Ethical hacking with Python tools

JOSE MANUEL ORTEGA
@JMOlORTEGAC
• Introduction Python pentesting
• Modules(Sockets, Requests, BeautifulSoup, Shodan)
• Analysis metadata
• Port scanning & Checking vulnerabilities
• Advanced tools
• Pentesting-tool
Python Pentesting

- Multi platform
- Prototypes and proofs of concept (POC)
- Many tools and libraries focused on security
- OSINT and Pentesting tools
- Very good documentation
Python Pentesting

**sqlmap**

Automatic SQL injection and database takeover tool

**Introduction**

sqlmap is an open source penetration testing tool for discovering, exploiting, and exploiting SQL injection flaws and taking over of databases. It features many niche features for the ultimate penetration testing experience such as SQL fingerprinting, data fetching from the database, and executing commands on the operating system via an injected shell.
http://sparta.secforce.com/
Usage: theharvester options

-d: Domain to search or company name
-b: data source: google, googleCSE, bing, bingapi, pgp, linkedin, google-profiles, jigsaw, twitter, googleplus, all
-s: Start in result number X (default: 0)
-v: Verify host name via dns resolution and search for virtual hosts
-f: Save the results into an HTML and XML file
-n: Perform a DNS reverse query on all ranges discovered
-c: Perform a DNS brute force for the domain name
-t: Perform a DNS TLD expansion discovery
-e: Use this DNS server
-l: Limit the number of results to work with (bing goes from 50 to 50 results, google 100 to 100, and pgp doesn’t use this option)
-h: use SHODAN database to query discovered hosts

Examples:
theHarvester.py -d microsoft.com -l 500 -b google
theHarvester.py -d microsoft.com -b pgp
theHarvester.py -d microsoft -l 200 -b linkedin
theHarvester.py -d apple.com -b googleCSE -l 500 -s 300
The Harvester

```python
python theHarvester.py -d nasa.gov -l 500 -b google

[-] Searching in Google:
  Searching 0 results...
  Searching 100 results...
  Searching 200 results...
  Searching 300 results...
  Searching 400 results...
  Searching 500 results...

[+] Emails found:
------------------
mobile@mail.nasa.gov
robert.j.gutro@nasa.gov

[+] Hosts found in search engines:
----------------------------------
[-] Resolving hostname IPs...
87.248.214.97:www.nasa.gov
198.117.0.121:mail.nasa.gov
198.116.65.32:www.hq.nasa.gov
87.248.214.97:www.jsc.nasa.gov
129.164.179.249:modis.gsfc.nasa.gov
192.68.196.38:eol.jsc.nasa.gov
69.58.188.49:go.nasa.gov
137.78.99.24:www.jpl.nasa.gov
54.192.61.72:mars.jpl.nasa.gov
198.116.65.32:oiir.hq.nasa.gov
128.183.4.33:daa.giss.nasa.gov
128.183.4.33:pubs.giss.nasa.gov
198.118.248.108:sohowww.nascom.nasa.gov
169.154.198.218:iswa.ccmc.gsfc.nasa.gov
128.183.20.84:space-geodesy.nasa.gov
```
Use all the available techniques in w3af to fingerprint the remote Web infrastructure.
• Scapy
  ○ Capturing and analysing network packets
• FiMap
  ○ Detecting RFI/LFI vulnerabilities
• XSSScrapy
  ○ Detecting XSS vulnerabilities
import socket

#TCP
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
result = sock.connect_ex(('127.0.0.1',80))
if result == 0:
    print "Port is open"
else:
    print "Port is filtered"
# Port Scanner

```python
from socket import *

ip = raw_input("Enter IP to scan : ")
start = input("Enter starting port number : ")
end = input("Enter ending port number : ")

print "Scanning IP: ", ip

for port in range(start,end):
    print "Testing port "+str(port)+"...."
    s = socket(AF_INET, SOCK_STREAM)
    s.settimeout(5)
    if(s.connect_ex((ip, port))==0):
        print "Port ", port, "is open"
    s.close()

print "Scanning completed !! "
```
# Imports socket module
# Asks user to enter IP
# Asks user to enter start
# Asks user to enter end
# For loop from startin
# Creates a socket s
# set timeout
# If connection to port
# Prints open port
# Closes socket s
# Printing completion message
Socket resolving IP/domain

```python
import socket
print(socket.gethostbyaddr("136.243.32.71"))
print(socket.gethostbyname("ep2016.europython.eu"))
```

