

## MÓDULO: VULNERABILIDADES WEB COMUNES

Investigar sobre las herramientas de reconocimiento web pasivo disponibles, realizar la instalación de las mismas en la máquina de ataque y realizar una prueba de funcionamiento de cada una.

Ejemplo de alguna de estas herramientas:

- Hardcidr
  - Amass
  - Metadata
  - Pymeta
  - Github
  - Trufflehog
  - Reposcanner
  - Linkedint
-

# UTILIZACIÓN DE LAS HERRAMIENTAS EN LA MÁQUINA KALI

En este ejercicio utilizaremos las distintas herramientas mencionadas en el anterior párrafo.

Para ello iniciaremos nuestra máquina KALI y la máquina víctima para hacer los escaneos sin necesidad de atacar a otra máquina fuera de nuestra red y causar problemas externos.

## HARDCIDR.

En primer lugar, utilizaremos la herramienta “Hardcidr”, para poder utilizarla nos vamos a su [documentación](https://github.com/trustedsec/hardcidr.gitsass).

```
(kali㉿ kali)-[~]  
$ git clone https://github.com/trustedsec/hardcidr.gitsass|
```

Hacemos un git clone y la url de la herramienta de Github.

Seguidamente iniciamos nuestra máquina víctima y obtenemos la ip de la máquina con un ifconfig.

```
ubuntu@ubuntu2004:~/Desktop$ ifconfig  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  
    inet 192.168.1.23  netmask 255.255.255.0  b  
    inet6 fe80::c6e:61a8:833f:9ac0  prefixlen 6  
    ether 08:00:27:9f:2f:af  txqueuelen 1000  (C  
    RX packets 1039  bytes 1347620 (1.3 MB)
```

Se trata de un script Bash para descubrir los bloques de red, o rangos, (en notación CIDR) propiedad de la organización objetivo durante la fase de recopilación de inteligencia de una prueba de penetración. Esta información es mantenida por los cinco Registros Regionales de Internet (RIR):

ARIN (Norteamérica)

RIPE (Europa/Asia/Oriente Medio)

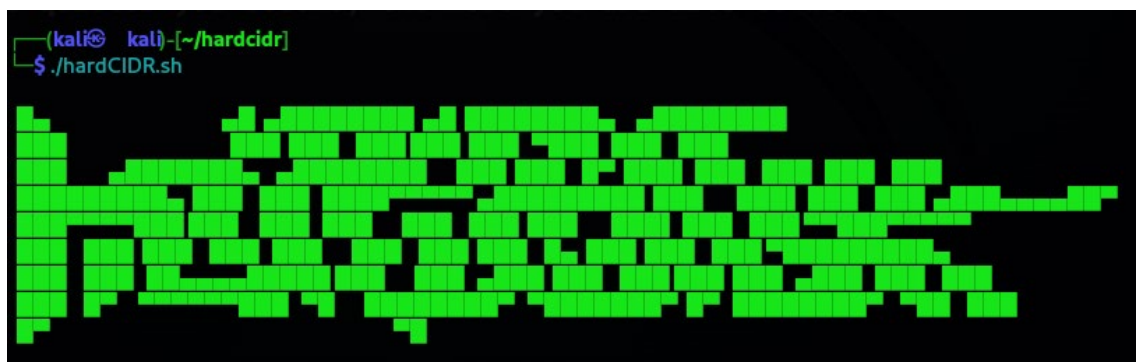
APNIC (Asia/Pacífico)

LACNIC (América Latina)

AfriNIC (África)

#### UTILIZACIÓN.

Iniciamos el programa con el comando `./hardCIDR.sh`



```
(kali) [~]/hardcidr
$ ./hardCIDR.sh
```

The output of the script is a large, dense block of green text on a black background, representing a list of IP ranges.

El script sin opciones especificadas consultará ARIN y un grupo de servidores de rutas BGP. El servidor de rutas se selecciona aleatoriamente en tiempo de ejecución. La opción `-h` muestra la ayuda:

### A tool for locating target Organization CIDRs

```
usage: ./hardCIDR.sh -r -l -f -p -h -u
-r = Query [R]IPE NCC (Europe/Middle East/Central Asia)
-l = Query [L]ACNIC (Latin America & Caribbean)
-f = Query A[f]riNIC (Africa)
-p = Query A[P]NIC (Asia/Pacific)
-u = Update LACNIC data file <- dont run with other options
-h = help
```

Introduzca el nombre de la organización, el dominio de correo electrónico y si se utilizan códigos de país como parte del correo electrónico. Si responde S a los códigos de país, se le preguntará si van antes del nombre de dominio o se añaden al TLD. Se creará un directorio para los archivos de salida en /tmp/. Si el directorio ya existe, se le preguntará si desea sobrescribirlo. Si la respuesta es N, se añadirá una marca de tiempo al nombre del directorio.

```
Enter Client Name: Microsoft
Enter Client Email Primary Domain (domain.tld): microsoft.com
Does Microsoft use country codes in email addresses? Y or N: N
[*] Enumerating CIDRs for Microsoft Org Handles via ARIN
[-] Found Org Handles for Microsoft
|
```

