



EUROPYTHON
2016 Bilbao, 17-24 July



Ethical hacking with Python tools

JOSE MANUEL ORTEGA
@JMORTEGAC

<https://speakerdeck.com/jmortega>



Testing Android Security extended
Nov 30, 2015 by speakerOrtega



Scraping the web with python
Nov 30, 2015 by speakerOrtega



Python Comparing ORM
Nov 22, 2015 by speakerOrtega



Python Seguridad & Criptografia
Nov 21, 2015 by speakerOrtega



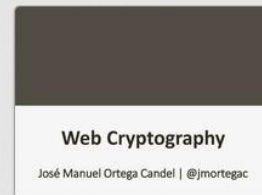
Seguridad dispositivos móviles
Sep 24, 2015 by speakerOrtega



Python Cryptography & Security
Jul 22, 2015 by speakerOrtega



Comparing JVM languages
Jun 28, 2015 by speakerOrtega



Web Cryptography
May 10, 2015 by speakerOrtega



Mobile Backend as a Service
Apr 26, 2015 by speakerOrtega



INDEX



- 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 tool that exploits SQL injection flaws and taking over of database servers. It has many niche features for the ultimate penetration tester like: automatic fingerprinting, over data fetching from the database, and executing commands on the operating system via ODBC.



SocialEngineer
T o o l k i t

http://sparta.secforce.com/



SPARTA 1.0 (BETA) - untitled - /root/Desktop/

File Help

Scan Brute

Hosts Services Tools

OS	Host
?	10.0.0.0
?	10.0.0.1
🐧	10.0.0.12
?	10.0.0.50
🌐	10.0.0.121
?	10.0.0.154
?	10.0.0.222

Services Scripts Information Notes nikto (80/tcp) smtp-enum-vrfy (25/tcp) mysql-default (3306/tcp)

Port	Protocol	State	Name	Version
21	tcp	open	ftp	vsftpd 2.3.4
22	tcp	open	ssh	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23	tcp	open	telnet	Linux telnetd
25	tcp	open	smtp	Postfix smtpd
53	tcp	open	domain	ISC BIND 9.4.2
80	tcp	open	http	Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111	tcp	open	rpc	rpcbind 2 (RPC #100000)
137	udp	open	nbns	Microsoft Windows XP netbios-ssn
139	tcp	open	nbss	Samba smbd 3.X (workgroup: WORKGROUP)

- Open with netcat
- Open with telnet
- Send to Brute
- Open in browser
- Take screenshot
- Grab banner
- Launch dirbuster
- Launch weblayer
- Run nikto
- Run nmap (scripts) on port
- Run sslscan

Log

Progress	Tool	Start time	Status
██████████	x11screen (6000/tcp)	015 14:45:18	Finished
██████████	nikto (8180/tcp)	015 14:45:18	Finished
██████████	x11screen (6000/tcp)	015 14:45:18	Finished

The Harvester



Usage: `theharvester options`

```
-d: Domain to search or company name
-b: data source: google, googleCSE, bing, bingapi, ppg, linkedin,
    google-profiles, jigsaw, twitter, googleplus, all

-s: Start in result number X (default: 0)
-u: 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,
-h: use SHODAN database to query discovered hosts
    google 100 to 100, and ppg doesn't use this option)
```

Examples:

```
theHarvester.py -d microsoft.com -l 500 -b google
theHarvester.py -d microsoft.com -b ppg
theHarvester.py -d microsoft -l 200 -b linkedin
theHarvester.py -d apple.com -b googleCSE -l 500 -s 300
```

The Harvester



```
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 hostnames 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:data.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
```


W3AF



The screenshot shows the W3AF application window with the following components:

- Title Bar:** w3af - Web Application Attack and Audit Framework
- Menu Bar:** Profiles, Edit, View, Tools, Configuration, Help
- Toolbar:** Contains icons for home, add, save, play, list, search, settings, help, print, search, and close.
- Navigation Tabs:** Scan config (selected), Log, Results, Exploit
- Profiles List:**
 - empty_profile
 - OWASP_TOP10
 - audit_high_risk
 - bruteforce
 - fast_scan
 - full_audit
 - full_audit_manual_disc
 - sitemap
 - web_infrastructure** (highlighted)
- Target:** A text input field containing "Insert the target URL here" and a "Start" button.
- Plugin List:**

Plugin	Active
▶ audit	<input type="checkbox"/>
▶ bruteforce	<input type="checkbox"/>
▶ discovery	<input checked="" type="checkbox"/>
▶ evasion	<input type="checkbox"/>
▶ grep	<input type="checkbox"/>
▶ mangle	<input type="checkbox"/>
- Main Panel:** A large grey area containing the text: "Use all the available techniques in w3af to fingerprint the remote Web infrastructure."
- Status Bar:** Shows three indicators: information (i) 0, warning (⚠) 0, and success (✓) 0.

