
# Hands-on Bug Hunting

## Active and Passive Reconnaissance and Dark Web Research

DAY 1 LAB GUIDE

https://darknetrecon.com

Instructors:
Omar Santos (@santosomar)
Joseph Mlodzianowski (@cedoxx)

# Introduction

This guide is a collection of exercises for the O'Reilly Live Training Hands-on Bug Hunting: Active and Passive Reconnaissance and Dark Web Research" authored and delivered by Omar Santos and Joseph Mlodzianowski.
For more information about the training visit https://darknetrecon.com

## Learning Path: Additional Cybersecurity Training (Free with your O'Reilly Subscription)

This training is part of a learning path of numerous live training sessions and video courses that are available with your O'Reilly subscription. To access the learning path go to: https://h4cker.org/learning-path

## Training Summary

This live and interactive training is designed to help you perform passive and active reconnaissance in ethical hacking and bug bounty hunting engagements. You will learn intermediate-to-advanced recon methodologies using open source intelligence (OSINT). In this training you will also learn how to perform dark web research and reconnaissance. You will learn how to use Tor, proxies and proxychains, and even how to create your own VPN servers in cloud environments.

## Helpful Resources Prior to Taking the Live Training:

- Security Penetration Testing The Art of Hacking Series LiveLessons (video)
- Wireless Networks, IoT, and Mobile Devices Hacking (video)
- Enterprise Penetration Testing and Continuous Monitoring (video)
- Hacking Web Applications The Art of Hacking Series LiveLessons: Security Penetration Testing for Today's DevOps and Cloud EnvironmentsWeb (video)
- Security Fundamentals (video)

# Lab Setup

Setup WebSploit Labs (https://websploit.org).
WebSploit Labs is a learning environment created by Omar Santos for different Cybersecurity Ethical Hacking (Web Penetration Testing) training sessions. WebSploit includes several intentionally vulnerable applications running in Docker containers on top of Kali Linux or Parrot Security OS, several additional tools, and over 8,000 cybersecurity resources. WebSploit comes with over 400 distinct exercises!

- Step 1: Download Kali or Parrot and install it on a VM
- Step 2: After you have installed Kali Linux, run the following command from a terminal window to setup your environment:

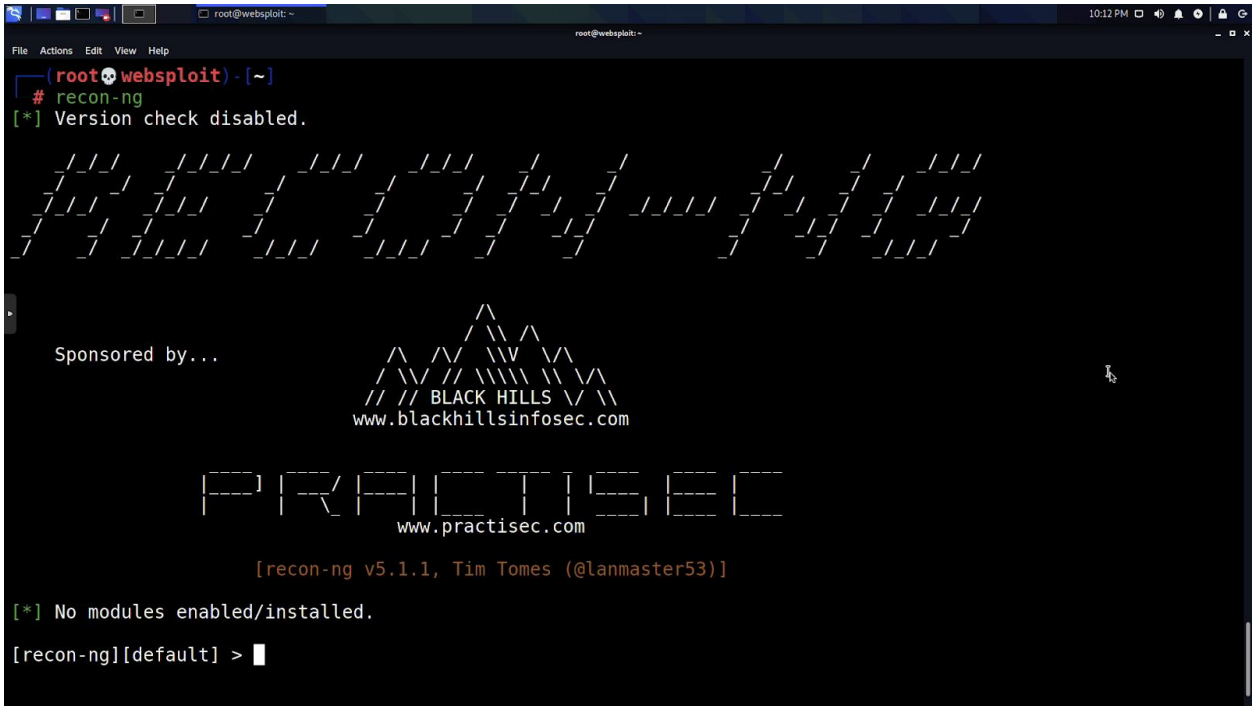
```
curl -sSL https://websploit.org/install.sh | sudo bash
```

This command will install all the tools, Docker, the intentionally vulnerable containers, and numerous cybersecurity resources.

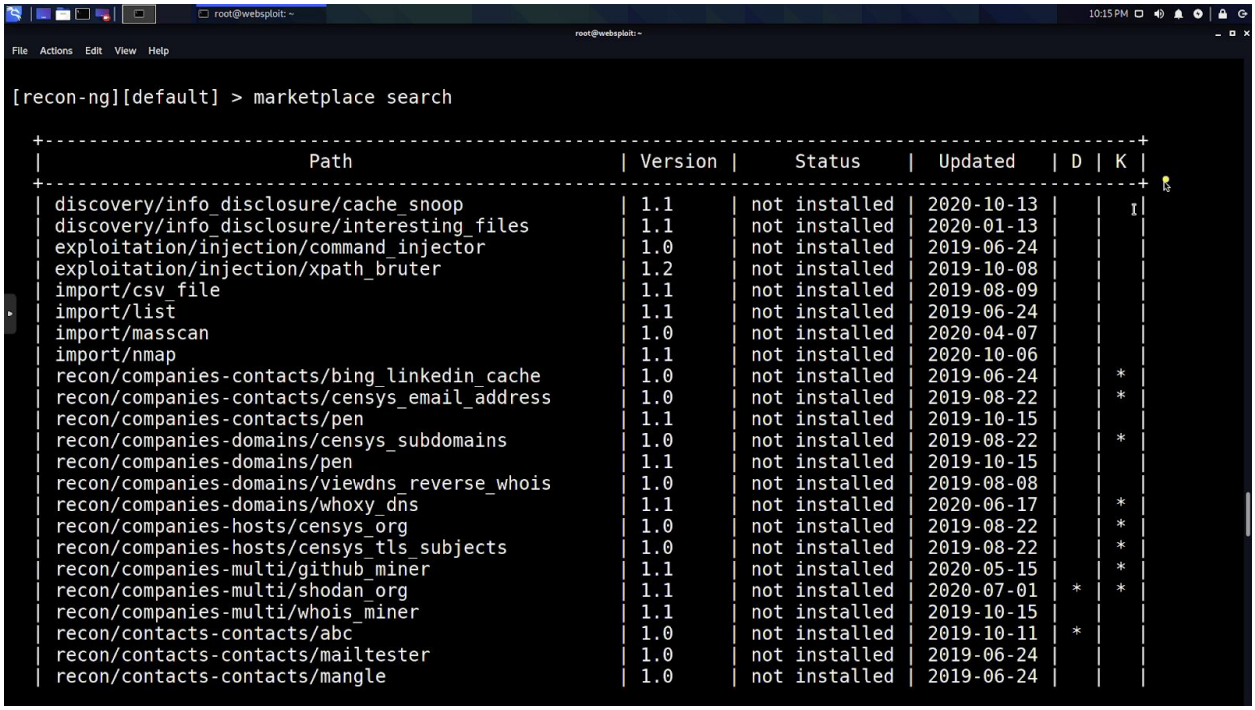
A quick demo can be accessed at: https://websploit.org/install.gif or at websploit.org main website (landing page).

# Exercise 1: Recon-NG

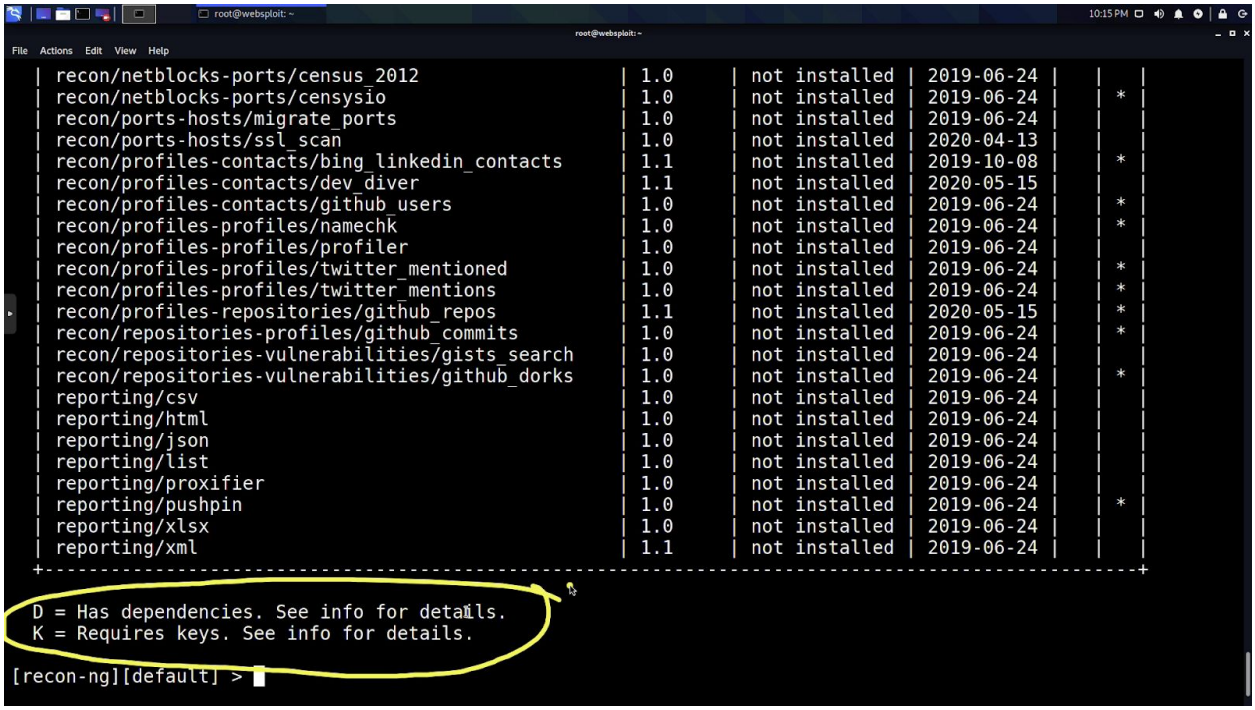
1. Start **Recon-NG** with by just typing **recon-ng** in a terminal Window:



2. Recon-NG has numerous modules that can be installed and activated from the “market place”. You can search all the modules by using the “**marketplace search**” command, as shown below:




The **D** and the **K** in the last two columns of the table shown above indicate that the module has dependencies or that it requires an **API key**. The screenshot below shows the legend:



