



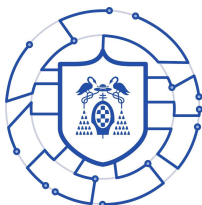
Universidad
de Alcalá

Honeypot

Máster en ciberseguridad

Asignatura: Sistemas de información para la
ciberseguridad

Kevin van Liebergen



2021

Índice general

Índice general	1
1 Enunciado	2
2 Endlessh SSH tarpit	3
2.1. Diagrama de arquitectura	3
2.2. Instalación y configuración	3
2.3. Ejemplo real	4
2.4. Misceláneo	6
3 Laboratorio T-pots Honeypots	7
3.1. Diagrama de arquitectura	7
3.2. Instalación y configuración	7
3.3. Pruebas honeypot	9
4 Conclusiones	15

1 Enunciado

Sobre el Honeypot se requiere:

- Explicar qué tipo es
- Instalación
- Configuración del mismo
- El alumno deberá realizar ejemplos del funcionamiento, realizando ataques para comprobar el funcionamiento del mismo

El alumno deberá entregar un documento que contenga lo siguiente:

- Explicación detallada de cada módulo y su funcionamiento
- Diagrama de arquitectura de la implementación
- Imágenes de captura de la implementación en funcionamiento
- Ejemplos demostrativos de casos de uso del honeypot
- Las capturas de imágenes deben contar con el nombre del alumno en el prompt del sistema para poder comprobarse la autenticidad

2 Endlessh SSH tarpit

2.1. Diagrama de arquitectura

Vamos a instalar el Honeypot **Endlessh SSH tarpit**, Honeypot de servicio de baja interacción que simula un servicio SSH legítimo, para ello hemos instalado en una máquina virtual Centos 7, el diagrama de arquitectura de la implementación es la que se muestra en la imagen 1, donde tenemos como Centos el sistema Operativo con el servicio falso, y en la misma red NAT un sistema Linux, que en este caso será un Kali Linux que intentará entrar por SSH al centos.

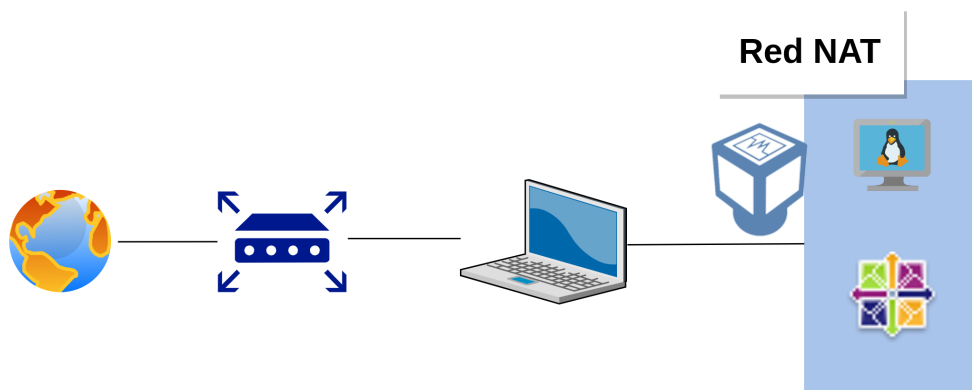


Figura 1: Diagrama de arquitectura

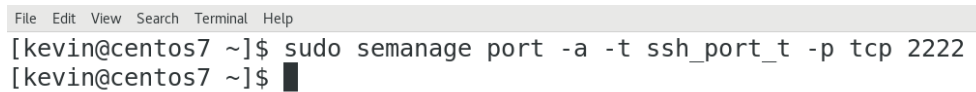
2.2. Instalación y configuración

La instalación se va a detallar a continuación, después de realizar una actualización del sistema (`$ sudo yum update`) y la instalación de las herramientas de administración (`polycoreutils-python`, `git` y `gcc`) se ha procedido a cambiar el puerto SSH legítimo como se muestra en la imagen 2, modificando el fichero `/etc/ssh/sshd_config`.

```
# If you want to change the port
# SELinux about this change.
# semanage port -a -t ssh_port_t
#
Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

Figura 2: Puerto SSH legítimo modificado

Mediante el comando de la imagen 3 establecemos al sistema que el puerto 2222 lo considere como un puerto ssh y que lo acepte como un protocolo tcp.



```
File Edit View Search Terminal Help
[kevin@centos7 ~]$ sudo semanage port -a -t ssh_port_t -p tcp 2222
[kevin@centos7 ~]$
```

Figura 3: Cambiar políticas de configuración de puertos

Seguidamente se ha procedido a configurar el firewall como aparecen en los apuntes de clase.

Después de ello se ha procedido a instalar el servicio *endlesssh* mediante los comandos:

```
$ git clone https://github.com/skeeto/endlesssh
$ make
$ sudo mv endlesssh /usr/local/bin
```

Una vez instalado se ha configurado el servicio para que escuche por el puerto 22 como se muestra en la figura 4, simulando que dicho servicio es un servicio real que escucha legítimamente.