Tools



- Scapy
 - Capturing and analysing network packets
- FiMap
 - Detecting RFI/LFI vulnerabilities
- XSScrapy
 - Detecting XSS vulnerabilities

Sockets Port scan



```
import socket

#TCP
sock = 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"
```

Sockets Port scan



```
# Port Scanner
from socket import * # Imports socket module
ip=raw_input("Enter IP to scan : ") # Asks user to enter IP
start=input("Enter starting port number : ") # Asks user to enter st
end=input("Enter ending port number : ") # Asks user to enter en
print "Scanning IP: " , ip
for port in range(start,end): # For loop from startin
    print "Testing port "+str(port)+"...."
    s=socket(AF_INET, SOCK_STREAM) # Creates a socket s
    s.settimeout(5) # set timeout
    if(s.connect_ex((ip,port))==0): # If connection to port
        print "Port " , port, "is open" # Prints open port
    s.close() # Closes socket s
print "Scanning completed !! "
```

Socket resolving IP/domain



```
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
```

Banner server



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

http_get = b"GET / HTTP/1.1\nHost: "+parsed_args.target+"\n\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 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\r\nServer: nginx\r\nDate:  
hon.eu/\r\n\r\n<html>\r\n<head><title>301 Moved Permanently  
ml>\r\n', (0, '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00  
closing connection  
Server: nginx
```

Requests



Requests

Star 16,936

Requests is an elegant and simple HTTP library for Python, built for human beings.

Buy Requests Pro

Get Updates

Receive updates on new releases and upcoming projects.

[Subscribe to Newsletter](#)

Translations

[English](#)

[French](#)

[German](#)

[Japanese](#)

Requests: HTTP for Humans

Release v2.9.1. ([Installation](#))

Requests is an [Apache2 Licensed](#) HTTP library, written in Python, for human beings.

Python's standard `urllib2` 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.

```
>>> 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()
{'private_gists': 419, '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. Keep-alive and HTTP connection pooling are 100% automatic, powered by [urllib3](#), which is embedded within Requests.

Checking headers



```
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 PSDa 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')
```

Requests



```
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)
```

Requests Authentication



```
import requests
encoded = base64.encodestring(user+' : '+passwd)

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

```
import requests
from requests.auth import HTTPDigestAuth

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

BeautifulSoup



```
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'))
```

Internal/external links



```
#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

<https://ep2016.europython.eu/p3/schedule/ep2016/>
<https://ep2016.europython.eu/p3/schedule/ep2016/list/>
<http://djangogirls.org/europython2016/>
<https://ep2016.europython.eu/p3/ep2016/whos-coming?speaker=on>
<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/](#)
<https://ep2016.europython.eu/p3/schedule/ep2016/>
<https://ep2016.europython.eu/p3/schedule/ep2016/list/>
[/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/](#)

Extract images and documents



```
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



Scrapy

[Download](#)

[Documentation](#)

[Community](#)

[Companies](#)

[Commercial Support](#)

[FAQ](#)

[Fork on Github](#)



Scrapy

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

PyPI [v1.0.4](#) downloads [43k/month](#) wheel [yes](#) PY3 [72%](#) coverage [82%](#)

Install the latest version of Scrapy

 **Scrapy 1.0**

`$ pip install scrapy`

[PyPI](#)

[Conda](#)

[APT](#)

[Source](#)

Build and run your web spiders

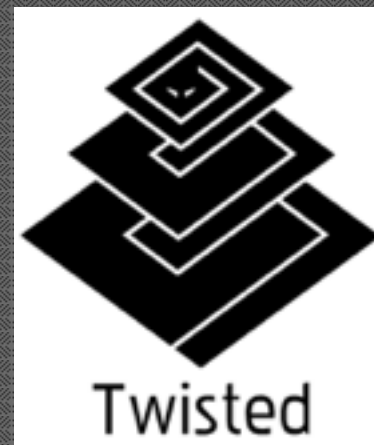
Terminal

```
$ pip install scrapy
$ cat > myspider.py <<EOF
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'.*/\d/\d/\d/$'):
            yield scrapy.Request(response.urljoin(url), self.parse_titles)

    def parse_titles(self, response):
        for post_title in response.css('div.entries > ul > li a::text').extract():
            yield {'title': post_title}
EOF
$ scrapy runspider myspider.py
```



Web Scraping



Scraping the Web

the workshop



José Manuel Ortega

@jmortegac

{CODEMOTION}

Python tools for webscraping

José Manuel Ortega

@jmortegac



Shodan



Shodan Developers Book View All...

