Why Is HTTPS Important?

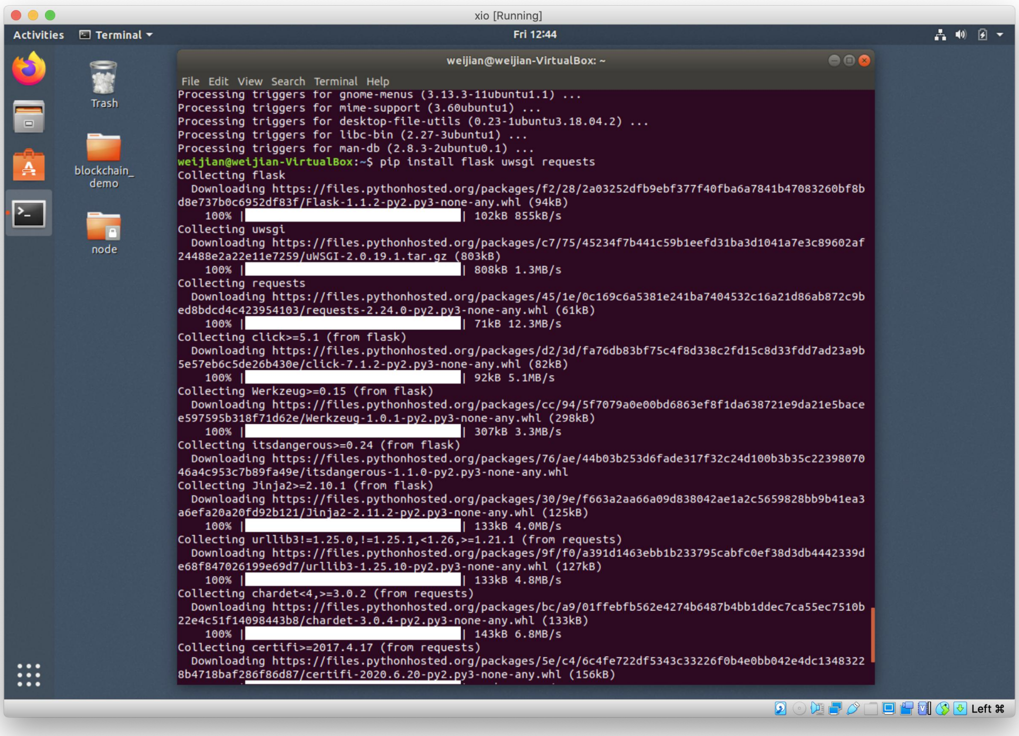
Secure communications are critical in providing a safe online environment. As more of the world moves online, including banks and healthcare sites, it's becoming more and more important for developers to create Python HTTPS applications. Again, HTTPS is just HTTP over TLS or SSL. TLS is designed to provide privacy from eavesdroppers. It can also provide authentication of both the client and the server.

In this section, you'll explore these concepts in depth by doing the following:

1. Creating a Python HTTPS server
2. Communicating with your Python HTTPS server
3. Capturing these communications
4. Analyzing those messages

Now let's creating an example application.

