Source: *Stable Diffusion AI*

Laboratory 2. Information Discovery and OSINT

DEPARTMENT OF COMPUTER SCIENCE. UNIVERSITY OF OVIEDO

Computer Security | 2022 – 2023 (V2.7 *"S-81 Isaac Peral"*)

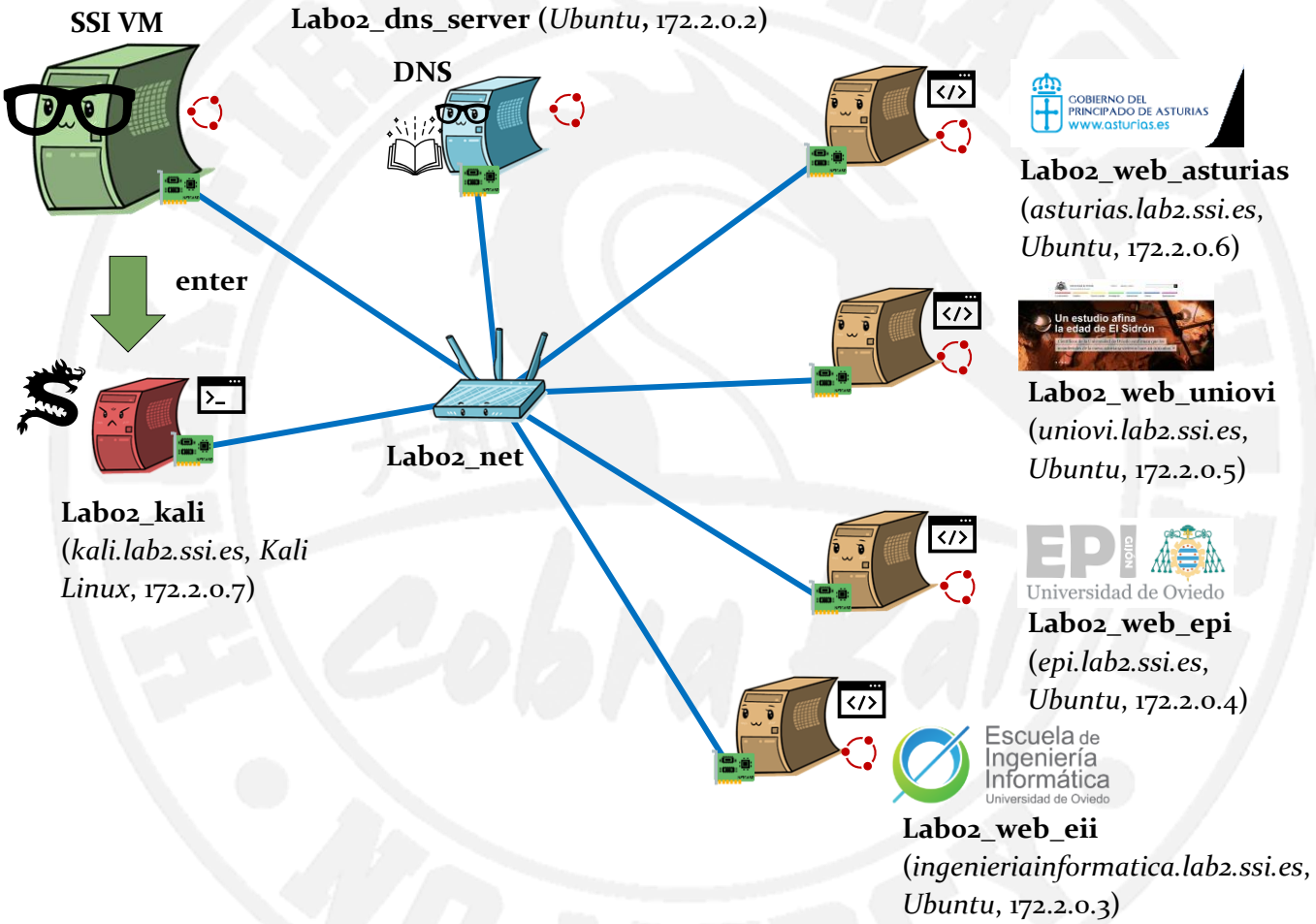
CONTENT

THE INFRASTRUCTURE OF THIS LABORATORY

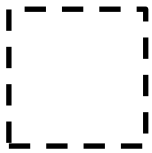
First, ensure that you have the `ssi_labs` folder structure we provided to you inside your VM. This lab (`lab_02`) consists of an infrastructure composed by several containers. Not all activities require using all of them. Please see the following diagram to fully understand the infrastructure layout:

- **An internal LAN**: (*labo2_net*, `172.2.0.0/16`)
- **4 different web servers**
- **A DNS**: providing names for each web server in the domain `lab2.ssi.es`.
- **A Kali machine** (`lab2_kali`) with a couple of tools installed **for enumeration purposes only**. These are described through this document. This also means you do not have to install them if you do not want. This *Kali* machine is the starting point of the lab.

IMPORTANT: **Make sure you have read and understood the instructions of the "*SSI Automated Infrastructures*" document to understand how to work with this type of automated lab infrastructures.**



BLOCK 1: EXPLORING WEB CONTENTS



Robots.txt file

Practical application: *You need to know if a website is revealing "secret" routes through this public file*

Part of the **Robot Exclusion Standard**, this file is used to tell search engines that follow this standard to not to index the URLs corresponding to certain entries found under the keyword `Disallow` in a file that is always named `robots.txt`. These files are always accessible using this URL pattern: `<My URL>/robots.txt`. Some developers use it so that *Google* or other search engines do not index "sensitive" content. The problem is that this file is public and accessible so engines can read its contents, and if engines can read it, you can too! 😊. You can find more information here: https://www.synopsys.com/blogs/software-security/robots-txt/

Expected Results: To complete this activity you need to pick any web you like and analyze its robots.txt file, determining if you can find something interesting or compromising security and why.

Document Metadata

Practical application: *You need to know if documents, images, etc. of a website are revealing too much information about their owners or authors*

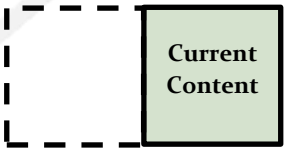
Metadata is data stored attached to a file, **but not as part of its contents, but normally in their file headers**. Metadata can be attached to a file in clear text or ciphered. Lots of people ignore that files may have metadata. Even worse, lots of people also do not know that lots of programs automatically put metadata in the files they generate without their consent or being aware of it. The importance of this (meta)data may vary from trivial to truly compromising information (tell that to *Parler* users!). Nowadays, lots of services automatically remove metadata from files you upload (social networks, except *Parler*, do), but you should not trust them to do it 100% of the time (bugs!). They may keep the original metadata from their own use (marketing), even if they remove it from the files displayed to the public. Therefore, you should remove metadata from the files you upload using appropriate programs just in case 😊.

FOCA is a *Windows* program that extracts and analyzes metadata from different file types, that also implements searching in domains, DNS servers, URLs, and published documents. FOCA can be downloaded from here: https://github.com/ElevenPaths/FOCA. There is a free open-source version and a commercial version. Unfortunately, it requires to set up a *SQL Database* (*SQL Express* or *SQL Server*) that takes time and effort to install. This is not part of this course objectives so, instead of this tool, we are going to use a simpler online one from the same vendor that analyzes the metadata of individual documents and other file types: ***Metashield Clean-up Online*** (https://metashieldclean-up.elevenpaths.com/).

This web page accepts any file of a list of accepted file types and extracts its metadata to see if there is something useful or suspicious in them.

Expected Results: This activity will be completed when you pick any local or downloaded file of the accepted types and analyze it with this online tool to see if you find something interesting.

BLOCK 2: SEARCHING FOR GOLD



Google Hacking

Practical application: *You can use the Google Search engine to find compromising information about any target on Internet*

Google Hacking is the ability to do certain manual search operations in the *Google Search Engine*, normally combined with the `site:` operator (also valid with other search engines, although each engine uses their own operators). These search operations aim to detect vulnerabilities, configuration problems, or information that may lead to an attack. There are lots of predefined search queries that cover different types of attacks / information gathering procedures in the ***Google Hacking Database*** **(GHDB)**: https://www.exploit-db.com/google-hacking-database. The procedure to use this database is:

- Choose a search category and a query
- Perform the query over a concrete target (using the `site:` search operator, https://support.google.com/websearch/answer/2466433?hl=es) and see if the result matches the expected one according to the search you chose.