SHODAN Explore Downloads Reports Enterprise Access Contact Us

The search engine for Buildings

Shodan is the world's first search engine for Internet-connected devices.

[Create a Free Account](#) [Getting Started](#)



SHANGHAI

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.

Sign in with Shodan

Username

Password

[Log in](#)

[Forgot your password?](#)

Sign in with



https://developer.shodan.io



🏠 shodan-python
latest

Search docs

☰ Getting Started

- Installation
- Connect to the API
- Searching Shodan
- Looking up a host

Basic Shodan Search

- Collecting Summary Information using Facets
- Access SSL certificates in Real-Time shodan

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:

```
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:

Shodan



```
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'])
```

Shodan



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

EPSU

MDTM

SSCN

TUFS

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

22 25 80

Services

22	SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6
tcp	Key type: ssh-rsa
ssh	Key: AAAAB3NzaC1yc2EAAAADAQABAAQDnSrmXiS8UwrH3IC2ZV/t05QWeGRNtFhq7YyFSC/gBKRXzm1Ldaga3kZQgTS3Qn2mPLaRvXeOkp1Iu/CeGvNHIAVNi6mBGT+uLqZPToSrveGR8ngGjU9y59GZhy1j21Lp79R2pP2mMKNtwevLSMTrqZyHnSPJmuszxlc95E3715q7zkhd9SVpBCwiEYmMzrSXoll7/KF6x07DyAxhRLQqo7vH+yArTk8ci+YKeRORDp6xrufG/VTBIDNzxG4yoDUzxcV1zPtOh/6Cz+oxg60HS+13xGRNoCrBbk1CsiMNdRGycUUmnp87od1HOW1/+0zSw7zLDN1x+DLIQnNZnlx Fingerprint: 18:73:8e:78:d0:a0:f5:d4:37:d4:a8:fc:43:be:38:b7
	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

Shodan