First we need to install some dependencies,

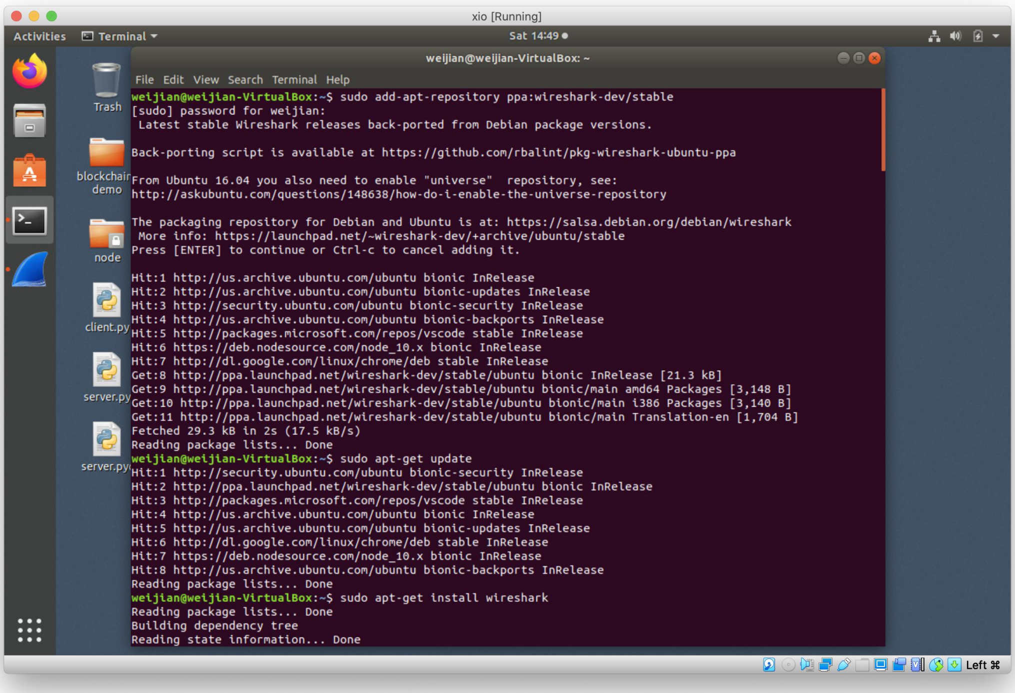


```
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
weijian@weijian-VirtualBox:~$ pip install flask uwsgi requests
Collecting flask
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8b
d8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 855kB/s
Collecting uwsgi
  Downloading https://files.pythonhosted.org/packages/c7/75/45234f7b441c59b1eefd31ba3d1041a7e3c89602af
24488e2a22e11e7259/uWSGI-2.0.19.1.tar.gz (803kB)
    100% |████████████████████████████████| 808kB 1.3MB/s
Collecting requests
  Downloading https://files.pythonhosted.org/packages/45/1e/0c169c6a5381e241ba7404532c16a21d86ab872c9b
ed8bdcd4c423954103/requests-2.24.0-py2.py3-none-any.whl (61kB)
    100% |████████████████████████████████| 71kB 12.3MB/s
Collecting click>=5.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b
5e57eb6c5de26b430e/click-7.1.2-py2.py3-none-any.whl (82kB)
    100% |████████████████████████████████| 92kB 5.1MB/s
Collecting Werkzeug>=0.15 (from flask)
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bace
e597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298kB)
    100% |████████████████████████████████| 307kB 3.3MB/s
Collecting itsdangerous>=0.24 (from flask)
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c22398070
46a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting Jinja2>=2.10.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/30/9e/f663a2aa66a09d838042ae1a2c5659828bb9b41ea3
a6efa20a20fd92b121/Jinja2-2.11.2-py2.py3-none-any.whl (125kB)
    100% |████████████████████████████████| 133kB 4.0MB/s
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 (from requests)
  Downloading https://files.pythonhosted.org/packages/9f/f0/a391d1463ebb1b233795cabfc0ef38d3db4442339d
e68f847026199e69d7/urllib3-1.25.10-py2.py3-none-any.whl (127kB)
    100% |████████████████████████████████| 133kB 4.8MB/s
Collecting chardet<4,>=3.0.2 (from requests)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b
22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |████████████████████████████████| 143kB 6.8MB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading https://files.pythonhosted.org/packages/5e/c4/6c4fe722df5343c33226f0b4e0bb042e4dc1348322
8b4718baf286f86d87/certifi-2020.6.20-py2.py3-none-any.whl (156kB)
```

Setting Up Wireshark

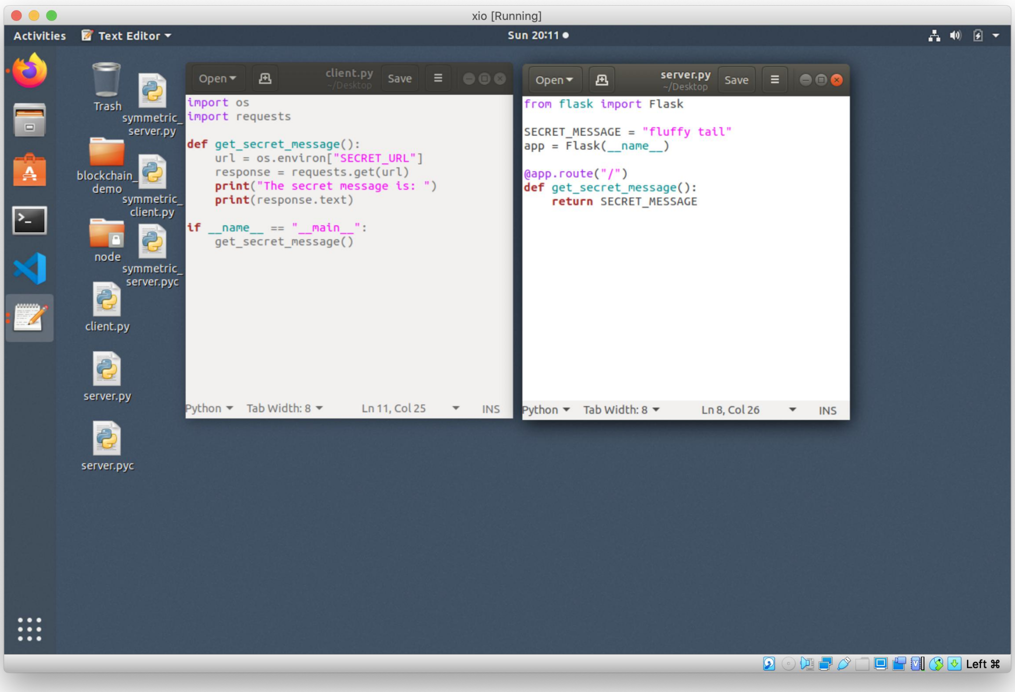
Wireshark is a widely used tool for network and protocol analysis. What this means is that it can help you see what's happening over network connections.

