

Introducción

He estado investigando estas herramientas, y he de decir que me han dejado gratamente sorprendido. Como sabéis el OSINT es el estudio de la información que se puede sacar de fuentes abiertas, es decir accesibles a todo el mundo y en teoría gratis.

Forma parte del ciclo del hacking, dentro de la parte de reconocimiento. Es lo en los años 80 y 90 se llamaba "hacer un dossier". Cuando en una redacción de un periódico de investigación se iba a seguir un caso sobre una persona, siempre se empieza por "hacer el dossier" o sumario de esa persona. Que era normalmente un bloque de documentos variopintos que nos decían todo lo que se sabía públicamente de esa persona.

Bien pues el OSINT es lo mismo pero utilizando medios electrónicos. Existen infinidad de herramientas de OSINT y es un tema por lo menos igual de extenso que el propio hacking. En este post, nos vamos a centrar en un par de herramientas. Una bastante extensa, nuclei, y otra que hace solo una cosa muy concreta, pero que puede ser de gran utilidad: udon. Ambas herramientas están escritas en lenguaje go, por lo que habremos de instalar dicho lenguaje en la máquina de ataque de nuestro laboratorio.

Para ello arrancamos la máquina Parrot Security que estoy usando últimamente y que podéis descargar aquí.

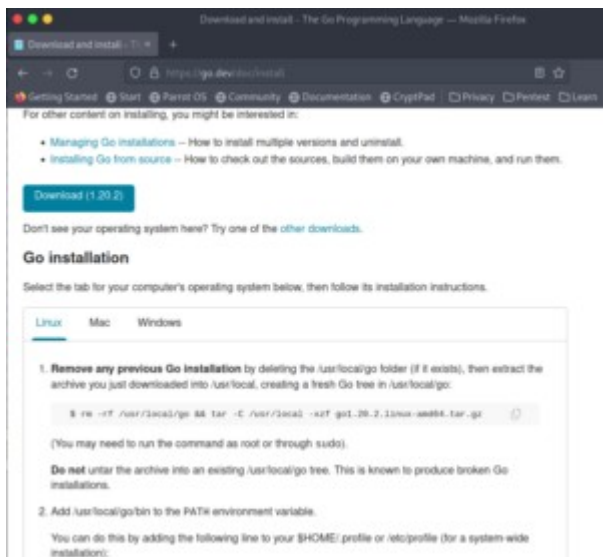
La contraseña para dicha máquina es: usuario, y el usuario es "usuario". Como véis poco originales.

Instalación De Lenguaje GO

Una vez dentro de Parrot Security, procedemos a instalar el lenguaje GO:

Navegamos a: <https://go.dev/doc/install>

Nos vamos a la opción "Linux". Pulsamos el botón descargar:



Esto nos dejará el archivo:

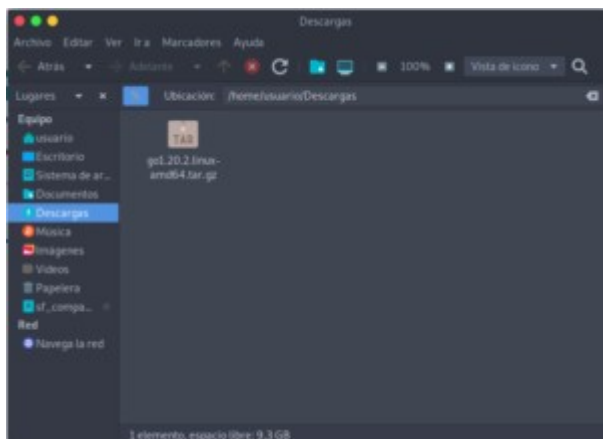
`go1.20.2.linux-amd64.tar.gz`

En la carpeta:

`/home/usuario/Descargas`

Ni que decir tiene que si ha pasado mucho tiempo cuando leáis esto, la versión habrá cambiado y será otro archivo.