('cloud1.europython.io', [], ['136.243.32.71'])

136.243.32.71
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(((parsed_args.target, 80))

http_get = b"GET / HTTP/1.1
Host: " + parsed_args.target + "
\n"
data = ''
try:
    sock.sendall(http_get)
data = sock.recvfrom(1024)
    print(data)
except socket.error:
    print("Socket error", socket.errno)
finally:
    print("closing connection")
    sock.close()

strdata = data[0]
# looks like one long line so split it at newline into multiple strings
headers = strdata.splitlines()
# use regular expression splitlines library to look for the one line we like
for s in headers:
    if re.search('Server:', s):
        print(s)
Banner server

usage: BannerServer.py [-h] -target TARGET [-proxy PROXY]

Obtain server banner

optional arguments:
  -h, --help    show this help message and exit
  -target TARGET target IP / domain
  -proxy PROXY  Proxy[IP:PORT]

python BannerServer.py -target ep2016.europython.eu -port 80

{'proxy-agent': 'Fortinet-Proxy/1.0'}
{'HTTP/1.1 301 Moved Permanently

Server: nginx
Requests: HTTP for Humans

Release v2.9.1. ([Installation](https://github.com/psf/requests/releases/tag/v2.9.1))

Requests is an [Apache2 Licensed](https://github.com/psf/requests/blob/master/LICENSE) HTTP library, written in Python, for human beings.

Python’s standard [urllib2](https://docs.python.org/3/library/urllib2.html) module provides most of the HTTP capabilities you need, but the API is thoroughly broken. It was built for a different time — and a different web. It requires an enormous amount of work (even method overrides) to perform the simplest of tasks.

Things shouldn’t be this way. Not in Python.

```python
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u"type":"User"
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

See similar code, without Requests.

Requests takes all of the work out of Python HTTP/1.1 — making your integration with web services seamless. There’s no need to manually add query strings to your URLs, or to form-encode your POST data. Keepalive and HTTP connection pooling are 100% automatic, powered by [urllib3](https://urllib3.readthedocs.io), which is embedded within Requests.
Checking headers

```python
response = requests.get("https://ep2016.europython.eu/", timeout=5)

print "Status code: "+str(response.status_code)

print "Headers response: 
for header, value in response.headers.items():
    print(header, '-->', value)

print "Headers request : 
for header, value in response.request.headers.items():
    print(header, '-->', value)
```
Checking headers

Status code: 200
Headers response:
(['Server', '-->', 'nginx'],
 ['Date', '-->', 'Mon, 04 Jul 2016 12:30:59 GMT'],
 ['Content-Type', '-->', 'text/html; charset=utf-8'],
 ['Transfer-Encoding', '-->', 'chunked'],
 ['Connection', '-->', 'keep-alive'],
 ['Content-Language', '-->', 'en'],
 ['Vary', '-->', 'Accept-Language, Cookie'],
 ['X-Frame-Options', '-->', 'SAMEORIGIN'],
 ['Set-Cookie', '-->', 'django_language=en; expires=Tue, 04-Jul-2017 12:30:59 GMT; Max-Age=31536000; Path=/'],
 ['P3P', '-->', 'CP="ALL DSP COR PSAa PSDe OUR NOR ONL UNI COM NAV"'],
 ['Strict-Transport-Security', '-->', 'max-age=31536000; includeSubdomains'],
 ['Content-Encoding', '-->', 'gzip'])
Headers request:
(['Connection', '-->', 'keep-alive'],
 ['Accept-Encoding', '-->', 'gzip, deflate'],
 ['Accept', '-->', '*/*'],
 ['User-Agent', '-->', 'python-requests/2.10.0'])
import requests

http_proxy = "http://10.10.10.10:3000"
https_proxy = "https://10.10.10.10:3000"

proxyDict = {
"http" : http_proxy,
"https" : https_proxy
}

r = requests.get(url, proxies=proxyDict)
import requests
encoded = base64.b64encode(user+':'+passwd)

response = requests.get(protectedURL, auth=(user, passwd))

import requests
from requests.auth import HTTPDigestAuth

response = requests.get(protectedURL, auth=HTTPDigestAuth(user, passwd))
```python
from bs4 import BeautifulSoup