You can install Wireshark with the following commands:



```
weijian@weijian-VirtualBox:~$ sudo add-apt-repository ppa:wireshark-dev/stable
[sudo] password for weijian:
 Latest stable Wireshark releases back-ported from Debian package versions.

Back-porting script is available at https://github.com/rbalint/pkg-wireshark-ubuntu-ppa

From Ubuntu 16.04 you also need to enable "universe"  repository, see:
http://askubuntu.com/questions/148638/how-do-i-enable-the-universe-repository

The packaging repository for Debian and Ubuntu is at: https://salsa.debian.org/debian/wireshark
 More info: https://launchpad.net/~wireshark-dev/+archive/ubuntu/stable
Press [ENTER] to continue or Ctrl-c to cancel adding it.

Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:5 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:6 https://deb.nodesource.com/node_10.x bionic InRelease
Hit:7 http://dl.google.com/linux/chrome/deb stable InRelease
Get:8 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu bionic InRelease [21.3 kB]
Get:9 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu bionic/main amd64 Packages [3,148 B]
Get:10 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu bionic/main i386 Packages [3,140 B]
Get:11 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu bionic/main Translation-en [1,704 B]
Fetched 29.3 kB in 2s (17.5 kB/s)
Reading package lists... Done
weijian@weijian-VirtualBox:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu bionic InRelease
Hit:3 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:6 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:7 https://deb.nodesource.com/node_10.x bionic InRelease
Hit:8 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
weijian@weijian-VirtualBox:~$ sudo apt-get install wireshark
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

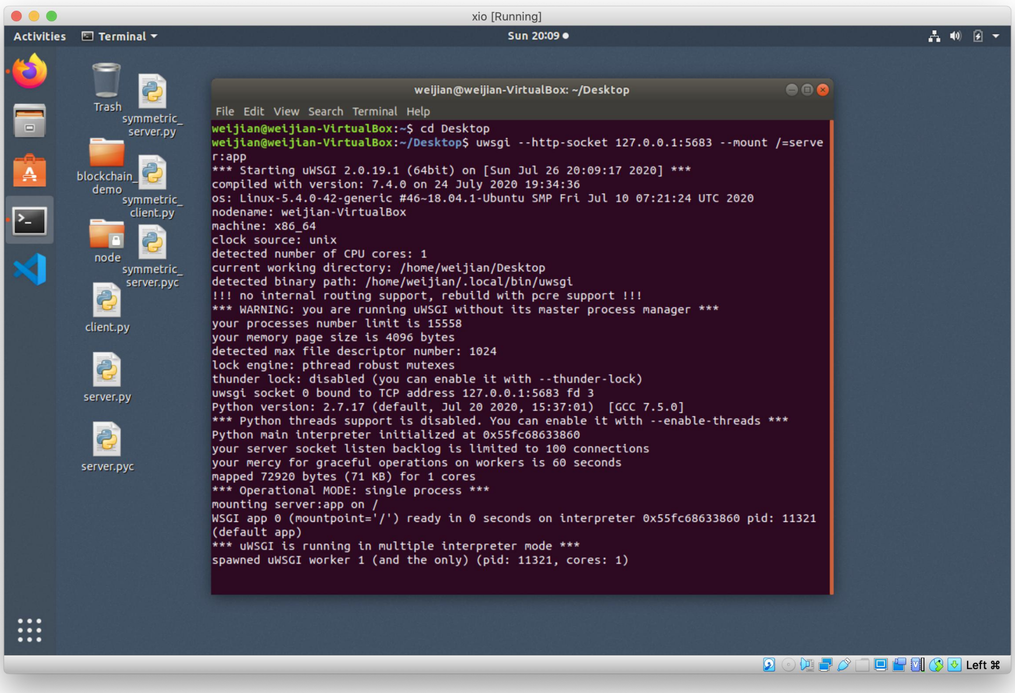
Now we can begin to write our application. We create server.py and client.py files.



```
import os
import requests

def get_secret_message():
    url = os.environ["SECRET_URL"]
    response = requests.get(url)
    print("The secret message is: ")
    print(response.text)

if __name__ == "__main__":
    get_secret_message()
```

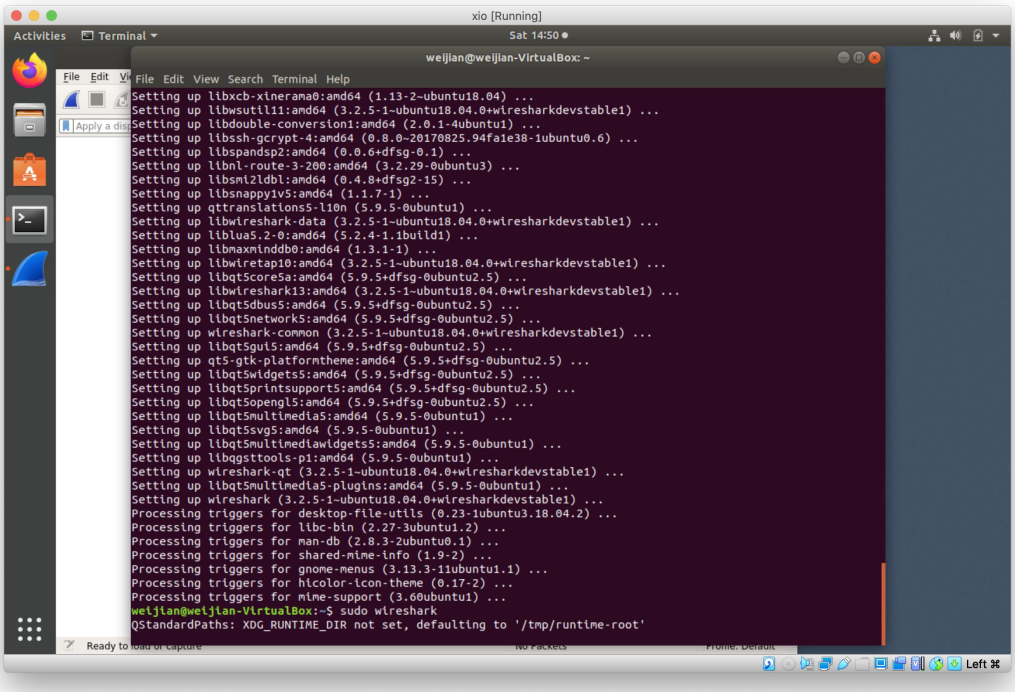
```
from flask import Flask