3. You can search the market place by using keywords. In the example below, we are searching for modules related to "**whois**".

```
[recon-ng][default] > marketplace search whois
[*] Searching module index for 'whois'...

  +-------------------------------------------------------------------------------------------+
  |                      Path                     | Version |     Status    |  Updated   | D | K |
  +-------------------------------------------------------------------------------------------+
  | recon/companies-domains/viewdns_reverse_whois | 1.0     | not installed | 2019-08-08 |   |   |
  | recon/companies-multi/whois_miner             | 1.1     | not installed | 2019-10-15 |   |   |
  | recon/domains-companies/whoxy_whois           | 1.1     | not installed | 2020-06-24 |   | * |
  | recon/domains-contacts/whois_pocs             | 1.0     | not installed | 2019-06-24 |   |   |
  | recon/netblocks-companies/whois_orgs          | 1.0     | not installed | 2019-06-24 |   |   |
  +-------------------------------------------------------------------------------------------+

  D = Has dependencies. See info for details.
  K = Requires keys. See info for details.

[recon-ng][default] >
```

The following is an example of modules related to the "**dns**" keyword. However, there are many other modules that can be used to perform DNS recon which are not listed below. This is because the "**marketplace search**" command is just using a keyword.

```
[recon-ng][default] > marketplace search dns
[*] Searching module index for 'dns'...

  +-------------------------------------------------------------------------------------------+
  |                      Path                     | Version |     Status    |  Updated   | D | K |
  +-------------------------------------------------------------------------------------------+
  | recon/companies-domains/viewdns_reverse_whois | 1.0     | not installed | 2019-08-08 |   |   |
  | recon/companies-domains/whoxy_dns             | 1.1     | not installed | 2020-06-17 |   | * |
  +-------------------------------------------------------------------------------------------+

  D = Has dependencies. See info for details.
  K = Requires keys. See info for details.
```

For instance, the Netcraft module is used to search domains and subdomains using DNS records in the Netcraft database.

```
[recon-ng][default] > marketplace search netcraft
[*] Searching module index for 'netcraft'...

  +-------------------------------------------------------------------------------+
  |             Path             | Version |     Status    |  Updated   | D | K |
  +-------------------------------------------------------------------------------+
  | recon/domains-hosts/netcraft | 1.1     | not installed | 2020-02-05 |   |   |
  +-------------------------------------------------------------------------------+

  D = Has dependencies. See info for details.
  K = Requires keys. See info for details.
```

4. Install the Netcraft module by using the following command:

```
[recon-ng][default] > marketplace install recon/domains-hosts/netcraft
```

5. You should be able to search for the installed module and “load it” to be able to run and use it to query its database, as demonstrated below:

```
[recon-ng][default] > modules search netcraft
[*] Searching installed modules for 'netcraft'...

  Recon
  -----
    recon/domains-hosts/netcraft

[recon-ng][default] > modules load recon/domains-hosts/netcraft
[recon-ng][default][netcraft] > info

      Name: Netcraft Hostname Enumerator
    Author: thrapt (thrapt@gmail.com)
   Version: 1.1

Description:
  Harvests hosts from Netcraft.com. Updates the 'hosts' table with the results.

Options:
  Name    Current Value  Required  Description
  ------  -------------  --------  -----------
  SOURCE  default        yes       source of input (see 'info' for details)

Source Options:
  default        SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

[recon-ng][default][netcraft] >
```

6. Set the SOURCE to any domain you would like to identify subdomains. H4cker.org is used in the following example. However, be creative and use any other domain you would like to get subdomains and additional information.

   ***NOTE**: You are NOT hacking anyone here, the tool is just using public DNS records to perform these actions.*

```
[recon-ng][default][netcraft] > options set SOURCE h4cker.org
SOURCE => h4cker.org
[recon-ng][default][netcraft] > run

----------
H4CKER.ORG
----------
[*] URL: http://searchdns.netcraft.com/?restriction=site%2Bends%2Bwith&host=h4cker.org
[*] Country: None
[*] Host: bootcamp.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ------------------------------------------------

-------
SUMMARY
-------
[*] 1 total (1 new) hosts found.
[recon-ng][default][netcraft] >
```

7. Install the **bing_domain_web** module, as shown below:

```
[recon-ng][default] > marketplace install recon/domains-hosts/bing_domain_web
[*] Module installed: recon/domains-hosts/bing_domain_web
[*] Reloading modules...
[recon-ng][default] >
```

8. Load the module and show the options:

```
[recon-ng][default] > modules load recon/domains-hosts/bing_domain_web
[recon-ng][default][bing_domain_web] > info

      Name: Bing Hostname Enumerator
    Author: Tim Tomes (@lanmaster53)
   Version: 1.1

Description:
  Harvests hosts from Bing.com by using the 'site' search operator. Updates the 'hosts' table with the
  results.

Options:
  Name    Current Value  Required  Description
  ------  -------------  --------  -----------
  SOURCE  default        yes       source of input (see 'info' for details)

Source Options:
  default        SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

[recon-ng][default][bing_domain_web] >
```

9. Set the SOURCE to the domain(s) you entered before (when you were running the Netcraft module).

```
[recon-ng][default][bing_domain_web] > options set SOURCE h4cker.org
SOURCE => h4cker.org
[recon-ng][default][bing_domain_web] > run
```

10. Did you find more information and subdomains like I did below?

```
H4CKER.ORG
----------
[*] URL: https://www.bing.com/search?first=0&q=domain%3Ah4cker.org
[*] Country: None
[*] Host: lpb.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ------------------------------------------------
[*] Country: None
[*] Host: webapps.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ------------------------------------------------
[*] Country: None
[*] Host: bootcamp.h4cker.org
[*] Ip_Address: None
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ------------------------------------------------
[*] Sleeping to avoid lockout...
```

11. Install and load the brute_hosts module:

```
[recon-ng][default] > marketplace install recon/domains-hosts/brute_hosts
[*] Module installed: recon/domains-hosts/brute_hosts
[*] Reloading modules...
[recon-ng][default] > modules load recon/domains-hosts/brute_hosts
```

12. This module uses wordlists. You can use any wordlist of your choosing. WebSploit comes with dozens of wordlists (the ones that come with Kali/Parrot and several under

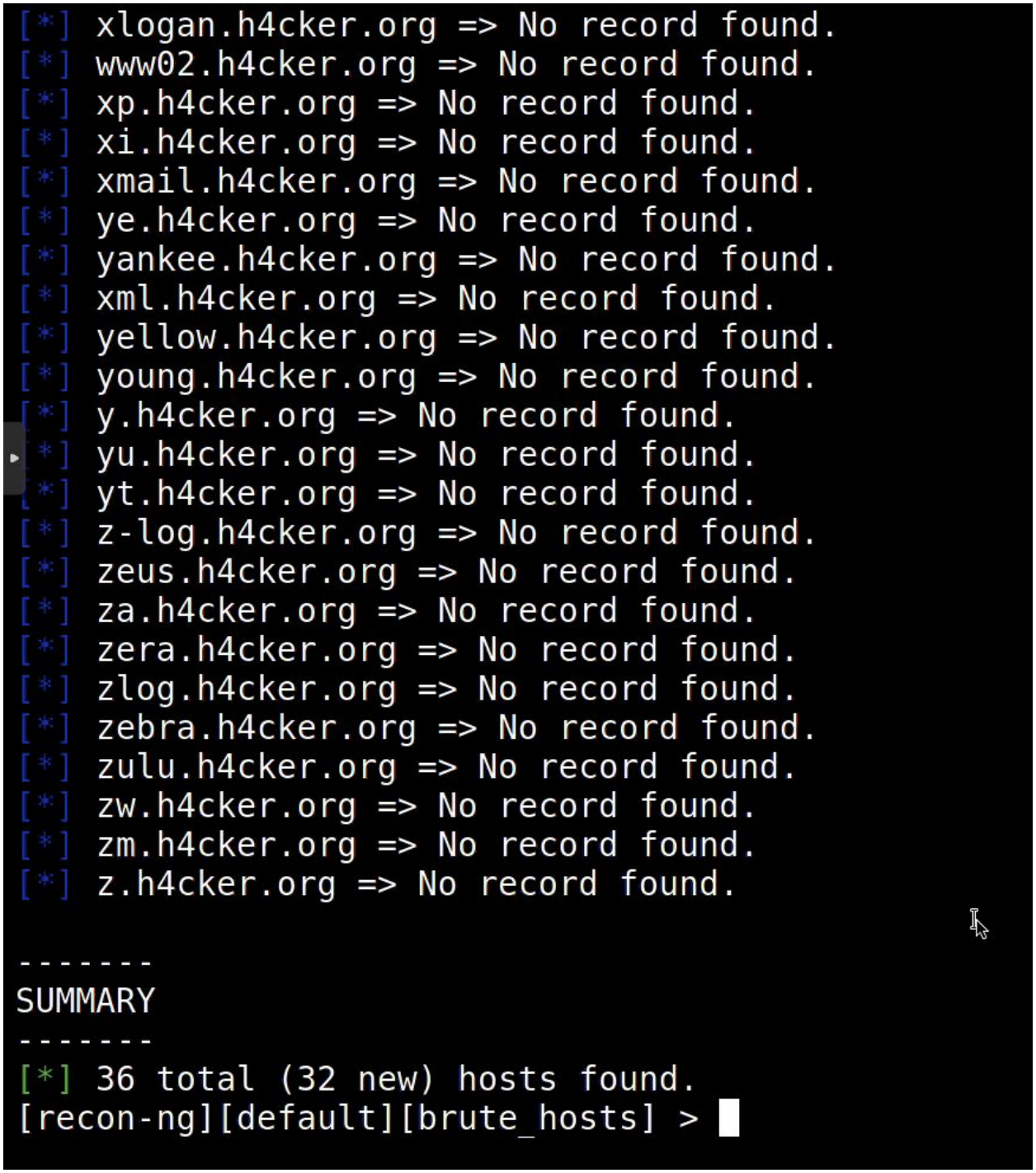
**/root/SecLists** directory. In the exa

```
[recon-ng][default] > modules load recon/domains-hosts/brute_hosts
[recon-ng][default][brute_hosts] > info

      Name: DNS Hostname Brute Forcer
    Author: Tim Tomes (@lanmaster53)
   Version: 1.0

Description:
  Brute forces host names using DNS. Updates the 'hosts' table with the results.

Options:
  Name      Current Value                     Required  Description
  --------  -------------                     --------  -----------
  SOURCE    default                           yes       source of input (see 'info' for details)
  WORDLIST  /root/.recon-ng/data/hostnames.txt  yes       path to hostname wordlist

Source Options:
  default        SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

[recon-ng][default][brute_hosts] >
```