Nos situamos en la carpeta: `/home/usuario/Descargas`



Y desde allí abrimos un terminal

`#sudo su`

Y seguidamente introducimos la siguiente instrucción que desinstala antiguas versiones de GO e instala la nueva de forma limpia:

```
>rm -rf /usr/local/go && tar -C /usr/local -xzf  
go1.20.2.linux-amd64.tar.gz
```

Y crea un árbol GO en:

`/usr/local/go`

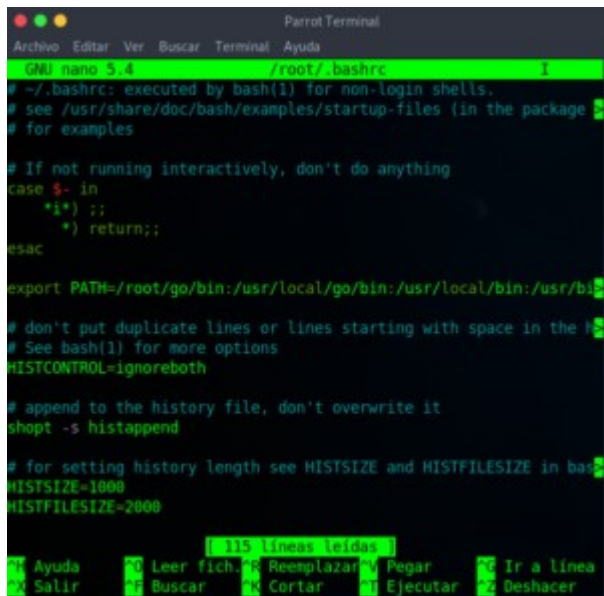
Los siguientes path deberemos añadirlos al la variable de entorno PATH:

`/root/go/bin:/usr/local/go/bin`

Editamos el archivo:

`#nano ~/.bashrc`

Y añadimos el path al principio según se ve en la imagen



```
Parrot Terminal
Archivo Editar Ver Buscar Terminal Ayuda
(RN) nano 5.4 /root/.bashrc I
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package
# for examples

# If not running interactively, don't do anything
case $- in
  *i*) ;;
  *) return;;
esac

export PATH=/root/go/bin:/usr/local/go/bin:/usr/local/bin:/usr/bi
# don't put duplicate lines or lines starting with space in the
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in ba
HISTSIZE=1000
HISTFILESIZE=2000

115 líneas leídas
Ayuda Leer fich. Reemplazar Pagar Ir a línea
Salir Buscar Cortar Ejecutar Deshacer
```

Aunque cada uno puede usar el editor que le parezca oportuno. Salimos del archivo grabando y reiniciamos Parrot Security.

Volvemos a abrir un terminal y comprobamos que GO está correctamente instalado preguntando por la versión:

```
#go version
```

Si nos devuelve el valor correcto, es que todo ha ido bien.

Instalación De Nuclei

Como hemos dicho antes nuclei requiere que esté instalado lenguaje GO. Una vez instalado Go, tecleamos el siguiente comando para instalar la versión mas reciente:

```
#go install -
v github.com/projectdiscovery/nuclei/v2/cmd/nuclei@latest
```

Bien, una vez instalado, la filosofía OSINT de nuclei es muy simple. Tomando como objetivo una página o aplicación web, le efectuamos una serie de escaneos basados en plantillas (templates). Algo parecido a los scripts de Nmap, pero a lo que mas se parece son a las expresiones regulares.

La buena noticia: hay infinidad de plantillas (miles) que nos van a dar prácticamente toda la información que se pueda extraer de una aplicación web. Es como coger a la web por los tobillos, ponerla boca a bajo y agitar hasta que salga todo.

La mala noticia: Si usamos todas las plantillas para hacer un dossier completo de la web, pues el tiempo que tarda puede ser muy alto.

Solución: Como es obvio, no usar todas las plantillas a la vez. Esto lo vamos a entender con varios ejemplos.