SECRET_MESSAGE = "fluffy tail"
app = Flask(__name__)

@app.route("/")
def get_secret_message():
    return SECRET_MESSAGE
```

This server file will display the secret message whenever someone visits the / path of your server. With that out of the way, you deploy your application on your secret server and run it:



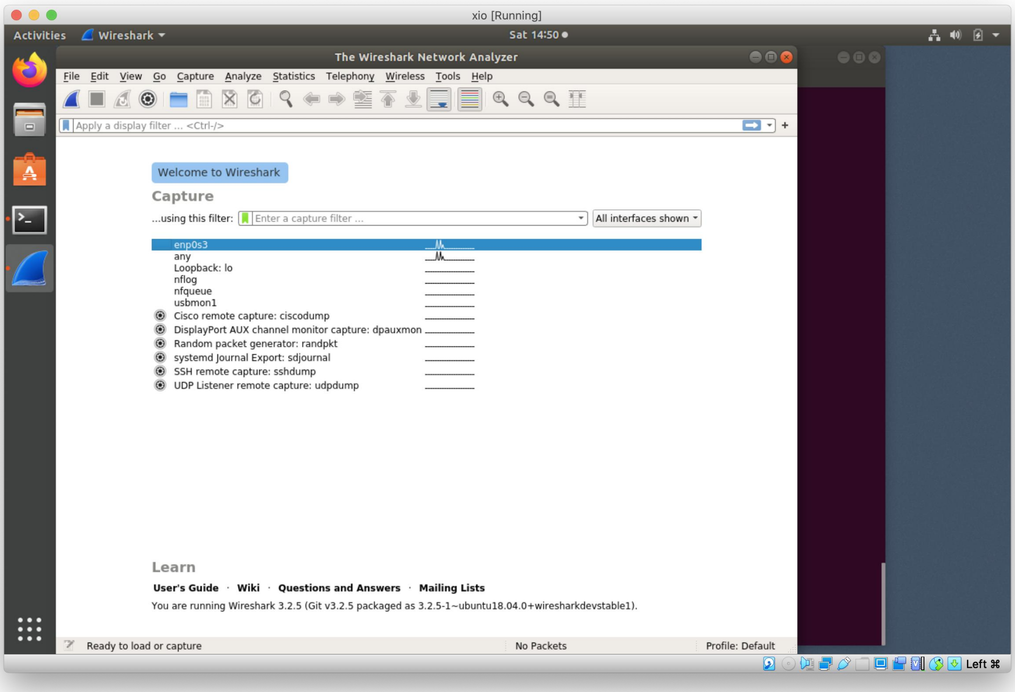
```
weijian@weijian-VirtualBox:~$ cd Desktop
weijian@weijian-VirtualBox:~/Desktop$ uwsgi --http-socket 127.0.0.1:5683 --mount /=serve
r:app
*** Starting uWSGI 2.0.19.1 (64bit) on [Sun Jul 26 20:09:17 2020] ***
compiled with version: 7.4.0 on 24 July 2020 19:34:36
os: Linux-5.4.0-42-generic #46~18.04.1-Ubuntu SMP Fri Jul 10 07:21:24 UTC 2020
nodename: weijian-VirtualBox
machine: x86_64
clock source: unix
detected number of CPU cores: 1
current working directory: /home/weijian/Desktop
detected binary path: /home/weijian/.local/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 15558
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 127.0.0.1:5683 fd 3
Python version: 2.7.17 (default, Jul 20 2020, 15:37:01)  [GCC 7.5.0]
*** Python threads support is disabled. You can enable it with --enable-threads ***
Python main interpreter initialized at 0x55fc68633860
your server socket listen backlog is limited to 100 connections
your mercy for graceful operations on workers is 60 seconds
mapped 72920 bytes (71 KB) for 1 cores
*** Operational MODE: single process ***
mounting server:app on /
WSGI app 0 (mountpoint='/') ready in 0 seconds on interpreter 0x55fc68633860 pid: 11321
(default app)
*** uWSGI is running in multiple interpreter mode ***
spawned uWSGI worker 1 (and the only) (pid: 11321, cores: 1)
```

Then we open Wireshark,