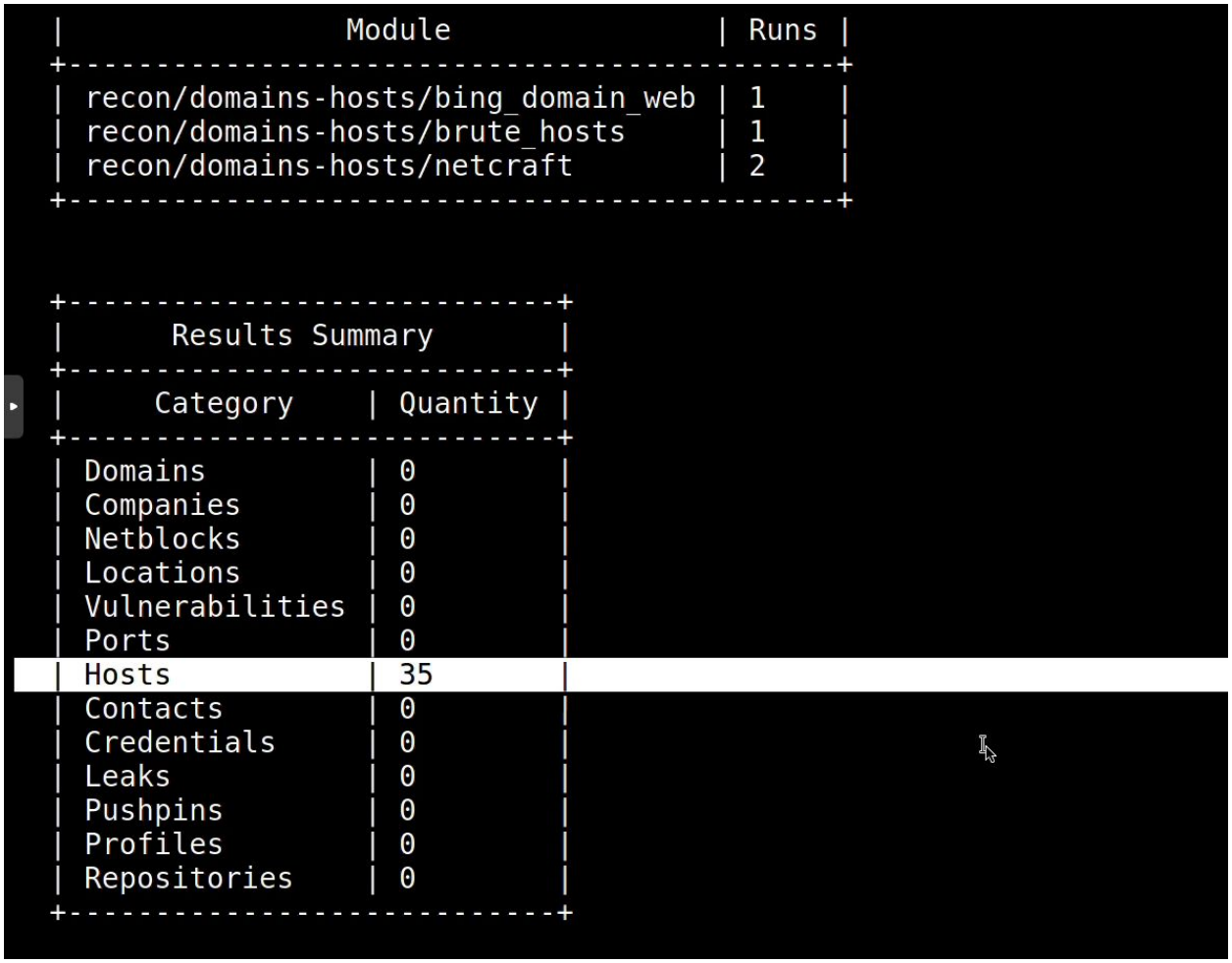
13. Set the SOURCE to the target domain (h4cker.org is used in the following example:

```
[recon-ng][default][brute_hosts] > options set SOURCE h4cker.org
```

14. You should be able to see the tool going through the wordlist to enumerate additional hosts. Look at the summary in the example below:



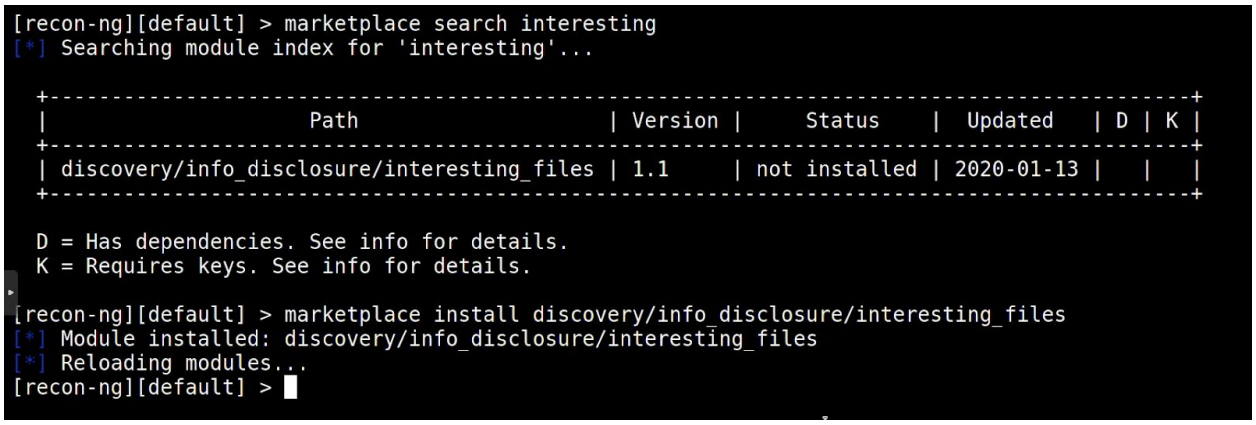
15. Enter the **dashboard** command to obtain a summary of all the findings (all modules), as demonstrated below:



```
|                Module                | Runs |
+--------------------------------------+
| recon/domains-hosts/bing_domain_web | 1    |
| recon/domains-hosts/brute_hosts     | 1    |
| recon/domains-hosts/netcraft        | 2    |
+--------------------------------------+


+---------------------------+
|      Results Summary      |
+---------------------------+
|     Category    | Quantity |
+---------------------------+
| Domains         | 0        |
| Companies       | 0        |
| Netblocks       | 0        |
| Locations       | 0        |
| Vulnerabilities | 0        |
| Ports           | 0        |
| Hosts           | 35       |
| Contacts        | 0        |
| Credentials     | 0        |
| Leaks           | 0        |
| Pushpins        | 0        |
| Profiles        | 0        |
| Repositories    | 0        |
+---------------------------+
```

16. Use the **show hosts** command to list all the enumerated hosts and respective information:


17. Now let's look for interesting files. Search the marketplace for the word "interesting" and you will see the "interesting_files" module. Install it as demonstrated below:




18. Load the module:

19. Type **info** to show all the options. The following are the options after you entered the **info** command:

```
    Author: Tim Tomes (@lanmaster53), thrapt (thrapt@gmail.com), Jay Turla (@shipcod3), and Mark Jeffery
   Version: 1.1

Description:
  Checks hosts for interesting files in predictable locations.

Options:
  Name      Current Value  Required  Description
  --------  -------------  --------  -----------
  DOWNLOAD  True           yes       download discovered files
  PORT      80             yes       request port
  PROTOCOL  http           yes       request protocol
  SOURCE    default        yes       source of input (see 'info' for details)

Source Options:
  default        SELECT DISTINCT host FROM hosts WHERE host IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

Comments:
  * Files: robots.txt, sitemap.xml, sitemap.xml.gz, crossdomain.xml, phpinfo.php, test.php, elmah.axd,
  server-status, jmx-console/, admin-console/, web-console/
  * Google Dorks:
    - inurl:robots.txt ext:txt
    - inurl:elmah.axd ext:axd intitle:"Error log for"
    - inurl:server-status "Apache Status"

[recon-ng][default][interesting_files] >
```

20. Set the PORT, SOURCE, and PROTOCOL:

```
[recon-ng][default][interesting_files] > options set PORT 443
PORT => 443
[recon-ng][default][interesting_files] > options set SOURCE h4cker.org
SOURCE => h4cker.org
[recon-ng][default][interesting_files] > options set PROTOCOL https
PROTOCOL => https
[recon-ng][default][interesting_files] >
```

21. Run the module. What information were you able to see?

```
[recon-ng][default][interesting_files] > run
[*] https://h4cker.org:443/robots.txt => 200. 'robots.txt' found but unverified.
[*] https://h4cker.org:443/sitemap.xml => 404
[*] https://h4cker.org:443/sitemap.xml.gz => 404
[*] https://h4cker.org:443/crossdomain.xml => 404
[*] https://h4cker.org:443/phpinfo.php => 200. 'phpinfo.php' found but unverified.
[*] https://h4cker.org:443/test.php => 200. 'test.php' found but unverified.
[*] https://h4cker.org:443/elmah.axd => 404
[*] https://h4cker.org:443/server-status => 404
[*] https://h4cker.org:443/jmx-console/ => 404
[*] https://h4cker.org:443/admin-console/ => 200. 'admin-console/' found but unverified.
[*] https://h4cker.org:443/web-console/ => 404
[*] 0 interesting files found.
[recon-ng][default][interesting_files] >
```

# Exercise 2: SpiderFoot

Description from SpiderFoot's GitHub Repo: "SpiderFoot is an open source intelligence (OSINT) automation tool. It integrates with just about every data source available and utilises a range of methods for data analysis, making that data easy to navigate.
SpiderFoot has an embedded web-server for providing a clean and intuitive web-based interface but can also be used completely via the command-line. It's written in Python 3 and GPL-licensed."

- Download SpiderFoot by cloning the GitHub repository at: https://github.com/smicallef/spiderfoot

```
git clone https://github.com/smicallef/spiderfoot.git
```

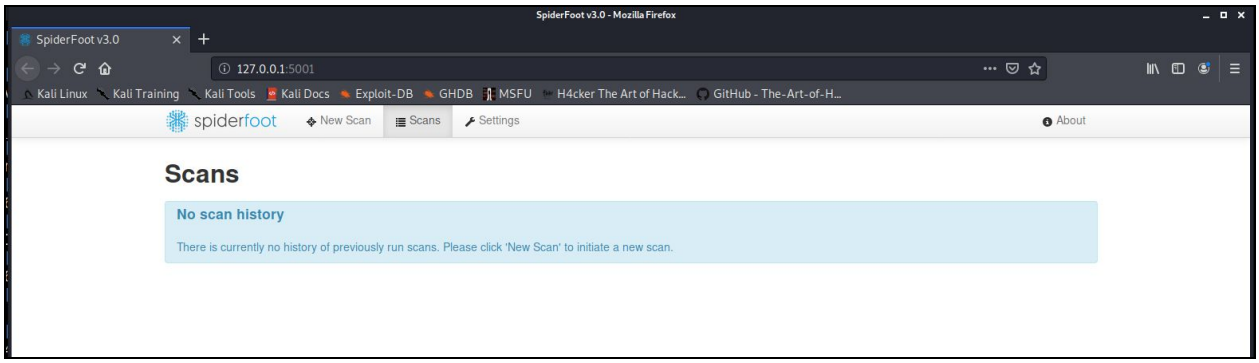
- Then make sure that all the necessary Python 3 modules are installed by using the command below and demonstrated in the following figures:

```
# python3 -m pip3 install  -r requirements.txt
```