import requests

url = raw_input("Enter a website to extract the URL's from: ")

r = requests.get("http://" + url)

data = r.text

soup = BeautifulSoup(data, "lxml")

for link in soup.find_all('a'):
    print(link.get('href'))
```
# Retrieves a list of all Internal links found on a page

def getInternalLinks(bsObj, includeUrl):
    internalLinks = []
    # Finds all links that begin with a "/"
    for link in bsObj.findAll("a", href=re.compile("^/(?=.*"+includeUrl+)")):
        if link.attrs['href'] is not None:
            if link.attrs['href'] not in internalLinks:
                internalLinks.append(link.attrs['href'])
    return internalLinks

# Retrieves a list of all external links found on a page

def getExternalLinks(bsObj, excludeUrl):
    externalLinks = []
    # Finds all links that start with "http" or "www" that do
    # not contain the current URL
    for link in bsObj.findAll("a", href=re.compile("^(http|www)((?!"+excludeUrl+)
        if link.attrs['href'] is not None:
            if link.attrs['href'] not in externalLinks:
                externalLinks.append(link.attrs['href'])
    return externalLinks
Internal/external links

External links
-------------------
http://djangogirls.org/europython2016/
http://europython.tv/
http://pyss.org/
http://www.europython-society.org/
https://ep2015.europython.eu/
https://ep2014.europython.eu/
https://ep2013.europython.eu/ep2013/
https://ep2013.europython.eu/ep2012/
http://www.europython-society.org/europython
http://blog.europython.eu/
https://twitter.com/europython
https://www.facebook.com/europython
https://mail.python.org/mailman/listinfo/europython-announce
https://www.python.org/psf-landing/
http://www.bilbao.net/
https://sites.google.com/site/bbvagroupateuropython/home
https://hired.com/
http://www.intel.com/
https://www.microsoft.com/
http://www.ehu.eus/

Internal links
-------------------
/en/
/en/registration/
/registration/
/en/registration/volunteers/
/en/registration/financial-aid/
/en/registration/tips-for-attendees/
/en/events/
/en/events/keynotes/
/en/events/conference-app/
/en/events/sessions/
/en/events/sprints/
/en/events/pydata/
/en/events/beginners-day/
/en/events/maker-area/
/en/events/social-event/
/en/speakers/
```python
def scrapingImagesPdf(self, url):
    print("\nScraping the server for images and pdfs.... " + url)

    try:
        response = requests.get(url)
        parsed_body = html.fromstring(response.text)

        # Grab links to all images
        images = parsed_body.xpath('//img/@src')

        # Grab links to all pdf
        pdfs = parsed_body.xpath('//a[@href[contains(., ".pdf")]]/@href')

    except Exception, e:
        print e
        print "Error to connect with " + url + " for scraping the site";
```
Scrapy

An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

Build and run your web spiders

```
$ pip install scrapy
$ cat > myspider.py
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['http://blog.scrapinghub.com']

def parse(self, response):
    for url in response.css('ul li a::attr(href)').re(r'^https?://[^ ]+/$'):
        yield scrapy.Request(url, self.parse_items)

def parse_items(self, response):
    for post_title in response.css('div.entries > ul > li a::text').extract():
        yield {'title': post_title}

$ scrapy runspider myspider.py
```
Web Scraping

Scraping the Web
the workshop

José Manuel Ortega
@jmortegac

Python tools for webscraping

José Manuel Ortega
@jmortegac
Welcome
Shodan lets you search for devices that are connected to the Internet. And a Shodan account means you get more access, more features and the ability to check out the latest developments.

More Results
With a free Shodan account you can access more results!

Developer API
The Shodan API makes it easy to access the data from within your own scripts.

New Filters
Once you're logged in you have access to a lot more filters that help you find exactly what you're looking for.
Installation

To get started with the Python library for Shodan, first make sure that you've received your API key. Once that's done, install the library via the cheeseshop using:

```
$ easy_install shodan
```

Or if you already have it installed and want to upgrade to the latest version:

