PYCONES 7-9 OCT 2016

OSINT tools for security auditing

José Manuel Ortega
@jmortegac

http://jmortega.github.io

GitHub repository

https://github.com/jmortega/osint_tools_security_auditing

- Shodan
- TOR
- emails
- google+API
- images
- ip_map_position
- links
- maltego_python
- mapping
- metadata
- panoramio
- twitter
- youtube
- BuiltWith.py
- GeoLite2-City.mmdb
- README.md
- censys_data.py
- checkFullContactAPI.py
- checkIpDetails.py
- checkLinkedLinProfile.py
- check_social_networks.py
- github_repositories.py

Agenda

- OSINT introduction
- Server information(Censys,Shodan)
- OSINT tools developed with python
- Geolocation,Metadata
- Twitter,Footprinting,FullContact

OSINT

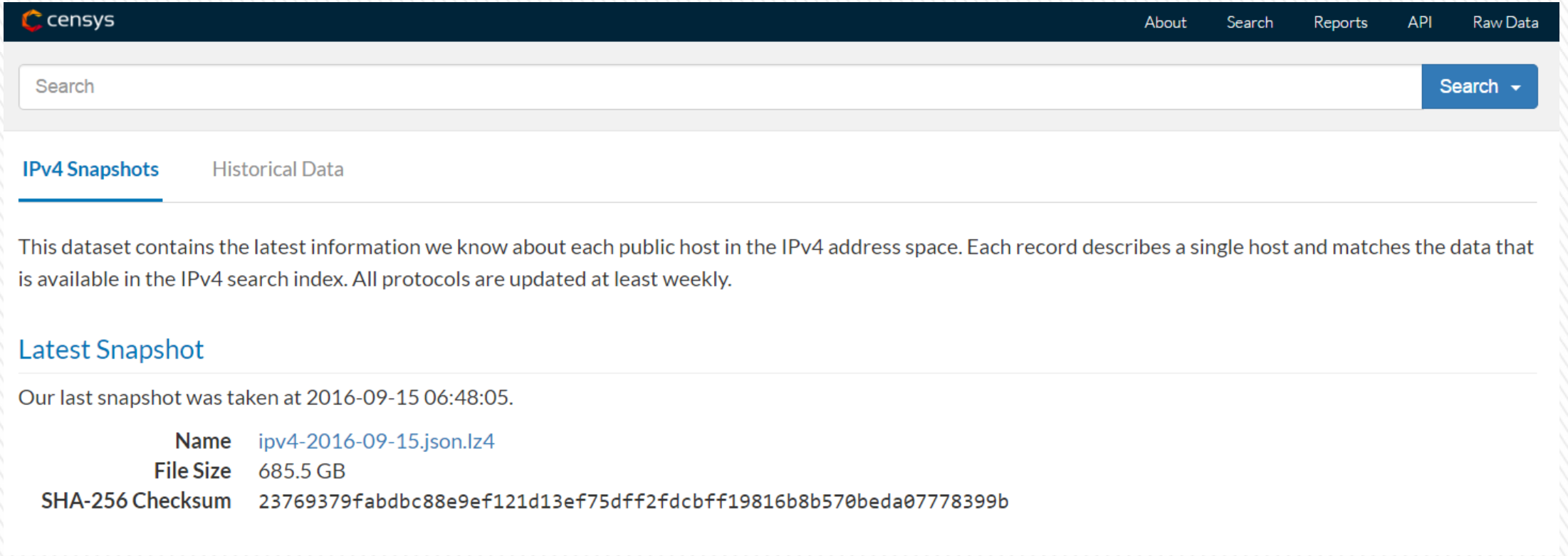
- Define a specific target and data you wish to obtain

- Technical-Accounts,servers,services,software
- Social-Social Media,Email,Photos
- Physical-Address,Home IP address,Footprinting
- Logical-Network,Operational intelligence

OSINT

- GeoLocation
- IP address
- Email address
- Telephone Number
- Usernames in social network profiles
- Metadata information from images
- Server information & vulnerabilities

Censys.io


censys
About
Search
Reports
API
Raw Data
Search
Search
IPv4 Snapshots
Historical Data
This dataset contains the latest information we know about each public host in the IPv4 address space. Each record describes a single host and matches the data that is available in the IPv4 search index. All protocols are updated at least weekly.
Latest Snapshot
Our last snapshot was taken at 2016-09-15 06:48:05.
Name
ipv4-2016-09-15.json.lz4
File Size
685.5 GB
SHA-256 Checksum
23769379fabdbc88e9ef121d13ef75dff2fdcbff19816b8b570beda07778399b

Censys.io

- https://www.censys.io/api/v1/view/ipv4/<ip address>
- https://www.censys.io/api/v1/view/websites/<domain>

```
def get_censys_data(ip,domain):

    if ip is not None:
        res = requests.get(API_URL + "/view/ipv4/"+ip, auth=(UID,SECRET))

    if domain is not None:
        res = requests.get(API_URL + "/view/websites/"+domain, auth=(UID,SECRET))

    if res.status_code != 200:
        print("error occurred: %s" % res.json()["error"])
        sys.exit(1)

    print(json.dumps(res.json(), indent=4))

    with open('data_censys.txt', 'w') as file:
        json.dump(res.json(), file, ensure_ascii=False,indent=4)
```

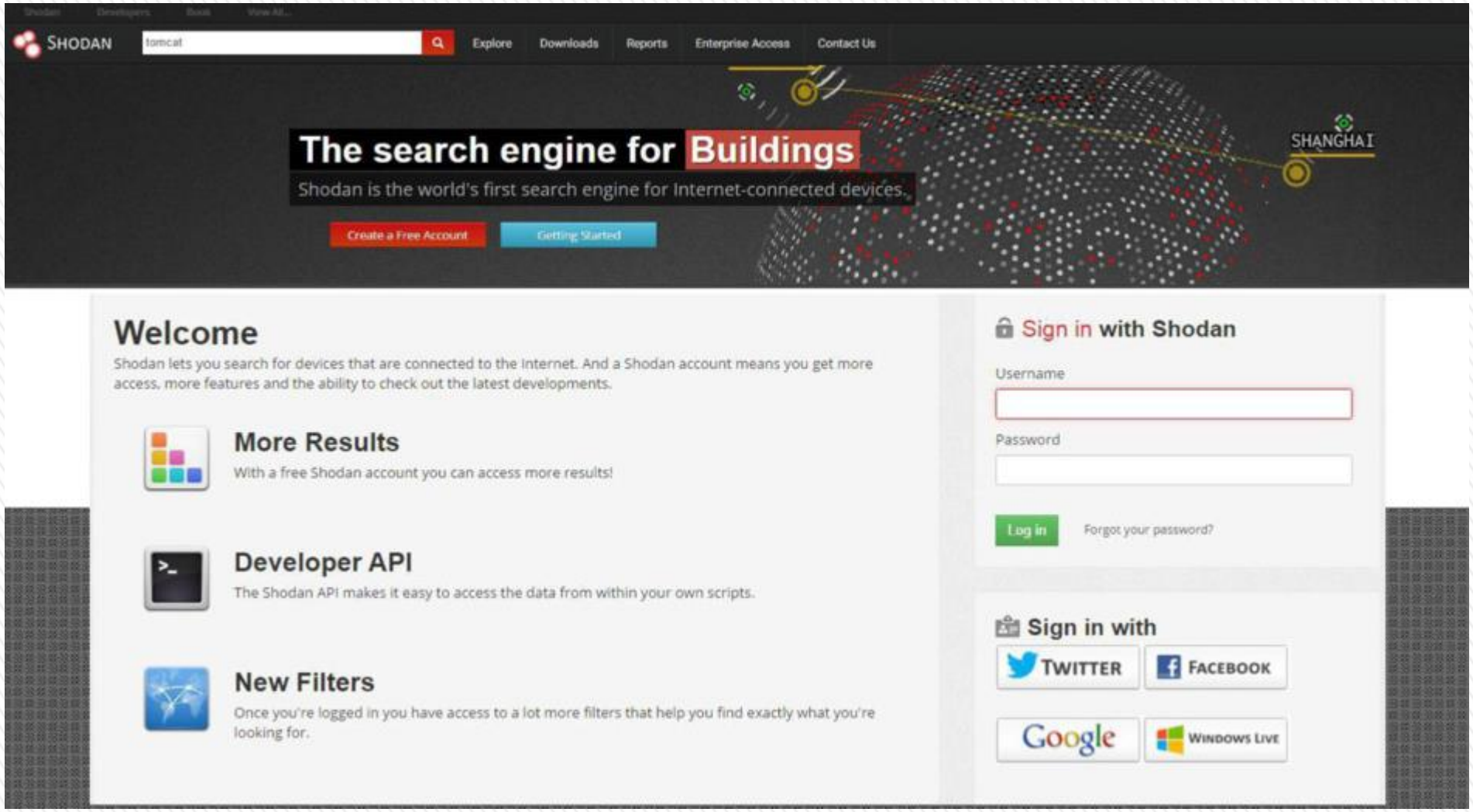
Censys.io

```
python censys_data.py -d python.org
```


```
            "mail.python.org"
        ],
        "organization": [
            "Python Software Foundation"
        ],
        "locality": [
            "Beaverton"
        ]
    },
    "issuer": {
        "country": [
            "IL"
        ],
        "organization": [
            "StartCom Ltd."
        ],
        "common_name": [
            "StartCom Class 2 Primary Intermediate Server CA"
        ],
        "organizational_unit": [
            "Secure Digital Certificate Signing"
        ]
```

```
        },
        "organization": "Rackspace Hosting",
        "asn": 27357,
        "rir": "unknown",
        "description": "RACKSPACE - Rackspace Hosting,US",
        "name": "RACKSPACE",
        "routed_prefix": "104.130.0.0/18"
    },
    "updated_at": "2016-09-24T10:30:10+00:00",
    "location": {
        "province": "Texas",
        "registered_country_code": "US",
        "postal_code": "78218",
        "latitude": 29.4889,
        "registered_country": "United States",
        "country": "United States",
        "longitude": -98.3987,
        "country_code": "US",
        "city": "San Antonio",
        "continent": "North America",
        "timezone": "America/Chicago"
    }
```

Shodan


SHODAN
tomcat
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
TWITTER
FACEBOOK
Google
WINDOWS LIVE