```
Setting up libxcb-xinerama0:amd64 (1.13-2~ubuntu18.04) ...
Setting up libwsutil11:amd64 (3.2.5-1~ubuntu18.04.0+wiresharkdevstable1) ...
Setting up libdouble-conversion1:amd64 (2.0.1-4ubuntu1) ...
Setting up libssh-gcrypt-4:amd64 (0.8.0~20170825.94fa1e38-1ubuntu0.6) ...
Setting up libspandsp2:amd64 (0.0.6+dfsg-0.1) ...
Setting up libnl-route-3-200:amd64 (3.2.29-0ubuntu3) ...
Setting up libsmi2ldbl:amd64 (0.4.8+dfsg2-15) ...
Setting up libsnappy1v5:amd64 (1.1.7-1) ...
Setting up qttranslations5-l10n (5.9.5-0ubuntu1) ...
Setting up libwireshark-data (3.2.5-1~ubuntu18.04.0+wiresharkdevstable1) ...
Setting up liblua5.2-0:amd64 (5.2.4-1.1build1) ...
Setting up libmaxminddb0:amd64 (1.3.1-1) ...
Setting up libwiretap10:amd64 (3.2.5-1~ubuntu18.04.0+wiresharkdevstable1) ...
Setting up libqt5core5a:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up libwireshark13:amd64 (3.2.5-1~ubuntu18.04.0+wiresharkdevstable1) ...
Setting up libqt5dbus5:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up libqt5network5:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up wireshark-common (3.2.5-1~ubuntu18.04.0+wiresharkdevstable1) ...
Setting up libqt5gui5:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up qt5-gtk-platformtheme:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up libqt5widgets5:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up libqt5printsupport5:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up libqt5opengl5:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
Setting up libqt5multimedia5:amd64 (5.9.5-0ubuntu1) ...
Setting up libqt5svg5:amd64 (5.9.5-0ubuntu1) ...
Setting up libqt5multimediawidgets5:amd64 (5.9.5-0ubuntu1) ...
Setting up libqgsttools-p1:amd64 (5.9.5-0ubuntu1) ...
Setting up wireshark-qt (3.2.5-1~ubuntu18.04.0+wiresharkdevstable1) ...
Setting up libqt5multimedia5-plugins:amd64 (5.9.5-0ubuntu1) ...
Setting up wireshark (3.2.5-1~ubuntu18.04.0+wiresharkdevstable1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for libc-bin (2.27-3ubuntu1.2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for shared-mime-info (1.9-2) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for mime-support (3.60ubuntu1) ...
weijian@weijian-VirtualBox:~$ sudo wireshark
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

You should be met with a screen that looks something like this below. This will start up your Flask application on port 5683. Next, you'll start a packet capture in Wireshark. This packet capture will help you see all the traffic going to and from the server. Begin by selecting the Loopback:lo interface on Wireshark:


xio [Running]
Activities
Wireshark
Sat 14:50
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Left ⌘

Apply a display filter … <Ctrl-/>

Welcome to Wireshark

Capture

…using this filter: Enter a capture filter … All interfaces shown

```
enp0s3
any
Loopback: lo
nflog
nfqueue
usbmon1
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
```

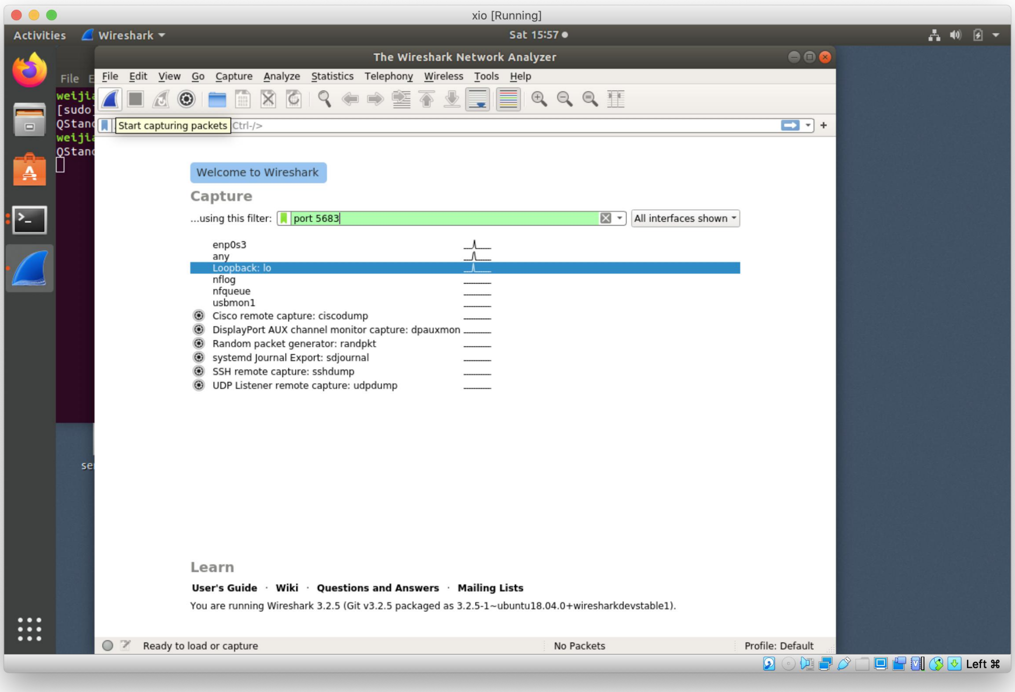
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 3.2.5 (Git v3.2.5 packaged as 3.2.5-1~ubuntu18.04.0+wiresharkdevstable1).