El script consulta cada RIR, así como un servidor de rutas BGP, preguntando a lo largo del proceso si se han localizado registros. Al finalizar, se generarán tres archivos: un CSV basado en Org Handle, un CSV basado en PoC Handle, y un archivo delimitado por líneas de todas las rutas localizadas en notación CIDR.

```

[-]Found ARIN email Records for microsoft.com

[*] Enumerating CIDsRs for Microsoft BGP Prefixes via Route Server Query
[-]Found ASN Records for Microsoft

[*] Enumerating CIDsRs for Microsoft Org Names via RIPE NCC
[-]Found RIPE NCC Records for Microsoft

[*] Enumerating CIDsRs for Microsoft Org Names via APNIC
[-]Found APNIC Records for Microsoft

[*] Enumerating CIDsRs for Microsoft Org Owner via LACNIC
[-]Found LACNIC Records for Microsoft

[*] Enumerating CIDsRs for Microsoft Org Names via AfriNIC
connect: Network is unreachable
[-]Found AfriNIC Records for Microsoft

!! Finally finished !!

[*] Org & Customer related CIDsRs located in microsoft/orgcidrs.csv
[*] PoC related CIDsRs located in microsoft/poccidrs.csv
[*] CIDR only list located in microsoft/cidrlist.txt

```

Si cancelamos el script en cualquier momento, se eliminarán los archivos de trabajo temporales y el directorio creado para los archivos de salida resultantes.

```

[*] Enumerating CIDsRs for Example Dot Com Org Handles via ARIN
[-]No Org Handles found for Example Dot Com

[*] Enumerating CIDsRs for Example Dot Com Customer Handles via ARIN
[-]No Customer Handles found for Example Dot Com

[*] Enumerating CIDsRs for PoCs with the example.com Email Domain via ARIN
[-]Found ARIN email Records for example.com
^C
Caught ctrl+c, aborting & cleaning up my mess. . .

```

Debe tenerse en cuenta que, debido a la similitud de los nombres de algunas organizaciones, podrían obtenerse resultados no relacionados con el objetivo. Los archivos CSV proporcionarán las direcciones y URL asociadas para su posterior validación cuando sea necesario. También es posible que los empleados de la organización objetivo hayan utilizado su dirección de correo electrónico corporativa para registrar sus propios dominios. Estos también se encontrarán en los resultados.

AMASS.

El proyecto OWASP Amass realiza el mapeo de redes de superficies de ataque y el descubrimiento de activos externos utilizando técnicas de recopilación de información de código abierto y reconocimiento activo.

Técnicas de recopilación de información utilizadas:

- APIs
- Certificates
- DNS
- Routing
- Scraping
- Web Archives
- WHOIS

Hay una serie de razones más técnicas, que explicaremos a continuación y demostraremos con más detalle más adelante:

Viene con 3 subcomandos, es decir, funciones:

- `amass intel` -- Descubre los espacios de nombres de destino para las enumeraciones.
- `amass enum` -- Realiza enumeraciones y mapeo de red.
- `amass db` -- Manipular la base de datos gráfica de Amass.

El subcomando `amass intel`, o módulo si lo desea, puede ayudarle a recopilar inteligencia de código abierto sobre la organización y permitirle encontrar más nombres de dominio raíz asociados a la organización. Para ver las opciones disponibles de este subcomando, simplemente escríbalo en el terminal:

```
$ amass intel
[...]
Usage: amass intel [options] [-whois -d DOMAIN] [-addr ADDR -asn ASN -cidr CIDR]
  -active
    Attempt certificate name grabs
  -addr value
    IPs and ranges (192.168.1.1-254) separated by commas
  -asn value
    ASNs separated by commas (can be used multiple times)
[...]
```

Para utilizar el programa escribimos en nuestra terminal de Kali el comando `amass`. Pero si no lo tenemos en nuestro repositorio hacemos un `sudo apt update & sudo apt install amass`.

Para iniciar un mapeo utilizamos el comando: `amass intel -d owasp.org -whois`

```
$ amass intel -d owasp.org -whois
blockster.com
modeltime.com
```

PYMETA.

PyMeta es una reescritura en Python3 de la herramienta PowerMeta, creada por dafthack en PowerShell. Utiliza consultas de búsqueda especialmente diseñadas para identificar y descargar los siguientes tipos de archivos (pdf, xls, xlsx, csv, doc, docx, ppt, pptx) de un dominio dado utilizando Google y Bing scraping.

Una vez descargados, los metadatos se extraen de estos archivos mediante la herramienta `exiftool` de Phil Harvey y se añaden a un

informe .csv. Alternativamente, Pymeta puede apuntarse a un directorio para extraer los metadatos de los archivos descargados manualmente utilizando el argumento de línea de comandos -dir.

Cómo instalar y usar.