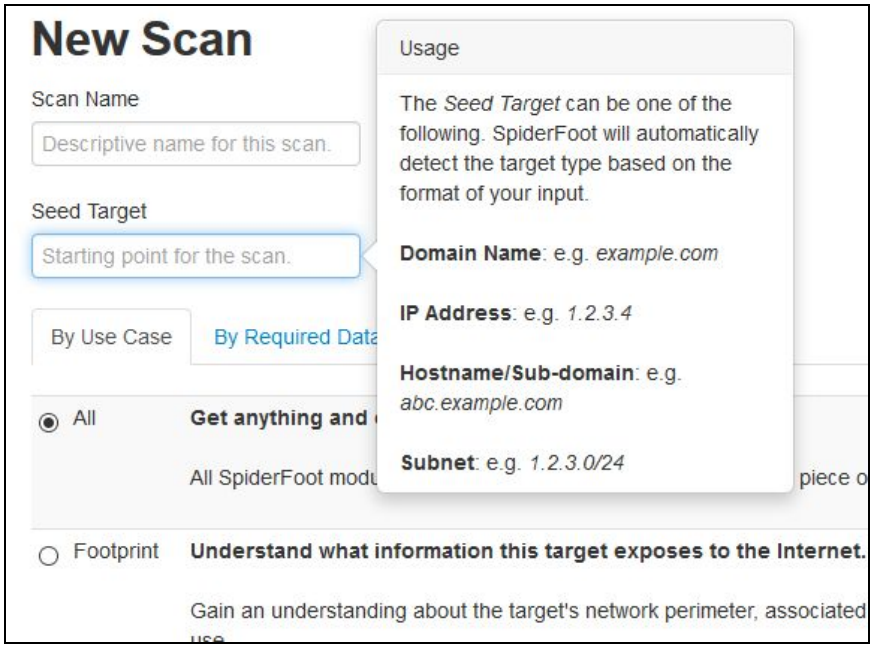
- Start SpiderFoot by using the python3 sf.py -l 127.0.0.1:5001 command as shown below:

```
root@websploit:~/spiderfoot# python3 sf.py -l 127.0.0.1:5001
Starting web server at http://127.0.0.1:5001 ...


*************************************************************
 Use SpiderFoot by starting your web browser of choice and
 browse to http://127.0.0.1:5001
*************************************************************
```

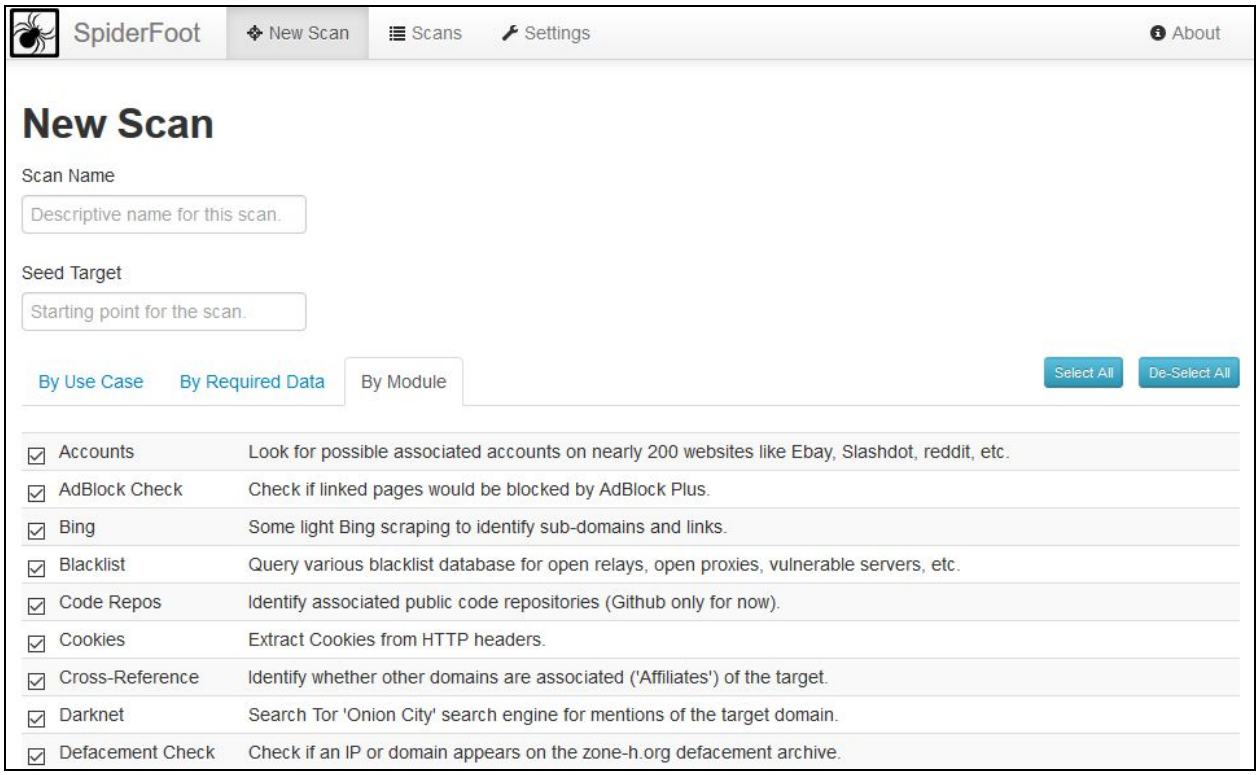
- Open the browser and go to http://127.0.0.1:5001 to access the SpiderFoot GUI.
- When you run SpiderFoot for the first time, there is no historical data, so you should be presented with a screen like the following:



- Click on the 'New Scan' button in the top menu bar. You will then need to define a name for your scan (these are non-unique) and a target (also non-unique):




- You can then define how you would like to run the scan - either by use case (the tab selected by default), by data required or by module.
- Module-based scanning is for more advanced users who are familiar with the behavior and data provided by different modules, and want more control over the scan:



Tip: If you select a module that depends on event types only provided by other modules, but those modules are not selected, you will get no results.

- Run your scan. Choose any arbitrary victim (your own website, your own name, etc.)

---

# Exercise3: Sublist3r

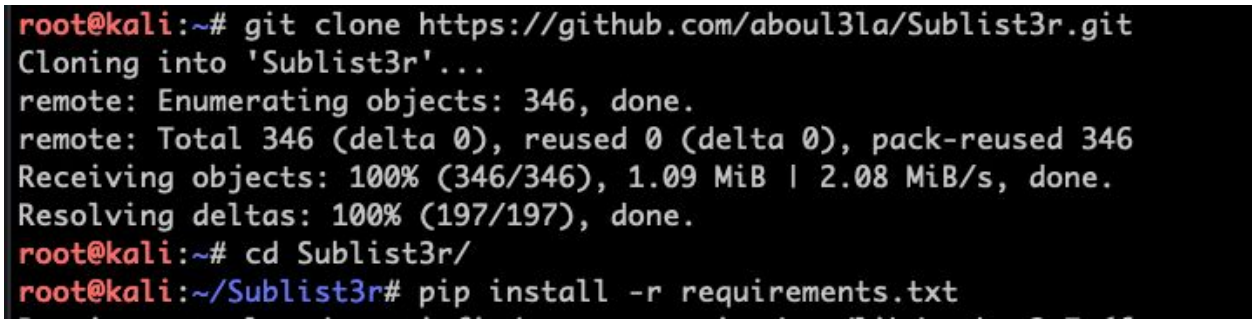
As it reads in their GitHub repository “Sublist3r is a python tool designed to enumerate subdomains of websites using OSINT. It helps penetration testers and bug hunters collect and gather subdomains for the domain they are targeting. Sublist3r enumerates subdomains using many search engines such as Google, Yahoo, Bing, Baidu, and Ask. Sublist3r also enumerates subdomains using Netcraft, Virustotal, ThreatCrowd, DNSdumpster, and ReverseDNS. subbrute was integrated with Sublist3r to increase the possibility of finding more subdomains using bruteforce with an improved wordlist. The credit goes to TheRook who is the author of subbrute.”

1. Download Sublist3r:

```
# git clone https://github.com/aboul3la/Sublist3r.git
```

2. Install the dependencies/requirements by using pip:

```
# cd Sublist3r
# pip3 install -r requirements.txt
```




3. Run Sublist3r to your own domain; or against h4cker.org or theartofhacking.com.

```
# python3 sublist3r.py -v -d your-domain.com -b -p 80,443
```

The following example is against the h4cker.org domain.

```
┌──(root💀websploit)-[~/Sublist3r]
└─# python3 sublist3r.py -v -d h4cker.org -b -p 80,443

                 ____        _     _ _     _   _____
                / ___| _   _| |__ | (_)___| |_|___ / _ __
                \___ \| | | | '_ \| | / __| __| |_ \| '__|
                 ___) | |_| | |_) | | \__ \ |_ ___) | |
                |____/ \__,_|_.__/|_|_|___/\__|____/|_|

                # Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for h4cker.org
[-] verbosity is enabled, will show the subdomains results in realtime
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[!] Error: Virustotal probably now is blocking our requests
Bing: bootcamp.h4cker.org
Bing: webapps.h4cker.org
Bing: websploit.h4cker.org
Bing: lpb.h4cker.org
Yahoo: lpb.h4cker.org
Yahoo: webapps.h4cker.org
Yahoo: bootcamp.h4cker.org
```

# Exercise 4: Maltego

This is an optional exercise. Omar will provide a demo and walkthrough of Maltego. Maltego requires you to register a free account at https://www.maltego.com/ce-registration/ in order to use the Maltego Community Edition. You will not be able to complete this exercise if you do not register. This is why this exercise is optional.

Maltego comes installed by default in Kali and Parrot. However, you can install it in any Windows, macOS, or Linux system. The following are the hardware requirements:

## Minimum:

- 4GB RAM,
- Intel i3

- 10Mbps Internet access speed
- 720p display

### Recommended:

- 16GB RAM
- Intel i7
- 20Mbps or more Internet access speed
- 1080p display

When working with large graphs, it's best to have as much memory and CPU as possible. Adding new results to a large graph, as well as calculating new graph layouts require a lot of computing power.

## Maltego Concepts

There are a few important concepts in Maltego that need to be understood before using the tool. The following definitions are directly from their documentation:

### Maltego Desktop Client / Desktop Client / Client

References to Maltego software used on the desktop is referred to as the Desktop Client, the Client or the Maltego Desktop Client. The Desktop Client editions available are Community, Classic, XL and One.

## Entity

An Entity is a piece of information shown as a node on the graph. Different Entity types are used to differentiate between the different pieces of information that can be represented in Maltego.

Entities can be anything from a DNS name, Person name, Phone number, etc. The Maltego Client comes with about 20 Entities targeted for use in online investigations, however, you can create your own custom Entities.

## Transform

A Transform is a piece of code that searches for information related to an Entity on the graph. Transforms allow you to query an API or database to show related info on the graph.

The idea is that we are "transforming" one type of information into another type. For example we could have the website "www.maltego.com" and transform it into the IP address "104.248.60.43".

By default Maltego has Transforms that can query information from data sources like DNS servers, search engines, social networks, WHOIS information, etc.