```
$ easy_install -U shodan
```

It's always safe to update your library as backwards-compatibility is preserved. Usually a new version of the library simply means there are new methods/features available.

Connect to the API

The first thing we need to do in our code is to initialize the API object:

```python
import shodan

SHODAN_API_KEY = "insert your API key here"

api = shodan.Shodan(SHODAN_API_KEY)
```

Searching Shodan

Now that we have our API object all good to go, we're ready to perform a search:
import shodan

SHODAN_API_KEY = "insert your API key here"
api = shodan.Shodan(SHODAN_API_KEY)

# Lookup the host
host = api.host(hostname)

# Print general info
print ""
    IP: %s
    Organization: %s
    Operating System: %s
""" % (host['ip_str'], host.get('org', 'n/a'), host.get('os', 'n/a'))

# Print all banners
for item in host['data']:
    print """Port: %s
    Banner: %s""" % (item['port'], item['data'])
Port: 21

Banner: 220 ProFTPD 1.3.5a Server (ProFTPD) [192.168.55.76]
550 SSL/TLS required on the control channel
550 SSL/TLS required on the control channel
211-Features:
   PBSZ
   AUTH TLS
   MFF modify;UNIX.group;UNIX.mode;
   REST STREAM
   MLST modify×;perm×;size×;type×;unique×;UNIX.group×;UNIX.mode×;UNIX.owner×;
   UTF8
   LANG en-US×
   EPRT
   EPSV
   MDTM
   SSCN
   TDFS
   MFMT
   SIZE
   PROT
   CCC
https://www.shodan.io/host/136.243.32.71

136.243.32.71 cloud1.europython.io

Country
Germany

Organization
Server Block

ISP
Server Block

Last Update
2016-07-03T12:32:39.991534

Hostnames
cloud1.europython.io

Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>25</td>
<td>ssh</td>
</tr>
<tr>
<td>80</td>
<td>http</td>
</tr>
</tbody>
</table>

Services

SSH-1.2-OpenSSH_6.6.1p1 Ubuntu-1ubuntu1.6

Key Type: ssh-rsa

Key:
AAAAB3NzaC1yc2EAAAADAQABAAABgQB35GqohPlAz/AtCpItU/CeP4/HYVYjY+9e688+uLqRT6b+5rvPCC+i8R9008Iy59FH
hyi1j21i+7k42pZbMK/twPnLHfRZ01y6P3SbHuw02como9913716q7/1sho95y9BdY6y/j11dXk7/7/EY07079/v/kKBLqg9p8HyAr1408v1+YkHE40P66nuxfSf/VT18tNh+6ybDqtr/vcC12fP0m/6C+u
xgDRG-z3vGRDCo-BaO4C1s9NvKycUmpqB7di1H012/x+tU0701L1=+c0L12v/t2m/k


Key Algorithms:
curve25519-sha256@libssh.org
ecdh-sha2-nistp256
ecdh-sha2-nistp384
ecdh-sha2-nistp521
diffie-hellman-group-exchange-sha256
diffie-hellman-group-exchange-sha1
diffie-hellman-group1-sha

diffie-hellman-group1-sha1

Server Host Key Algorithms:
ssh-rsa
ssh-dss
ecdsa-sha2-nistp256
Shodan
Port: 80
Banner: HTTP/1.1 404 Not Found
Server: nginx/1.4.6 (Ubuntu)
Date: Tue, 19 Jul 2016 19:11:33 GMT
Content-Type: text/html
Content-Length: 573
Connection: keep-alive

Port: 22
Banner: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6
Key type: ssh-rsa
Key: AAAAB3NzaC1yc2EAAAADQAAAABAAABAADn5raXisS8w=H31CZ2U/t05QNeGRNhFlhqTYfFG/aBKRKzmlDagg3KZgS3On2nPLxom0kpl1u/CeBjNHJAVNi6mG8t+uLqZPT0sSrweGR8ngGJU9yJ9GZhu1lJ2l1B9R2pPzwmKNTweuL5MTrq2yHN5PJMwuzxxMc95E3715q7z2khGSp6cBCwiEVzMzrX0N7KFGxBo7QOa5xRLLQ07uHVykOtTk8ei+YKhEOQ0p6xrufGJ/T5IDagx94yoUDzxc1izP0h/6Cz+oXg60H5+i13xGRNoCp8BfKICsiHNdRGuCUmnp87od1H0W1/9Z5W7zLDM1x+DL10nZ2nWx

Kex Algorithms:
- curve25519-sha256@libssh.org
- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521
- diffie-hellman-group-exchange-sha256
- diffie-hellman-group-exchange-sha1
- diffie-hellman-group14-sha1
- diffie-hellman-group1-sha1

Server Host Key Algorithms:
- ssh-rsa
- ssh-dss
- ecdsa-sha2-nistp256
- ssh-ed25519

Encryption Algorithms:
- aes128-ctr
- aes192-ctr
- aes256-ctr
- arcfour256
- arcfour128
- aes128-qcm@openssh.com
pip install builtwith

builtwith.parse('https://ep2016.europython.eu')