```
[kevin@centos7 endlesssh]$ sudo !!
sudo mkdir -p /etc/endlesssh
[kevin@centos7 endlesssh]$ sudo vim /etc/endlesssh/config
[kevin@centos7 endlesssh]$ cat /etc/endlesssh/config
Port 22
_
```

Figura 4: Puerto del servicio endlesssh configurado

2.3. Ejemplo real

Como ejemplo, una vez hemos arrancado el servicio SSH endlesssh procedemos a intentar conectarlos a los puertos 2222 y 22. Como podemos observar en la imagen 5 si nos conectamos por SSH hacia el puerto 2222 nos permite loguear sin problemas con las credenciales correctas (centos:centos).

```

(kali㉿kali)-[~]
$ ssh kevin@10.0.2.7 -p 2222
The authenticity of host '[10.0.2.7]:2222 ([10.0.2.7]:2222)' can't be established.
ECDSA key fingerprint is SHA256:KqsNgEz/uRBG8R3Cvc+9T3VugCBho8P8rTPJ/92kPd4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.0.2.7]:2222' (ECDSA) to the list of known hosts.
+-----+
| LINUXVMIMAGES.COM |
+-----+
User Name: centos
Password: centos (sudo su -)
kevin@10.0.2.7's password:
Last login: Fri Dec 25 06:17:59 2020
+-----+
| LINUXVMIMAGES.COM |
+-----+
User Name: centos
Password: centos (sudo su -)
[kevin@centos7 ~]$

```

Figura 5: Conexión correcta hacia el puerto 2222

Sin embargo, si intentamos conectarnos mediante el puerto 22 nos quedaremos espernado de manera indefinida, debido a que no realiza ninguna otra acción para que realicemos ninguna interacción, tampoco nos da un error de Timeout.

```

(kali㉿kali)-[~]
$ ssh kevin@10.0.2.7 -p 22

```

Figura 6: Conexión incorrecta hacia el puerto 22

Por otro lado, si observamos el fichero de configuración de **endlesssh** observamos los intentos por conectarse de algún dispositivo, en la figura 7 se puede observar que se ha intentado conectar la dirección IP 10.0.2.15 (Kali Linux), y se muestra información acerca de la hora de entrada (12:00:02 y 16:07:04) y la hora de salida (12:00:11 y 16:07:50) respectivamente.

```

2020-12-25T11:59:30.900Z BindFamily IPv4 Mapped IPv6
2020-12-25T12:00:02.419Z ACCEPT host::ffff:10.0.2.15 port=47598 fd=4 n=1/4096
2020-12-25T12:00:11.433Z CLOSE host::ffff:10.0.2.15 port=47598 fd=4 time=9.014 bytes=135
2020-12-25T16:07:04.575Z ACCEPT host::ffff:10.0.2.15 port=47606 fd=4 n=1/4096
2020-12-25T16:07:50.665Z CLOSE host::ffff:10.0.2.15 port=47606 fd=4 time=46.090 bytes=820

```

Figura 7: Logs del Honeypot

Si lanzamos el ssh de manera que nos lance información (con **-vvv**) podemos observar el intento de intercambio de claves, de manera que nos intente engañar de que se están intercambiando claves cuando en realidad no es así.

```

debug1: kex_exchange_identification: banner line 0: +d?<$
debug1: kex_exchange_identification: banner line 1: ]u%xeun?RILS\\QSFx$ywm#1L
debug1: kex_exchange_identification: banner line 2: EAR%;P_n9"~waNE~*RuQ0[/b\\y59
debug1: kex_exchange_identification: banner line 3: iNPlf,0j|xCv'8
debug1: kex_exchange_identification: banner line 4: h@AOyzto@S&<7WAKJ
debug1: kex_exchange_identification: banner line 5: J[jncdhV%gTs:3=lDiu
debug1: kex_exchange_identification: banner line 6: FxY>Wp^*0T2eS/JL
debug1: kex_exchange_identification: banner line 7: 8":'?scWVon){ltNx'Dn`A
debug1: kex_exchange_identification: banner line 8: u;vsXys`_GDiq]%-pygPH?i3$K
debug1: kex_exchange_identification: banner line 9: VdXp"k*bW.WlL$

```

Figura 8: Conexión con verbose (-vvv) hacia el Honeypot

2.4. Misceláneo

Por curiosidad, se ha realizado un escaneo de puertos a toda la red, y ni si quiera me ha encontrado el sistema Centos (dirección 10.0.2.7), avisando de que es posible que exista un firewall rechazando paquetes.

Realizando un escaneo únicamente al puerto 22 sin el parámetro `-Pn` lo detecta como cerrado, al igual que antes nos sugiere que lo escaneemos con el parámetro `-Pn`.