```
(u'info', '-->', u'protocol 2.0')
(u'hash', '-->', -107126894)
(u'ip', '-->', 2297634887L)
(u'isp', '-->', u'Server Block')
(u'transport', '-->', u'tcp')
(u'ip_str', '-->', u'136.243.32.71')
(u'data', '-->', u'SSH-2.0-OpenSSH_6.6.1p1-Ubuntu-2ubuntu2.6\nKey type: ssh-rsa\nKey: AAAAB3NzaC1yc2EAAAADAQABAAQBNsrMxiS8Uwr
vNHIAyNii6mBGt+uLqZPToSrueGR8ngGjU9y59GZ\nhy11j21Lp79R2p2mMKNtweuL5MTrqZyHnSPJHmuszxlw95E37i5q7zkhd9SUpBCwiEYmMzrSXoN\n7/KfGx07
60HS+13xGRNoCrBbK1CsiMNdRGycUUmnpB7od1H0W1/+0zSW7zLDN1x+DLIQnNZnWx\nFingerprint: 18:73:8e:78:d0:a0:f5:d4:37:d4:a8:fc:43:be:38:b7
6\n\tecdh-sha2-nistp384\n\tecdh-sha2-nistp521\n\tdiffie-hellman-group-exchange-sha256\n\tdiffie-hellman-group-exchange-sha1\n\td
Key Algorithms:\n\tssh-rsa\n\tssh-dss\n\ttecdsa-sha2-nistp256\n\tssh-ed25519\n\nEncryption Algorithms:\n\taes128-ctr\n\taes192-ct
taes256-gcm@openssh.com\n\tchacha20-poly1305@openssh.com\n\taes128-cbc\n\t3des-cbc\n\tblowfish-cbc\n\tcast128-cbc\n\taes192-cbc\
s:\n\tthmac-md5-etm@openssh.com\n\tthmac-sha1-etm@openssh.com\n\tumac-64-etm@openssh.com\n\tumac-128-etm@openssh.com\n\tthmac-sha2-
tm@openssh.com\n\tthmac-sha1-96-etm@openssh.com\n\tthmac-md5-96-etm@openssh.com\n\tthmac-md5\n\tthmac-sha1\n\tumac-64@openssh.com\n\
0\n\tthmac-ripemd160@openssh.com\n\tthmac-sha1-96\n\tthmac-md5-96\n\nCompression Algorithms:\n\tnone\n\tzlib@openssh.com\n\n')
(u'port', '-->', 22)
(u'hostnames', '-->', [u'cloud1.europython.io'])
(u'location', '-->', {u'city': None, u'region_code': None, u'area_code': None, u'longitude': 9.0, u'country_code3': u'DEU', u'la
u'DE', u'country_name': u'Germany'})
(u'timestamp', '-->', u'2016-07-03T12:39.991534')
(u'domains', '-->', [u'europython.io'])
(u'org', '-->', u'Server Block')
(u'os', '-->', None)
(u'_shodan', '-->', {u'crawler': u'122dd688b363c3b45b0e7582622da1e725444808', u'options': {}, u'module': u'ssh', u'id': None})
(u'opts', '-->', {u'ssh': {u'fingerprint': u'18:73:8e:78:d0:a0:f5:d4:37:d4:a8:fc:43:be:38:b7', u'mac': u'hmac-sha2-256', u'ciphe
31C22u/t0SQWeGRNtFhq7YyfSC/gBkRXz\nm1Ldagq3kZQgTS3Qn2mPLaRuXeoKp1Iu/CeGuNHIAyNii6mBGt+uLqZPToSrueGR8ngGjU9y59GZ\nhy11j21Lp79R2pP
yAxWRLQ0o7uH+yArTk8ci+YKeR0Rdp6xruFG/UTBIDNzxG4yoDUzXuC1zPt0h/6Cz+o\nxg60HS+13xGRNoCrBbK1CsiMNdRGycUUmnpB7od1H0W1/+0zSW7zLDN1x+D
sh-rsa', u'ssh-dss', u'ecdsa-sha2-nistp256', u'ssh-ed25519'], u'encryption_algorithms': [u'aes128-ctr', u'aes192-ctr', u'aes256-
gcm@openssh.com', u'chacha20-poly1305@openssh.com', u'aes128-cbc', u'3des-cbc', u'blowfish-cbc', u'cast128-cbc', u'aes192-cbc',
s': False, u'languages': [u'], u'kex_algorithms': [u'curve25519-sha256@libssh.org', u'ecdh-sha2-nistp256', u'ecdh-sha2-nistp384
fie-hellman-group-exchange-sha1', u'diffie-hellman-group14-sha1', u'diffie-hellman-group1-sha1'], u'compression_algorithms': [u'
com', u'hmac-sha1-etm@openssh.com', u'umac-64-etm@openssh.com', u'umac-128-etm@openssh.com', u'hmac-sha2-256-etm@openssh.com', u
```


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: 579
Connection: keep-alive

Port: 22
Banner: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6
Key type: ssh-rsa
Key: AAAAB3NzaC1yc2EAAAADAQABAAQADnSrmXiS8UwrH31CZ2U/t0$QWeGRNtFhq7YyFSC/gBkRXz
m1Ldagq3kZ0gTS3Qn2mPLaRvXe0kp1Iu/CeGvNHIAyNii6mBGt+uLqZPToSrueGR8ngGjU9y59GZ
hy1lj21Lp79R2pP2mMKNtweuL5MTrqZyHnSPJmmszxc95E37i5q7zkh9SUpBCwiEYmMzrSXoN
7/KfGx07DyAxwRLQ0o7vH+yArTk8ci+vKerORDp6xrufG/UTBIDNzxG4yoDUzxvC1zPt0h/6Cz+o
xg60HS+13xGRNoCrBbK1CsiMNdRGycUUmpB7od1H0W1/+0zS7zLDN1x+DLIQnNznWx
Fingerprint: 18:73:8e:78:d0:a0:f5:d4:37:d4:a8:fc:43:be:38:b7

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-gcm@openssh.com
```

BuiltWith



- **pip install builtwith**
- `builtwith.parse('https://ep2016.europython.eu')`

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

Analysis metadata



```
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
```

Analysis metadata



```
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: 2BF3A9E97BC886678DE12E6EB8835720
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': -131.46757777777778}
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



Python-nmap



- Automating port scanning
- Synchronous and asynchronous modes

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

NmapScanner



```
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 "[*] Execuing 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
```


NmapScanner



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 .....
[*] Execuing command: nmap -oX - -p 21 -sU 192.168.56.101
[+] 192.168.56.101 tcp/21 open
Checking port 22 .....
[*] Execuing command: nmap -oX - -p 22 -sU 192.168.56.101
[+] 192.168.56.101 tcp/22 open
Checking port 23 .....
[*] Execuing command: nmap -oX - -p 23 -sU 192.168.56.101
[+] 192.168.56.101 tcp/23 open
Checking port 24 .....
[*] Execuing command: nmap -oX - -p 24 -sU 192.168.56.101
[+] 192.168.56.101 tcp/24 closed
Checking port 25 .....
[*] Execuing command: nmap -oX - -p 25 -sU 192.168.56.101
[+] 192.168.56.101 tcp/25 open
Checking port 80 .....
[*] Execuing command: nmap -oX - -p 80 -sU 192.168.56.101
[+] 192.168.56.101 tcp/80 open
```

