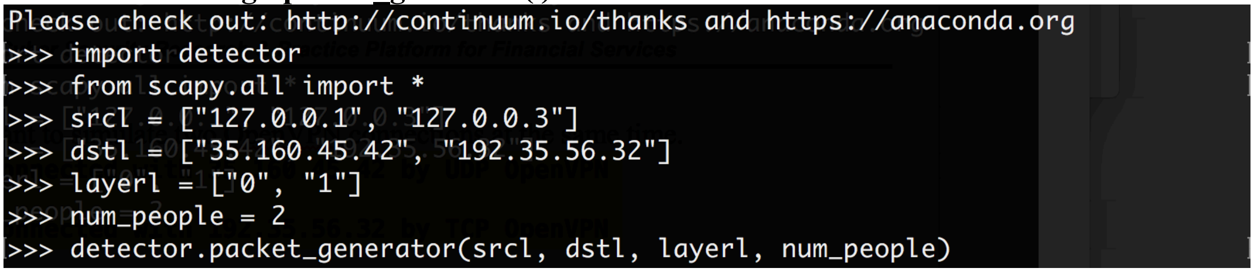
```
127.0.0.1 is connected with 35.160.45.42 by UDP OpenVPN

127.0.0.3 is connected with 192.35.56.32 by TCP OpenVPN
```

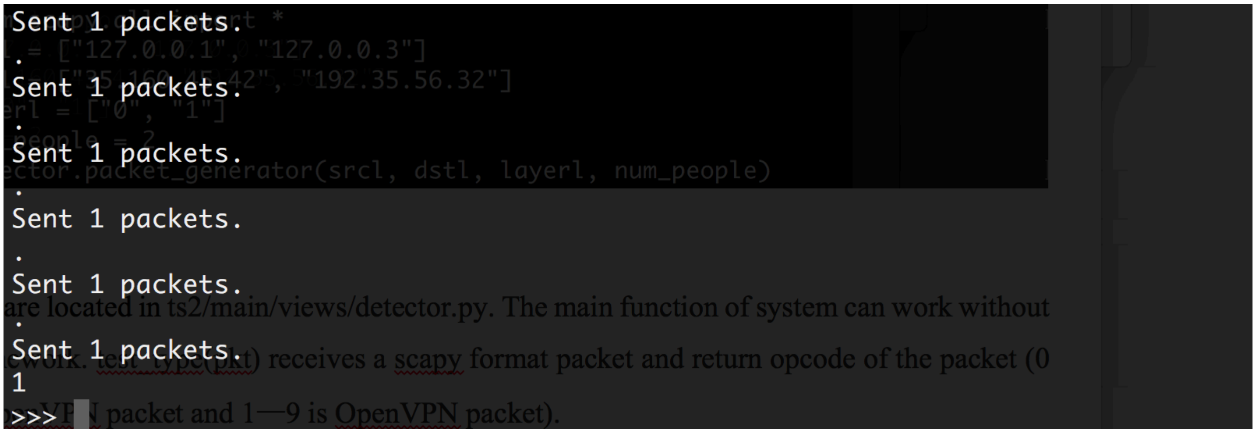


```
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import detector
>>> from scapy.all import *
>>> srcl = ["127.0.0.1", "127.0.0.3"]
>>> dstl = ["35.160.45.42", "192.35.56.32"]
>>> layerl = ["0", "1"]
>>> num_people = 2
>>>
```