Ready to load or capture | No Packets | Profile: Default

You can see that the Loopback:lo portion is highlighted. This instructs Wireshark to monitor this port for traffic. You can do better and specify which port and protocol you'd like to capture. You can type port 5683 in the capture filter and http in the display filter:


xio [Running]
Activities
Wireshark
Sat 15:57
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Start capturing packets
Left ⌘

```
weijt
[sudo
QStan
weijt
QStan
```

Ctrl-/>

Welcome to Wireshark

Capture

…using this filter: port 5683 All interfaces shown

```
enp0s3
any
Loopback: lo
nflog
nfqueue
usbmon1
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
```

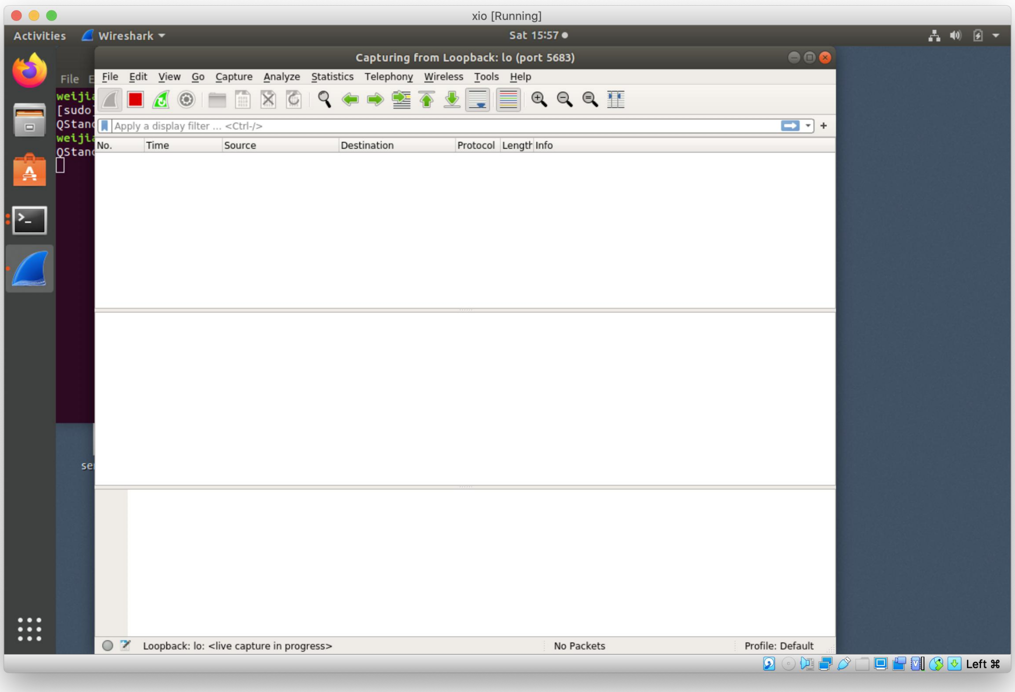
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 3.2.5 (Git v3.2.5 packaged as 3.2.5-1~ubuntu18.04.0+wiresharkdevstable1).

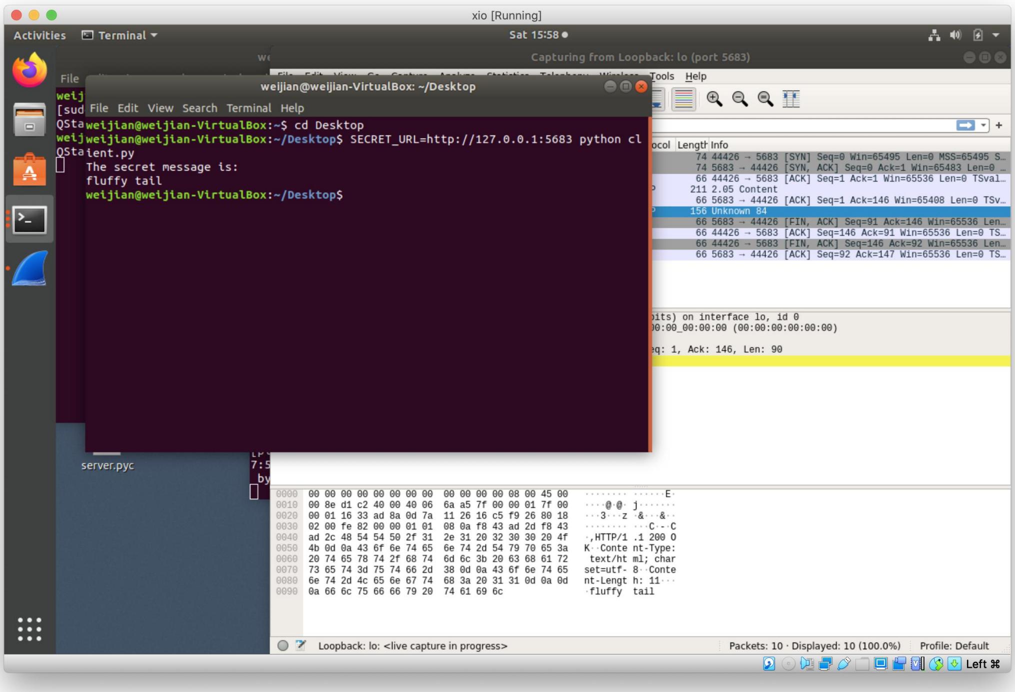
Ready to load or capture | No Packets | Profile: Default