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

```
python demoShodanSearch.py -target 2016.es.pycon.org
```

```
IP: 144.76.246.116
Organization: Server Block
Operating System: None

Port: 22
Banner: SSH-2.0-OpenSSH_6.7p1 Debian-3
Key type: ssh-rsa
Key: AAAAB3NzaC1yc2EAAAADAQABAAABAQDeCws6s5+nMDJvTXElKw1HTQMHlDOVSCTMu6Ck4XlxOBhz
wAu1jh1mvD7Cm1gD/PmotlS8qiJUiwoXdjf8viRSkHAxDCp2ZN4wRM/V+yzkuL1CnLkoHya9FqKE
qdxirhyqUrzsNVEO2IGs5Yik4p7i/TGQKzKY+8BQpHfqdB84BOOgigX6S3TPIaYHqnkwuDUtG5H
Ap2u1zoNH4ap/K73Oq92UxeGU5xB+oq5V6q42QDA1/abRmxwYwzty7AjBd3rcf7x1p88qCRiRZzd
TSv4W88WoosTxBQeD/1vvmOPEfIMnKIh2ugnmzm9QynMxCdkeLXr7juIOXXTotdWdK/h
Fingerprint: 95:9f:ff:eb:58:86:57:18:18:05:f4:18:b4:60:71:61

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

Shodan

- **Checking data with ip address**
- **https://www.shodan.io/host/144.76.246.116**

144.76.246.116 static.116.246.76.144.clients.your-server.de

Country	Germany
Organization	Server Block
ISP	Server Block
Last Update	2016-09-21T02:03:35.595690
Hostnames	static.116.246.76.144.clients.your-server.de
ASN	AS24940

Ports

22 111

Services

22 tcp ssh

OpenSSH Version: 6.7p1 Debian 3

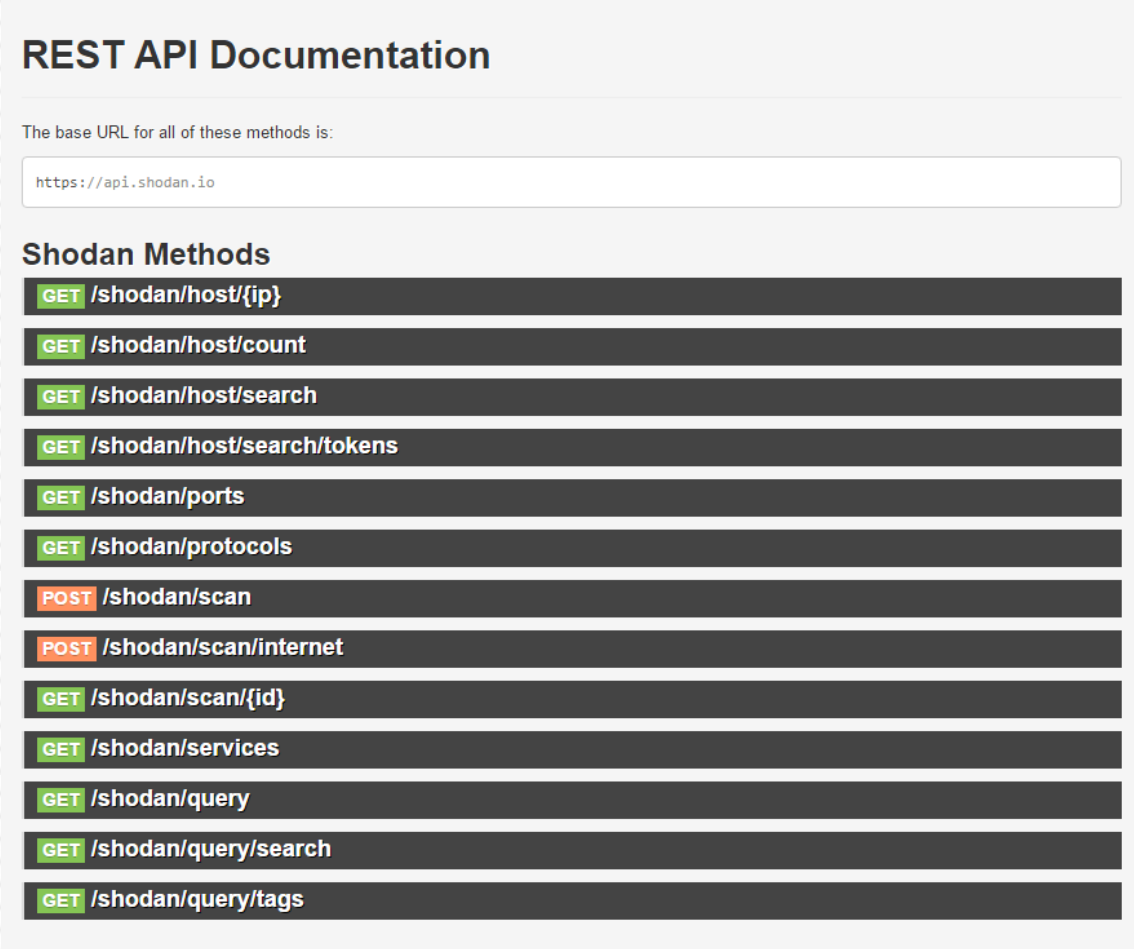
```
SSH-2.0-OpenSSH_6.7p1 Debian-3
Key type: ssh-rsa
Key: AAAAB3NzaC1yc2EAAAADAQABAAABAQDeCws6s5+nMDJvTXE1Kw1HTQMH1DOVSCTMu6Ck4X1x0Bhz
wAu1jh1mvD7Cm1gD/Pmot1S8qiJUiwoXdjf8viRSkHAxDCp2ZN4wRM/V+yzkuL1CnLkoHya9FqKE
qdxirhyqUrzsNVEO2IGs5Yik4p7i/TGQKzKY+8BQpHfqdB84B00OgigX6S3TPIaYHqnkwuDUtG5H
Ap2u1zoNH4ap/K730q92UxeGU5xB+oq5V6q42QDA1/abRmxwYwzty7AjBd3rcf7x1p88qCRiRZzd
TSv4W88WoosTxBQeD/1vvm0PEfIMnKIh2ugnmzm9QynMxCdkeLXr7juIOXXTotdWdK/h
Fingerprint: 95:9f:ff:eb:58:86:57:18:18:05:f4:18:b4:60:71:61

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

Shodan CVE vulns

```
# Print vuln information
for item in host['vulns']:
    CVE = item.replace('!','')
    print 'Vulns: %s' % item
    exploits = api.exploits.search(CVE)
    for item in exploits['matches']:
        if item.get('cve')[0] == CVE:
            print item.get('description')
```

Shodan Developer API

https://developer.shodan.io/api



Recon-ng

- https://bitbucket.org/LaNMaSteR53/recon-ng
- **Open Source OSINT toolkit written in python**
- **Actively maintained**
- **Uses modules and saves all recollected information in databases**

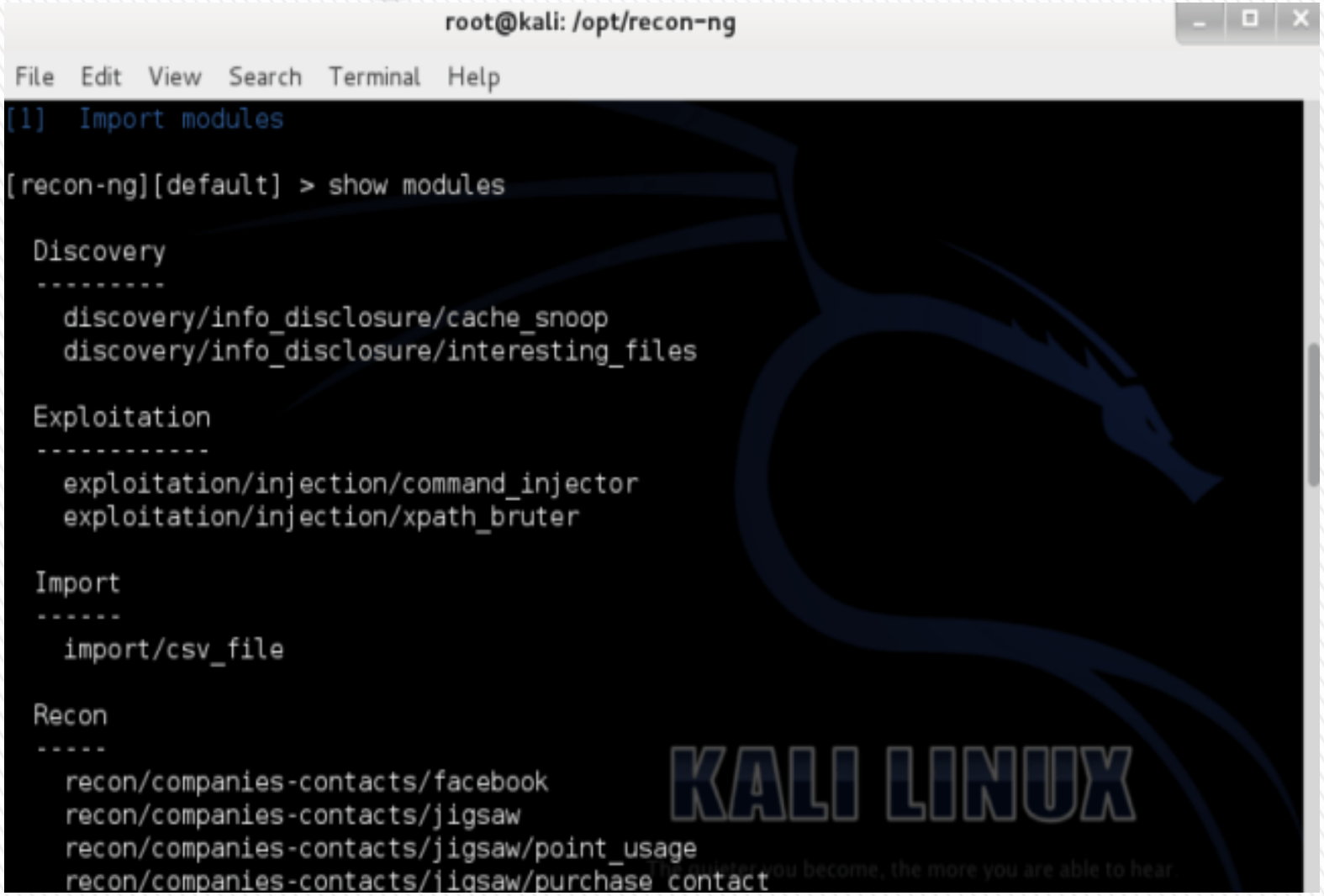
Recon-ng dependences

- dnspython - http://www.dnspython.org/
- dicttoxml - https://github.com/quandyfactory/dicttoxml/
- jsonrpclib - https://github.com/joshmarshall/jsonrpclib/
- lxml - http://lxml.de/
- slowaes - https://code.google.com/p/slowaes/
- XlsxWriter - https://github.com/jmcnamara/XlsxWriter/
- Mechanize
- PyPDF2
- sqlite3

Recon-ng


Consulting | Research | Development | Training
http://www.blackhillsinfosec.com
[recon-ng v4.4.0, Tim Tomes (@LaNMaSteR53)]
[65] Recon modules
[7] Reporting modules
[2] Import modules
[2] Exploitation modules
[2] Discovery modules
[recon-ng][default] >
KALI LINUX