```python
>>> builtwith.parse("https://ep2016.europython.eu")
{u'javascript-frameworks': [u'jQuery', u'Modernizr', u'jQuery UI'], u'web-servers': [u'Nginx']}
```
from PyPDF2 import PdfFileReader, PdfFileWriter
import os

def printMeta():
    for dirpath, dirnames, files in os.walk("pdf"):  
        for name in files:
            ext = name.lower().rsplit('.', 1)[-1]
            if ext in ['pdf']:
                print "[+] Metadata for file: %s " %(dirpath+os.path.sep+name)
                pdfFile = PdfFileReader(file(dirpath+os.path.sep+name, 'rb'))
                docInfo = pdfFile.getDocumentInfo()
                for metaItem in docInfo:
                    print ' [+] ' + metaItem + ': ' + docInfo[metaItem]
                print "\n"

[+] Metadata for file: pdf\python.pdf
[+] /Title:Guía de aprendizaje de Python
[+] /Author:Guido van Rossum, Fred L. Drake, Jr., editor
[+] /Producer:pdfTeX-0.13d
[+] /CreationDate:D:20001124213800
[+] /Creator:TeX
```python
from PIL.ExifTags import TAGS, GPSTAGS
from PIL import Image
import os

def get_exif_metadata(image_path):
    ret = {}
    image = Image.open(image_path)
    if hasattr(image, '_getexif'):
        exifinfo = image._getexif()
        if exifinfo is not None:
            for tag, value in exifinfo.items():
                decoded = TAGS.get(tag, tag)
                ret[decoded] = value
        decode_gps_info(ret)
    return ret
```
Analysis metadata

Metadata: 42016 - Value: 2BF3A9E97BC886678DE12E6E8835720
Metadata: YResolution - Value: (300, 1)
Metadata: ResolutionUnit - Value: 2
Metadata: Copyright - Value: Frank Noort
Metadata: Artist - Value: Frank Noort
Metadata: Make - Value: Canon
Metadata: GPSInfo - Value: {'Lat': 32.07874722222222, 'Lng': -131.4675777777778}
Metadata: XResolution - Value: (300, 1)
Metadata: ExifOffset - Value: 146
Metadata: ExifVersion - Value: 0220
Metadata: DateTimeOriginal - Value: 2002:10:28 11:05:09
Metadata: Model - Value: Canon EOS-5
Metadata: DateTime - Value: 2008:03:09 22:00:01
Metadata: Software - Value: Adobe Photoshop CS2 Windows
Port Scanning

NMAP.ORG
Python-nmap

- Automating port scanning
- Synchronous and asynchronous modes