You can find an image with the steps that this procedure requires **in the first seminar** of the course. We have the following categories of predefined searches

Footholds	Files Containing Juicy Info
Files Containing Usernames	Files Containing Passwords
Sensitive Directories	Sensitive Online Shopping Info
Web Server Detection	Network or Vulnerability Data
Vulnerable Files	Pages Containing Login Portals
Vulnerable Servers	Various Online Devices
Error Messages	Advisories and Vulnerabilities

You can also find URLs containing more examples **in the first seminar**.

Expected Results: This activity will be completed when you choose a website you like and perform a couple of search operations with the *Google Hacking Database* to familiarize with the process, analyzing if you find something suspicious.

Shodan

Practical application: *You can use the Shodan machine search engine to know information about any public endpoint on internet*

Shodan (http://www.shodanhq.com/) is a search engine able to find devices instead of web pages: routers, servers, CCTV recording cameras, traffic lights, ... More details about its functionality and usage can be found **in the first seminar** (also including URLs with usage examples).

Expected Results: This activity will be completed when you can perform a query in *Shodan* over a single URL or IP, analyzing the output you obtain and why you think this information may be a danger for the target. You have more freedom if you create a free account, but it is not necessary to do this exercise if you limit yourself to single IPs.

Censys

Practical application: *You can use the Censys machine search engine to know information about any public endpoint on internet*

Censys is like *Shodan*, but it presents information in a more structured way. It allows you to do search operations based on the data it obtains from the servers it analyzes all over the Internet. **The first seminar** offers more details about it. *Censys* usage examples can be found here:

- https://search.censys.io/search/examples?resource=hosts
- https://support.censys.io/hc/en-us/articles/360059720271-Search-2-0-Example-Host-Queries

Expected Results: This activity will be completed when you can use *Censys* to scan an IP or network range (for example one range that belongs to Uniovi). Then, you must analyze the output it provides and the information that is displayed about each host. Analyze this information to determine why it could be dangerous from an attack point of view.

The Wayback machine

Practical application: *You can use the Internet Archive (aka "Wayback Machine") search engine to know information about past published websites on many internet domains*

*(**NOTE**: The Wayback Machine is massive, and therefore is sometimes quite slow. It is not your connection, or your ISP, it is just like that, and there is nothing you can do about it 😊)*

Most people (including a sizable amount of web administrators!) are used to search for compromising content in web pages to remove it, but they do not consider what happens with the content that **WAS** there, but no longer do. This content may prove especially useful to attackers, and this lab section will teach you how to deal with it.

There is a reference web page to know about the "history" of any website on Internet and its name is "***The Wayback Machine***" (aka *The Internet Archive*): https://archive.org/web/. This website is a massive archive of previous contents displayed by any public website that was active on Internet in certain dates in the past, but it also has "hacking" usages. For example, you can inspect ALL the URLs that has been extracted from a particular domain using a query like this: http://web.archive.org/web/*/<URL>/* (e. g. http://web.archive.org/web/*/http://www.uniovi.es/*). Please note that the URL load is asynchronous, so it may take a while. Do not worry if you initially find an empty table (for Uniovi, it usually takes 30s to load).

Once we have the historic URL list of a site, we can use this information to obtain a series of remarkably interesting information about it. You can find more details here: https://www.elladodelmal.com/2013/04/hacking-con-archivecom-wayback-machine.html

- The URL table may be processed with code to extract all the URLs, or the **URLs of a certain type** that may be interesting for a certain purpose. For example, you can extract images or documents so you can inspect its contents (or metadata! 😄 😄). This can also be done in the *Wayback Machine* web page, as it has a filter textbox.

100,000 URLs have been captured for this domain.

Filter results (i.e. '.txt'): pdf

URL	MIME TYPE	FROM	TO	CAPTURES	DUPLICATES	UNIQUES
http://www.uniovi.es/-/defecto.pdf	text/html	Jun 16, 2013	Jul 15, 2013	2	0	2
http://www.uniovi.es/-/file.pdf	text/html	Jun 16, 2013	Jul 15, 2013	2	0	2
http://www.uniovi.es/-/preistabelle.pdf	text/html	Jun 16, 2013	Jul 15, 2013	2	0	2
http://www.uniovi.es/-/presitabelle.pdf	text/html	Jun 16, 2013	Jul 15, 2013	2	0	2
http://www.uniovi.es/aal/archivos_pdf/neutro_materia.pdf	text/html	Jun 6, 2011	Dec 15, 2018	7	6	1
http://www.uniovi.es/accesoyayudas/estudios/373/Oficio+de+Adjunto+Plantilla+Solicitud.pdf/5412a9a3-9730-40a4-8	text/plain	Sep 4, 2014	Sep 4, 2014	1	0	1
http://www.uniovi.es/accesoyayudas/tramites/tramite/-/asset_publisher/v9UAvM9qJpFc/content/wwww.uniovi.es/documents/31582/243104/INSTANCIA+VICERRECTOR+ESTUDIANTES_131028.pdf/d7cdc975-d56c-404f-b5a0-ef50be637791	text/html	Sep 8, 2014	Sep 8, 2014	1	0	1
http://www.uniovi.es/Alfonso_Garcia_Leal/1993478.pdf	text/html	Jan 19, 2012	Apr 12, 2012	2	1	1
http://www.uniovi.es/Areas/Mecanica.Fluidos/becasempleo/investigacionaerodinamica_paniaguaSMALL.pdf	unk	Nov 11, 2014	Nov 11, 2014	1	0	1
http://www.uniovi.es/Areas/Mecanica.Fluidos/docencia/_asignaturas/maquinas_de_fluidos/Lecc6_r1.pdf	unk	Apr 12, 2017	Apr 12, 2017	1	0	1
http://www.uniovi.es/Areas/Mecanica.Fluidos/docencia/_asignaturas/maquinas_de_fluidos/Presenta_Leccion3.pdf	unk	Apr 12, 2017	Apr 12, 2017	1	0	1
http://www.uniovi.es/Areas/Mecanica.Fluidos/docencia/_asignaturas/mecanica_de_fluidos/05_06/8.%20FLUJO_CONDUCTOS.pdf	unk	Jul 29, 2015	Jul 29, 2015	1	0	1

- You can also obtain the contents of **interesting files** of a web. E. g.: the `robots.txt` file
- **History of any file on the website**: The URL table column shows the number of different copies that any file has had over time. This means that we can locate any version of any file that has been indexed. You can iterate through them in the calendar view once you click in the file

Saved **4 times** between June 28, 2012 and July 11, 2013.

1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 **2013** 2014 2015 2016 2017 2018

JAN

	1	2	3	4	5	
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

FEB

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

MAR

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

APR

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

MAY

		1	2	3	4	
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

JUN

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

JUL

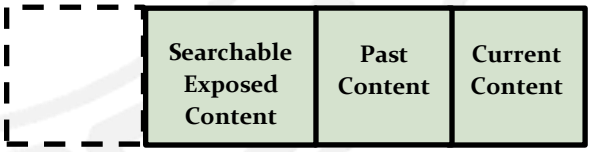
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

AUG

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Expected results: This activity will be completed when you are able to use the *Wayback Machine* to locate the historic URLs list of any website you like, any interesting file you choose, and the different historic versions of any document or webpage that was in the site. Once you can do that, you also must think about what this could mean from the security point of view (the questions at the end of the lab may help you to analyze it).