Recon-ng modules



Recon-ng modules

master | Recon-ng / modules /

- ..
- discovery/info_disclosure
- exploitation/injection
- import
- recon
- reporting

master | Recon-ng / modules / recon /

- ..
- companies-contacts
- companies-multi
- contacts-contacts
- contacts-credentials
- contacts-domains
- contacts-profiles
- credentials-credentials
- domains-contacts
- domains-credentials/pwnedlist

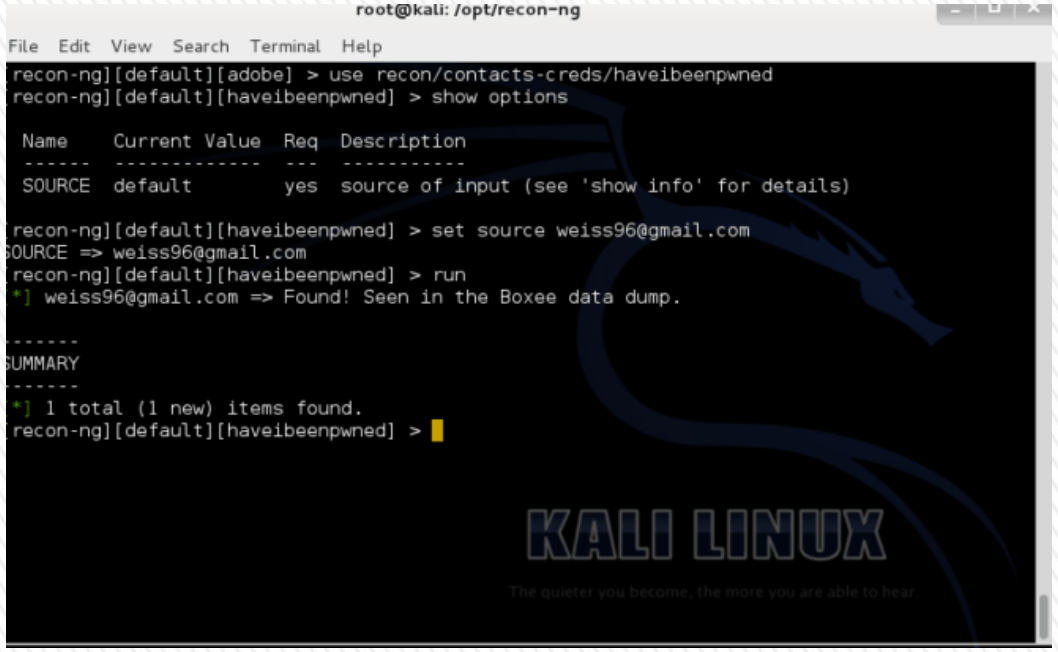
Recon-ng mailtester

```
[recon-ng][default][mailtester] > show options

  Name    Current Value  Required  Description
  ------  -------------  --------  -----------
  REMOVE  False          yes       remove invalid email addresses
  SOURCE  default        yes       source of input (see 'show
info' for details)

[recon-ng][default][mailtester] > run
[*] steve@holdenweb.com => E-mail address is valid
[recon-ng][default][mailtester] >
```

```
[recon-ng][default][mailtester] > run
[*] DATABASE => /home/bbking/.recon-ng/workspaces/default/data.db
[*] QUERY => SELECT DISTINCT email FROM contacts WHERE email IS NOT NULL
send: 'POST /testmail.php HTTP/1.1\r\nAccept-Encoding: identity\r\nContent-Length:
35\r\nHost: www.mailtester.com\r\nContent-Type: application/x-www-form-
urlencoded\r\nConnection: close\r\nUser-Agent: Recon-
ng/v4\r\n\r\nlang=en&email=steve%40holdenweb.com'
reply: 'HTTP/1.1 200 OK\r\n'
. . .
[*] steve@holdenweb.com => E-mail address is valid
[*] DATABASE => /home/bbking/.recon-ng/workspaces/default/data.db
[*] QUERY => INSERT OR REPLACE INTO dashboard (module, runs) VALUES ('recon/contacts-
contacts/mailtester', COALESCE((SELECT runs FROM dashboard WHERE
module='recon/contacts-contacts/mailtester')+1, 1))
```

```python
class Module(BaseModule):

    meta = {
        'name': 'MailTester Email Validator',
        'author': 'Tim Tomes (@LaNMaSteR53)',
        'description': 'Leverages MailTester.com to validate email addresses.',
        'query': 'SELECT DISTINCT email FROM contacts WHERE email IS NOT NULL',
        'options': (
            ('remove', False, True, 'remove invalid email addresses'),
        ),
    }

    def module_run(self, emails):
        url = 'http://www.mailtester.com/testmail.php'
        error = 'Too many requests from the same IP address.'
        payload = {'lang':'en'}
        for email in emails:
            payload['email'] = email
            resp = self.request(url, method='POST', payload=payload)
            if error in resp.text:
                self.error(error)
                break
            tree = fromstring(resp.text)
            # clean up problematic HTML for debian based distros
            tree.forms[0].getparent().remove(tree.forms[0])
            msg_list = tree.xpath('//table[last()]/tr[last()]/td[last()]/text()')
            msg = ' '.join([x.strip() for x in msg_list])
            output = self.alert if 'is valid' in msg else self.verbose
            output('%s => %s' % (email, msg))
            if 'does not exist' in msg:
                self.query('UPDATE contacts SET email=NULL where email=?', (email,))
```

Recon-ng breach search




```python
from recon.core.module import BaseModule
import urllib

class Module(BaseModule):

    meta = {
        'name': 'Have I been pwned? Breach Search',
        'author': 'Tim Tomes (@LaNMaSteR53) & Tyler Halfpop (@tylerhalfpop)',
        'description': 'Leverages the haveibeenpwned.com API to determine if email addresses are associa
        'query': 'SELECT DISTINCT email FROM contacts WHERE email IS NOT NULL',
    }

    def module_run(self, accounts):
        # retrieve status
        base_url = 'https://haveibeenpwned.com/api/v2/%s/%s'
        endpoint = 'breachedaccount'
        for account in accounts:
            resp = self.request(base_url % (endpoint, urllib.quote(account)))
            rcode = resp.status_code
            if rcode == 404:
                self.verbose('%s => Not Found.' % (account))
            elif rcode == 400:
                self.error('%s => Bad Request.' % (account))
                continue
            else:
                for breach in resp.json:
                    self.alert('%s => Breach found! Seen in the %s breach that occurred on %s.' % (accou
                self.add_credentials(account)
```

Recon-ng subdomains

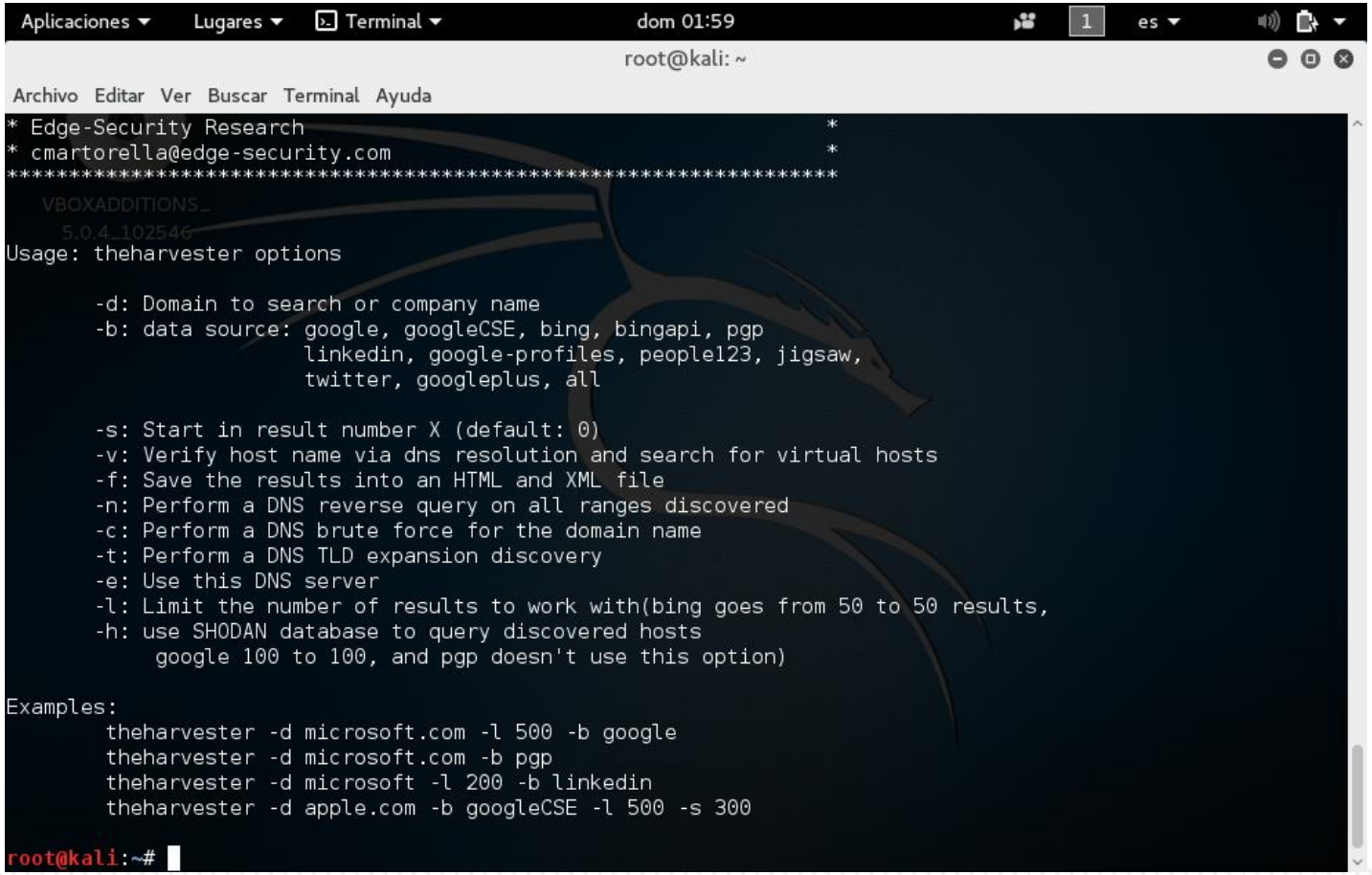
```
class Module(BaseModule):
    meta = {
        'name': 'HackerTarget Lookup',
        'author': 'Michael Henriksen (@michenriksen)',
        'description': 'Uses the HackerTarget.com API to find host names. Updates the \'hosts\' table with the results.',
        'query': 'SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL',
    }

    def module_run(self, domains):
        for domain in domains:
            self.heading(domain, level=0)
            url = 'https://api.hackertarget.com/hostsearch/'
            payload = {'q': domain}
            resp = self.request(url, payload=payload)
            if resp.status_code is not 200:
                self.error('Got unexpected response code: %i' % resp.status_code)
                continue
            if resp.text == '':
                self.output('No results found.')
                continue
            for line in resp.text.split("\n"):
                line = line.strip()
                if line == '':
                    continue
                host, address = line.split(",")
                self.add_hosts(host=host, ip_address=address)
```