6. Simulate through packet_generator( )

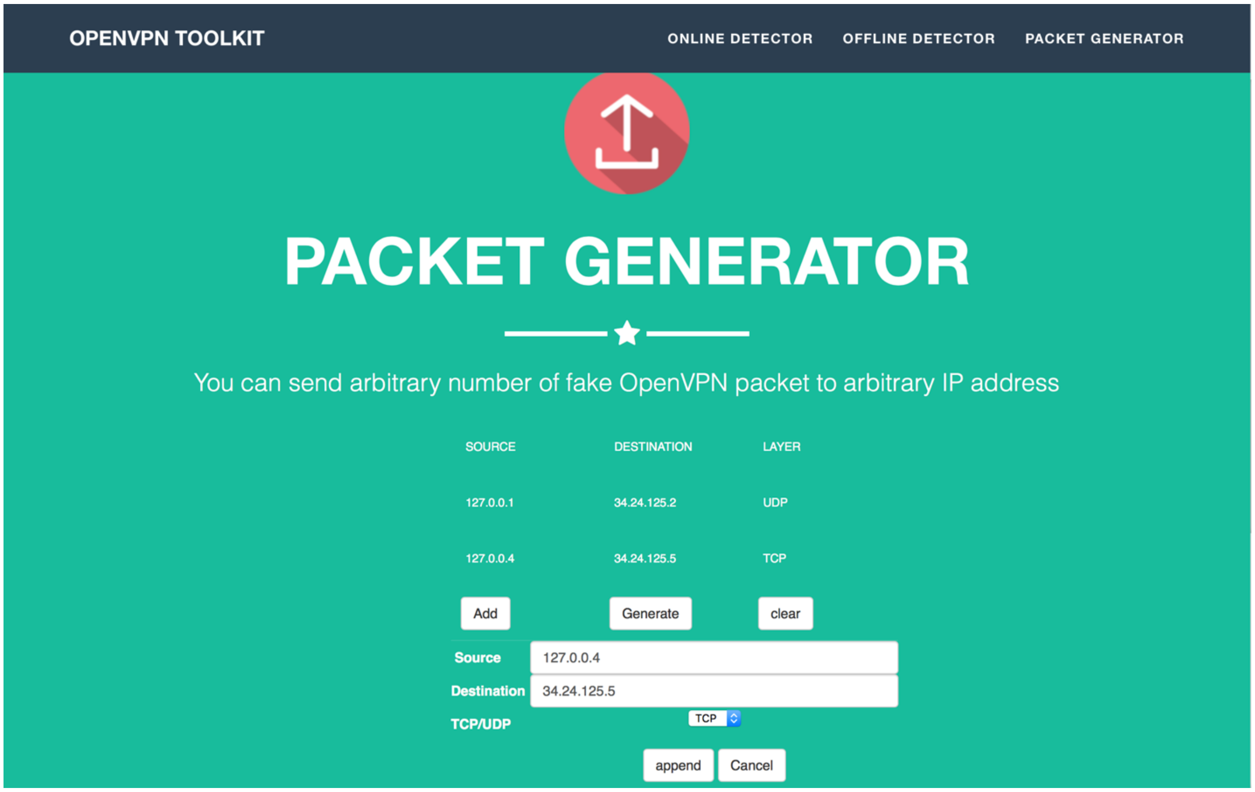


```
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import detector
>>> from scapy.all import *
>>> srcl = ["127.0.0.1", "127.0.0.3"]
>>> dstl = ["35.160.45.42", "192.35.56.32"]
>>> layerl = ["0", "1"]
>>> num_people = 2
>>> detector.packet_generator(srcl, dstl, layerl, num_people)
```



```
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
1
>>>
```

1.3.4 Results when using A2 APIs



PACKET GENERATOR

You can send arbitrary number of fake OpenVPN packet to arbitrary IP address

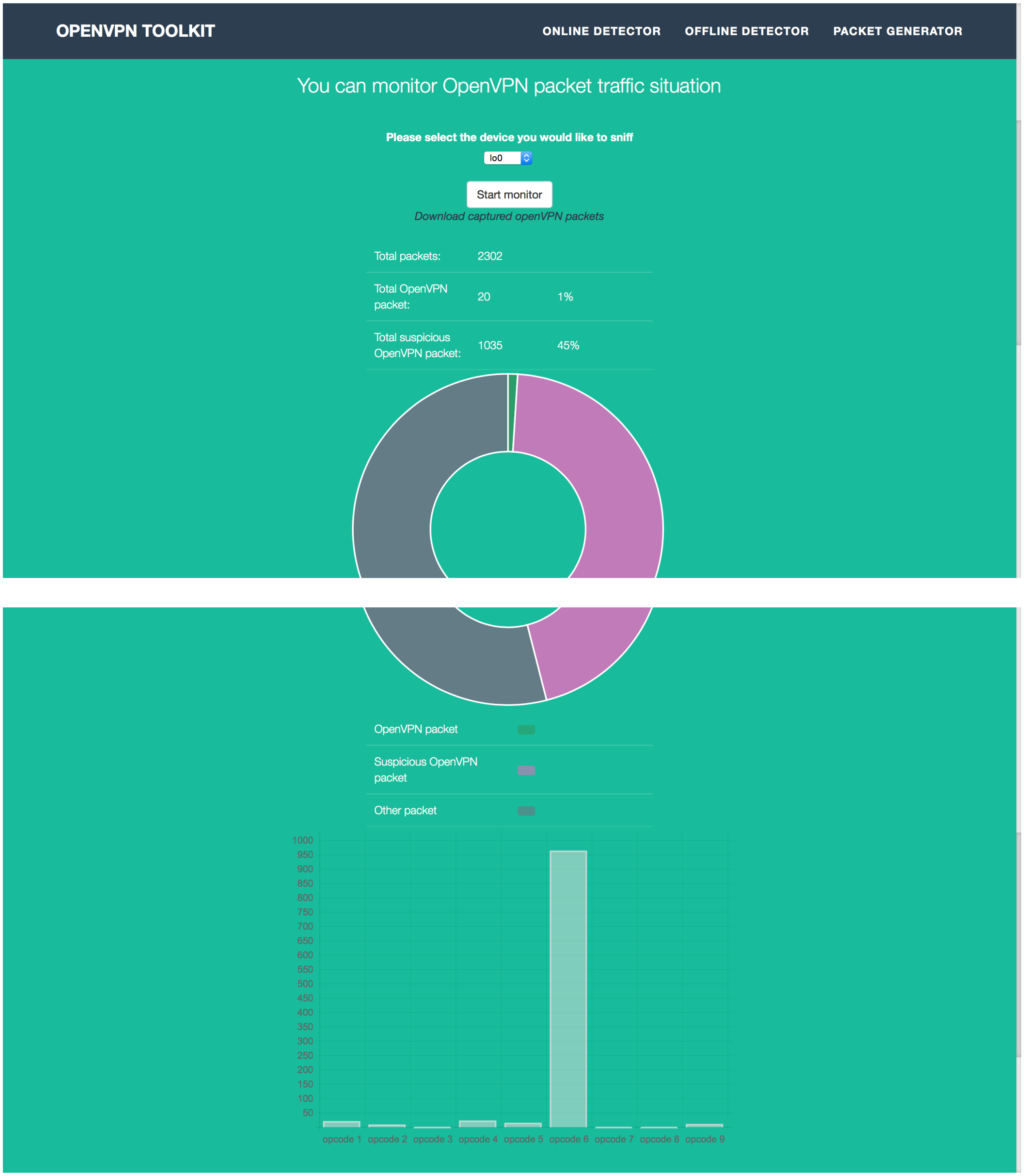
SOURCE	DESTINATION	LAYER
127.0.0.1	34.24.125.2	UDP
127.0.0.4	34.24.125.5	TCP