NmapScanner Async



```
#Asynchronous
```

```
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)
```

NmapScanner Async



```
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 .....
Command linenmap -oX - -A -sU -p21 --script ftp-brute.nse 192.168.56.101
Script ftp-brute -->
  Accounts:
  user:user - Valid credentials
  Statistics: Performed 1937 guesses in 602 seconds, average tps: 3
Checking ftp-libopie.nse .....
Checking ftp-proftpd-backdoor.nse .....
Checking ftp-usftpd-backdoor.nse .....
Command linenmap -oX - -A -sU -p21 --script ftp-usftpd-backdoor.nse 192.168.56.101
Script ftp-usftpd-backdoor -->
  UULNERABLE:
  usFTPD version 2.3.4 backdoor
  State: UULNERABLE (Exploitable)
  IDs: OSVDB:73573 CUE:CUE-2011-2523
  usFTPD 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:
  https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/usftpd_234_backdoor.rb
  http://osvdb.org/73573
  https://cve.mitre.org/cgi-bin/cvename.cgi?name=CUE-2011-2523
  http://scarybeastsecurity.blogspot.com/2011/07/alert-usftpd-download-backdoored.html
```

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-geo
afp-ls.nse              ganglia-info.nse           ip-geolocation-geo
afp-path-vuln.nse       giop-info.nse              ip-geolocation-ip
afp-serverinfo.nse     gkrellm-info.nse           ip-geolocation-ma
afp-showmount.nse      gopher-ls.nse              ipidseq.nse
ajp-auth.nse           gpsd-info.nse              ipv6-node-info.nse
ajp-brute.nse           hadoop-datanode-info.nse   ipv6-ra-flood.nse
ajp-headers.nse        hadoop-jobtracker-info.nse irc-botnet-channel
ajp-methods.nse        hadoop-namenode-info.nse   irc-brute.nse
ajp-request.nse        hadoop-secondary-namenode-info.nse irc-info.nse
amqp-info.nse          hadoop-tasktracker-info.nse irc-sasl-brute.nse
asn-query.nse          hbase-master-info.nse     irc-unrealircd-ba
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     jdwp-exec.nse
banner.nse             hostmap-robtex.nse        jdwp-info.nse
bitcoin-getaddr.nse   http-adobe-coldfusion-apsa1301.nse jdwp-inject.nse
bitcoin-info.nse       http-affiliate-id.nse     jdwp-version.nse
bitcoinrpc-info.nse   http-apache-negotiation.nse krb5-enum-users.n
bittorrent-discovery.nse http-auth-finder.nse     ldap-brute.nse
```

Scripts Nmap



- 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")
```

Mysql Scripts Nmap



```
#mysql
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.nmasync.scan(hostname,
arguments="-A -sV -p3306 --script mysql-audit.nse",callback=callbackMySQL)
    self.scanning()
    print 'Checking mysql-brute.nse.....'
    self.nmasync.scan(hostname,
arguments="-A -sV -p3306 --script mysql-brute.nse",callback=callbackMySQL)
    self.scanning()
    print 'Checking mysql-databases.nse.....'
    self.nmasync.scan(hostname,
arguments="-A -sV -p3306 --script mysql-databases.nse",callback=callbackMySQL)
    self.scanning()
    print 'Checking mysql-databases.nse.....'
    self.nmasync.scan(hostname,
```

Check FTP Login Anonymous



Shodan Developers Book View All...

 SHODAN port:21 Anonymous user logged in

 Exploits  Maps  Share Search  Download Results

TOP COUNTRIES



United States	192,483
China	42,635
Germany	32,027
United Kingdom	12,000
Korea, Republic of	10,477