The green box indicates that Wireshark is happy with the filter you typed. Now you can begin the capture by clicking on the fin in the top left:


xio [Running]
Activities
Wireshark
Sat 15:57
Capturing from Loopback: lo (port 5683)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
Profile: Default
Left ⌘

```
Loopback: lo: <live capture in progress>          No Packets
```

This new window is fairly plain, but the message at the bottom says <live capture in progress>, which indicates that it's working. Don't worry that nothing is being displayed, as that's normal. In order for Wireshark to report anything, there has to be some activity on your server. To get some data, try running your client:


xio [Running]
Activities
Terminal
Sat 15:58
weijian@weijian-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
Capturing from Loopback: lo (port 5683)
Tools Help
server.pyc
Profile: Default
Left ⌘

```
weijian@weijian-VirtualBox:~$ cd Desktop
weijian@weijian-VirtualBox:~/Desktop$ SECRET_URL=http://127.0.0.1:5683 python cl
ient.py
The secret message is:
fluffy tail
weijian@weijian-VirtualBox:~/Desktop$
```

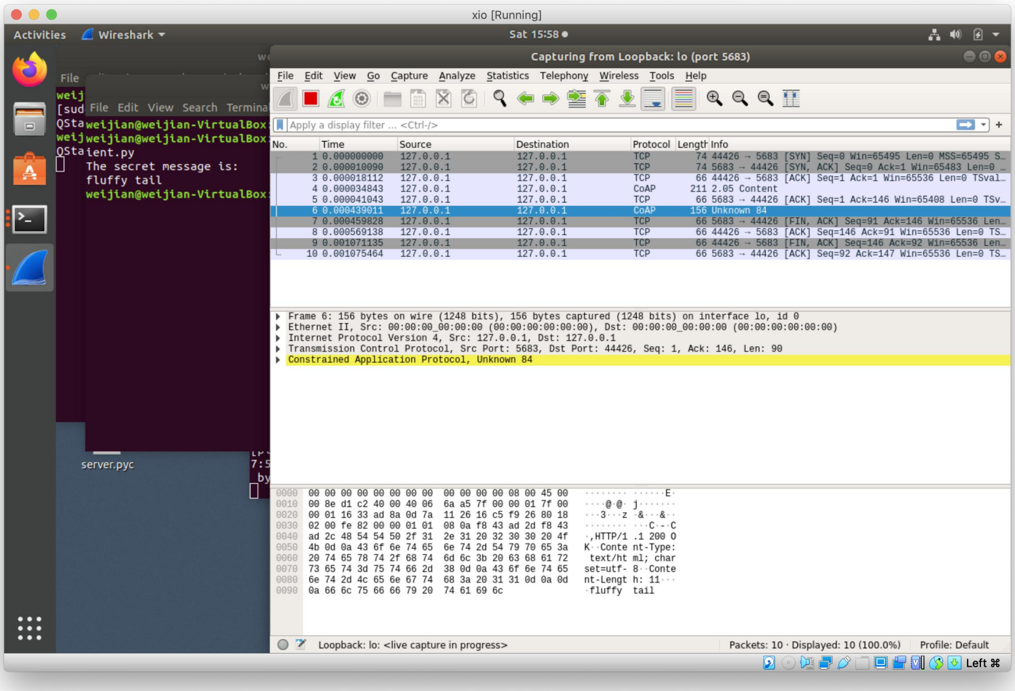
```
ocol  Length Info
         74 44426 → 5683 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 S…
         74 5683 → 44426 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 …
         66 44426 → 5683 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval…
        211 2.05 Content
         66 5683 → 44426 [ACK] Seq=1 Ack=146 Win=65408 Len=0 TSv…
        156 Unknown 84
         66 5683 → 44426 [FIN, ACK] Seq=91 Ack=146 Win=65536 Len…
         66 44426 → 5683 [ACK] Seq=146 Ack=91 Win=65536 Len=0 TS…
         66 44426 → 5683 [FIN, ACK] Seq=146 Ack=92 Win=65536 Len…
         66 5683 → 44426 [ACK] Seq=92 Ack=147 Win=65536 Len=0 TS…
```

```
bits) on interface lo, id 0
00:00_00:00:00 (00:00:00:00:00:00)

eq: 1, Ack: 146, Len: 90
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 8e d1 c2 40 00 40 06  6a a5 7f 00 00 01 7f 00   ····@·@· j·······
0020  00 01 16 33 ad 8a 0d 7a  11 26 16 c5 f9 26 80 18   ···3···z ·&···&··
0030  02 00 fe 82 00 00 01 01  08 0a f8 43 ad 2d f8 43   ········ ···C·-·C
0040  ad 2c 48 54 54 50 2f 31  2e 31 20 32 30 30 20 4f   ·,HTTP/1 .1 200 O
0050  4b 0d 0a 43 6f 6e 74 65  6e 74 2d 54 79 70 65 3a   K··Conte nt-Type:
0060  20 74 65 78 74 2f 68 74  6d 6c 3b 20 63 68 61 72    text/ht ml; char
0070  73 65 74 3d 75 74 66 2d  38 0d 0a 43 6f 6e 74 65   set=utf- 8··Conte
0080  6e 74 2d 4c 65 6e 67 74  68 3a 20 31 31 0d 0a 0d   nt-Lengt h: 11···
0090  0a 66 6c 75 66 66 79 20  74 61 69 6c               ·fluffy  tail
```

```
Loopback: lo: <live capture in progress>     Packets: 10 · Displayed: 10 (100.0%)
```