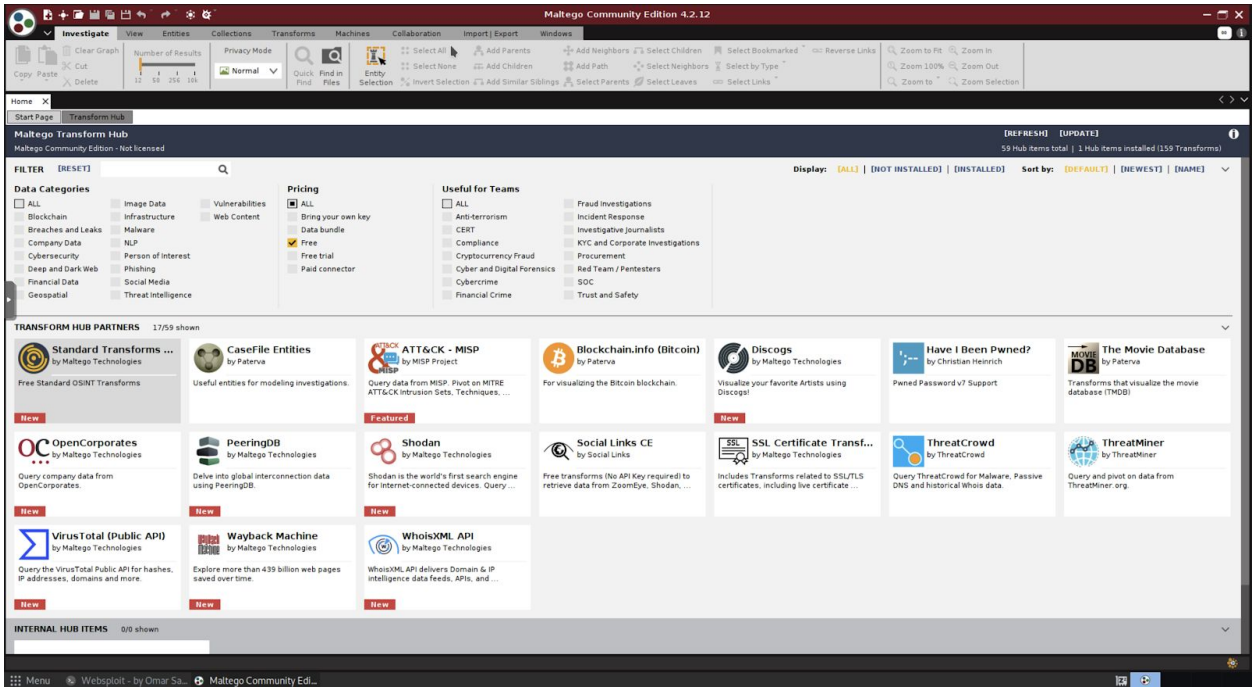
# Machine

Machines are the Maltego equivalent of macros. Machines allow you to chain together multiple Transforms, filters and actions in order to automate common and tedious tasks.

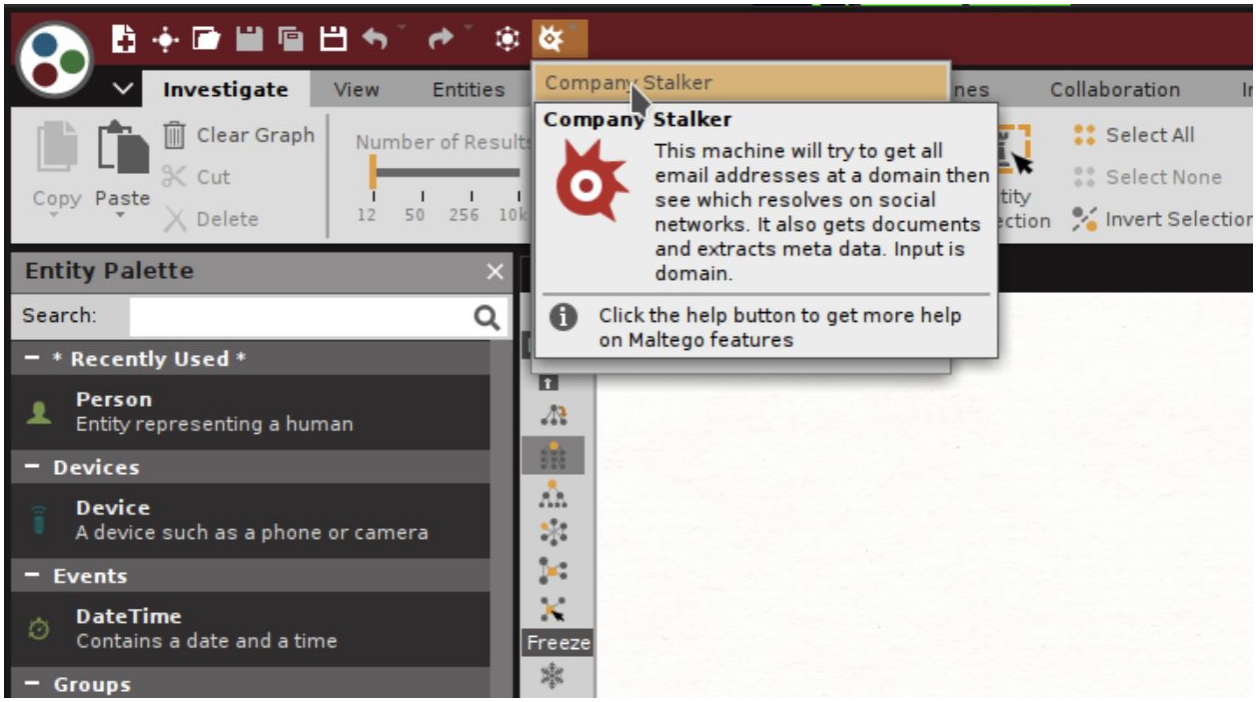
# Hub Item

Transforms and the Entity types that they query need to be stored on a server that can be accessed by the Maltego Client.
Hub items allow Maltego users to install combinations of Transforms, Entities and Machines from a server. By default, Maltego installs the Hub item called Standard Transforms which contains the Transforms, Entities and Machines that are developed and maintained by the developers of Maltego.
Additional Hub items can be installed to get 3rd party functionality built by the community.

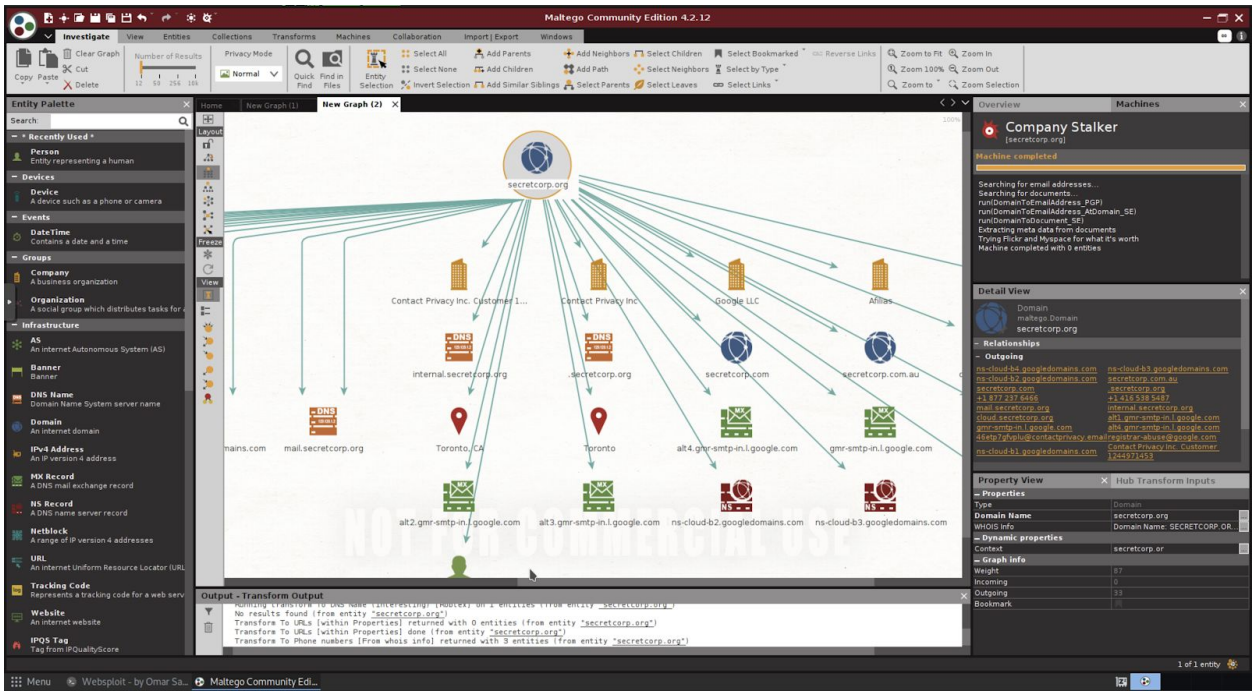
## The Transform Hub




Wizards like the Company Stalker allow you to automate the search of information about a given company.

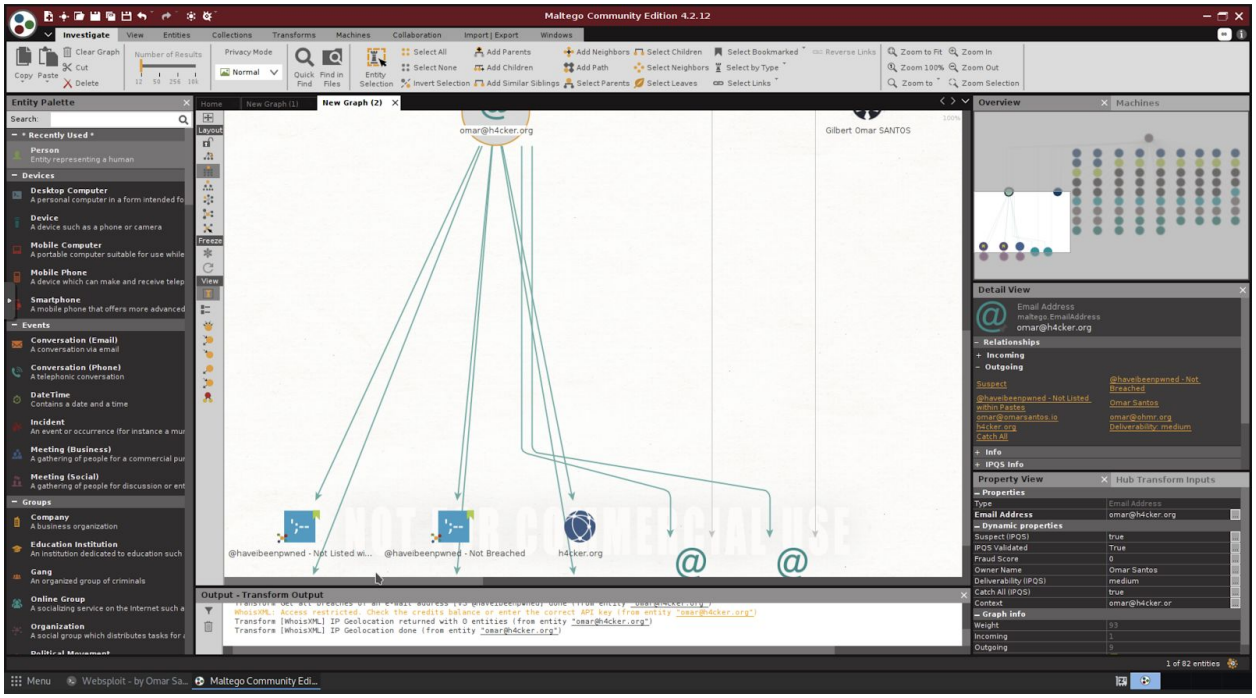



In the following example, I am using secretcorp.org (a domain owned by Omar). Feel free to use any other domain that you would like to get more information about.




In the results you see different subdomains (e.g., mail.secretcorp,org, internal.secretcorp.org, etc.) that were discovered using DNS and other information.

Try to find information about yourself. You can install and take advantage of the "HaveIBeenPwned" transform to see if your email or information has been exposed in a breach. You can use a "Pearson" search or an email search, as demonstrated in the example below:




# Exercise 5: TheHarvester

Use TheHarvester to find information about a domain of your choosing. We are only searching public information, so you can pick any domain.