BLOCK 3: LOOKING BACK OVER MY DNS



Sub-domains Discovery

Practical application: *You need to know if a particular domain name also has subdomains registered*

Domain names are very curious because normally most people know the main one but, most of the times, there are secondary subdomains that are not so popular, only known by some people, or just forgotten. In any case, they suppose more vectors of enumeration, and hence the potential of discovering more things about a target. There are lots of ways of discovering this information automatically. These are some of them:

- `knockpy` (https://github.com/guelfoweb/knock). *Python* tool designed to enumerate subdomains on a target domain through a wordlist. On *Ubuntu*, you can install it with `sudo apt install knockpy` and run a basic scan with `./knockpy <target>`. You can find full usage instructions it their repository README.
- `dnsenum` (https://github.com/fwaeytens/dnsenum): E. g. `dnsenum uniovi.es`. If no `-f` tag is supplied, it will default to `/usr/share/dnsenum/dns.txt` or a `dns.txt` file in the same directory as `dnsenum.pl`

NOTE: A few students have reported that `dnsenum` may freeze their labs when is launched. If it is your case, quickly cancel the operation (CRTL+C) and use another tool. This is due to a *VirtualBox* installation problem, but we could not trace its exact origin, as it does not happen to every student.

dnsenum 1.2.6 Cheatsheet (ingenieriainformatica.uniovi.es)
multithreaded perl script to enumerate DNS information of a domain

https://github.com/fwaeytens/dnsenum

GENERAL USAGE

```
dnsenum [Options] <domain>
```

NOTES

If no -f tag supplied will default to /usr/share/dnsenum/dns.txt or the dns.txt file in the same directory as dnsenum.pl

OPTIONS

GENERAL OPTIONS

- **--dnsserver <server>**: Use this DNS server for A, NS and MX queries.
- **--enum**: Shortcut option equivalent to --threads 5 -s 15 -w.
- **-h, --help**: Print this help message.
- **--nocolor**: Disable ANSIColor output.
- **--noreverse**: Skip the reverse lookup operations.
- **--private**: Show and save private ips at the end of the file domain_ips.txt.
- **--subfile <file>**: Write all valid subdomains to this file.
- **-t, --timeout <value>**: The tcp and udp timeout values in seconds (default: 10s).
- **--threads <value>**: The number of threads that will perform different queries.
- **-v, --verbose**: Be verbose: show all the progress and all the error messages.

GOOGLE SCRAPING OPTIONS

- **-p, --pages <value>**: The number of google search pages to process when scraping names, the default is 5 pages, the -s switch must be specified.
- **-s, --scrap <value>**: The maximum number of subdomains that will be scraped from Google (default 15).

BRUTE FORCE OPTIONS

- **-f, --file <file>**: Read subdomains from this file to perform brute force. (Takes priority over default dns.txt)
- **-r, --recursion**: Recursion on subdomains, brute force all discovered subdomains that have an NS record.
- **-u, --update <a|g|r|z>**: Update the file specified with the -f switch with valid subdomains.
  - a (all) Update using all results.
  - g Update using only google scraping results.
  - r Update using only reverse lookup results.
  - z Update using only zonetransfer results.

```
operario@kali:~$ dnsenum uniovi.es
dnsenum VERSION:1.2.6

-----   uniovi.es   -----

Host's addresses:
__________________

uniovi.es.                               1216     IN    A        156.35.233.105

Name Servers:
_____________

enol.si.uniovi.es.                       172800   IN    A        156.35.14.2
chico.rediris.es.                        6186     IN    A        162.219.54.2
zeus.etsimo.uniovi.es.                   1800     IN    A        156.35.23.24
sun.rediris.es.                          3781     IN    A        199.184.182.1
solid.net.uniovi.es.                     1800     IN    A        156.35.11.170

Mail (MX) Servers:
__________________

mx02.puc.rediris.es.                     30       IN    A        130.206.19.162
mx02.puc.rediris.es.                     30       IN    A        130.206.19.130
mx01.puc.rediris.es.                     30       IN    A        130.206.19.162
mx01.puc.rediris.es.                     30       IN    A        130.206.19.130

Trying Zone Transfers and getting Bind Versions:
________________________________________________


Trying Zone Transfer for uniovi.es on enol.si.uniovi.es ...
AXFR record query failed: REFUSED

Trying Zone Transfer for uniovi.es on chico.rediris.es ...
AXFR record query failed: REFUSED
```

WHOIS NETRANGE OPTIONS:

- **-d, --delay <value>**: The maximum value of seconds to wait between whois queries, the value is defined randomly, default: 3s.
- **-w, --whois**: Perform the whois queries on c class network ranges. **Warning**: this can generate very large netranges and it will take lot of time to perform reverse lookups.

REVERSE LOOKUP OPTIONS:

- **-e, --exclude <regexp>**: Exclude PTR records that match the regexp expression from reverse lookup results, useful on invalid hostnames.

OUTPUT OPTIONS:

- **-o --output <file>**: Output in XML format. Can be imported in MagicTree (www.gremwell.com)

- https://dnsdumpster.com/: Website that allow us to locate subdomains of a given domain.
- https://searchdns.netcraft.com/: Another website that allow us to locate subdomains of a given domain.

- https://www.virustotal.com : If we go to "Search", provide a DNS name (e. g. uniovi.es), and perform a URL detection scan like the one we previously saw, the "Details" tab also gives its associated subdomains.
- https://crt.sh/: Providing a domain name (e. g. uniovi.es) also gives us associated subdomains.

Finally, it is possible that some of these domains may be, as we said, **abandoned**. In this case we have two options:

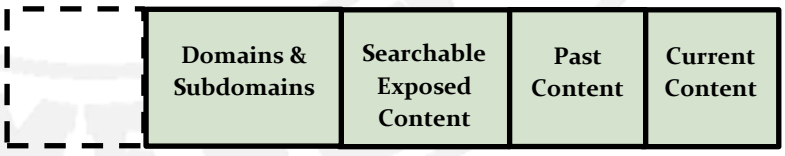
- A **truly abandoned** domain (no content)
- An **unmaintained domain** (potentially vulnerable contents, old versions of services, unpatched, unfixed…).

If we have lots of domains to analyze, distinguishing truly empty ones from others with contents may take a lot of time. We can save time if we visually inspect the contents of any of the domains we found, to see what type of contents they serve. The `eyewitness` (https://github.com/ChrisTruncer/EyeWitness) tool accepts a file with a list of sub-domains, and it will give you a report with screenshots of each sub-domain. E. g. `eyewitness.py -f subdomains.txt`. However, we will not use it in this lab.

These are not the only tools that can do this work, but they cover most use cases. `Knockpy` and `dnsenum` require installation to use them, but they are available in the standard *Kali Linux* packages. You can find these installed in the *Kali* container of the **Lab 2 infrastructure**.

Expected Results: This activity will be completed when you are able to enumerate all the subdomains of `uniovi.es` with any of the web-based provided tools. Command-line tools must be used against the domain included in the infrastructure of this lab, `lab02.ssi.es`. We recommend using at least one of the command-line tools and one of the websites provided to gain better understanding of how to perform this activity. Evaluating all the mentioned tools is NOT REQUIRED.

BLOCK 4: INSPECTING IPS



Target an IP Range

Practical application: *You need to know the assigned IP ranges of any internet domain*

Once we know the DNS domains and subdomains of any organization, we can search its assigned public IP ranges to know which IPs could be used by this organization machines. This enables us to inspect for machines or services exposed belonging to this organization. For .com sites, we can check https://whois.arin.net. To do this, we can input the organization name (e. g. "Microsoft", "Yahoo") in the *SEARCH WHOIS-RWS* textbox and scroll down to see the IP ranges this organization have assigned.

For .es domains, we can use www.nic.es for the same purpose. Clicking in the details of a domain (*view data*) shows us the DNS servers that serve the different networks assigned to the organization, from which we can easily extract IPs and IP ranges.

Expected Results: This activity will be completed when you can check the IP range assigned to uniovi.es or any .com organization you want.

Global scans with Zmap

Practical application: *You can perform internet-wide scans for services in a concrete port*

`zmap` (https://zmap.io/) is a fast single-packet network scanner optimized for Internet-wide network scans. On a computer with a gigabit connection, *ZMap* can scan the **entire public IPv4 address space in under 45 minutes**. Faster connections mean less scan time. You can install `zmap` on *Kali* and *Ubuntu* with `sudo apt install zmap`.