After executing the client.py code from above, you should now see some entries in Wireshark. If all has gone well, then you'll see some entries that look something like this:


xio [Running]
Activities
Wireshark
Sat 15:58
Capturing from Loopback: lo (port 5683)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
server.pyc

```
weijian@weijian-VirtualBox:
weijian@weijian-VirtualBox:
ient.py
The secret message is:
fluffy tail
weijian@weijian-VirtualBox:
```

```
No. Time        Source     Destination Protocol Length Info
  1 0.000000000 127.0.0.1  127.0.0.1   TCP       74 44426 → 5683 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 S…
  2 0.000010090 127.0.0.1  127.0.0.1   TCP       74 5683 → 44426 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 …
  3 0.000018112 127.0.0.1  127.0.0.1   TCP       66 44426 → 5683 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval…
  4 0.000034843 127.0.0.1  127.0.0.1   CoAP     211 2.05 Content
  5 0.000041043 127.0.0.1  127.0.0.1   TCP       66 5683 → 44426 [ACK] Seq=1 Ack=146 Win=65408 Len=0 TSv…
  6 0.000439011 127.0.0.1  127.0.0.1   CoAP     156 Unknown 84
  7 0.000459828 127.0.0.1  127.0.0.1   TCP       66 5683 → 44426 [FIN, ACK] Seq=91 Ack=146 Win=65536 Len…
  8 0.000569138 127.0.0.1  127.0.0.1   TCP       66 44426 → 5683 [ACK] Seq=146 Ack=91 Win=65536 Len=0 TS…
  9 0.001071135 127.0.0.1  127.0.0.1   TCP       66 44426 → 5683 [FIN, ACK] Seq=146 Ack=92 Win=65536 Len…
 10 0.001075464 127.0.0.1  127.0.0.1   TCP       66 5683 → 44426 [ACK] Seq=92 Ack=147 Win=65536 Len=0 TS…
```

```
▶ Frame 6: 156 bytes on wire (1248 bits), 156 bytes captured (1248 bits) on interface lo, id 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 5683, Dst Port: 44426, Seq: 1, Ack: 146, Len: 90
▶ Constrained Application Protocol, Unknown 84
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 8e d1 c2 40 00 40 06  6a a5 7f 00 00 01 7f 00   ····@·@· j·······
0020  00 01 16 33 ad 8a 0d 7a  11 26 16 c5 f9 26 80 18   ···3···z ·&···&··
0030  02 00 fe 82 00 00 01 01  08 0a f8 43 ad 2d f8 43   ········ ···C·-·C
0040  ad 2c 48 54 54 50 2f 31  2e 31 20 32 30 30 20 4f   ·,HTTP/1 .1 200 O
0050  4b 0d 0a 43 6f 6e 74 65  6e 74 2d 54 79 70 65 3a   K··Conte nt-Type:
0060  20 74 65 78 74 2f 68 74  6d 6c 3b 20 63 68 61 72    text/ht ml; char
0070  73 65 74 3d 75 74 66 2d  38 0d 0a 43 6f 6e 74 65   set=utf- 8··Conte
0080  6e 74 2d 4c 65 6e 67 74  68 3a 20 31 31 0d 0a 0d   nt-Lengt h: 11···
0090  0a 66 6c 75 66 66 79 20  74 61 69 6c               ·fluffy  tail
```

```
Loopback: lo: <live capture in progress>          Packets: 10 · Displayed: 10 (100.0%)    Profile: Default
```

These two entries represent the several parts of the communication that occurred. When you click the one that I highlighted, you'll see the communication content, "fluffy tail".

If you look carefully, then you'll see the secret message in plain text! This is a big problem for the Secret Squirrels. What this means is that anyone with some technical know-how can very easily see this traffic if they're interested. So, how do you solve this problem? The answer is cryptography.