Check FTP Login Anonymous



```
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>

Branch: `master` ▾ [pywebfuzz](#) / [pywebfuzz](#) / [data](#) / [Discovery](#) / **PredictableRes** /

nhamiel Added the updated data directory ...

..	
CMS	Added the updated data directory
Apache.fuzz.txt	Added the updated data directory
ApacheTomcat.fuzz.txt	Added the updated data directory
CGI_HTTP_POST.fuzz.txt	Added the updated data directory
CGI_HTTP_POST_Windows.fuzz.txt	Added the updated data directory
CGI_Microsoft.fuzz.txt	Added the updated data directory
CGI_XPlatform.fuzz.txt	Added the updated data directory
ColdFusion.fuzz.txt	Added the updated data directory
HTTP_POST_Microsoft.fuzz.txt	Added the updated data directory
Hyperion.fuzz.txt	Added the updated data directory
JBoss.fuzz.txt	Added the updated data directory
JavaServlets_Common.fuzz.txt	Added the updated data directory
KitchensinkDirectories.fuzz.txt	Added the updated data directory
Logins.fuzz.txt	Added the updated data directory

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
['/admin.asp', '/admin.aspx', '/admin.cfm', '/admin.jsp', '/admin.php', '/a
ator.cfm', '/administrator.jsp', '/administrator.php', '/administrator.php4
fault.asp', '/exchange/logon.asp', '/gs/admin', '/index.php?u=', '/login.as
sp', '/logon.aspx', '/logon.jsp', '/logon.php', '/logon.php3', '/logon.php4
Testing..http://192.168.56.101/admin.asp
Testing..http://192.168.56.101/admin.aspx
Testing..http://192.168.56.101/admin.cfm
Testing..http://192.168.56.101/admin.jsp
Testing..http://192.168.56.101/admin.php
Testing..http://192.168.56.101/admin.php4
Testing..http://192.168.56.101/admin.pl
Testing..http://192.168.56.101/admin.py
Testing..http://192.168.56.101/admin.rb
Testing..http://192.168.56.101/administrator
Testing..http://192.168.56.101/administrator.asp
Testing..http://192.168.56.101/administrator.aspx
Testing..http://192.168.56.101/administrator.cfm
Testing..http://192.168.56.101/administrator.jsp
Testing..http://192.168.56.101/administrator.php
Testing..http://192.168.56.101/administrator.php4
Testing..http://192.168.56.101/administrator.pl
Testing..http://192.168.56.101/administrator.py
Testing..http://192.168.56.101/administrator.rb
Testing..http://192.168.56.101/admnistrator.php3
Testing..http://192.168.56.101/cgi-bin/sqwebmail?noframes=1
Testing..http://192.168.56.101/default.asp
Testing..http://192.168.56.101/exchange/logon.asp
Testing..http://192.168.56.101/gs/admin
Testing..http://192.168.56.101/index.php?u=
[+] Found Login Resource: /index.php?u=
Testing..http://192.168.56.101/login.asp
Testing..http://192.168.56.101/login.aspx
Testing..http://192.168.56.101/login.cfm
Testing..http://192.168.56.101/login.php
```

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>

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)
```

```
Choose an option:Port 443 open
```

```
##### Started scanning for checking OPENSSL Heartbleed vulnerability '176.34.114.90' #####
```

```
Connecting with ...176.34.114.90 Port: 443
```

```
Sending Client Request...
```

```
Waiting for Server Request...
```

```
... received message: type = 22, ver = 0302, length = 58
```

```
Sending heartbeat request...
```

```
... received message: type = 22, ver = 0302, length = 754
```

```
... received message: type = 22, ver = 0302, length = 525
```

```
... received message: type = 22, ver = 0302, length = 4
```

```
... received message: type = 24, ver = 0302, length = 16384
```

```
Received heartbeat response:
```

Heartbleed



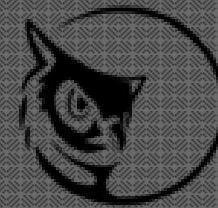
```
3ef0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3f90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3fa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

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[®]



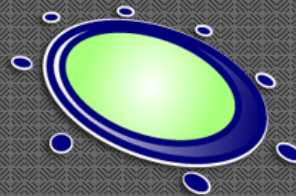
w3af



nexpose[®]



Nessus[®]
vulnerability scanner



OpenVAS
Open Vulnerability Assessment System

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

```
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

Nexpose