La sintaxis de nuclei es extensísima. Se puede ver con:

```
#nuclei -h
```

Vamos a instalar udon, para poder hacer los ejemplos concretos

Instalación De Udon

Bueno, una vez instalado GO, la instalación de udon es muy fácil. Desde un terminal en nuestro Parrot Security:

```
#go install github.com/dhn/udon@latest
```

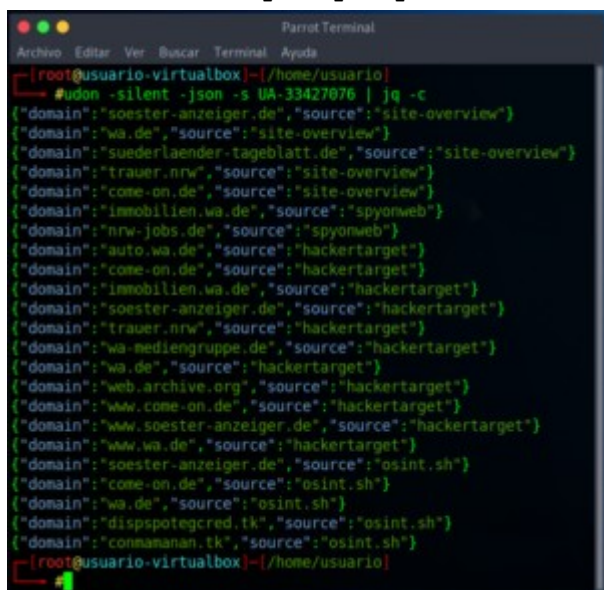
También deberemos instalar jq. El comando JQ se usa para transformar los datos JSON en un formato más legible e imprimirlos en la salida estándar en Linux. El comando JQ se basa en filtros que se utilizan para buscar e imprimir solo los datos necesarios de un archivo JSON.

```
#apt install jq -y
```

Una vez instalado udon y jq en nuestro sistema, veremos que su uso es muy sencillo, voy a poner el ejemplo que viene por defecto:

```
#udon -silent -json -s UA-33427076 | jq -c
```

Y la salida que proporciona es:



```
Parrot Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[root@usuario-virtualbox] ~/home/usuario
[+] #udon -silent -json -s UA-33427076 | jq -c
[{"domain":"soester-anzeiger.de","source":"site-overview"}]
[{"domain":"wa.de","source":"site-overview"}]
[{"domain":"suederlaender-tageblatt.de","source":"site-overview"}]
[{"domain":"trauer.nrw","source":"site-overview"}]
[{"domain":"come-on.de","source":"site-overview"}]
[{"domain":"immobilien.wa.de","source":"spyonweb"}]
[{"domain":"nrw-jobs.de","source":"spyonweb"}]
[{"domain":"auto.wa.de","source":"hackertarget"}]
[{"domain":"come-on.de","source":"hackertarget"}]
[{"domain":"immobilien.wa.de","source":"hackertarget"}]
[{"domain":"soester-anzeiger.de","source":"hackertarget"}]
[{"domain":"trauer.nrw","source":"hackertarget"}]
[{"domain":"wa-mediengruppe.de","source":"hackertarget"}]
[{"domain":"wa.de","source":"hackertarget"}]
[{"domain":"web.archive.org","source":"hackertarget"}]
[{"domain":"www.come-on.de","source":"hackertarget"}]
[{"domain":"www.soester-anzeiger.de","source":"hackertarget"}]
[{"domain":"www.wa.de","source":"hackertarget"}]
[{"domain":"soester-anzeiger.de","source":"osint.sh"}]
[{"domain":"come-on.de","source":"osint.sh"}]
[{"domain":"wa.de","source":"osint.sh"}]
[{"domain":"dispspotegcred.tk","source":"osint.sh"}]
[{"domain":"comananan.tk","source":"osint.sh"}]
[root@usuario-virtualbox] ~/home/usuario
```

Es decir, ¿Qué hace esta utilidad?, ¿Qué hace udon?. Toma como base un id de Google Analytics, hace un escaneo que ayuda a identificar las webs asociadas a ese código de GA.