In the following example, I am querying information about the h4cker.org domain using all the supported sources.

```
┌──(root💀websploit)-[~]
└─# theHarvester -d h4cker.org -b google

*******************************************************************
*                                                                 *
* | |_| |__   ___     /\  /\__ _ _ ____   _____  ___| |_ ___ _ __   *
* | __|  _ \ / _ \   / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* theHarvester 3.2.0                                            *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*                                                                 *
*******************************************************************


[*] Target: h4cker.org
```

```
[*] Hosts found: 5
---------------------
bootcamp.h4cker.org:185.199.110.153, 185.199.111.153, 185.199.108.153, 185.199.109.153
lpb.h4cker.org:185.199.110.153, 185.199.111.153, 185.199.108.153, 185.199.109.153
store.h4cker.org:185.199.110.153, 185.199.111.153, 185.199.108.153, 185.199.109.153
websploit.h4cker.org:127.0.1.1
www.h4cker.org
```

You can limit top query only one source byu using the -b argument with any of the options shown below:

```
                        Save the results to an HTML and/or XML file.
  -b SOURCE, --source SOURCE
                        baidu, bing, bingapi, bufferoverun, certspotter, crtsh, dnsdumpster, duckduckgo, exalead,
                        github-code, google, hackertarget, hunter, intelx, linkedin, linkedin_links, netcraft, otx,
                        pentesttools, projectdiscovery, qwant, rapiddns, securityTrails, spyse, sublist3r,
                        threatcrowd, threatminer, trello, twitter, urlscan, virustotal, yahoo
```

- What subdomains did you find?
- Did you find any emails?

# Exercise 6: Finding Usernames

Finding available usernames in numerous platforms:

- https://namechk.com/
- https://knowem.com/
- https://usersearch.org/


That information is useful to try to figure out the target's social footprint and other associations.

## WhatsMyName

You can also use tools such as WhatsMyName to find specific usernames in numerous sites and platforms.
First clone the GitHub repo github.com/WebBreacher/WhatsMyName, as shown below:

```
┌─[root@parrot]─[/]
└──╼ #git clone https://github.com/WebBreacher/WhatsMyName.git
Cloning into 'WhatsMyName'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 1766 (delta 4), reused 4 (delta 2), pack-reused 1755
Receiving objects: 100% (1766/1766), 840.19 KiB | 13.55 MiB/s, done.
Resolving deltas: 100% (1080/1080), done.
┌─[root@parrot]─[/]
└──╼ #cd WhatsMyName/
```

Install the required packages using pip, as demonstrated below:

```
┌─[root@parrot]─[/WhatsMyName]
└──╼ #python3 -m pip install -r requirements.txt
Requirement already satisfied: requests==2.24.0 in /usr/local/lib/python3.9/dist-packages (from -r requirements.t
xt (line 1)) (2.24.0)
Requirement already satisfied: urllib3==1.25.10 in /usr/local/lib/python3.9/dist-packages (from -r requirements.t
xt (line 2)) (1.25.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/lib/python3/dist-packages (from requests==2.24.0->-r req
uirements.txt (line 1)) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/lib/python3/dist-packages (from requests==2.24.0->-r requirem
ents.txt (line 1)) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/lib/python3/dist-packages (from requests==2.24.0->-r re
quirements.txt (line 1)) (2020.6.20)
┌─[root@parrot]─[/WhatsMyName]
└──╼ #
```

Use the web_accounts_list_checker.py Python script to find out if such username exists in numerous sites and platforms:

```
┌─[✗]─[root@parrot]─[/WhatsMyName]
└──╼ #python3 web_accounts_list_checker.py  -u johndoe
-  286 sites found in file.
>  Looking up https://www.7cups.com/@johndoe
>  Looking up https://learn.acloud.guru/profile/johndoe
>  Looking up https://asciinema.org/~johndoe
>  Looking up https://audiojungle.net/user/johndoe
>  Looking up https://www.biggerpockets.com/users/johndoe
>  Looking up https://www.bookcrossing.com/mybookshelf/johndoe
>  Looking up https://www.buymeacoffee.com/johndoe
>  Looking up https://www.championat.com/user/johndoe/
>  Looking up https://community.cloudflare.com/u/johndoe
>  Looking up https://www.cnet.com/profiles/johndoe/
>  Looking up https://www.coroflot.com/johndoe
>  Looking up https://www.codewars.com/users/johndoe
>  Looking up https://coderwall.com/johndoe/
>  Looking up https://johndoe.crevado.com/
>  Looking up https://dating.ru/johndoe/
>  Looking up https://www.designspiration.com/johndoe/
>  Looking up https://dev.to/johndoe
>  Looking up https://ello.co/johndoe
>  Looking up https://www.eyeem.com/u/johndoe
>  Looking up https://fancy.com/johndoe
>  Looking up https://www.gamepedia.com/members/johndoe
```

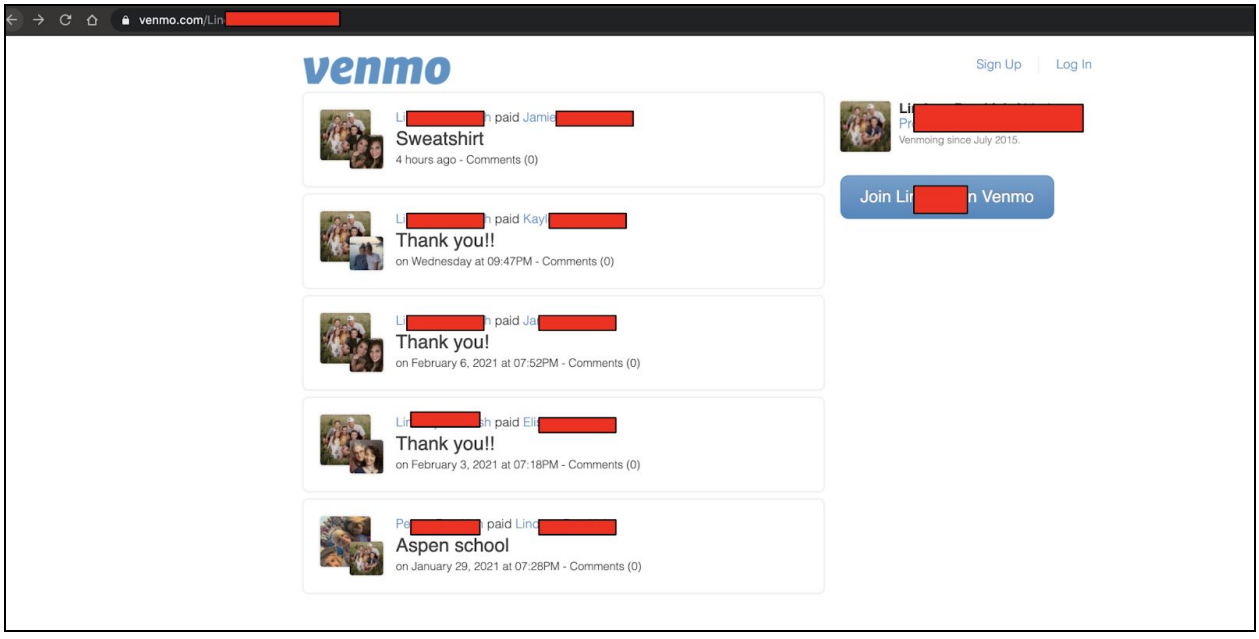
## Targeting Nerds Like Us

You can take advantage of public APIs like the GitHub API to obtain detailed information about the activities of their users: https://api.github.com/users/<user_name>/events/public

```
┌─[root@parrot]─[~]
└──╼ #curl -sSL https://api.github.com/users/santosomar/events/public
[
  {
    "id": "15235596035",
    "type": "PushEvent",
    "actor": {
      "id": 1690898,
      "login": "santosomar",
      "display_login": "santosomar",
      "gravatar_id": "",
      "url": "https://api.github.com/users/santosomar",
      "avatar_url": "https://avatars.githubusercontent.com/u/1690898?"
    },
    "repo": {
      "id": 94801380,
      "name": "The-Art-of-Hacking/h4cker",
      "url": "https://api.github.com/repos/The-Art-of-Hacking/h4cker"
    },
    "payload": {
      "push_id": 6558369521,
      "size": 1,
      "distinct_size": 1,
      "ref": "refs/heads/master",
      "head": "3983206dd4a1b29d235ceb10d79b65267a77a0ed",
      "before": "1e6112f10f08519d297d8c8e6a564bec920e0c6a",
      "commits": [
```

## Examples of Public Financial Transactions

You can use publicly available information in sites like venmo to see how people send money to their friends and family. You can also learn different habits, etc.



# Exercise 7: Finding Emails and Breach Information

You can use many tools (including TheHarvester, Infoga and others to find emails related to a domain and see if the person's information may be in dumps from different breaches.

To install the tool clone the repository and do a basic Python setup, as demonstrated below:

```
$ git clone https://github.com/m4ll0k/Infoga.git
$ cd Infoga
$ python setup.py install
$ python infoga.py
```

The following is the tool (Infoga) usage:

```
Usage: infoga.py [OPTIONS]

        -d --domain     Target URL/Name
        -s --source     Source data, default "all":

                all     Use all search engine
                google  Use google search engine
                bing    Use bing search engine
                yahoo   Use yahoo search engine
                ask     Use ask search engine
                baidu   Use baidu search engine
                dogpile Use dogpile search engine
                exalead Use exalead search engine
                pgp     Use pgp search engine

        -b --breach     Check if email breached
        -i --info       Get email informations
        -r --report     Simple file text report
        -v --verbose    Verbosity level (1,2 or 3)
        -H --help       Show this help and exit

Example:
        infoga.py --domain site.gov -v 3
        infoga.py --info admin@site.gov -v 3
        infoga.py --domain site.gov --source pgp --breach -v 1
        infoga.py --domain site.gov --source google --breach --report
site_gov.txt -v 3
```

The following is an example of a query using infoga:

```
┌─[root@parrot]─[~/Infoga]
└──╼ #python3 infoga.py --domain example.com --source all --breach -v 1
_____________________________________
---[ Infoga - Email OSINT
---[ Momo (m4ll0k) Outaadi
---[ https://github.com/m4ll0k
_____________________________________

[*] Searching "example.com" in Ask...
[i] Found 0 emails in Ask
[*] Searching "example.com" in Baidu...
[i] Found 4 emails in Baidu
[*] Searching "example.com" in Bing...
[i] Found 0 emails in Bing
[*] Searching "example.com" in DogPile...
[i] Found 0 emails in Dogpile
[*] Searching "example.com" in Exalead...
[i] Found 5 emails in Exalead
[*] Searching "example.com" in Google...
[i] Found 9 emails in Google
[*] Searching "example.com" in PGP...
```

# Exercise 8: Nmap

WebSploit is running several intentionally vulnerable applications in Docker containers.