```python
import nmap
# Synchronous
nm = nmap.PortScanner()
# nm.scan('ip/range','port_list')
results = nm.scan('127.0.0.1', '22,25,80,443')
```
class NmapScanner:

    def __init__(self):
        self.nmSc = nmap.PortScanner()

    def nmapScan(self, host, port):
        try:
            print "Checking port " + port + " ............"
            self.nmSc.scan(host, port)

            # Command info
            print "[*] Executing command: %s" % self.nmSc.command_line()
            self.state = self.nmSc[host]["tcp"][int(port)]["state"]
            print "[+] " + host + " tcp/" + port + " " + self.state
        except Exception,e:
            print "Error to connect with " + host + " for port scanning"
            pass
for port in port_list:
    NmapScanner().nmapScan(ip, port)

python NmapScanner.py -target 192.168.56.101 -ports 21,22,23,24,25,80

Checking port 21 ...........
[×] Executing command: nmap -oX -p 21 -sU 192.168.56.101
[+] 192.168.56.101 tcp/21 open

Checking port 22 ...........
[×] Executing command: nmap -oX -p 22 -sU 192.168.56.101
[+] 192.168.56.101 tcp/22 open

Checking port 23 ...........
[×] Executing command: nmap -oX -p 23 -sU 192.168.56.101
[+] 192.168.56.101 tcp/23 open

Checking port 24 ...........
[×] Executing command: nmap -oX -p 24 -sU 192.168.56.101
[+] 192.168.56.101 tcp/24 closed

Checking port 25 ...........
[×] Executing command: nmap -oX -p 25 -sU 192.168.56.101
[+] 192.168.56.101 tcp/25 open

Checking port 80 ............
[×] Executing command: nmap -oX -p 80 -sU 192.168.56.101
[+] 192.168.56.101 tcp/80 open
# Asynchronous

```python
nm_async = nmap.PortScannerAsync()
def callback_result(host, scan_result):
    print '------------------'
    print host, scan_result

nm_async.scan(hosts='192.168.1.0/30', arguments='-sP', callback=callback_result)

while nm_async.still_scanning():
    print("Waiting >>>")
    nm_async.wait(2)
```
python NmapScannerAsync.py -target 192.168.56.101 -ports 21

Checking port 21 ............
[*] 192.168.56.101 tcp/21 open
Checking ftp port with nmap scripts.......
Checking ftp-anon.nse ......
Command linenmap -oX -A -sU -p21 --script ftp-anon.nse 192.168.56.101
Script ftp-anon --> Anonymous FTP login allowed (FTP code 230)
Checking ftp-bounce.nse ......
Checking ftp-brute.nse ......
Script ftp-brute -->
  Accounts:
    user:user - Valid credentials
Statistics: Performed 193T guesses in 602 seconds, average tps: 3
Checking ftp-libopie.nse ......
Checking ftp-proftpd-backdoor.nse ......
Checking ftp-vsftpd-backdoor.nse ......
Script ftp-vsftpd-backdoor -->
VULNERABLE:
vsFTPD version 2.3.4 backdoor
  State: VULNERABLE (Exploitable)
  IDs: OSVDB:73573 CUE:CUE-2011-2523
  vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-04.
Disclosure date: 2011-07-03
Exploit results:
  Shell command: id
  Results: uid=0(root) gid=0(root)
References:
http://osvdb.org/73573
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CUE-2011-2523
### Scripts Nmap

```
$ ls /usr/share/nmap/scripts/
acarsd-info.nse          ftp-proftpd-backdoor.nse          informix-tables.nse
address-info.nse          ftp-vsftpd-backdoor.nse          ip-forwarding.nse
afp-brute.nse             ftp-vuln-cve2010-4221.nse          ip-geolocation-gateway.nse
afp-ls.nse                ganglia-info.nse                   ip-geolocation-ipv4.nse
afp-path-vuln.nse          giop-info.nse                     ip-geolocation-ipv6.nse
afp-serverinfo.nse         gkrellm-info.nse                   ip-geolocation-map.nse
afp-showmount.nse          gopher-ls.nse                     ipidseq.nse
afp-auth.nse              gpsd-info.nse                      ipv6-node-info.nse
afp-brute.nse             hadoop-datanode-info.nse           ipv6-ra-flood.nse
afp-headers.nse            hadoop-jobtracker-info.nse         irc-botnet-channel.nse
afp-methods.nse            hadoop-namenode-info.nse           irc-brute.nse
afp-request.nse            hadoop-secondary-namenode-info.nse   irc-info.nse
amap-info.nse              hadoop-tasktracker-info.nse         irc-sasl-brute.nse
asn-query.nse              hbase-master-info.nse               irc-unrealircd-back.nse
auth-owners.nse            hbase-region-info.nse               iscsi-brute.nse
auth-spoof.nse             hddtemp-info.nse                   iscsi-info.nse
backorifice-brute.nse      hostmap-bfk.nse                    isns-info.nse
backorifice-info.nse       hostmap-ip2hosts.nse                jdkw-exec.nse
banner.nse                 hostmap-robotex.nse                 jdkw-info.nse
bitcoin-getaddr.nse        http-adobe-coldfusion-apsa1301.nse   jdkw-inject.nse
bitcoin-info.nse           http-affiliate-id.nse               jdkw-version.nse
bitcoinrpc-info.nse        http-apache-negotiation.nse        krb5-enum-users.nse
bittorrent-discovery.nse   http-auth-finder.nse                ldap-brute.nse
```
• Programming routines allow to find potential vulnerabilities in a given target
• First check if the port is open
• Detect vulnerabilities in the service port opened