Source: 127.0.0.4

Destination: 34.24.125.5

TCP/UDP: TCP

Generator



OPENVPN TOOLKIT

You can monitor OpenVPN packet traffic situation

Please select the device you would like to sniff

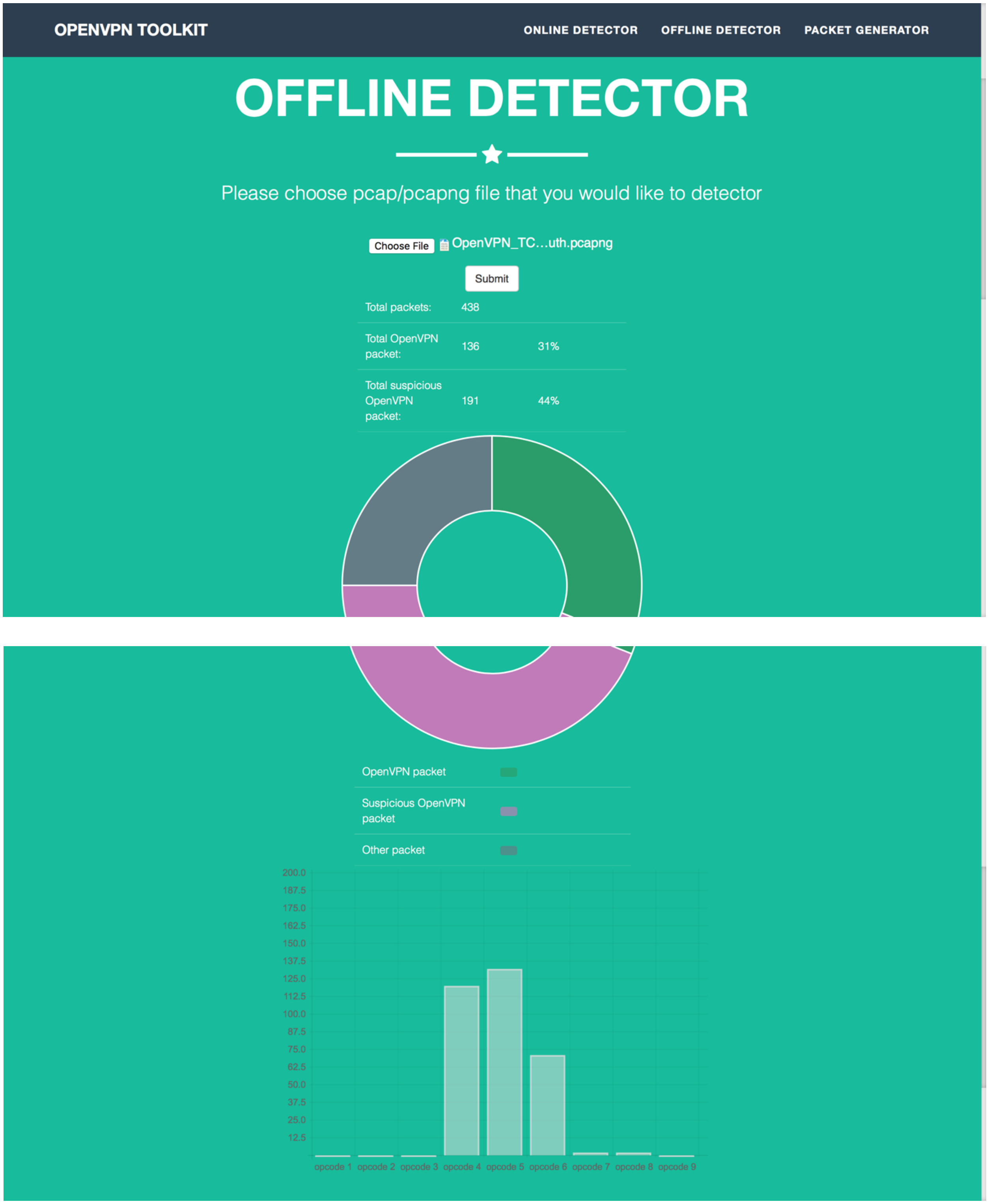
lo0

Download captured openVPN packets

Total packets:	2302	
Total OpenVPN packet:	20	1%
Total suspicious OpenVPN packet:	1035	45%

OpenVPN packet	🟩
Suspicious OpenVPN packet	🟪
Other packet	⬛

Online detector



Please choose pcap/pcapng file that you would like to detector

Total packets:	438	
Total OpenVPN packet:	136	31%
Total suspicious OpenVPN packet:	191	44%

Offline detector

1.3.5 Passed testing cases

We compute the recall as the performance matrix, where

$$recall = \frac{\text{# of OpenVPN packets detected by the detector}}{\text{# of OpenVPN packets}}.$$

Thus, the following table shows the performances of Wireshark and our OpenVPN detector when detecting OpenVPN packets. We can see that our detector can efficiently detect OpenVPN packets, especially for testing case 1-b and 2-b. The problem that Wireshark detects OpenVPN packets in a low recall is that Wireshark depends on a fixed official port number, 1194. However, in the real situations, the official port number is not fixed. Thus the OpenVPN packets with other port numbers are dismissed by Wireshark.

	Description of testing case	Testing data	Recall (Wireshark)	Recall (Our detector)
1-a	TCP OpenVPN packets	Total 438 TCP packets, which contains 327 real OpenVPN packets (Data source: Wireshark official website https://wiki.wireshark.org/OpenVPN)	100%	100%
1-b	TCP OpenVPN packets	Total 103 real OpenVPN packets, which contains 91 real OpenVPN packets (Data source: captured by scapy, we manually label the packet as either an OpenVPN packet or not)	0%	88.35%