Ejecutamos pip3 install pymetasec en la terminal de Kali.

```
—$ pip3 install pymetasec
Defaulting to user installation because normal site-packages is not writeable
Collecting pymetasec
  Downloading pymetasec-1.1.1-py3-none-any.whl (19 kB)
Collecting bs4 (from pymetasec)
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from bs4)
Requirement already satisfied: beautifulsoup4 in /usr/lib/python3/dist-packages (from bs4)
Requirement already satisfied: soupsieve>1.2 in /usr/lib/python3/dist-packages (from beautifulsoup4)
Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
```

Opciones.

```
options:
  -h, --help            show this help message and exit
  -T MAX_THREADS        Max threads for file download (Default=5)
  -t TIMEOUT            Max timeout per search (Default=8)
  -j JITTER             Jitter between requests (Default=1)

Search Options:
  -s ENGINE, --search ENGINE  Search Engine (Default='google,bing')
  --file-type FILE_TYPE      File types to search (default=pdf,xls,xlsx,csv,doc,docx,ppt,pptx)
  -m MAX_RESULTS           Max results per type search

Proxy Options:
  --proxy PROXY          Proxy requests (IP:Port)
  --proxy-file PROXY     Load proxies from file for rotation

Output Options:
  -o DOWNLOAD_DIR        Path to create downloads directory (Default: ./)
  -f REPORT_FILE         Custom report name ("pymeta_report.csv")

Target Options:
  -d DOMAIN              Target domain
  -dir FILE_DIR          Pre-existing directory of file
```

Para usar la herramienta utilizamos el comando:

```
Python3 pymeta.py -d <url>
```



```
(kali@kali) ~/pymeta
$ python3 pymeta.py
pymeta.py: error: one of the arguments -d -dir is required

(kali@kali) ~/pymeta
$ python3 pymeta.py -d dropbox.com

PyMeta v1.2.0 - by @m8sec

[*]Target Domain : dropbox.com
[*]Search Engines(s) : google, bing
[*]File Types(s) : pdf, xls, xlsx, csv, doc, docx, ppt, pptx
[*]Max Downloads : 50

[*]Searching google, bing for 8 file type(s) on "dropbox.com"
[*]0 | pdf - https://www.google.com/search?q=site:dropbox.com+filetype:pdf&num=100&start=0
[*]14 | pdf - http://www.bing.com/search?q=site:dropbox.com%20filetype:pdf&first=0
[*]28 | pdf - http://www.bing.com/search?q=site:dropbox.com%20filetype:pdf&first=14
[*]40 | pdf - http://www.bing.com/search?q=site:dropbox.com%20filetype:pdf&first=28 (200)
|
```

Nos empieza a hacer el escaneo a la página indicada. Al finalizar, tendremos un archivo .csv para hojear más a fondo los datos obtenidos.

```
[*]Adding source URL's in report
[+]Report complete: ./drop_meta/pymeta_report.csv
```

REPOSCANNER.

Reposcanner es un script python para buscar en el historial de commits de repositorios Git cadenas interesantes como claves API, inspirado en truffleHog.

```
(kali@kali) ~/reposcanner
$ python3 reposcanner.py -h
usage: reposcanner.py [-h] -r REPO [-c COUNT] [-e ENTROPY] [-l LENGTH] [-b BRANCH] [-v]

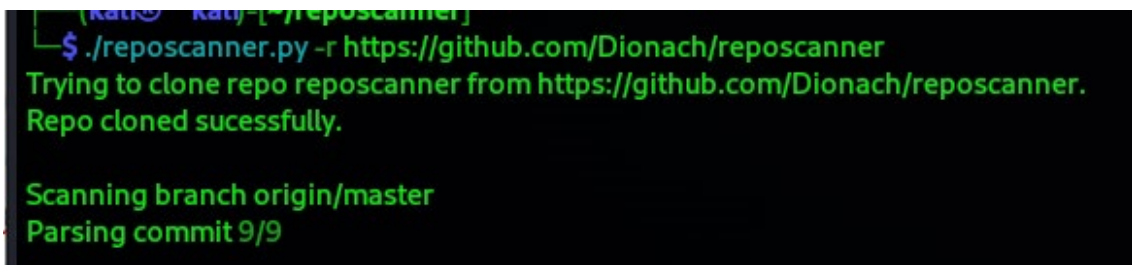
options:
  -h, --help            show this help message and exit
  -r REPO, --repo REPO  Repo to scan
  -c COUNT, --count COUNT  Number of commits to scan (default all)
  -e ENTROPY, --entropy ENTROPY  Minimum entropy to report (default 4.3)
  -l LENGTH, --length LENGTH  Maximum line length (default 500)
  -b BRANCH, --branch BRANCH  Branch to scan
  -v, --verbose          Verbose output

(kali@kali) ~/reposcanner
```

Ejemplo:

```
./reposcanner.py -r https://github.com/Dionach/reposcanner -v -c
```

30



```
(kati@kati) ~/reposcanner
$ ./reposcanner.py -r https://github.com/Dionach/reposcanner
Trying to clone repo reposcanner from https://github.com/Dionach/reposcanner.
Repo cloned successfully.

Scanning branch origin/master
Parsing commit 9/9
```