1. Use Nmap to perform a basic TCP SYN scan.

```
┌──(root💀websploit)-[~]
└─# nmap -sS 127.0.0.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-20 22:34 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000060s latency).
Not shown: 992 closed ports
PORT     STATE SERVICE
88/tcp   open  kerberos-sec
3306/tcp open  mysql
8080/tcp open  http-proxy
8888/tcp open  sun-answerbook
9000/tcp open  cslistener
9001/tcp open  tor-orport
9002/tcp open  dynamid
9090/tcp open  zeus-admin

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
```

2. Why are you not able to see all the ports running on WebSploit? What about all the ports shown by using the **containers.sh** script under /root can be used to show the containers that are running in your system:

```
┌──(root💀websploit)-[~]
└─# ./containers.sh
```

```
WebSploit
by Omar Santos @santosomar

The following are the WebSploit vulnerable containers and associated exposed ports
+--------------------+------+
| Vuln App Container | Port |
+--------------------+------+
| webgoat            | 8881 |
| webwolf            | 9090 |
| juice-shop         | 8882 |
| dvwa               | 8883 |
| mutillidae_2       | 8884 |
| bwapp2             | 8885 |
| dvna               | 8886 |
| hackazon           | 8887 |
| hackme-rtov        | 8888 |
| mayhem             | 8889 |
| rtv-safemode       | 9000 |
| grayhat-mmxx       | 9001 |
| yascon-hackme      | 9002 |
+--------------------+------+

The following are the running containers:
NAMES               PORTS                                                 STATUS
yascon-hackme       0.0.0.0:9002->80/tcp                                  Up 7 days
grayhat-mmxx        8000/tcp, 0.0.0.0:9001->8001/tcp                      Up 7 days
rtv-safemode        0.0.0.0:3306->3306/tcp, 0.0.0.0:9000->80/tcp          Up 7 days
mayhem              0.0.0.0:88->22/tcp, 0.0.0.0:8889->80/tcp              Up 7 days
hackme-rtov         0.0.0.0:8888->80/tcp                                  Up 7 days
hackazon            0.0.0.0:8887->80/tcp                                  Up 7 days
dvna                0.0.0.0:8886->9090/tcp                                Up 7 days
bwapp2              3306/tcp, 0.0.0.0:8885->80/tcp                        Up 7 days
mutillidae_2        3306/tcp, 0.0.0.0:8884->80/tcp                        Up 7 days
dvwa                0.0.0.0:8883->80/tcp                                  Up 7 days
juice-shop          0.0.0.0:8882->3000/tcp                                Up 7 days
webgoat             0.0.0.0:9090->9090/tcp, 0.0.0.0:8881->8080/tcp        Up 7 days
```

3. Modify your Nmap TCP SYN Scan to find all ports (hint: leverage the [cheatsheet at the GitHub repository]).
4. Now try to find out what is the version of the underlying operating system.
5. Can you enumerate other information using the Nmap Scripting Engine (NSE)?

# Exercise 9: Directory Enumeration with Gobuster

Let's try to enumerate directories/folders and files in the web applications running in WebSploit's Docker containers based on the information you revealed using Nmap. You can use tools like Gobuster (https://github.com/OJ/gobuster). Gobuster is a very fast tool written in GoLang used to brute-force:

- URIs (directories and files) in web sites.
- DNS subdomains (with wildcard support).
- Virtual Host names on target web servers.
- Open Amazon S3 buckets

Use Gobuster to find the web directory on at least three (3) of the web applications that are running in the containers.

I will give you one hint:

```
┌──(root💀 websploit)-[~]
└─# gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://127.0.0.1:8889
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://127.0.0.1:8889
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2021/02/20 22:22:36 Starting gobuster
===============================================================
/s (Status: 301)
/admin (Status: 301)
/assets (Status: 301)
/wp-login (Status: 301)
/wp-admin (Status: 301)
```

1. Did you find the admin folder?
2. What other folders/directories did you enumerate in the other web applications running in WebSploit?
3. Can you enumerate files?
4. Try using other wordlists. There are many awesome wordlists under **/root/SecLists**:

```
┌──(root💀 websploit)-[~/SecLists]
└─# pwd
/root/SecLists

┌──(root💀 websploit)-[~/SecLists]
└─# ls
CONTRIBUTING.md  CONTRIBUTORS.md  Discovery  Fuzzing  IOCs  LICENSE  Miscellaneous  Passwords  Pattern-Matching  Payloads  README.md  Usernames  Web-Shells

┌──(root💀 websploit)-[~/SecLists]
└─#
```

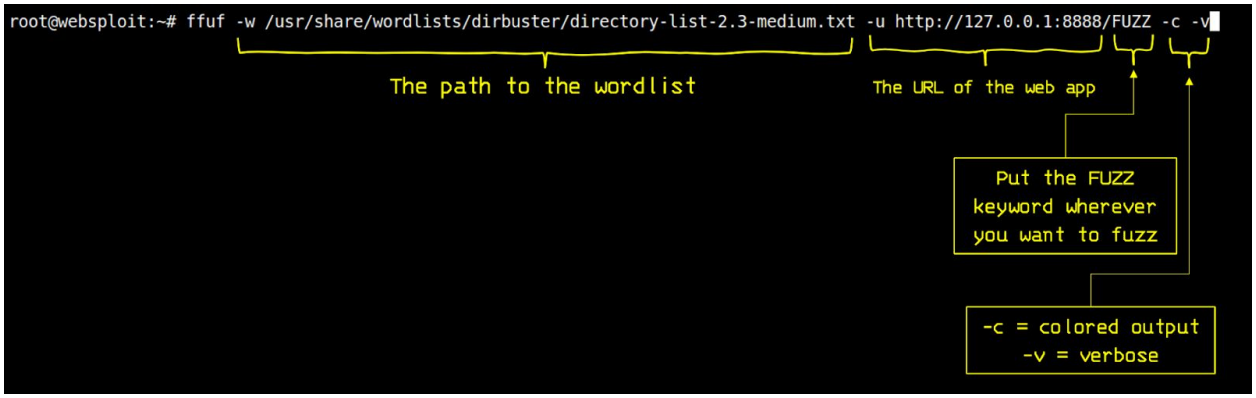
5. Did you find anything different with your new wordlist?

# Exercise 10: More Web Fuzzing with ffuf

ffuf and Burp Suite are two of my favorite tools for web application penetration testing and bug hunting. ffuf is a very fast web application fuzzer written in Go that is pretty popular among pen testers and bug bounty hunters. Not only it is fast, but it also has tons of great functionality that can help to integrate it with other tools like Burp Suite.
If you are reading this article, you may have a passion for hacking web applications or bug bounties. You may already know what Burp (or Burp Suite) is; however, here is a quick 2-second introduction. Burp is a very popular web application proxy, scanner, and overall awesome web penetration testing tool with tons of plugins. There are three versions of Burp: community edition (free), professional, and enterprise. I am using the community edition in the following examples.
I am using the learning environment that I created called WebSploit Labs for the next demonstrations. WebSploit is basically Kali Linux + several additional tools and tons of Docker containers running intentionally vulnerable applications. WebSploit also has over 8,000 cybersecurity resources (a clone of my pretty popular GitHub repository).

## A Basic Example

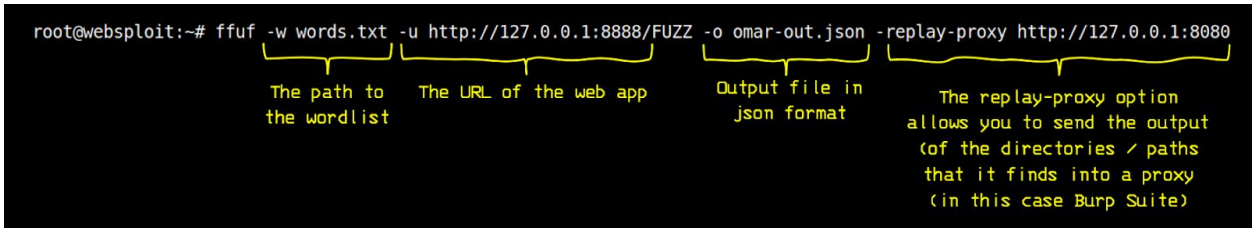
The following is a very basic example of running ffuf to enumerate and discover different directories in one of the intentionally vulnerable web applications (running on port 8888). The following is the explanation of the command syntax. I am using the wordlist that comes with dirbuster in Kali (directory-list-2.3-medium.txt).

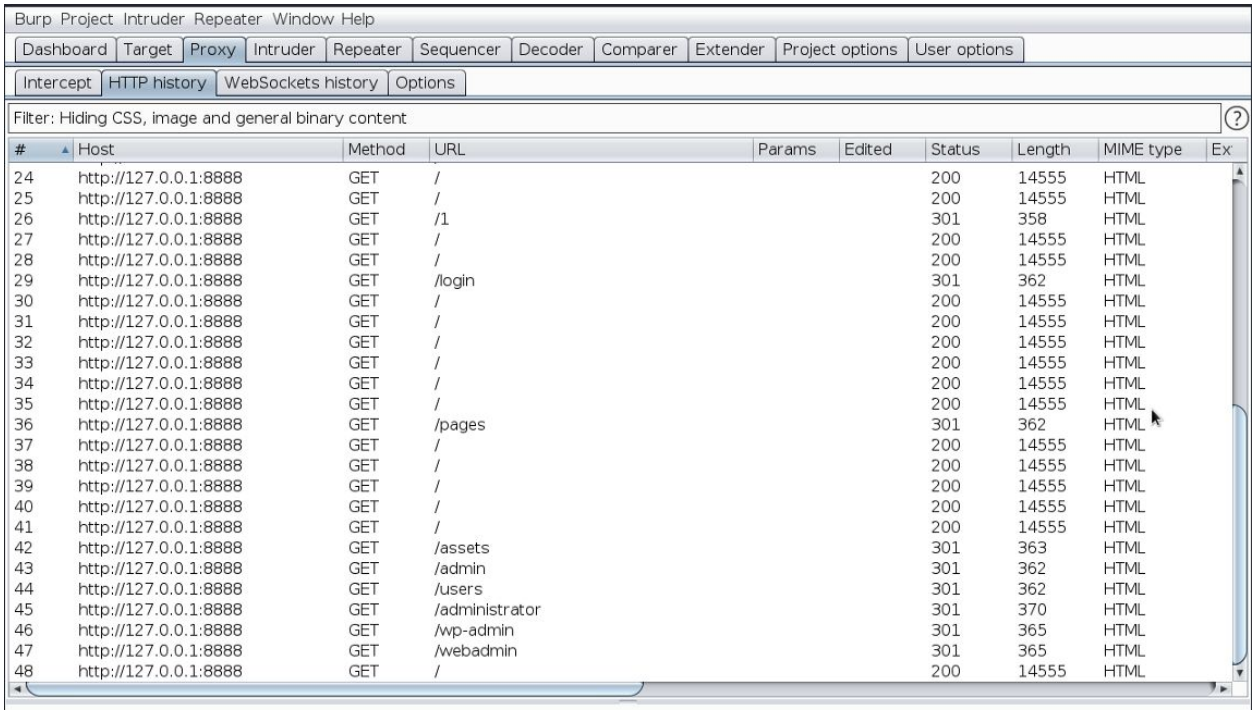



HINT: This GIF has a good demo..