`Zmap` will not be used here to scan all the Internet, as it is overkill and have no real purpose in this lab. However, it can be used to find services listening on a concrete port fast and easy in a known network or IP range (that may be obtained from the previous activity). For example, if we know that the network `88.151.16.0/24` belongs to the domain *asturias.es* from the previous activity, it is quite easy to locate active web servers in this network (HTTP, port 80) with `zmap -p80 88.151.16.0/24`. As the output may be large, it is recommended to store it to a file with the `-o` option.

The following cheatsheet provide an overview of all `zmap` options. The **Lab 2 infrastructure** provide a *Kali Linux* with `zmap` installed.

Expected Results: This activity will be completed when you can check the active web servers (or any other service) on a network, for example asturias.es, or uniovi.es (by default Zmap does not work on local networks).

Zmap 2.1.1 Cheatsheet (ingenieriainformatica.uniovi.es)

Fast internet-wide scanner

https://zmap.io/

GENERAL USAGE

```
zmap [OPTION]... [SUBNETS]…
```

```
operario@kali:~$ sudo zmap -p 80 88.151.16.0/24 -o asturias.txt
Oct 29 18:33:29.149 [WARN] blacklist: ZMap is currently using the default blacklist located at /etc/zmap/blacklist.
conf. By default, this blacklist excludes locally scoped networks (e.g. 10.0.0.0/8, 127.0.0.1/8, and 192.168.0.0/16
). If you are trying to scan local networks, you can change the default blacklist by editing the default ZMap confi
guration at /etc/zmap/zmap.conf.
Oct 29 18:33:29.153 [INFO] zmap: output module: csv
 0:00 0%; send: 10 0 p/s (261 p/s avg); recv: 0 0 p/s (0 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.00%
 0:01 13%; send: 256 done (6.23 Kp/s avg); recv: 17 16 p/s (16 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 6.64%
 0:02 26%; send: 256 done (6.23 Kp/s avg); recv: 76 56 p/s (36 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 29.69%
 0:03 38%; send: 256 done (6.23 Kp/s avg); recv: 76 0 p/s (24 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 29.69%
 0:04 51%; send: 256 done (6.23 Kp/s avg); recv: 76 0 p/s (18 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 29.69%
 0:05 64% (3s left); send: 256 done (6.23 Kp/s avg); recv: 76 0 p/s (14 p/s avg); drops: 0 p/s (0 p/s avg); hitrate
: 29.69%
 0:06 77% (2s left); send: 256 done (6.23 Kp/s avg); recv: 76 0 p/s (12 p/s avg); drops: 0 p/s (0 p/s avg); hitrate
: 29.69%
 0:07 89% (1s left); send: 256 done (6.23 Kp/s avg); recv: 76 0 p/s (10 p/s avg); drops: 0 p/s (0 p/s avg); hitrate
: 29.69%
Oct 29 18:33:37.339 [INFO] zmap: completed
```

NOTES

Probe-module (tcp_synscan): Probe module that sends a TCP SYN packet to a specific port. Possible classifications are: synack and rst. A SYN-ACK packet is considered a success and a reset packet is considered a failed response.
Output-module (csv): By default, ZMap prints out unique, successfulIP addresses (e.g., SYN-ACK from a TCP SYN scan) in ASCII form (e.g., 192.168.1.5) to stdout or the specified output file. Internally this is handled by the "csv" output module and is
equivalent to running zmap --output-module=csv --output-fields=saddr --output-filter="success = 1 && repeat = 0".

OPTIONS

BASIC ARGUMENTS

-b, --blacklist-file=path: File of subnets to exclude, in CIDR notation, e.g. 192.168.0.0/16
-o, --output-file=name: Output file
-p, --target-port=port: port number to scan (for TCP and UDP scans)
-w, --whitelist-file=path: File of subnets to constrain scan to, in CIDR notation, e.g. 192.168.0.0/16

SCAN OPTIONS

--retries=n: Max number of times to try to send packet if send fails (default=`10')
--shard=n: Set which shard this scan is (0 indexed) (default=`0')
--shards=N: Set the total number of shards (default=`1')
-B, --bandwidth=bps: Set send rate in bits/second (supports suffixes G, M and K)
-c, --cooldown-time=secs: How long to continue receiving after sending last probe (default=`8')
-d, --dryrun: Don't actually send packets
-e, --seed=n: Seed used to select address permutation
-N, --max-results=n: Cap number of results to return
-n, --max-targets=n: Cap number of targets to probe (as a number or a percentage of the address space)
-P, --probes=n: Number of probes to send to each IP (default=`1')
-r, --rate=pps: Set send rate in packets/sec
-t, --max-runtime=ses: Cap length of time for sending packets

ADDITIONAL OPTIONS

--cores=STRING: Comma-separated list of cores to pin to
--ignore-invalid-hosts: Ignore invalid hosts in whitelist/blacklist file
--max-sendto-failures=n: Maximum NIC sendto failures before scan is aborted (default=`-1')
--min-hitrate=n: Minimum hitrate that scan can hit before scan is aborted (default=`0.0')
-C, --config=filename: Read a configuration file, which can specify any of these options (default=`/etc/zmap/zmap.conf')
-h, --help: Print help and exit
-T, --sender-threads=n: Threads used to send packets (default=`1')
-V, --version: Print version and exit

EXAMPLES

zmap -p 80 (scan the Internet for hosts on tcp/80 and output to stdout)
zmap -N 5 -B 10M -p 80 (find 5 HTTP servers, scanning at 10 Mb/s)
zmap -p 80 10.0.0.0/8 192.168.0.0/16 -o (scan both subnets on tcp/80)
zmap -p 80 1.2.3.4 10.0.0.3 (scan 1.2.3.4, 10.0.0.3 on tcp/80)

NETWORK OPTIONS

--source-mac=addr: Source MAC address
-G, --gateway-mac=addr: Specify gateway MAC address
-i, --interface=name: Specify network interface to use
-S, --source-ip=ip|range: Source address(es) for scan packets
-s, --source-port=port|range: Source port(s) for scan packets
-X, --vpn: Sends IP packets instead of Ethernet (for VPNs)

PROBE MODULES

--list-probe-modules: List available probe modules
--probe-args=args: Arguments to pass to probe module
-M, --probe-module=name: Select probe module (default=`tcp_synscan')

DATA OUTPUT

--list-output-fields: List all fields that can be output by selected probe module
--list-output-modules: List available output modules
--output-args=args: Arguments to pass to output module
--output-filter=filter: Specify a filter over the response fields to limit what responses get sent to the output module
-f, --output-fields=fields: Fields that should be output in result set
-O, --output-module=name: Select output module (default=`default')

LOGGING AND METADATA

--disable-syslog: Disables logging messages to syslog
--notes=notes: Inject user-specified notes into scan metadata
--user-metadata=json: Inject user-specified JSON metadata into scan metadata
-L, --log-directory=directory: Write log entries to a timestamped file in this directory
-l, --log-file=name: Write log entries to file
-m, --metadata-file=name: Output file for scan metadata (JSON)
-q, --quiet: Do not print status updates
-u, --status-updates-file=name: Write scan progress updates to CSV file
-v, --verbosity=n: Level of log detail (0-5) (default=`3')

LAN scans with Nmap

Practical application: *You need to inspect a Local Area Network for "alive" machines*

The purpose of this activity is to locate all active machines in a LAN network (not from Internet!). To do that, we provide it the LAN network of the **Lab 2 infrastructure**. This extends the lab 1 activity, that only determines if a server is "alive". Use a *list scan* to see if the DNS names of the hosts are also provided and this cheatsheet as reference.