Recon-ng Shodan API

```
def search_shodan_api(self, query, limit=0):
    api_key = self.get_key('shodan_api')
    url = 'https://api.shodan.io/shodan/host/search'
    payload = {'query': query, 'key': api_key}
    results = []
    cnt = 1
    page = 1
    self.verbose('Searching Shodan API for: %s' % (query))
    while True:
        resp = self.request(url, payload=payload)
        if resp.json == None:
            raise framework.FrameworkException('Invalid JSON response.\n%s' % (resp.text))
        if 'error' in resp.json:
            raise framework.FrameworkException(resp.json['error'])
        if not resp.json['matches']:
            break
        # add new results
        results.extend(resp.json['matches'])
        # check limit
        if limit == cnt:
            break
        cnt += 1
        # next page
        page += 1
        payload['page'] = page
    return results
```

The harvester

- https://github.com/laramies/theHarvester



The harvester modules

›

Python modules

- httplib
- socket
- requests
- shodan

```
import string
import httplib
import sys
import os
from socket import *
import re
import getopt

try:
    import requests
except:
    print "Request library not found,
    sys.exit()

from discovery import *
from lib import htmlExport
from lib import hostchecker
```

The harvester

```
[-] Searching in Bing:
root@kali:~# theharvester -d pycon.org -l 500 -b google

*******************************************************************
*                                                                 *
* | |_| |__   ___     /\  /\__ _ _ ____   _____ ___| |_ ___ _ __  *
* | __| '_ \ / _ \   / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/  / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___|  \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* TheHarvester Ver. 2.6                                           *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*******************************************************************


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

```
[+] Emails found:
------------------
contacto2016@es.pycon.org
info@pycon.org
swc@za.pycon.org

[+] Hosts found in search engines:
------------------------------------
[-] Resolving hostnames IPs...
148.251.113.227:www.pycon.org
144.76.246.116:2016.es.pycon.org
69.164.212.224:za.pycon.org
151.101.60.223:us.pycon.org
82.166.0.152:il.pycon.org
151.101.60.133:na.pycon.org
106.187.52.176:tw.pycon.org
144.76.246.116:2015.es.pycon.org
205.147.96.46:in.pycon.org
188.166.192.28:ua.pycon.org
52.50.6.31:cz.pycon.org
94.23.84.75:es.pycon.org
87.204.59.165:pl.pycon.org
106.187.52.176:Tw.pycon.org
158.69.9.197:ar.pycon.org
```

OSR framework

- **pip install osrframework**
- **Developed in python 2.7**
- **Integrates with maltego transforms**
- https://pypi.python.org/pypi/osrframework/0.13.2
- https://github.com/i3visio/osrframework

OSR python modules

- **BeautifulSoup**
- **Requests**
- **Mechanize**
- **pyDNS**→resolving name servers
- **python-whois**→to recover the whois info from a domain
- **tweepy**→for connecting with Twitter API
- **Skype4Py**→ for connecting with Skype API
- **Python-emailahoy**→for checking email address
- **Multiprocessing**→import Process, Queue, Pool

OSR python scripts

alias_generator.py	Added osrfconsole script to control de utilities in the framework to ...
domainfy.py	Check information against DNS instead of making HTTP GET queries
entify.py	Added osrfconsole script to control de utilities in the framework to ...
enumeration.py	Added osrfconsole script to control de utilities in the framework to ...
mailfy.py	Added osrfconsole script to control de utilities in the framework to ...
osrfconsole.py	Fix issue #166 by adding a split to information in PLATFORMS
phonefy.py	Added osrfconsole script to control de utilities in the framework to ...
searchfy.py	Added osrfconsole script to control de utilities in the framework to ...
usufy.py	Added osrfconsole script to control de utilities in the framework to ...

OSR python scripts

```
python searchfy.py -q "pycon" -p "twitter"
```

i3visio_uri	i3visio_alias	i3visio_platform
http://twitter.com/uapycon	uapycon	Twitter
http://twitter.com/drakekin	drakekin	Twitter
http://twitter.com/PyConJ	PyConJ	Twitter
http://twitter.com/pyconit	pyconit	Twitter
http://twitter.com/PyConES	PyConES	Twitter
http://twitter.com/pyconvolunteers	pyconvolunteers	Twitter
http://twitter.com/PyConVE	PyConVE	Twitter
http://twitter.com/pycon	pycon	Twitter
http://twitter.com/PyConUK	PyConUK	Twitter
http://twitter.com/pyconindia	pyconindia	Twitter
http://twitter.com/PyConDE	PyConDE	Twitter

OSR python scripts

```
python usufy.py -n pycones
```

```
Platform selection arguments:
  Criteria for selecting the platforms where performing the search.

  -p <platform> [<platform> ...], --platforms <platform> [<platform> ...]
                        select the platforms where you want to perform the
                        search amongst the following: ['all', '500px',
                        'about', 'anarchy101', 'aporrealos', 'apsense',
                        'archive', 'arduino', 'ariva', 'armorgames',
                        'artbreak', 'artician', 'askfm', 'audiob', 'audioboo',
                        'authorstream', 'autospies', 'backyardchickens',
                        'badoo', 'behance', 'bennugd', 'bitbucket',
                        'bitcointa', 'bitcointalk', 'bitly', 'blackplanet',
                        'bladna', 'blip', 'blogspot', 'bookofmatches',
                        'boonex', 'bordom', 'boxedup', 'breakcom',
                        'bucketlistly', 'burbuja.info', 'burdastyle',
                        'buzznet', 'cafemom', 'carbonmade', 'cardomain',
                        'care2', 'castroller', 'causes', 'ccsinfo', 'chess',
                        'cockos', 'connectingsingles', 'couchsurfing',
                        'dailymail', 'dailymotion', 'deviantart',
                        'digitalspy', 'disqus', 'doodle', 'douban',
                        'dribbble', 'drugbuyersforum', 'drupal', 'ebay',
                        'echatta', 'ehow', 'elmundo', 'enfemenino',
                        'ethereum', 'etsy', 'facebook', 'fanbitcoin',
                        'fanpop', 'fark', 'favstar', 'flickr', 'flixster',
```

OSR python scripts

```
python usufy.py -n pycones
```

```
                        Starting search in 230 platform(s)... Relax!

[!] WARNING. Something happened when trying to attach OSRFramework to a valid Skype se
[!] Exception grabbed: Skype attach timeout
[!] In spite of this message, execution is going on.

A summary of the results obtained are shown in the following table:
Sheet Name: Profiles recovered (2016-9-25_16h52m).
+-----------------------------------------------------+---------------+-------------
|                      i3visio_uri                    | i3visio_alias | i3visio_plat
+=====================================================+===============+=============
| http://twicsy.com/u/pycones                         | pycones       | Twicsy
+-----------------------------------------------------+---------------+-------------
| http://www.klout.com/pycones                        | pycones       | Klout
+-----------------------------------------------------+---------------+-------------
| http://favstar.fm/users/pycones                     | pycones       | Favstar
+-----------------------------------------------------+---------------+-------------
| http://twitter.com/pycones                          | pycones       | Twitter
+-----------------------------------------------------+---------------+-------------
| http://www.ummahforum.com/member.php?username=pycones | pycones       | Ummahforum
+-----------------------------------------------------+---------------+-------------
```

OSR python scripts

checkIfEmailWasHacked.py	Added Tor search platform to searchfy.py
checkIfHashIsCracked.py	Added Tor search platform to searchfy.py
checkInSkype.py	Checked that mailfy.py work on Linux.
checkIpDetails.py	Added Tor search platform to searchfy.py
checkIpFromAlias.py	Checked that mailfy.py work on Linux.
checkPhoneDetails.py	Added Tor search platform to searchfy.py
getBitcoinAddressDetails.py	Added Tor search platform to searchfy.py

```
python checkIpDetails.py -q 2016.es.pycon.org
```

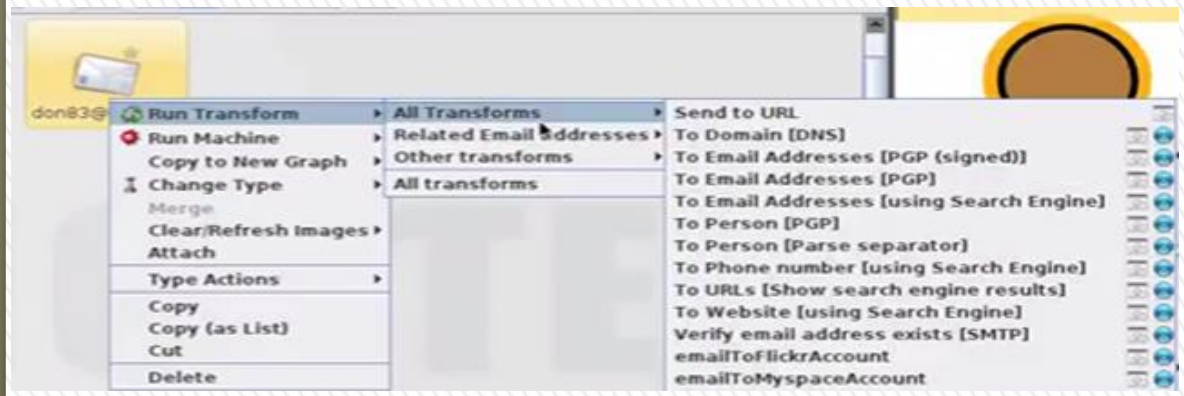
```
try:
    apiURL = "http://ip-api.com/json/" + query