```
nm.scan(arguments="-n -A -p3306 --script=/usr/share/nmap/scripts/mysql-info.nse")
```
```python
if (port==3306) and self.nmsync[hostname]["tcp"][int(port)]["state"]=='open':
    print 'Checking MYSQL port with nmap scripts......'

    #scripts for mysql:3306 open
    print 'Checking mysql-audit.nse......'
    self.nmsync.scan(hostname,
                     arguments="-A -sV -p3306 --script mysql-audit.nse",callback=callbackMySQL)
    self.scanning()
    print 'Checking mysql-brute.nse......'
    self.nmsync.scan(hostname,
                     arguments="-A -sV -p3306 --script mysql-brute.nse",callback=callbackMySQL)
    self.scanning()
    print 'Checking mysql-databases.nse......'
    self.nmsync.scan(hostname,
                     arguments="-A -sV -p3306 --script mysql-databases.nse",callback=callbackMySQL)
    self.scanning()
    print 'Checking mysql-databases.nse......'
    self.nmsync.scan(hostname,
                     arguments="-A -sV -p3306 --script mysql-databases.nse",callback=callbackMySQL)
    self.scanning()
```
Check FTP Login Anonymous

SHODAN

port:21 Anonymous user logged in

Exploits  Maps  Share Search  Download Results

TOP COUNTRIES

<table>
<thead>
<tr>
<th>Country</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>192,483</td>
</tr>
<tr>
<td>China</td>
<td>42,635</td>
</tr>
<tr>
<td>Germany</td>
<td>32,027</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>12,000</td>
</tr>
<tr>
<td>Korea, Republic of</td>
<td>10,477</td>
</tr>
</tbody>
</table>
def anonymousLogin(hostname):
    try:
        ftp = ftplib.FTP(hostname)
        ftp.login('anonymous', '')
        print '\n[*] ' + str(hostname) + ' FTP Anonymous Logon Succeeded.'
        return ftp
    except Exception, e:
        print '\n[-] ' + str(hostname) + ' FTP Anonymous Logon Failed.'
        return False
Check Webs sites

- `pip install pywebfuzz`
- [https://github.com/disassembler/pywebfuzz](https://github.com/disassembler/pywebfuzz)
from pywebfuzz import fuzzdb
import requests

logins = fuzzdb.Discovery.PredictableRes.Logins

domain = "http://192.168.56.101"

for login in logins:
    print "Checking... " + domain + login
    response = requests.get(domain + login)
    if response.status_code == 200:
        print "Login Resource: " + login
PyWebFuzz

[+] Get predictable urls
[+] Found Login Resource: /index.php?u=
Heartbleed

- Vulnerability in OpenSSL V1.0.1
- Multi-threaded tool for scanning hosts for CVE-2014-0160.
- https://github.com/musalbas/heartbleed-masstest
- https://filippo.io/Heartbleed
# construct heartbeat request packet
ver_chr = chr(ver&0xff)
hb = h2bin("18 03") + ver_chr + h2bin("40 00 01 3f fd") + \\x01*16381
hb += h2bin("18 03") + ver_chr + h2bin("00 03 01 00 00")

s.send(hb)
return hit hb(s)
Heartbleed

WARNING: server returned more data than it should - server is vulnerable!

Final Results

Server vulnerable found 2
Server vulnerable: 176.34.114.90
IP: 176.34.114.90
Country: Ireland
City: None
Latitude: 53.3478
Longitude: -6.2597
Hostnames: [u'mailout1.theframeworks.com']
Advanced tools

- Metasploit
- w3af
- Nexpose
- Nessus
- OpenVAS
Metasploit

• python-msfrpc