Nmap 7.80 Cheatsheet series (ingenieriainformatica.uniovi.es)

Part 1: Reconnaissance (Basic)

https://nmap.org/

GENERAL USAGE

`nmap [Scan Type(s)] [Options] {target specification}`

NOTES

Target specifications can be host names, IP addresses, ranges, networks, etc. (scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254)

Use these options to Locate "alive machines" (sometimes only returns that, sometimes they also return some port / service information)

```
operario@kali:~$ nmap -sn 192.168.20.10
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-21 18:38 CEST
Nmap scan report for 192.168.20.10
Host is up (0.00085s latency).
Nmap done: 1 IP address (1 host up) scanned in 13.01 seconds
operario@kali:~$ sudo nmap -PS22,80 192.168.20.10
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-21 18:42 CEST
Nmap scan report for 192.168.20.10
Host is up (0.00012s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:67:7A:EF (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.30 seconds
```

TARGET SPECIFICATION

- **-iL <inputfilename>**: Input from from file a list of hosts/networks
- **-iR <num hosts>**: Choose random targets
- **--exclude <host1[,host2][,host3],...>**: Exclude hosts/networks
- **--excludefile <exclude_file>**: Exclude list from file

RECONOISSANCE EXAMPLES

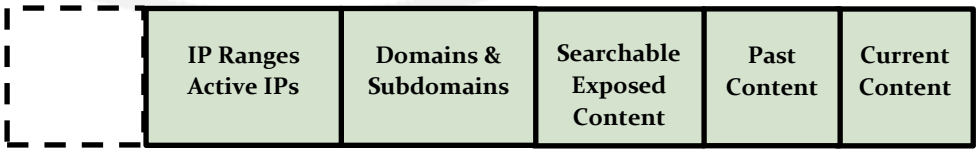
- *nmap -sn scanme.nmap.org*
- *sudo nmap -PS22,80 scanme.nmap.org*
- *sudo nmap --traceroute scanme.nmap.org*

HOST DISCOVERY OPTIONS (WAYS TO CHECK "ALIVE" MACHINES)

- **--dns-servers <serv1[,serv2],...>**: Specify custom DNS servers
- **-n/-R**: Never do DNS resolution/Always resolve [default: sometimes]
- **-PE/PP/PM**: ICMP echo, timestamp, and netmask request discovery probes
- **-Pn**: Treat all provided hosts as online -- skip host discovery
- **-PO[protocol list]**: IP Protocol Ping
- **-PS/PA/PU/PY[portlist]**: TCP SYN/ACK, UDP or SCTP discovery to given ports
- **-sL**: List Scan - simply list targets to scan
- **-sn**: Ping Scan - disable port scan
- **--system-dns**: Use OS's DNS resolver
- **--traceroute**: Trace hop path to each host

Expected Results: This activity will be completed when you are able to locate active hosts in the mentioned LAN and its DNS names.

BLOCK 5. LET'S MAKE IT PERSONAL: PRIVACY, BROWSING SECURITY AND SOCIAL NETWORKS



Blacklight: Website privacy inspector

Practical application: *You need to know how any internet website uses your navigation data for its own benefit*

Blacklight (https://themarkup.org/blacklight) scans and reveals the specific user-tracking technologies of any site. This means that it reveals who is getting your data.

Expected Results: This activity will be completed when you are able to use *Blacklight* to inspect how any site you choose is collecting your data and analyze the results.

Create a secure and truly private browsing environment on your VM

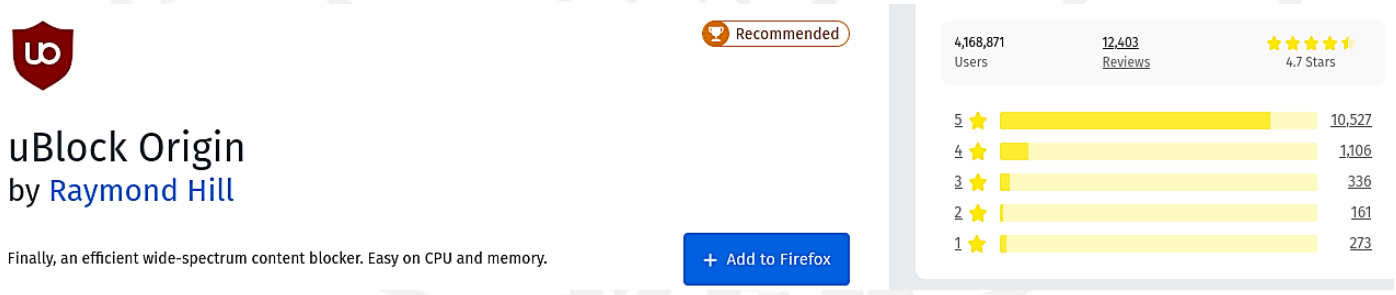
Practical application: *You can create a safer internet browser for your daily life*

*(**NOTE**: Secure does not mean private. Your connection attempts will be still logged by the destination servers, and your IP will be still revealed to them. To achieve a certain degree of browsing privacy you need other tools (like ToR), but these are beyond the scope of the course)*

The purpose of this section is to fight against the techniques you have discovered with *Blacklight*, as the amount of data collected by a website may be troublesome in certain usage scenarios. Apart from that, it is very handy to have a more secure browsing environment just in case you accidentally browse a malicious or trojanized web page, pages with malicious ads, and other web threats. This way, you can browse on a safer way if you use the following browser plugins. Combining it with a "browsing VM", you obtain a substantially more secure browsing environment. We will use *Firefox* as a browser example, but the same plugins are available in other browsers too (also on mobile versions). This also connects with the introduction topic of the theory.

To deal with browser extensions, you need to consider the following things:

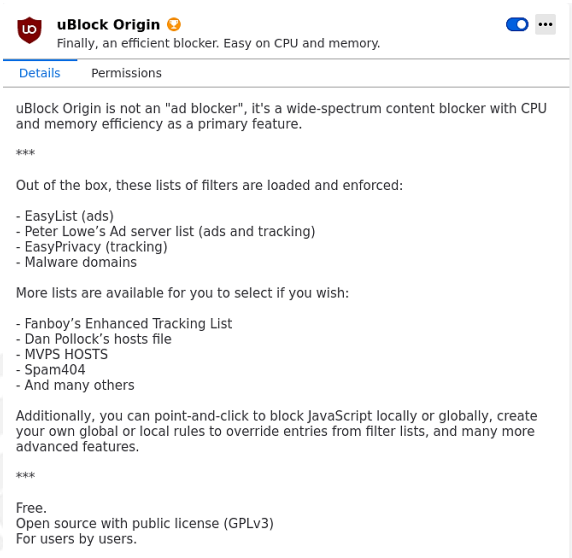
- **There are malicious extensions**: to prevent problems, install only the *Recommended* ones (inspected by the browser vendor), with high score, and lots of user recommendations.



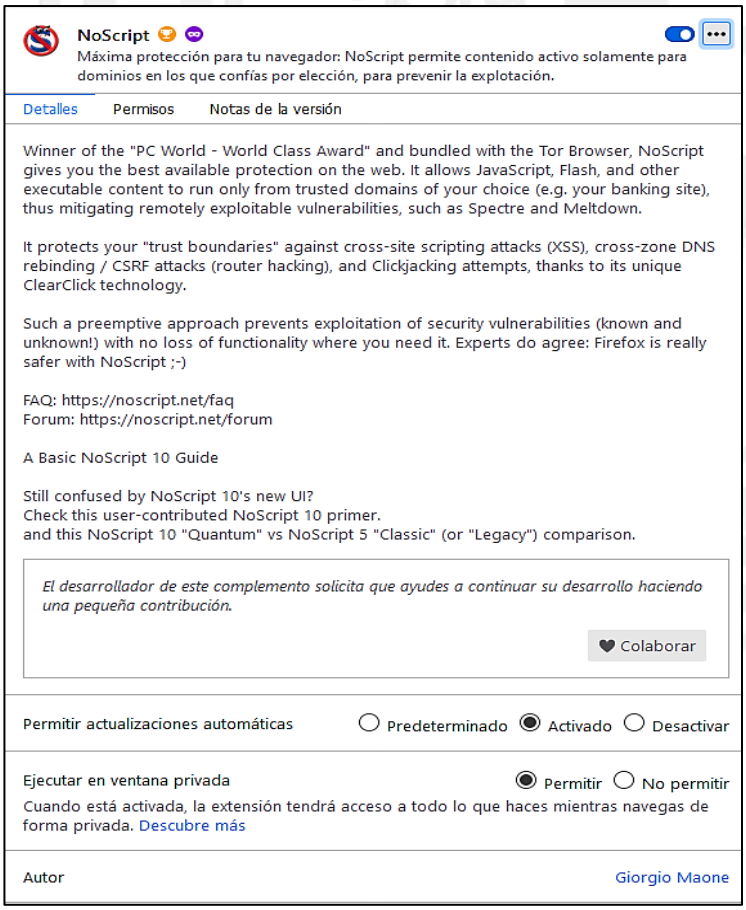
- Be wary of extensions asking for too many or unreasonable permissions.
- Extensions are normally installed using a specific browser option (*Add-ons*) or a market (*Google Chrome Store*) not externally.