2-a	UDP OpenVPN packets	Total 440 packets, which contains 439 real OpenVPN packets, 1 icmp packets (Data source: Wireshark official website https://wiki.wireshark.org/OpenVPN)	100%	100%
2-b	UDP OpenVPN packets	Total 122 packets, which contains 74 real OpenVPN packets (Data source: captured by scapy, we manually label the packet as either an OpenVPN packet or not)	0%	60.66%
3	Other protocol packets	Total 1608 non-openVPN packets, which contains 1385 UDP packets, 2 TCP packets, 1 OICP packet, 5 NBNS packets, 139 MDNS packets and etc. (Data source: captured by Wireshark)	100%	99.81%

2 DISCUSSIONS OF A2

2.1 THEORY OF DETECTING OPENVPN

The difficulty in OpenVPN detecting mainly relies on encrypted payload and poor information header. There are 9 types of OpenVPN packets in total.

P_CONTROL_HARD_RESET_CLIENT_V1
P_CONTROL_HARD_RESET_SERVER_V1
P_CONTROL_SOFT_RESET_V1
P_CONTROL_V1
P_ACK_V1
P_DATA_V1
P_CONTROL_HARD_RESET_CLIENT_V2
P_CONTROL_HARD_RESET_SERVER_V2
P_DATA_V2

Their headers are different. The header of P_CONTROL_HARD_RESET_SERVER_V2 has richest information which is

opcode 1 bytes
session ID 8 bytes
HMAC 20 bytes or 16 bytes
packet-id 4bytes
net time 4bytes
message packet-id array length 1bytes (n)
message packet-id array element 4*n bytes
remote session id 8 bytes
message packet-id 4 bytes

We first check if opcode is a valid value between [1, 9]. Then we calculate packet's length according to message packet-id array length and check if it is same as length described in udp header. However, for type like P_DATA_V1 whose header has poorest information like

opcode 1 bytes
encrypted data n bytes

It is hard to tell if it is an OpenVPN packet. Our detector works well on P_ACK_V1, P_CONTROL_HARD_RESET_CLIENT_V2, P_CONTROL_HARD_RESET_SERVER_V2 whose headers have more valuable information. It is meaningful to monitor an internet traffic because an OpenVPN traffic must contain those three type of packets.