    # Accessing the ip-api.com RESTful API
    data = urllib2.urlopen(apiURL).read()

    # Reading the text data onto python structures
    apiData = json.loads(data)

    # i3visio structure to be returned
    jsonData = []
```

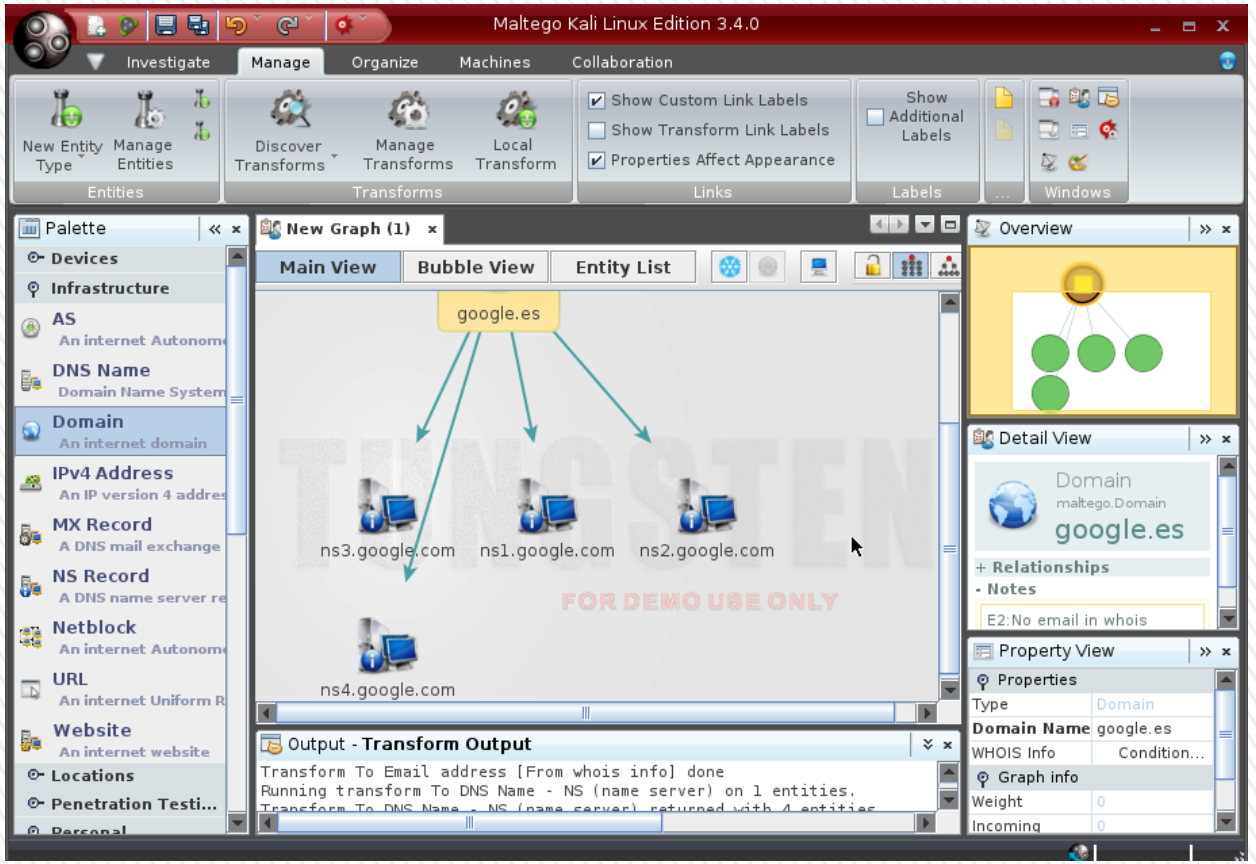

```
    ],
    "type": "i3visio.location.country",
    "value": "Germany"
  },
  {
    "attributes": [],
    "type": "i3visio.text",
    "value": "SERVER BLOCK"
  },
  {
    "attributes": [],
    "type": "i3visio.text",
    "value": "Europe/Berlin"
  },
  {
    "attributes": [],
    "type": "i3visio.ipv4",
    "value": "144.76.246.116"
  },
  {
    "attributes": [],
    "type": "i3visio.location.geo",
    "value": "49.1247, 10.7806"
```

Maltego transforms

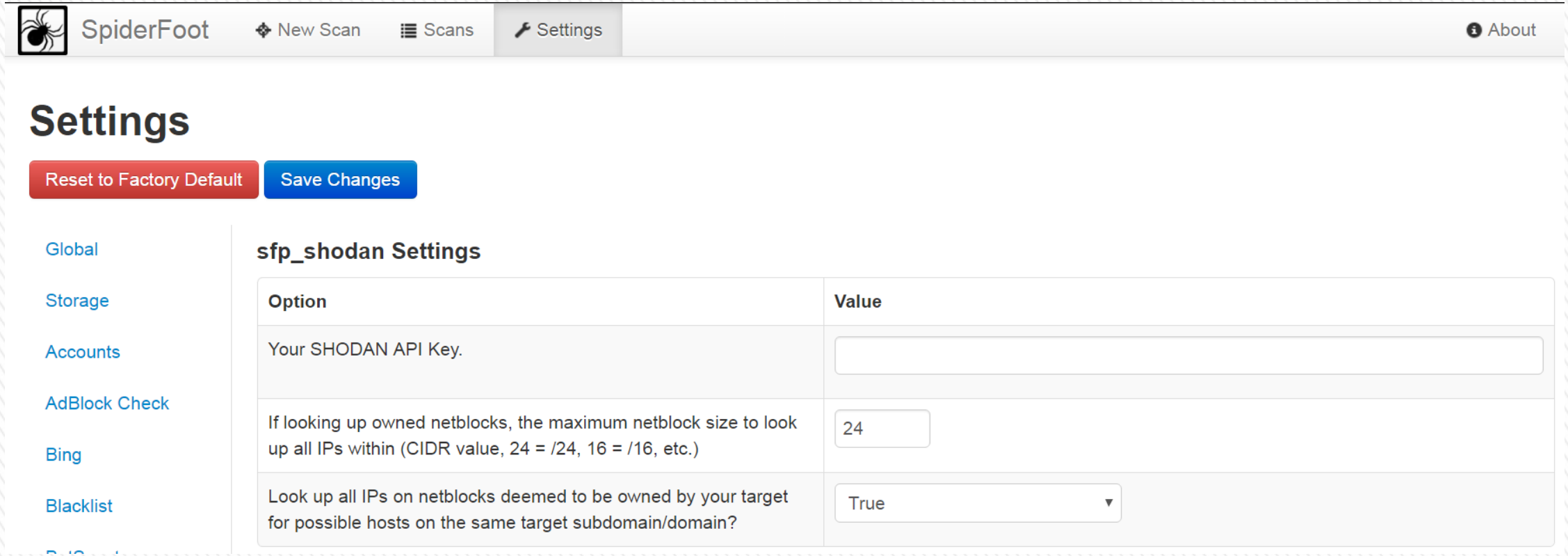
```
python maltego_emails.py -d http://2016.es.pycon.org

<MaltegoMessage>
<MaltegoTransformResponseMessage>
        <Entities>
<Entity Type="maltego.EmailAddress">
<Value>patrocinadores2016@es.pycon.org</Value>
</Entity>
<Entity Type="maltego.EmailAddress">
<Value>patrocinadores2016@es.pycon.org</Value>
</Entity>
<Entity Type="maltego.EmailAddress">
<Value>contacto2016@es.pycon.org</Value>
</Entity>
<Entity Type="maltego.EmailAddress">
<Value>contacto2016@es.pycon.org</Value>
</Entity>
</Entities>
</MaltegoTransformResponseMessage>
</MaltegoMessage>
```

Maltego transforms





SpiderFoot-Footprinting



SpiderFoot-data sources

Source	Location	Notes
abuse.ch	http://www.abuse.ch	Various malware trackers.
AdBlock	https://easylist-downloads.adblockplus.org/easylist.txt	AdBlock pattern matches
AlienVault	https://reputation.alienvault.com	AlienVault's IP reputation database.
Autoshun.org	http://www.autoshun.org	Blacklists.
AVG Site Safety Report	http://www.avgthreatlabas.com	Site safety checker.
Bing	http://www.bing.com	Scraping but future version to also use API.
Blocklist.de	http://lists.blocklist.de	Blacklists.
Checkusernames.com	http://www.checkusernames.com	Look up username availability on popular sites.
DNS	Your configured DNS server.	Defaults to your local DNS but can be configured to whatever IP address you supply SpiderFoot.
DomainTools	http://www.domaintools.com	
DroneBL	http://www.dronebl.org	
Facebook	http://www.facebook.com	Scraping but future version to also use API.
FreeGeoIP	http://freegeoip.net	
Google	http://www.google.com	Scraping but future version to also use API.
Google+	http://plus.google.com	Scraping but future version to also use API.
Google Safe Browsing	http://www.google.com/safebrowsing	Site safety checker.
LinkedIn	http://www.linkedin.com	Scraping but future version to also use API.
malc0de.com	http://malc0de.com	Blacklists.
malwaredomainlist.com	http://www.malwaredomainlist.com	Blacklists.

SpiderFoot-data sources

Source	Location	Notes
malwaredomains.com	http://www.malwaredomains.com	Blacklists.
McAfee SiteAdvisor	http://www.siteadvisor.com	Site safety checker.
NameDroppers	http://www.namedroppers.org	
Nothink.org	http://www.nothink.org	Blacklists.
OpenBL	http://www.openbl.org	Blacklists.
PasteBin	http://www.pastebin.com	Achieved through Google scraping.
PGP Servers	http://pgp.mit.edu/pks/	PGP public keys.
PhishTank	http://www.phishtank.org	Identified phishing sites.
Project Honeypot	http://www.projecthoneypot.org	Blacklists. API key needed.
RIPE/ARIN	http://stat.ripe.net/	
Robtex	http://www.robtex.com	
SANS ISC	http://isc.sans.edu	Internet Storm Center IP reputation database.
SHODAN	http://www.shodanhq.com	API key needed.
SORBS	http://www.sorbs.net	Blacklists.
SpamHaus	http://www.spamhaus.org	Blacklists.
ThreatExpert	http://www.threatexpert.com	Blacklists.
TOR Node List	http://torstatus.blutmagie.de	
TotalHash.com	http://www.totalhash.com	Domains/IPs used by malware.
UCEPROTECT	http://www.uceprotect.net	Blacklists.
VirusTotal	http://www.virustotal.com	Domains/IPs used by malware. API key needed.
Whois	Various	Whois servers for different TLDs.
Yahoo	http://www.yahoo.com	Scraping but future version to also use API.
Zone-H	http://www.zone-h.org	Easy to get black-listed. Log onto the site in a browser from the IP you're scanning from first and enter the CAPTCHA, then it should be fine.

SpiderFoot-modules

- **Python 2.7**
- **BeautifulSoup**
- **DNSPython**
- **Socks**
- **Socket**
- **SSL**
- **CherryPy**
- **M2MCrypto**
- **Netaddr**
- **pyPDF**

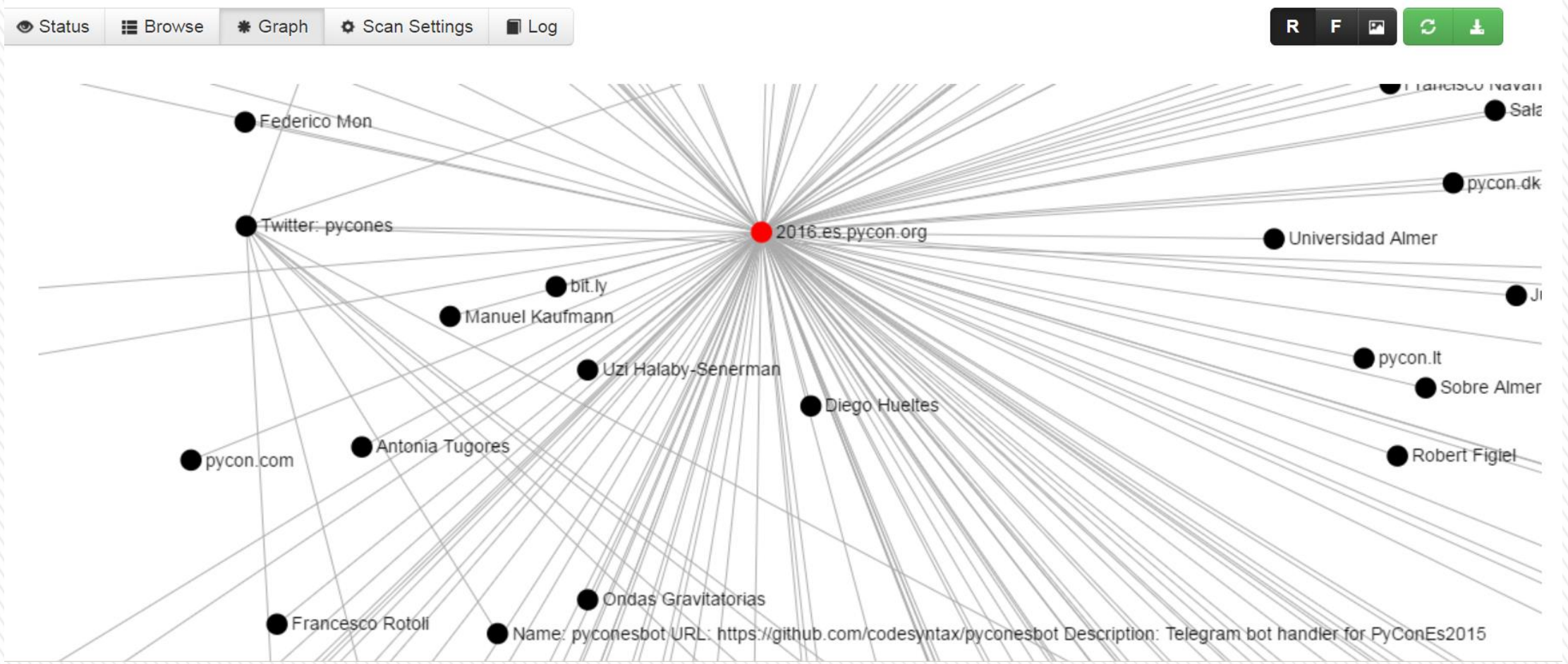
adblockparser	Added AdBlock parser module.
bs4	Spidering to utilise BeautifulSoup parsing and adjusted configuration.
dns	Added the DNSPython library.
exifread	Identify image meta data and extract software used from images, offic...
gexf	PyGEXF added
ispell	Identifying names in content.
metapdf	Extracting docx, xlsx, pptx and pdf meta data.
openxmllib	Extracting docx, xlsx, pptx and pdf meta data.
phonenumbers	Support for identifying phone numbers.
pyPdf	Extracting docx, xlsx, pptx and pdf meta data.
pythonwhois	Pythonwhois external module added.
stem	Stem license.
__init__.py	Module re-shuffle.
socks.py	TOR integration support.

SpiderFoot-BeautifulSoup

from bs4 import BeautifulSoup, SoupStrainer