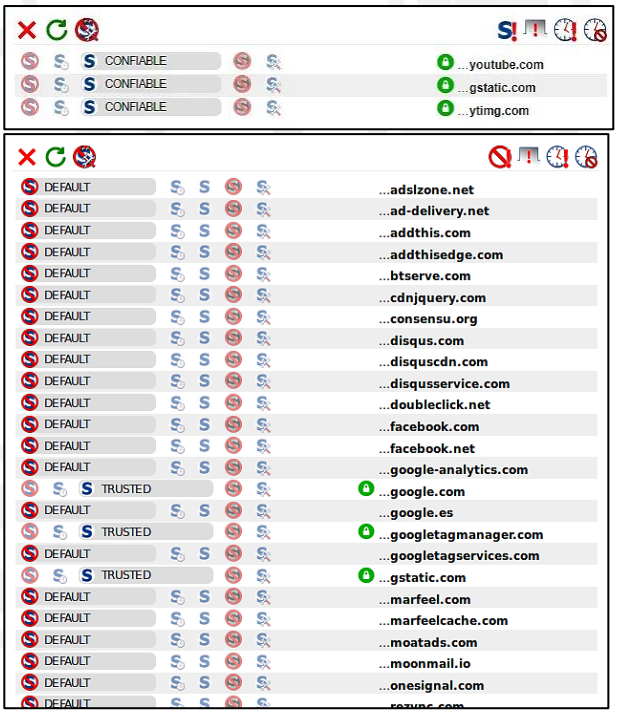
Once this is said, it is recommended to install extensions that deal with:

- **Ads**: as they can be obtrusive and/or malicious, even being a legal income source for page owners. *uBlock Origin* is the recommended add-on.



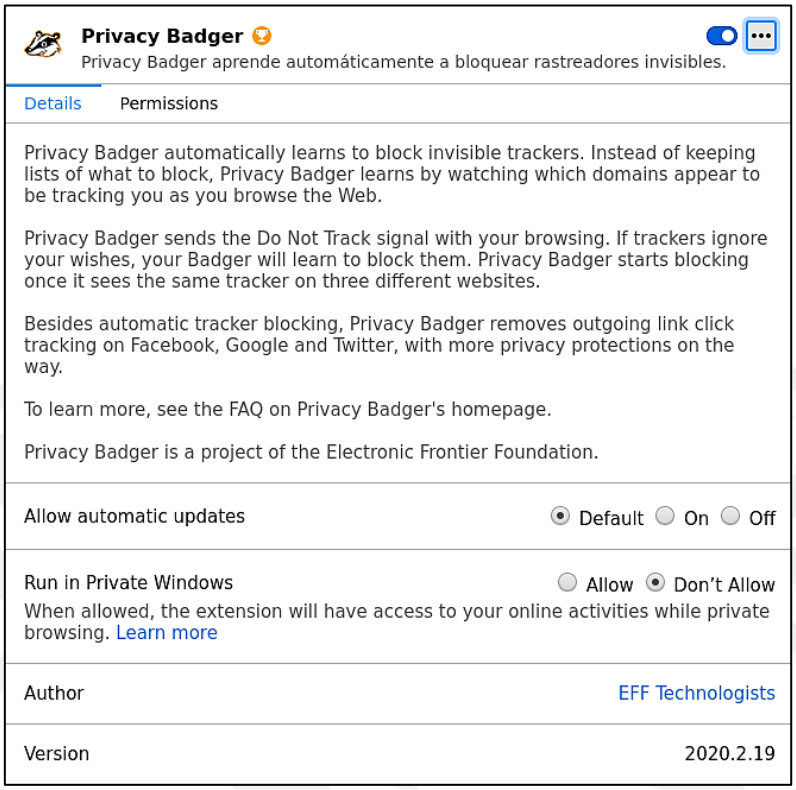
- **Script blockers**: Using *JavaScript* in webpages is quite common and legitimate, but sometimes scripting is also used for nefarious purposes. To prevent this, the *NoScript* extension is recommended. It not only blocks scripts, but also identifies malicious scripting usages in web pages. However, using this requires "training", as it initially blocks most scripts on a web page rendering it unusable (or making their contents invisible) most of the time. However, you can unlock script sources until the page is usable enough for your purposes. Do not unlock obvious ads or malicious domains! Normally you should start with script sources that are on the same DNS name than the page you are viewing. This kind of training is stored between sessions, so you only need to do this once per webpage.





- **Privacy**: *Privacy Badger* is the recommended extension that prevents webpages to track your browsing history and create a profile of your browsing behaviors. It also blocks popular social networks and

website tracking attempts (*Facebook*, *Google*, *Twitter*...) and implement other measures protecting user privacy.



Expected Results: This activity will be completed when your secure browser is operational and able to browse to webpages with all the mentioned plugins active.

Have I been pwnd?

Practical application: *You need to know if the password of any of your accounts in an internet service has been potentially leaked through one of the many data leaks that are produced nowadays*

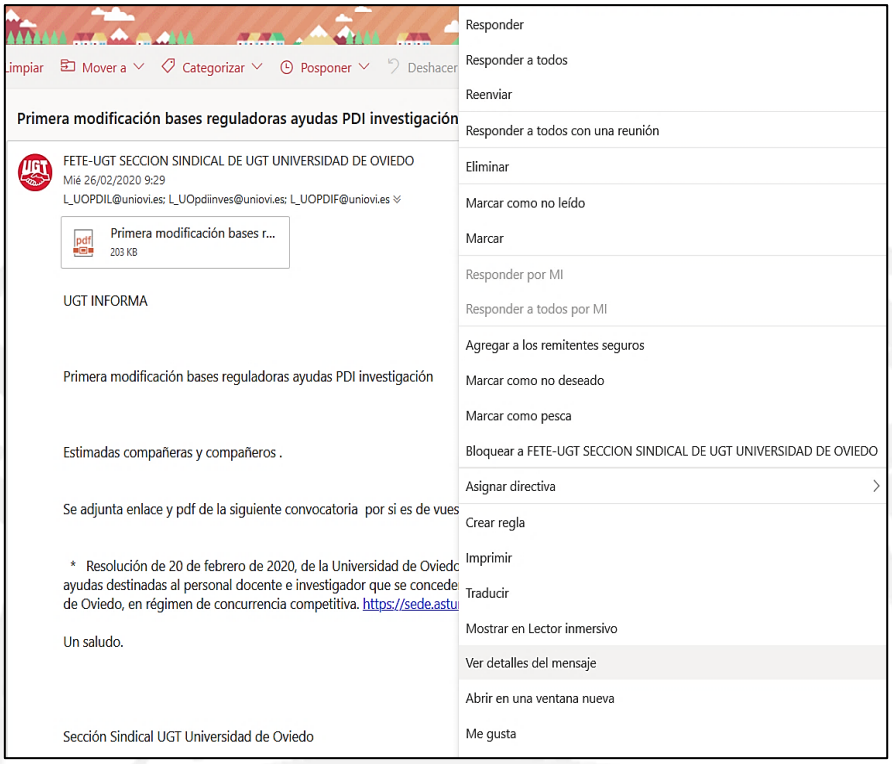
Have I been pwnd? (https://haveibeenpwned.com/) is a website that stores and queries user data leaks. It accepts email account names, and it tells if this account name has been leaked in any of the known user databases that have been stolen. Therefore, if there was an email account inside one of the multiple stolen user databases it contains, it tells you the site whose users has been stolen, when, and other details about the data leak. Due to the probability of using the same password and email in multiple sites, it is especially important to verify this to prevent multiple intrusions.

Expected Results: This activity will be completed when you inspect any email account you want to know if it is part of a known data leak.