Y la siguiente pregunta del millón claro está es: escogido un target, ¿Como puedo saber su id de Google Analytics para posteriormente conocer las webs asociadas?. Para responder a esa pregunta usaremos nuclei.

Ejemplos De Uso De Nuclei Y Udon

Vamos a hacer una recapitulación de las carpetas donde están instalados el lenguaje GO y estas utilidades OSINT: nuclei y udon.

Descarga del lenguaje GO:

/home/usuario/Descargas

Lenguaje GO:

/usr/local/go

/usr/local/go/bin

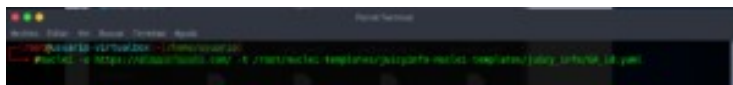
Ejecutables nuclei y udon:

/root/go/bin

Templates de nuclei:

/root/nuclei-templates

Supongamos que quiero utilizar una sola plantilla de nuclei que haga una cosa concreta. Voy a usar la plantilla que me dirá el id de Google Analytics, si lo tiene, de la web target.



```
#nuclei -u https://myweb.com/ -t  
/root/nuclei-templates/juicyinfo-nuclei-templates/juicy_info/G  
A_id.yaml
```

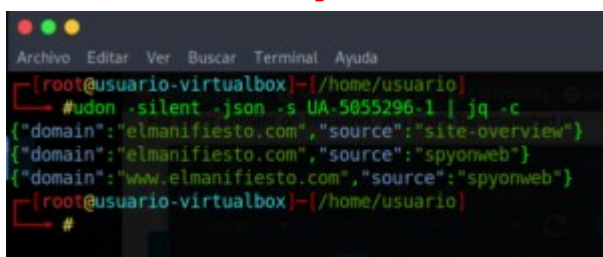
Es decir con el parámetro -u le digo la url target y con el parámetro -t la ruta completa del template o plantilla incluida la extensión que en este caso es yaml.

El resultado es:



Ahora tomaría el id de Google que nuclei ha identificado y usaría udon para tratar de hayar dominios y subdominios:

```
#udon -silent -json -s UA-5055296-1 | jq -c
```



Como vemos, udon ha encontrado un dominio y un subdominio (www).

Imaginemos que queremos saber si una aplicación web tiene un archivo robots.txt:

```
#nuclei -u juanjosevivas.es -t  
/root/nuclei-templates/miscellaneous/robots-txt.yaml
```

```
Parrot Terminal
nuclei -u juanjosevivas.es -t /root/nuclei-templates/miscellaneous/robots-txt.yaml

projectdiscovery.io

[INF] Using Nuclei Engine 2.8.9 (latest)
[INF] Using Nuclei Templates 9.3.9 (latest)
[INF] Templates added in last update: 61
[INF] Templates loaded for scan: 1
[INF] Targets loaded for scan: 1
[INF] Running httpx on input host
[INF] Found 1 URL from httpx
[robots.txt] (http) [info] https://juanjosevivas.es/robots.txt
~/root@osvuario-virtualbox: ~/home/usuario
```

Como podéis ver tengo el dichoso robots.txt en esta web. En fin, puede ser peligroso porque lo usan los hackers para extraer información valiosa sobre la superficie de ataque. Bien, pues yo creo que con estos pocos ejemplos tenéis una idea de cómo funcionan y para que sirven estas interesantes utilidades. Nuevamente os digo que es cuestión de que experimentéis vosotros mismos. Si tenéis alguna duda dejarlo en los comentarios, a ver si es posible que llegue alguno que no sea spam.