```
try:
    for t in tags.keys():
        for lnk in BeautifulSoup(data, "lxml",
            parse_only=SoupStrainer(t)).find_all(t):
            if lnk.has_attr(tags[t]):
                urlsRel.append([None, lnk[tags[t]]])
except BaseException as e:
    self.error("Error parsing with BeautifulSoup: " + str(e), False)
    return None
```

SpiderFoot-Results



VirusTotal - API

```
def get_data_virus_total(ip,domain):

    url_ip_address = 'https://www.virustotal.com/vtapi/v2/ip-address/report'
    url_domain = 'https://www.virustotal.com/vtapi/v2/domain/report'

    #parameters = {'ip': ip, 'apikey': '-- YOUR API KEY --'}
    parameters_ip = {'ip': ip, 'apikey': '1c2fb58f31b82e29a93c6a87392b21bc3b6424
    parameters_domain = {'domain': domain, 'apikey': '1c2fb58f31b82e29a93c6a8739

    if ip is not None:

        response = requests.get(url_ip_address,params=parameters_ip)
        response_dict = json.loads(response.text)
        print(json.dumps(response_dict, sort_keys=True,indent=4))

        with open('data_virus_total.txt', 'w') as file:
                json.dump(response_dict, file, ensure_ascii=False,indent=4)
```

VirusTotal - report


```
    "ftp.python.org",
    "blog-de.python.org",
    "warehouse.python.org",
    "packaging.python.org",
    "console.python.org",
    "testpypi.python.org",
    "doc.python.org",
    "status.python.org",
    "front.python.org",
    "jobs.python.org",
    "planet.python.org",
    "packages.python.org",
    "dinsdale.python.org",
    "bugs.python.org",
    "cheeseshop.python.org",
    "svn.python.org",
    "hg.python.org",
    "mail.python.org",
    "legacy.python.org",
    "blog.python.org",
    "wiki.python.org",
    "g.pypi.python.org",
    "pl.python.org",
    "e.pypi.python.org",
    "www.python.org",
    "pypi.python.org",
    "docs.python.org"
],
"Websense ThreatSeeker category": "information technology",
"detected_urls": [
    {
        "url": "http://python.org/",
        "positives": 1,
        "total": 53,
        "scan_date": "2014-07-02 10:55:30"
    },
    {
        "url": "http://python.org/ftp/python/2.7.3/python-2.7.3.msi",
        "positives": 1,
        "total": 39,
        "scan_date": "2013-06-15 02:56:10"
    }
],
```

```
],
"resolutions": [
    {
        "last_resolved": "2014-11-01 00:00:00",
        "ip_address": "104.130.43.121"
    },
    {
        "last_resolved": "2014-02-20 00:00:00",
        "ip_address": "140.211.10.69"
    },
    {
        "last_resolved": "2016-06-28 00:00:00",
        "ip_address": "23.253.135.79"
    },
    {
        "last_resolved": "2013-10-04 00:00:00",
        "ip_address": "82.94.164.162"
    }
],
"domain_siblings": [],
"whois": null,
"Alexa domain info": "python.org is one of the top 10,000 sites
"verbose_msg": "Domain found in dataset",
"BitDefender category": "computersandsoftware",
"undetected_referrer_samples": [
```


Extract Metadata

- **PDF→PyPDF2,PDFMiner**
- **Images→Pillow,pyexiv2(python 2.7),gexiv2(python 3)**

```
[+] Metadata for file: images\img.jpg
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': -131.4675777777778}
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
```

GeoLocation

http://dev.maxmind.com/geoip/geoip2/geolite2/

```
import geoip2
import geoip2.database
```

```
def getGeo(self, ip):
    response = ''
    if not os.path.exists('GeoLite2-City.mmdb'):
        msg = "[*] GeoLite2-City.mmdb is not found ..."
        logging.error(msg)
    else:
        try:
            reader = geoip2.database.Reader('GeoLite2-City.mmdb')
            response = reader.city(ip)
        except:
            #   *** need fixing exception messages ***
            msg = "[*] AddressNotFoundError for ip: %s" % (ip)
            logging.error(msg)
    return response
```

```
python GeoIP.py --target 8.8.8.8
```

```
[*] city: Mountain View
[*] region_code: CA
[*] area_code: 650
[*] time_zone: America/Los_Angeles
[*] dma_code: 807
[*] metro_code: San Francisco, CA
[*] country_code3: USA
[*] latitude: 37.3845
[*] postal_code: 94040
[*] longitude: -122.0881
[*] country_code: US
[*] country_name: United States
[*] continent: NA
[*] city: Mountain View
[*] region_code: CA
[*] area_code: 650
```

FootPrinting tools

- **Orb(Python 2.x)**
- **https://github.com/epsylon/orb**
- python-whois - Python module for retrieving WHOIS information
- python-dnspython - DNS toolkit for Python
- python-nmap - Python interface to the Nmap port scanner

- **InstaRecon(Python 2.x)**
- **https://github.com/vergl4s/instarecon**
- dnspython,ipaddress
- ipwhois,python-whois
- requests,shodan

InstaRecon

```
python instarecon.py -v -s <SHODAN_API_KEY> python.org
```

```
# _____________________ Scanning python.org _____________________ #

[-] WARNING: PTR lookup failed for 23.253.135.79 - NXDOMAIN
[*] Domain: python.org

[*] IPs & reverse DNS:
23.253.135.79

[*] NS records:
ns1.p11.dynect.net
        208.78.70.11 - ns1.p11.dynect.net
ns2.p11.dynect.net
        204.13.250.11 - ns2.p11.dynect.net
ns3.p11.dynect.net
        208.78.71.11 - ns3.p11.dynect.net
ns4.p11.dynect.net
        204.13.251.11 - ns4.p11.dynect.net

[*] MX records:
mail.python.org
        188.166.95.178 - mail.python.org
```

InstaRecon

```
status.python.org
        52.35.72.202 - ec2-52-35-72-202.us-west-2.compute.amazonaws.com
        54.149.249.224 - ec2-54-149-249-224.us-west-2.compute.amazonaws.com
        54.148.61.28 - ec2-54-148-61-28.us-west-2.compute.amazonaws.com
        https://status.python.org/
testpypi.python.org
        151.101.12.175
        https://testpypi.python.org/
warehouse.python.org
        151.101.60.175
        https://warehouse.python.org/
        https://warehouse.python.org/project/whitenoise/
wiki.python.org
        140.211.10.69 - virt-y8pzvf.psf.osuosl.org
        https://wiki.python.org/jython
        https://wiki.python.org/jython/JythonDeveloperGuide
        https://wiki.python.org/jython/JythonFaq
        https://wiki.python.org/jython/JythonUsers
        https://wiki.python.org/jython/PackageScanning
        https://wiki.python.org/jython/PoiExample
        https://wiki.python.org/jython/ReadlineSetup
        https://wiki.python.org/jython/RoadMap
        https://wiki.python.org/jython/WhyJython
```

Python modules

- **BeautifulSoup** for parsing web information
- **Requests,urllib3** for synchronous requests
- **Asyncio,aiohttp** for asynchronous requests
- **Robobrowser,Scrapy** for web crawling
- **PyGeoIP,geoip2,geojson** for GeoLocation
- **python-twitter,tweepy** for connecting with twitter
- **Shodan** for obtain information for servers
- **DNSPython,netaddr** for resolving ip address

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#
```

python™ 3

Tinfoleak-pycones

python tinfoleak.py pycones -i -s --sdate 2016/01/01 --hashtags --mentions --meta --media [d] --geo GEOFILE --top 10 -o report.html

TINFOLEAK V1.5

Screen Name: PyConES
Account Created at: 06/20/2012
Followers: 2,190
Following: 1
Tweets: 1,871 (1.21 tweets/day)
Twitter ID: 613492537
URL: http://2016.es.pycon.org
Location: España
Time Zone: Madrid
Geo enabled: False

PyCon España

Twitter oficial de PyCon España. Tweets por el equipo organizador. Contacta con nosotros en contacto2016@es.pycon.org #python #PyConES16 #pycon

Client Apps | Hashtags | User Mentions | Tweets | Metadata | Media | Geolocation

CLIENT APPLICATIONS

Source	Uses	Percentage	First Use	Last Use
Twitter for Android	72	36.0 %	07/13/2016	09/17/2016
TweetDeck	127	63.5 %	07/12/2016	09/16/2016
Twitter Web Client	1	0.5 %	07/14/2016	07/14/2016

Total: 3 results.