```
=[ metasploit v4.11.4-dev-b206de77 ]
+ -- --=[ 1488 exploits - 858 auxiliary - 251 post ]
+ -- --=[ 432 payloads - 37 encoders - 8 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > load msgrpc Pass=msfadmin
[*] MSGRPC Service: 127.0.0.1:55552
[*] MSGRPC Username: msf
[*] MSGRPC Password: msfadmin
[*] Successfully loaded plugin: msgrpc
msf >
```
Metasploit API call

• Calls in msgpack format

```ruby
cmdMysqlLogin=""""auxiliary/scanner/mysql/mysql_login
set RHOSTS """" + self.ip

    cmdMysqlLogin = cmdMysqlLogin + """"\nrun
    print
self.client.call('console.write',[self.console['id'],cmdMysqlLogin])
self.processData(self.console['id'])
```
Nexpose

- Tool developed by Rapid7 for scanning and vulnerability discovery.
- It allows programmatic access to other programs via HTTP/s requests.
- BeautifulSoup to obtain data from vulnerabilities server
try:
    if pyconnect == 0:
        pynexposeHttps = pynexposeHttps.NeXposeServer(serveraddr_nexpose, port_server_nexpose, user_nexpose, password_nexpose)
        pyconnect = 1
    except Exception, e:
        pyconnect = 0
        print "Error to connecting with NeXposeServer"
        print e

def vulnerabilityListing(self):
    print "\nVulnerabilities"
    print "----------------"
    bsoupVulnerabilityListing = BeautifulSoup(self.pynexposeHttps.vulnerability_listing(),'lxml')
    for vulnerability in bsoupVulnerabilityListing.findAll('vulnerabilitysummary'):
        attrs = dict(vulnerability.attrs)
        print "Id: " + attrs['id']
        print "Severity: " + attrs['severity']
        print "Title: " + attrs['title']
        bsoupVulnerabilityDetails = BeautifulSoup(self.pynexposeHttps.vulnerability_details(attrs['id']),'lxml')
        for vulnerability_description in bsoupVulnerabilityDetails.findAll('description'):
            print "Description: " + vulnerability_description.text
Pentesting tool

1. Check Open Ports [80, 8080 by default]
2. Port Scanning [It will scan over ports parameter, by default it will scan 80 and 8080]
3. Nmap Scanning Advanced
4. Check Option methods
5. Check DNS Servers info
6. Check Host info from Shodan Service
7. NMAP Port Scanning
8. Host Info by Socket Call
9. GeoLocation Host Info
10. Scraping for images and pdf & obtain metadata
11. Get Headers info
12. Get SSH user/password Brute Force [Requires port 22 opened]
13. Get FTP Anonymous access [Requires port 21 opened]
14. MetaSploitFrameWork
15. NexposeFramework
16. HTTP SCAN [Requires port 80 opened]
17. Check HeartBleed OpenSSL vulnerability [Requires port 443 opened]
18. Check FTP Server Buffer Overflow Vulnerability [Requires port 21 opened]
19. Check Vulnerabilities SQL, XSS, LFI in domain
20. Check Domains and obtain metadata [mails, hosts, servers, urls]
21. Check open ports with scapy
22. Check website libraries
23. Identify web server
https://github.com/jmortega/python-pentesting
https://github.com/jmortega/europython_ethical_hacking
References & libs

- http://docs.shodanhq.com
- http://docs.python-requests.org/en/master/
- http://scrapy.org
- http://www.pythonsecurity.org/libs
- https://github.com/dloss/python-pentest-tools
- http://kali-linux.co/2016/07/12/python-tools-for-penetration-testers%E2%80%8B/
- https://github.com/PacktPublishing/Effective-Python-Penetration-Testing
Books

Black Hat Python
Python Programming for Hackers and Pentesters
Justin Seitz
Foreword by Charlia Miller

Python Penetration Testing Essentials
Employ the power of Python to get the best out of pentesting
Mohit
Books

1. Gray Hat Python: Python Programming for Hackers and Reverse Engineers
   - Justin Seitz

2. Python para Pentesters
   - Daniel Echeverri Montoya
THANK YOU!