## Saving the Results to a File and also Sending the Directories Found to Burp

The **-o** option allows you to send the output to a JSON file (omar-out.json in the example below). The **-replay-proxy** is the cool option that allows you to send the paths of the directories found into Burp. Why is this useful? Well, the free version of Burp does not come with an automated scanner, spider, or fuzzer. This method, at least, allows you to send all the successful results right into Burp for further analysis.




| # | Host | Method | URL | Params | Edited | Status | Length | MIME type | Ex |
|---|---|---|---|---|---|---|---|---|---|
| 24 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 25 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 26 | http://127.0.0.1:8888 | GET | /1 | | | 301 | 358 | HTML | |
| 27 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 28 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 29 | http://127.0.0.1:8888 | GET | /login | | | 301 | 362 | HTML | |
| 30 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 31 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 32 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 33 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 34 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 35 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 36 | http://127.0.0.1:8888 | GET | /pages | | | 301 | 362 | HTML | |
| 37 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 38 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 39 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 40 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 41 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |
| 42 | http://127.0.0.1:8888 | GET | /assets | | | 301 | 363 | HTML | |
| 43 | http://127.0.0.1:8888 | GET | /admin | | | 301 | 362 | HTML | |
| 44 | http://127.0.0.1:8888 | GET | /users | | | 301 | 362 | HTML | |
| 45 | http://127.0.0.1:8888 | GET | /administrator | | | 301 | 370 | HTML | |
| 46 | http://127.0.0.1:8888 | GET | /wp-admin | | | 301 | 365 | HTML | |
| 47 | http://127.0.0.1:8888 | GET | /webadmin | | | 301 | 365 | HTML | |
| 48 | http://127.0.0.1:8888 | GET | / | | | 200 | 14555 | HTML | |

The following is the first few lines of the contents of the output file (omar-out.json):

```
root@websploit:~# less omar-out.json | python -m json.tool
{
    "commandline": "ffuf -w words.txt -u http://127.0.0.1:8888/FUZZ -o omar-out.json -replay-proxy http://127.0.0.1:8080",
    "config": {
        "autocalibration": false,
        "autocalibration_strings": [],
        "cmd_inputnum": 100,
        "cmdline": "ffuf -w words.txt -u http://127.0.0.1:8888/FUZZ -o omar-out.json -replay-proxy http://127.0.0.1:8080",
        "colors": false,
        "delay": {
            "value": "0.00"
        },
        "dirsearch_compatibility": false,
        "extensions": [],
        "filters": {},
        "follow_redirects": false,
        "headers": {},
        "ignore_wordlist_comments": false,
        "inputmode": "clusterbomb",
        "inputproviders": [
            {
                "keyword": "FUZZ",
                "name": "wordlist",
                "value": "words.txt"
            }
        ],
        "matchers": {
            "status": {
                "value": "200,204,301,302,307,401,403"
            }
        },
        "maxtime": 0,
```

You can find additional details of all the different supported options at the ffuf man page (**man ffuf**). I just wanted to introduce the concept of “integrating” both tools (ffuf and Burp) to perform reconnaissance and further analysis.

An additional cool trick to pull off with **-replay-proxy**

It works well with a reverse SSH tunnel as well, so you can run your ffuf on VPS while proxying the matches to Burp running on your local desktop.

**Your Turn!** Now try to do this to at least three (3) of the web applications running in WebSploit.

# Exercise 11: OWASP ZAP

The easiest way to start using ZAP is via the Quick Start tab. To run a Quick Start Automated Scan :

1. Start ZAP (you can use the **zaproxy** command in the Linux terminal).
2. Select all the default options.
3. Click the **Quick Start** tab of the **Workspace Window**.
4. Click the large **Automated Scan** button.

5. In the **URL** to attack text box, enter the full URL of the web application you want to attack. Enter **http://127.0.0.1:port** (**port** = the port of any of the web applications running in the different WebSploit containers).
6. Click the **Attack** button.

As ZAP spiders your web application, it constructs a map of your web applications' pages and the resources used to render those pages. Then it records the requests and responses sent to each page and creates alerts if there is something potentially wrong with a request or response.

## (Optional) Automating your workflow with the OWASP ZAP API

***This document*** provides example guides & API definitions for ZAP APIs. You can view code examples in the dark area to the right; switch the programming language of the examples with the tabs on the top right.
The following are some of the features provided by ZAP:

- Intercepting Proxy
- Active and Passive Scanners
- Traditional and Ajax Spiders
- Brute Force Scanner
- Port Scanner
- Web Sockets

This is an optional exercise. If you are a web developer, feel free to familiarize yourself with the ZAP API: https://www.zaproxy.org/docs/api/?python#introduction

The following is an example using Python (zapv2).

```
#!/usr/bin/env python
import time
from zapv2 import ZAPv2

# The URL of the application to be tested
target = 'https://public-firing-range.appspot.com'
# Change to match the API key set in ZAP, or use None if the API key is disabled
apiKey = 'changeMe'

# By default ZAP API client will connect to port 8080
zap = ZAPv2(apikey=apiKey)
# Use the line below if ZAP is not listening on port 8080, for example, if listening on port
8090
# zap = ZAPv2(apikey=apiKey, proxies={'http': 'http://127.0.0.1:8090', 'https':
'http://127.0.0.1:8090'})

print('Spidering target {}'.format(target))
# The scan returns a scan id to support concurrent scanning
scanID = zap.spider.scan(target)
while int(zap.spider.status(scanID)) < 100:
    # Poll the status until it completes
```

```
    print('Spider progress %: {}'.format(zap.spider.status(scanID)))
    time.sleep(1)

print('Spider has completed!')
# Prints the URLs the spider has crawled
print('\n'.join(map(str, zap.spider.results(scanID))))
# If required post process the spider results
```

# Exercise 12: Enumerating Users in Linux

After you compromise a system (post exploitation), you typically try to obtain information about what other users have access to such systems. Let's do a quick exploration of the different ways that you can enumerate users in a Linux system.
To make this exercise a little more interesting. Create a couple of users in your WebSploit system using the useradd command, as shown below:

```
┌──(root💀websploit)-[~]
└─# useradd -m superman -s /bin/bash

┌──(root💀websploit)-[~]
└─# useradd -m batman -s /bin/bash

┌──(root💀websploit)-[~]
└─# passwd superman
New password:
Retype new password:
passwd: password updated successfully

┌──(root💀websploit)-[~]
└─# passwd batman
New password:
Retype new password:
passwd: password updated successfully
```

**Of course, this is a very basic tutorial and if you already have root access to the system, you won't need to enumerate much ;-)

1. Get a List of All Users using the **cat /etc/passwd** file… Local user information is stored in the /**etc/passwd** file. Each line in this file represents login information for one user. To open the file you can either use **cat** or **less** :

```
┌──(omar㉿websploit)-[~]
└─$ cat /etc/passwd | egrep "superman|batman"
superman:x:1001:1001::/home/superman:/bin/bash
batman:x:1002:1002::/home/batman:/bin/bash
```

Each line in the file has seven fields delimited by colons that contain the following information:

- User name.
- Encrypted password (x means that the password is stored in the /etc/shadow file).
- User ID number (UID).
- User's group ID number (GID).
- Full name of the user (GECOS).
- User home directory.
- Login shell (defaults to /bin/bash).

If you want to display only the username you can use either **awk** or **cut** commands to print only the first field containing the username:

**awk -F: '{ print $1}' /etc/passwd**

```
┌──(omar㉿websploit)-[~]
└─$ awk -F: '{ print $1}' /etc/passwd
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-timesync
systemd-network
systemd-resolve
mysql
```

The following is an example using the **cut** command:
**cut -d: -f1 /etc/passwd**

```
┌──(omar㉿websploit)-[~]
└─$ cut -d: -f1 /etc/passwd
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
```

2. You can also use the **getent** command. The **getent** command displays entries from databases configured in **/etc/nsswitch.conf** file, including the passwd database, which can be used to query a list of all users.

```
┌──(omar㉿websploit)-[~]
└─$ getent passwd | grep superman
superman:x:1001:1001::/home/superman:/bin/bash
```

As you can see, the output is the same as when displaying the content of the **/etc/passwd** file. If you are using LDAP for user authentication, the getent will display all Linux users from both **/etc/passwd** file and LDAP database.

You can also use **awk** or **cut** to print only the first field containing the username:

- **getent passwd | awk -F: '{ print $1}'**
- **getent passwd | cut -d: -f1**

If you want to find out how many users accounts you have on your system, pipe the getent passwd output to the **wc** command:

```
┌──(omar㉿websploit)-[~]
└─$ getent passwd | wc -l
56
```

As you can see from the output above, my Linux system has 33 user accounts.
System and Normal Users

There is no real technical difference between the system and regular (normal) users. Typically system users are created when installing the OS and new packages. In some cases, you can create a system user that will be used by some applications.

Normal users are the users created by the root or another user with sudo privileges. Usually, a normal user has a real login shell and a home directory.

Each user has a numeric user ID called UID. If not specified when creating a new user with the useradd command, the UID will be automatically selected from the /etc/login.defs file depending on the UID_MIN and UID_MIN values.

To check the UID_MIN and UID_MIN values on your system, you can use the following command:

```
┌──(omar㉿websploit)-[~]
└─$ grep -E '^UID_MIN|^UID_MAX' /etc/login.defs
UID_MIN                  1000
UID_MAX                 60000
```

From the output above, we can see that all normal users should have a UID between 1000 and 60000. Knowing the minimal and maximal value allow us to query a list of all normal users in our system.
Your system UID_MIN and UID_MIN values may be different so the more generic version of the command above would be:

```
eval getent passwd {$(awk '/^UID_MIN/ {print $2}' /etc/login.defs)..$(awk '/^UID_MAX/ {print $2}' /etc/login.defs)}
```

For example:

```
┌──(omar㉿websploit)-[~]
└─$ eval getent passwd {$(awk '/^UID_MIN/ {print $2}' /etc/login.defs)..$(awk '/^UID_MAX/ {print $2}' /etc/login
.defs)}

omar:x:1000:1000:Omar Santos,,,:/home/omar:/usr/bin/zsh
superman:x:1001:1001::/home/superman:/bin/bash
batman:x:1002:1002::/home/batman:/bin/bash
```

In my system, there was also a user called “omar”.

If you want to print only the usernames just pipe the output to the cut command:

```
eval getent passwd {$(awk '/^UID_MIN/ {print $2}' /etc/login.defs)..$(awk '/^UID_MAX/ {print $2}' /etc/login.defs)} | cut -d: -f1
```

You can also use the **lastlog** command to find out the last time that a user has logged in to the system:

```
┌──(omar㉿websploit)-[~]
└─$ lastlog | grep superman
superman         pts/2    127.0.0.1        Sat Feb 20 23:25:41 -0500 2021
```

You may see ****Never logged in**** in your output, since you have not logged in to the system with the “superman” user.

## Additional Tips for Account Discovery

- **Local Accounts**: https://attack.mitre.org/techniques/T1087/001/
  In Windows, commands such as **net user** and **net** localgroup of the Net utility. Alsoyou can use **id** and **groupson** in macOS and Linux to list local users and groups.
- **Domain Accounts**: https://attack.mitre.org/techniques/T1087/002/
  Commands such as **net user /domain** and **net group /domain** of the Net utility, **dscacheutil -q groupon** macOS, and **ldapsearch** on Linux can list domain users and groups.
- **Email Accounts**: https://attack.mitre.org/techniques/T1087/003/
  In on-premises Exchange and Exchange Online, the **Get-GlobalAddressList** PowerShell cmdlet can be used to obtain email addresses and accounts from a domain using an authenticated session
- **Cloud Accounts**: https://attack.mitre.org/techniques/T1087/004/
  - Adversaries may attempt to get a listing of cloud accounts. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or SaaS application.
  - With authenticated access there are several tools that can be used to find accounts. The **Get-MsolRoleMember** PowerShell cmdlet can be used to obtain account names given a role or permissions group in Office 365. The Azure CLI (AZ CLI) also provides an interface to obtain user accounts with authenticated access to a domain. The command az ad user list will list all users within a domain.
  - The AWS command **aws iam list-users** may be used to obtain a list of users in the current account while **aws iam list-roles** can obtain IAM roles that have a specified path prefix.
  - In GCP, **gcloud iam service-accounts** list and **gcloud projects get-iam-policy** may be used to obtain a listing of service accounts and users in a project.

**Congratulations! You have completed DAY 1.**
Tomorrow you will learn a LOT about how to perform Dark Web reconnaissance.
See you tomorrow!