TOP HASHTAGS

Date (since)	Date (until)	RT's	FAV's	Count	#Hashtag
07/16/2016	09/25/2016	361	341	59	#PyConES16
09/01/2016	09/21/2016	58	6	16	#affectivePyCon2016
07/28/2016	09/23/2016	64	0	12	#Python
09/12/2016	09/24/2016	92	13	7	#PythonHack
07/18/2016	07/21/2016	53	33	6	#EuroPython
08/13/2016	09/21/2016	22	0	5	#pycones
09/08/2016	09/14/2016	19	0	4	#Django
09/07/2016	09/20/2016	17	0	3	#DjangoGirlsAL
09/01/2016	09/16/2016	13	0	3	#keynote
08/18/2016	09/20/2016	12	0	3	#Almería

Total: 10 results.

Tinfoleak-parameters

```
optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -t TWEETS, --tweets TWEETS
                        number of tweets to be analysed (default: 200)
  -i, --info            general information about a user
  -s, --sources         show the client applications used to publish the
                        tweets
  --sdate SDATE         filter the results for this start date (format:
                        yyyy/mm/dd)
  --edate EDATE         filter the results for this end date (format:
                        yyyy/mm/dd)
  --stime STIME         filter the results for this start time (format:
                        HH:MM:SS)
  --etime ETIME         filter the results for this end date (format:
                        HH:MM:SS)
  --hashtags            show info about hashtags included in tweets
  --mentions            show info about user mentions
  --meta                show metadata information from user images
  --media [D]           [no value]: show user images and videos, [d]: download
                        user images to "username" directory
  --friend FRIENDNAME   show friendship with the FRIENDNAME user
```

Tinfoleak-python dependences

- import tweepy→**Twitter API library for Python**
- from PIL import Image, ExifTags, ImageCms→**metadata from images**
- import pyexiv2→**metadata from images**
- import urllib2→**requests**
- from OpenSSL import SSL
- from jinja2 import Template, Environment, FileSystemLoader→**report**

Tinfoleak-get auth configuration

```
class Configuration():
    """Configuration information"""
    # ------------------------------------------------------------------
    def __init__(self):
        try:
            # Read tinfoleak configuration file ("tinfoleak.conf")
            config = ConfigParser.RawConfigParser()
            config.read('tinfoleak.conf')
            self.color = config.get('colors', 'INFO')
            self.color_hdr = config.get('colors', 'HEADER')
            self.color_fun = config.get('colors', 'FUNCTION')

            CONSUMER_KEY = config.get('Twitter OAuth', 'CONSUMER_KEY')
            CONSUMER_SECRET = config.get('Twitter OAuth', 'CONSUMER_SECRET')
            ACCESS_TOKEN = config.get('Twitter OAuth', 'ACCESS_TOKEN')
            ACCESS_TOKEN_SECRET = config.get('Twitter OAuth', 'ACCESS_TOKEN_SECRET')
```

Tinfoleak-Geolocation

```
# ---------------------------------------------------
def set_geofile_information(self, tweet, user):
    try:
        tweet_geo = 0
        place = ""
        geo = ""

        # Get place from tweet
        if tweet.place:
            place = tweet.place.name.encode('utf-8')

        # Get coordinates from tweet
        if tweet.geo:
            geo = tweet.geo['coordinates']
            tweet_geo = 1
```

Tinfoleak-Geolocation

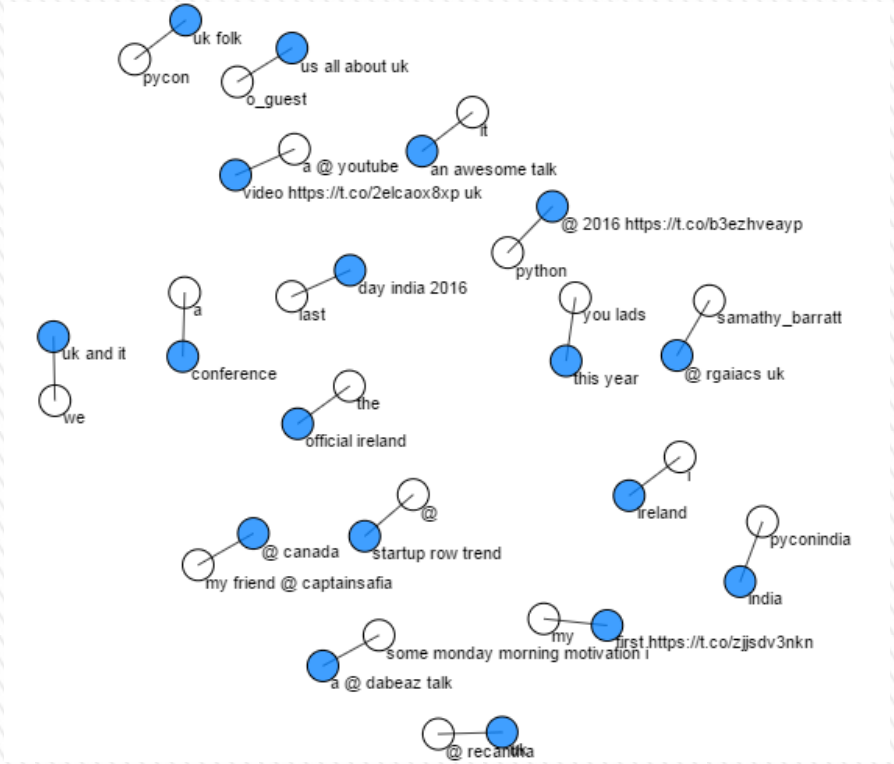
```
if tweet.geo:
    sgeo = tweet.geo['coordinates']
    add = 1
    lat = str(sgeo[0])[:str(sgeo[0]).find(".")+4]
    lon = str(sgeo[1])[:str(sgeo[1]).find(".")+4]
    location = "[" + lat + ", " + lon + "]"
    for i in range (1, 20-len(location)):
        location += " "
    location = location + "\t" + splace
```

OSINT Tweepy

```
class StreamListener(tweepy.StreamListener):
    def on_data(self, data):
        data = json.loads(data)
        try:
            #get data with no geolocation
            if data['geo'] is None:
                print('[>] {0}(@{1})\n{2}'.format(
                    data['user']['name'].encode('ascii', 'ignore'),
                    data['user']['screen_name'].encode('ascii', 'ignore'),
                    data['text'].encode('ascii', 'ignore')
                    )
                )
            #get data with geolocation
            if data['geo'] is not None:
                print('[>] {0}(@{1}) -- LOCATION/LATITUDE/LONGITUDE: {2} / {3}'.fo
                    data['user']['name'].encode('ascii', 'ignore'),
                    data['user']['screen_name'].encode('ascii', 'ignore'),
                    data['user']['location'].encode('ascii', 'ignore'),
                    data['geo']['coordinates']
                    )
                )

            return True

        except Exception, e:
            print('[!] Error: {0}'.format(e))
```

Pattern web & graph

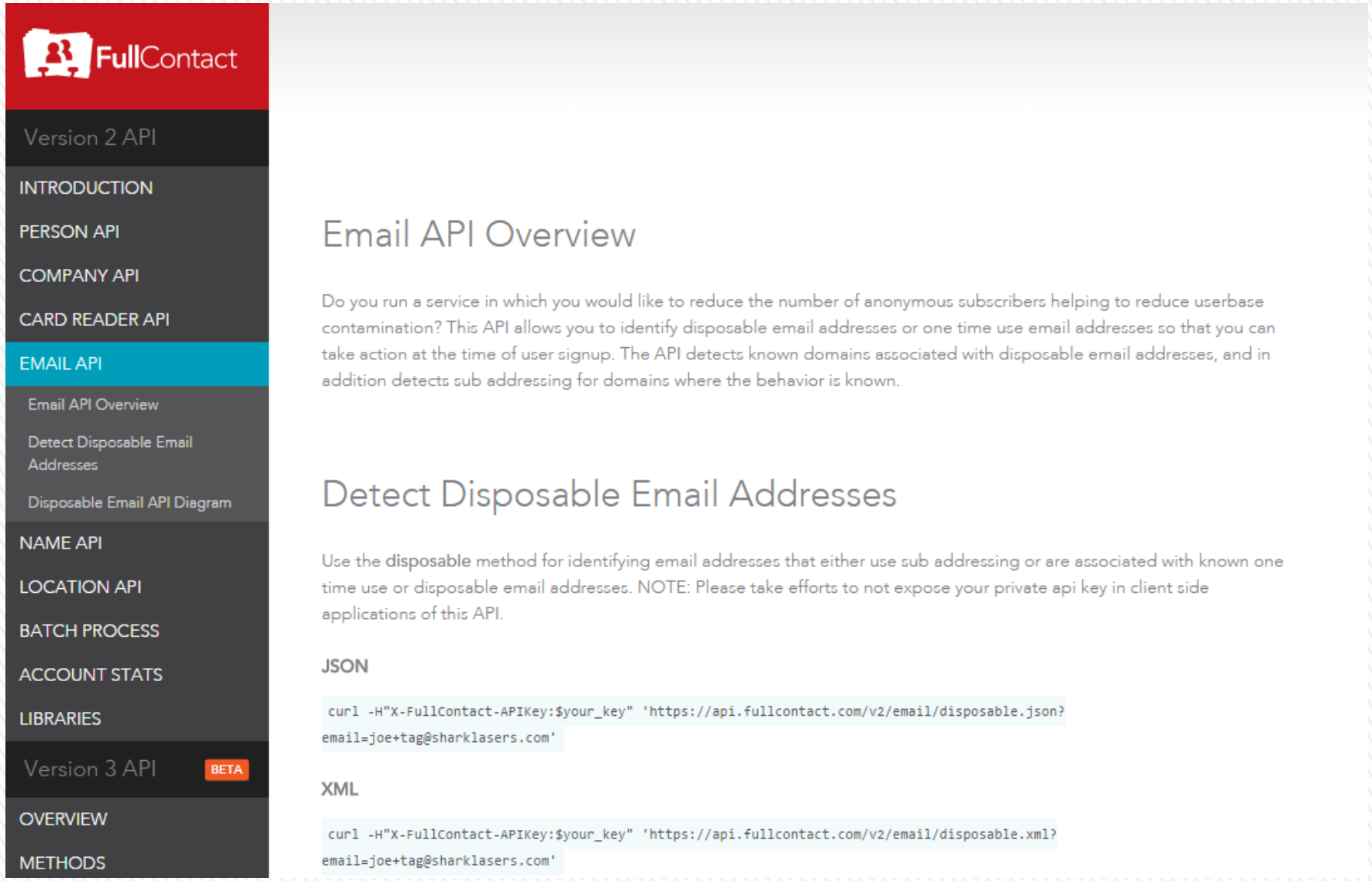
```
from pattern.web import Twitter, plaintext
from pattern.en import parsetree
from pattern.graph  import Graph
from pattern.search import search
import argparse

def get_pattern_data(search_param):

    twitter = Twitter(language='en')

    for tweet in twitter.search(search_param, cached=True):
        print(plaintext(tweet.text).encode('ascii', 'ignore').decode('utf-8'))


    g = Graph()
    for i in range(10):
        for result in twitter.search(search_param, start=i+1,count=10):
            s = result.text.lower()
            s = plaintext(s)
            s = parsetree(s)
            p = '{NP} (VP) ' +search_param+ ' {NP}'
            for m in search(p, s):
                x = m.group(1).string # NP left
                y = m.group(2).string # NP right
                if x not in g:
                    g.add_node(x)
                    if y not in g:
                        g.add_node(y)
                    g.add_edge(g[x], g[y], stroke=(0,0,0,0.75)) # R,G,B,A
```



FullContact API

- We know we have a valid email address
- What other profiles are associated with this address?
- Go to fullcontact.com for an API key…..