```
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



```
[0]-->EXIT
[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



python-pentesting-tool — Edit

11 commits 1 branch 0 releases
















Branch: **master** [New pull request](#) [New file](#) [Find file](#) **HTTPS** <https://github.com/jmortega>

jortega new options Lat

python-msfrpc	new options
python-pywebfuzz	new options
CheckFTPVulnerable.py	new options
CheckFTPVulnerable.pyc	new options
CheckOpenSslVulnerable.py	new options
CheckOpenSslVulnerable.pyc	new options
CheckVuln_SQL_XSS_LFI.py	new options
CheckVuln_SQL_XSS_LFI...	new options
Checker.py	new options
Checker.pyc	new options
ExtractMails.py	new options
ExtractMails.pyc	new options

https://github.com/jmortega/europython_ethical_hacking



 ftp_brute_force	europython examples
 geolP	europython examples
 requests	europython examples
 sockets	europython examples
 NmapScan.py	europython examples
 NmapScanner.py	europython examples
 NmapScannerAsync.py	europython examples
 NmapScannerAsync.pyc	europython examples
 ShodanSearch.py	europython examples
 WebSpider.py	europython examples
 builtWithDemo.py	europython examples
 checkFTPAnonymousLogin.py	europython examples
 demofuzzdb.py	europython examples
 extractDataFromImages.py	europython examples
 extractDataFromPDF.py	europython examples

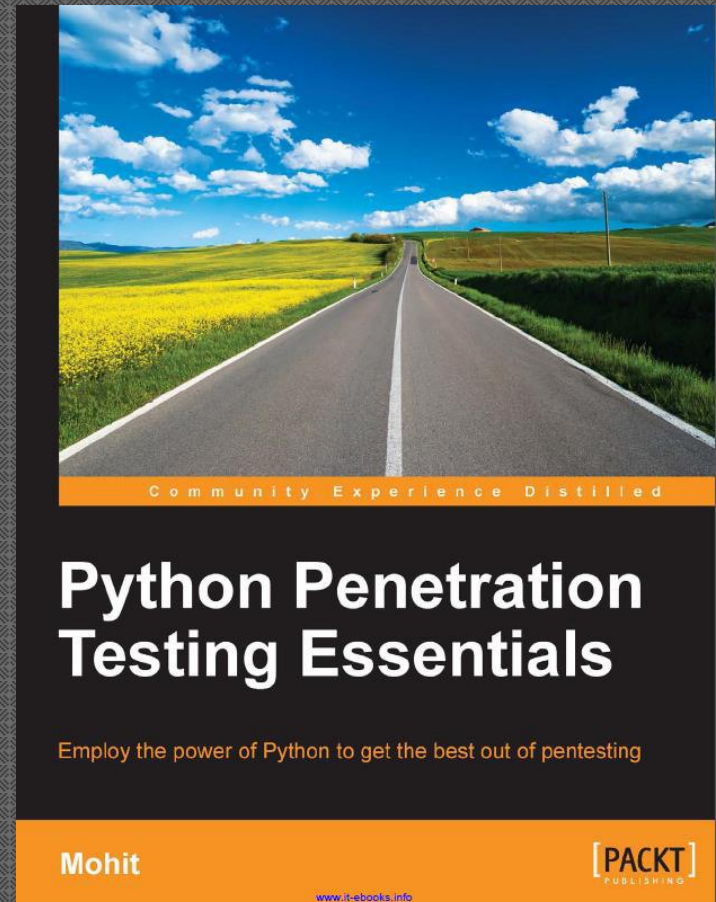
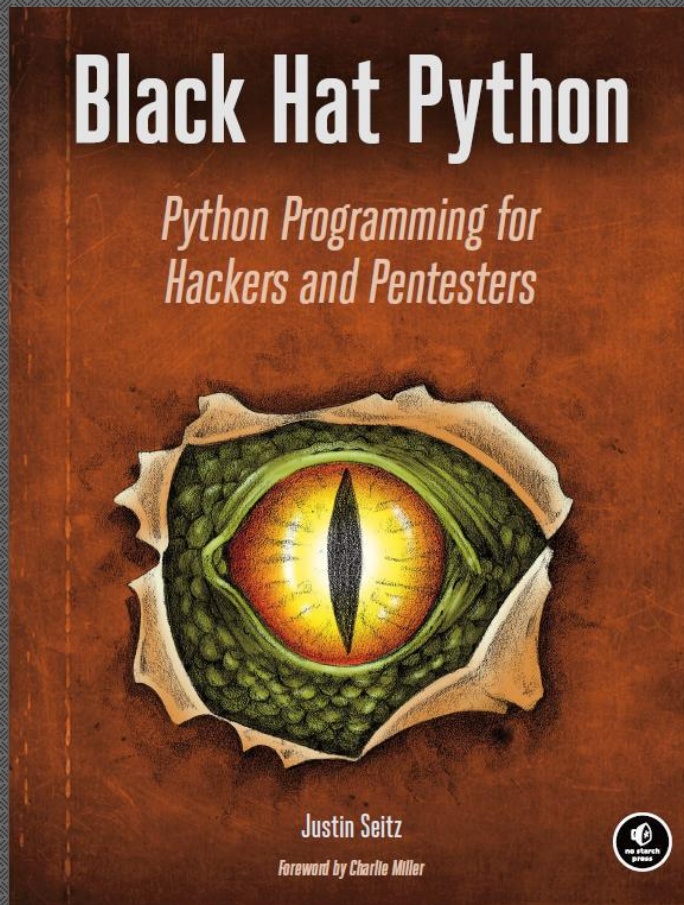
References & libs



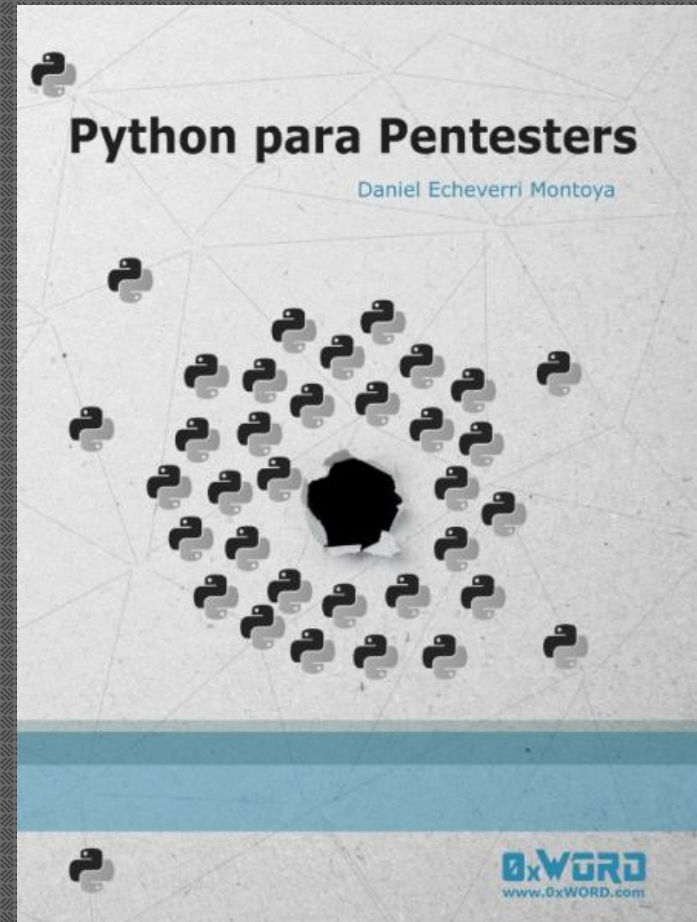
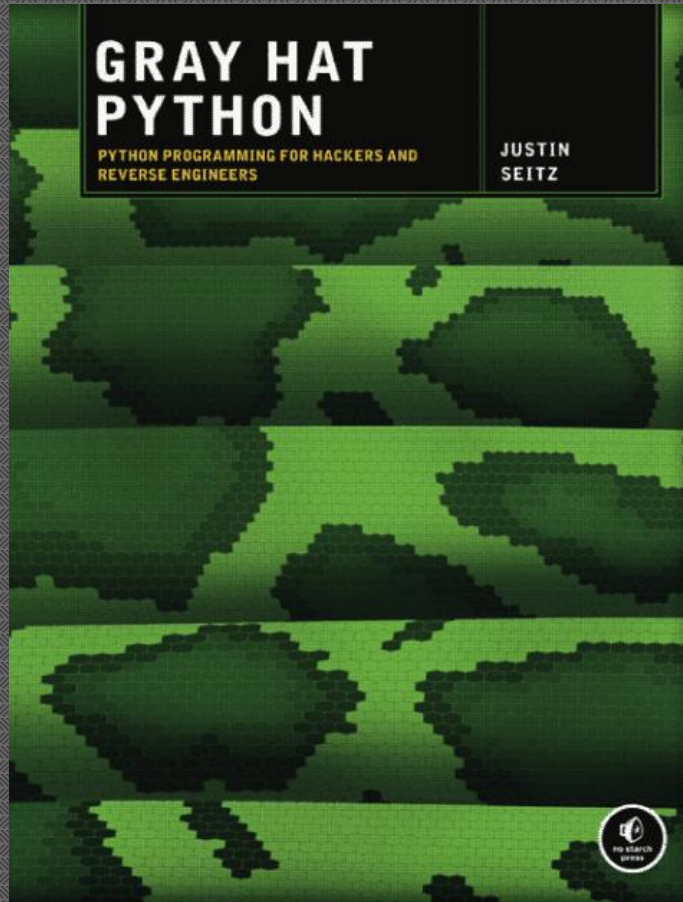
- <http://docs.shodanhq.com>
- <http://docs.python-requests.org/en/master/>
- <http://scrapy.org>
- <http://xael.org/pages/python-nmap-en.html>

- <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



Books





EUROPYTHON
2016 Bilbao, 17-24 July



THANK YOU!