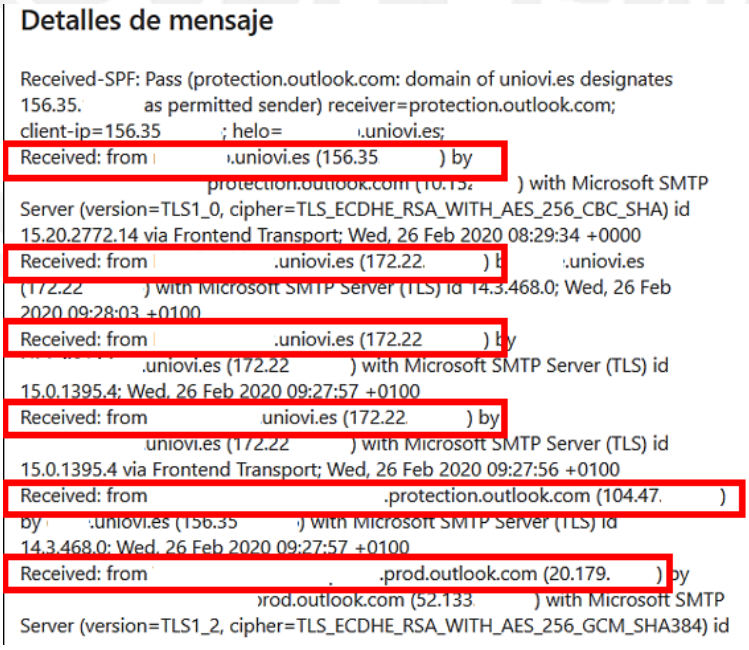
Geolocating an email sender (or any IP)

Practical application: *You need to geolocate any IP, including those that are part of email headers*

Geolocating an IP address is possible, although the precision we can obtain (without having special permissions given by a judge) is limited to district code areas, not particular coordinates. Any IP can be geolocated using online services like https://www.iplocation.net/. This gives us coordinates that can be later geolocated in popular applications like *Google Earth*, *Google Maps*, or similar ones. A special application of this can be used with emails. Some email providers preserve all email headers in the messages they manage. You can see email headers using advanced email client options. In the Uniovi webmail the option is this one:



There are lots of headers with different meanings, but they are not important for this purpose. One special header is `Received: from`, indicating the email servers that the email has passed through. Typically, an email "rebounds" through multiple email servers until reaching its destination, and you can track these servers using the IPs on these headers (if they are not removed).



However, the interesting header is not this one, as you will only obtain geolocation of server datacenters and not the original email source. The key header is `x-originating-ip` that again, depending on the email provider,

could have been removed. However, if present, it will provide us with the approximate coordinates (district level) of the email sender. Uniovi webmail usually do not remove this header from emails.

```
Authentication-Results-Original: uniovi.es; dkim=none (message not signed)
header.d=none;uniovi.es; dmarc=none action=none header.from=uniovi.es;
x-originating-ip: [156.35.      ]
x-ms-publictraffictype: Email
```

Expected Results: This activity will be completed when you are able to know the geolocation of any IP and, more concretely, of the IP of the author of an email you received, if this information is provided in the email headers.

Twint: Investigating Twitter

Practical application: *You need to investigate someone's behavior in Twitter*

Twint (https://github.com/twintproject/twint) is included in this laboratory as an example of the multiple tools that can be used to investigate people activity in social networks. It is used against *Twitter* accounts, but similar tools exist for other social networks too. The main advantage of *Twint* is that it uses web scraping techniques, so a *Twitter API Key* (and hence an account) is not necessary, like other similar tools like *Tinfoleak*. Anybody can investigate *Twitter* without an account in this social network!

To install *Twint* the following procedure is recommended, so you have the last possible version:

- Install the `pip3` package:

```
sudo apt-get install python3-pip
```

- Install *Twint* from the source (please for this lab read the next paragraph first!)

```
git clone --depth=1 https://github.com/twintproject/twint.git
cd twint
pip3 install . -r requirements.txt
```

However, the *Twitter* interface is known to change often, and this makes `twint` to fail frequently until it adapts to these changes. Therefore, **we recommend you to not to download it from the source** and use the version we provide you in the `ssi_labs\lab_sessions\lab_02\twint` folder. You just must access the `twint` folder of the lab2 in your VM and install the requirements with `pip3` as indicated.

This will install `twint` in a local folder of your current user account. If the user account is called `operario`, you can call `twint` like this: `/home/operario/.local/bin/twint --csv -o sanchez.csv --limit 20 -s "Pedro Sanchez" --verified`

This collects the first 20 tweets that contains "Pedro Sanchez" but tweeted by verified accounts. There are lots of different examples that you can find in the official webpage and in the following cheatsheet.

NOTE: If you really like to know how information is managed, processed, and interpreted in Internet, and what you can really do with this information, you should really consider taking the optional course **Web Information Systems** (https://twitter.com/SIW_ovi). It will give you a much more detailed background on these techniques!

Twint 2.1.21 Cheatsheet (ingenieriainformatica.uniovi.es)

An Advanced Twitter Scraping Tool

https://github.com/twintproject/twint

Install the latest version:

```
git clone --depth=1 https://github.com/twintproject/twint.git
cd twint
pip3 install . -r requirements.txt
```

GENERAL USAGE

```
python3 twint [options]
```

NOTES

You don't need a Twitter API Key to use this tool. If is recommended to install the latest version to avoid problems with scraping the Twitter UI

OPTIONS

Typical arguments

- **-g GEO, --geo GEO**: Search for geocoded Tweets.
- **-h, --help**: show this help message and exit
- **-l LANG, --lang LANG**: Search for Tweets in a specific language.
- **--location**: Show user's location (Experimental).
- **--near NEAR**: Near a specified city.
- **-s SEARCH, --search SEARCH**: Search for Tweets containing this word or phrase.
- **-u USERNAME, --username USERNAME**: User's Tweets you want to scrape.

EXAMPLES

To get only follower usernames/following usernames
twint -u username --followers
twint -u username --following
To get user info of followers/following users
twint -u username --followers --user-full
twint -u username --following --user-full
Userlist: To get only user info of user
twint -u username --user-full
To get user info of users from a userlist
twint --userlist inputlist --user-full
Tweet translation (experimental): To get 100 english tweets and translate them to italian
twint -u noneprivacy --csv --output none.csv --lang en --translate --translate-dest it --limit 100

Miscellaneous examples
twint -u username - Scrape all the Tweets of a user (doesn't include retweets but includes replies).
twint -u username -s pineapple - Scrape all Tweets from the user's timeline containing pineapple.
twint -s pineapple - Collect every Tweet containing pineapple from everyone's Tweets.
twint -u username --year 2014 - Collect Tweets that were tweeted before 2014.
twint -u username --since "2015-12-20 20:30:15" - Collect Tweets that were tweeted since 2015-12-20 20:30:15.
twint -u username --since 2015-12-20 - Collect Tweets that were tweeted since 2015-12-20 00:00:00.
twint -u username -o file.txt - Scrape Tweets and save to file.txt.
twint -u username -o file.csv --csv - Scrape Tweets and save as a csv file.
twint -u username --email --phone - Show Tweets that might have phone numbers or email addresses.
twint -s "Donald Trump" --verified - Display Tweets by verified users that Tweeted about Donald Trump.
twint -g="48.880048,2.385939,1km" -o file.csv --csv - Scrape Tweets from a radius of 1km around a place in Paris and export them to a csv file.
twint -u username -es localhost:9200 - Output Tweets to Elasticsearch
twint -u username -o file.json --json - Scrape Tweets and save as a json file.
twint -u username --database tweets.db - Save Tweets to a SQLite database.
twint -u username --followers - Scrape a Twitter user's followers.
twint -u username --following - Scrape who a Twitter user follows.
twint -u username --favorites - Collect all the Tweets a user has favorited (gathers ~3200 tweet).
twint -u username --following --user-full - Collect full user information a person follows
twint -u username --timeline - Use an effective method to gather Tweets from a user's profile (Gathers ~3200 Tweets, including retweets & replies).
twint -u username --retweets - Use a quick method to gather the last 900 Tweets (that includes retweets) from a user's profile.
twint -u username --resume resume_file.txt - Resume a search starting from the last saved scroll-id.