FullContact API



FullContact API

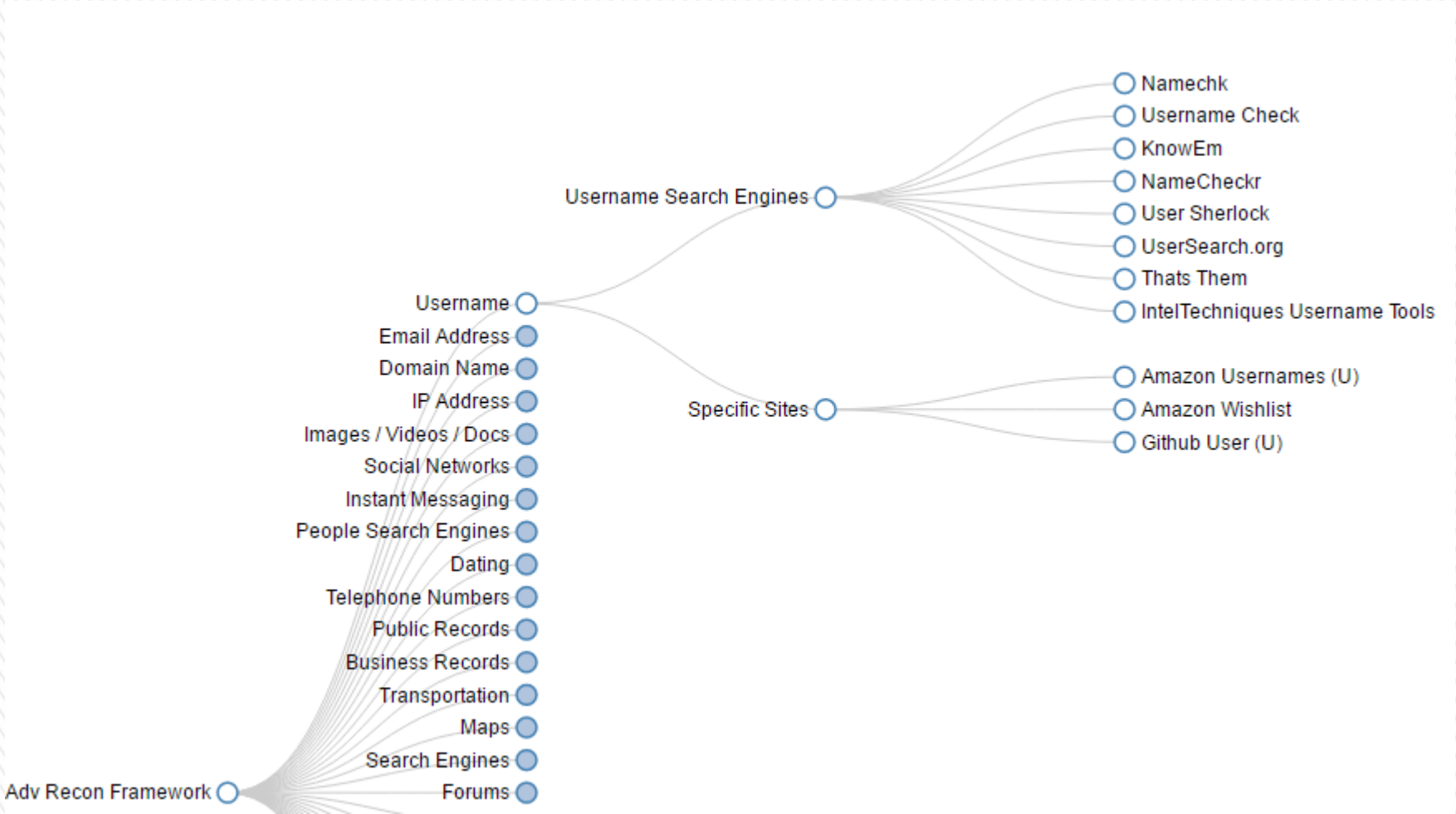
```
def get_fullcontact(email):
        api_key = 'a82ad9009f6b1b1'
        base_url = 'https://api.fullcontact.com/v2/person.json'
        payload = {'email':email, 'apiKey':api_key}
        resp = requests.get(base_url, params=payload)
        if resp.status_code == 200:
                # parse contact information
                if 'contactInfo' in resp.json():
```

FullContact API

```
python checkFullContactAPI.py -e gvanrossum@gmail.com
```

```
Guido van Rossum - gvanrossum@gmail.com
Staff Engineer at Dropbox
Employee at Google
Employee at Elemental Security
Employee at Zope Corporation
Employee at CNRI
Employee at BeOpen.com
Employee at CWI
Employee at SARA
San Francisco Bay Area
Github - https://github.com/gvanrossum
Twitter - https://twitter.com/gvanrossum
Quora - http://www.quora.com/Guido-van-Rossum
Gravatar - https://gravatar.com/gvanrossum
Flickr - https://www.flickr.com/people/gvanrossum
Yelp - http://gvanrossum.yelp.com
Blogger - http://blogger.com/profile/12821714508588242516
GooglePlus - https://plus.google.com/u/0/115212051037621986145
Klout - http://klout.com/gvanrossum
LinkedIn - https://www.linkedin.com/pub/guido-van-rossum/0/756/4a0
Confidence: 88%
```

http://osintframework.com/



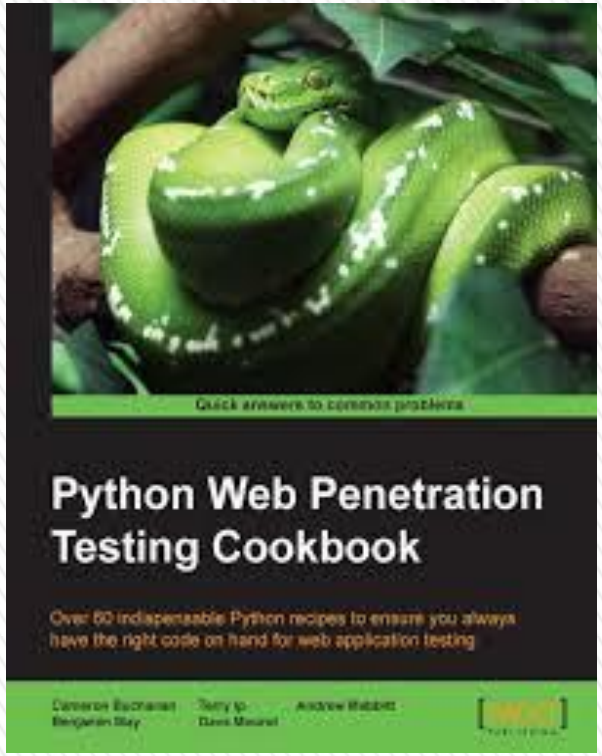
Kali Linux

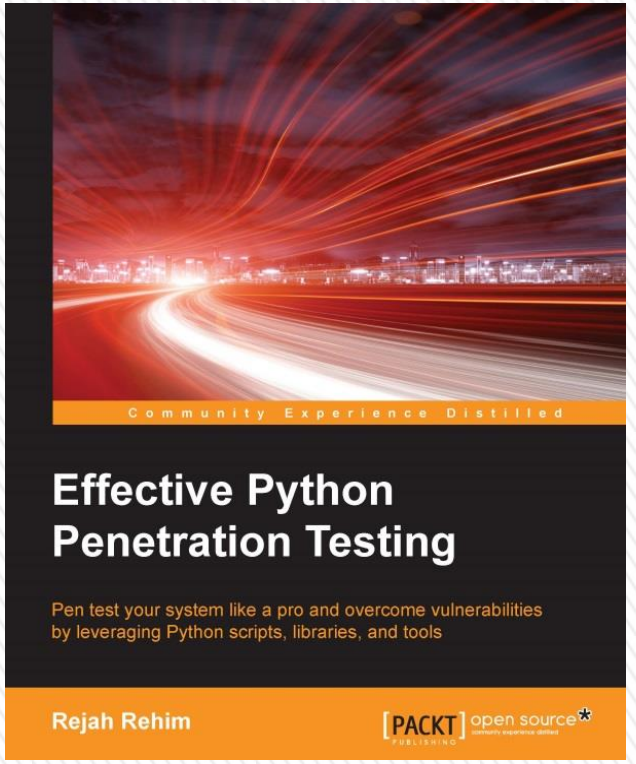

Aplicaciones
Lugares
vie 15:58
1
es
Escribir para buscar...
01-Information
02-Vulnerability
03-Web Apps
04-Database
05-Password
06-Wireless
intrace
irpas-ass
irpass-cdp
lbd
maltego
miranda
nbtscan
ncat
netdiscover
netmask
nmap
onesixtyone
p0f
recon-ng
smbmap
smtp-user-enum
snmp-check
sparta
Frecuentes
Todas

References

- https://sourceforge.net/projects/spiderfoot
- http://www.edge-security.com/theharvester.php
- https://developer.shodan.io/api
- http://www.clips.ua.ac.be/pattern
- http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines#OSINT
- http://www.vicenteaguileradiaz.com/tools
- https://github.com/automatingosint/osint_public
- http://www.automatingosint.com/blog/

Books


Python Web Penetration Testing Cookbook
Over 60 indispensable Python recipes to ensure you always have the right code on hand for web application testing


Community Experience Distilled
Effective Python Penetration Testing
Pen test your system like a pro and overcome vulnerabilities by leveraging Python scripts, libraries, and tools
Rejah Rehim
[PACKT] open source

Thanks!

@jmortegac