Other Options

- **--backoff-exponent BACKOFF_EXPONENT**: Specify a exponent for the polynomial backoff in case of errors.
- **-ho, --hide-output**: Hide output, no tweets will be displayed.
- **--min-wait-time MIN_WAIT_TIME**: specifiy a minimum wait time in case of scraping limit error. This value will be adjusted by twint if the value provided does not satisfy the limits constraints
- **-sc, --skip-certs**: Skip certs verification, useful for SSC.

Translation

- **--translate**: Get tweets translated by Google Translate.
- **--translate-dest TRANSLATE_DEST**: Translate tweet to language (ISO2).

Tweet filtering

- **--email**: Filter Tweets that might have email addresses
- **-fr, --filter-retweets**: Exclude retweets from the results.
- **--images**: Display only Tweets with images.
- **--links LINKS**: Include or exclude tweets containing one o more links. If not specified you will get both tweets that might contain links or not.
- **--media**: Display Tweets with only images or videos.
- **--members-list MEMBERS_LIST**: Filter the tweets sent by users in a given list.
- **--min-likes MIN_LIKES**: Filter the tweets by minimum number of likes.
- **--min-replies MIN_REPLIES**: Filter the tweets by minimum number of replies.
- **--min-retweets MIN_RETWEETS**: Filter the tweets by minimum number of retweets.
- **--phone**: Filter Tweets that might have phone numbers
- **--since DATE**: Filter Tweets sent since date (Example: "2017-12-27 20:30:15" or 2017-12-27).
- **--source SOURCE**: Filter the tweets for specific source client.
- **--until DATE**: Filter Tweets sent until date (Example: "2017-12-27 20:30:15" or 2017-12-27).
- **--verified**: Display Tweets only from verified users (Use with -s).
- **--year YEAR**: Filter Tweets before specified year.

Search modifiers

- **--all USERNAME**: Search all Tweets associated with a user.
- **-cq CUSTOM_QUERY, --custom-query CUSTOM_QUERY**: Custom search query.
- **--favorites**: Scrape Tweets a user has liked.
- **--followers**: Scrape a person's followers.
- **--following**: Scrape a person's follows
- **-nr, --native-retweets**: Filter the results for retweets only.
- **--profile-full**: Slow, but effective method of collecting a user's Tweets and RT.
- **-pt, --popular-tweets**: Scrape popular tweets instead of recent ones.
- **--replies**: Display replies to a subject.
- **--resume TWEET_ID**: Resume from Tweet ID.
- **--retweets**: Include user's Retweets (Warning: limited).
- **--to USERNAME**: Search Tweets to a user.
- **--user-full**: Collect all user information (Use with followers or following only).
- **--userlist USERLIST**: Userlist from list or file.
- **--videos**: Display only Tweets with videos.

Output Options

- **--cashtags**: Output cashtags in seperate column.
- **--count**: Display number of Tweets scraped at the end of session.
- **--csv**: Write as .csv file.
- **-db DATABASE, --database DATABASE**: Store Tweets in a sqlite3 database.
- **--debug**: Store information in debug logs
- **-es ELASTICSEARCH, --elasticsearch ELASTICSEARCH**: Index to Elasticsearch.
- **--essid [ESSID]**: Elasticsearch Session ID, use this to differentiate scraping sessions.
- **--format FORMAT**: Custom output format (See wiki for details).
- **--hashtags**: Output hashtags in seperate column.
- **-if [INDEX_FOLLOW], --index-follow [INDEX_FOLLOW]**: Custom Elasticsearch Index name for Follows.
- **-it [INDEX_TWEETS], --index-tweets [INDEX_TWEETS]**: Custom Elasticsearch Index name for Tweets.
- **-iu [INDEX_USERS], --index-users [INDEX_USERS]**: Custom Elasticsearch Index name for Users.
- **--json**: Write as .json file
- **--limit LIMIT**: Number of Tweets to pull (Increments of 20).
- **-o OUTPUT, --output OUTPUT**: Save output to a file.
- **--pandas-type [PANDAS_TYPE]**: Specify HDF5 or Pickle (HDF5 as default)
- **-pc PANDAS_CLEAN, --pandas-clean PANDAS_CLEAN**: Automatically clean Pandas dataframe at every scrape.
- **--stats**: Show number of replies, retweets, and likes.
- **--store-pandas STORE_PANDAS**: Save Tweets in a DataFrame (Pandas) file.
- **--userid USERID**: Twitter user id.

HTTP Proxies

- **--proxy-host PROXY_HOST**: Proxy hostname or IP.
- **--proxy-port PROXY_PORT**: The port of the proxy server.
- **--proxy-type PROXY_TYPE**: Socks5, HTTP, etc.

Expected Results: This activity will be completed when you familiarize with *Twint* and are able to extract from *Twitter* any information you want, understanding the security implications this may have.

BADGES AND SELF-ASSESSMENT

NOTE: You have a version of this badge table in an editable format available in the *Virtual Campus*. You can use this file to easily create a document in the format you want and take extended notes of your activities to create a log of what you did. Remember that this material is key to keep track of your self-evaluation.

Badge Level	Unlocked when	Unlocked?
	You can find and analyze any `robots.txt` file	
	Answer this question: *What do you think it will be the security problems that a badly written* `robots.txt` *could provoke?*	
	You can analyze the metadata of any individual file of an appropriate type you find. You can answer this question: *Why metadata can be a security threat?*	
	You can analyze a single URL or IP using *Shodan* and interpret the results. You can also answer this question: *what part of the results links with something that we saw in the previous lab, and may indicate a severe problem with the webpage?*	
	You can search all the historic URLs of a domain.	
	You can answer this question: *what security problems may happen if we have an historic archive of the different contents of a* `robots.txt` *file?*	
	You can answer this question: *why subdomains are interesting from a security point of view? What could happen if a subdomain (and its contents) has been forgotten for a long time?*	
	You can analyze how a website collect user data and deals with privacy using *Blacklight*.	
	You can check if a password has been potentially compromised in a data leak and answer this question: *what should you do if you find your email in one of these documented intrusions?*	
	You can use *Censys* to analyze the hosts of a network and interpret the results, both from the network and from any individual hosts. You can also answer this question: *are you able to find services running in the hosts? How can you find if some are vulnerable?*	
	You know how to operate the *Google Hacking Database* against a concrete target.	
	Answer this question: *What kind of security problem do you think finding past content could suppose to a website?*	
	You can locate subdomains of any domain using one of the provided tools and analyze the domain information they provide.	
	You can locate the IP networks of any *.com* or *.es* domain.	
	You can locate active services in public network ranges using `zmap`	
	You can locate active hosts in LAN networks.	

	You can setup a private and secure browsing environment	
	You can geolocate an IP (and, particularly, an email sender). You can answer this question: *after seeing the IP location precision you have, what is, in your opinion, the usefulness of IP geolocation?*	
	You can answer this question: *What happens if instead of an email of yours, you decide to investigate emails from other people? What happens if the email is reported as compromised?*	
	You can answer this question: *If I am able to find any image or document that is (or was) in any website, what can we do with these files that may reveal me more information? What happens if I can find the history of versions of any of these files? What kind of information it may provide?*	
	You can answer this question: *If I am able to find any version of any web page of that once was part of a website, what can we do with these files that may reveal me more information? Do you think this could allow locating potential vulnerabilities?*	
	You can answer this question: Imagine that I leave by mistake a compromising comment in a webpage (indicating a product name and/or version, part of the server code in a comment...). *What should we do to really eliminate the security problem this could suppose?*	
	You can investigate *Twitter* accounts looking for different information types via *Twint*. You also can answer this question: *What security implications may have that you are able to obtain past tweets from any user, even without a Twitter account?*	
	You can answer this question: *What do you think people in social networks do to locate and display embarrassing or compromising messages that users/famous people/public figures have tweeted in the past?*	
	You can answer this question: It is a fact that UDP scans are slow, especially in WANs / Internet. It is also known that `nmap` usually scans by default the topmost used ports worldwide (a finite list of known ports). Knowing this, if you want to create an UDP "hidden" service, which is difficult to locate, *what would you do?*	
	**I have seen things you people would not believe**: Use multiple enumeration and OSINT techniques to extract lots of information about machines, companies, and persons that you can use to analyze them.	