```

(kali@kali)-[~]
$ nmap 10.0.2.7 -p 22

Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-25 11:26 EST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 0.09 seconds

```

Figura 9: Nmap sin el parámetro `-Pn`

Lanzando el comando anterior con el parámetro `-Pn` (no hacer ping), si nos reconoce el puerto 22 como si estuviera abierto.

```

(kali@kali)-[~]
$ nmap 10.0.2.7 -p 22 -Pn

Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-25 11:27 EST
Nmap scan report for 10.0.2.7 (10.0.2.7)
Host is up (0.0015s latency).

PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds

```

Figura 10: Nmap con el parámetro `-Pn`

Por ello es importante saber este tipo de comportamiento de cara a enfrentarnos a un servicio similar siendo nosotros un equipo red team. Este honeypot de baja interacción no nos provee de más información debido a que

no admite una interacción mayor con el usuario, al contrario del honeypot que comentaremos en el siguiente capítulo.

3 Laboratorio T-pots Honeypots

3.1. Diagrama de arquitectura

Para este apartado vamos a implementar el **Honeypot T-Pot**, se ha descargado la imagen ISO que contenía debian con la arquitectura de T-pot. El diagrama de arquitectura es idéntica al de la figura 1 del apartado anterior, configurada una red NAT se he instalado el servidor Debian + T-Pot en lugar del Centos del ejercicio anterior.

3.2. Instalación y configuración

Una vez instalado según las indicaciones de las transparencias se ha procedido a comprobar la funcionalidad, a ver los contenedores docker como aparece en la imagen 11 en la Web UI de administración en el puerto 64294.

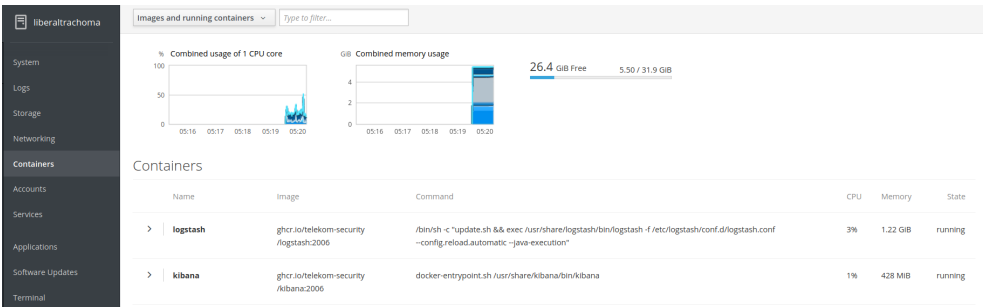


Figura 11: Docker instanciados

Cuando se ha intentado acceder al puerto 64297 para a Web UI de monitoreo no se permitía acceder a la página, debido a errores que se desconoce como aparece en la figura 12 por lo que se ha procedido a instalarlo de otra manera.

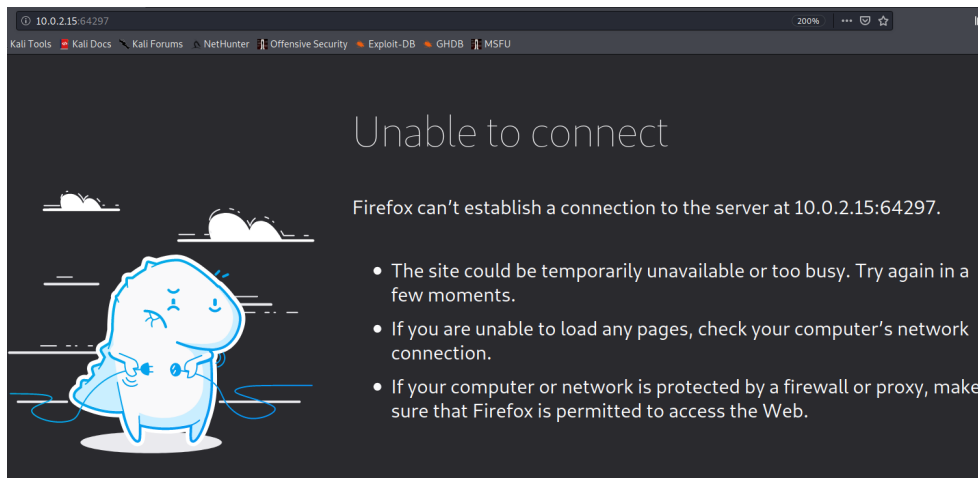


Figura 12: No es posible conectarse a la Web UI

El otro método de instalación ha sido el siguiente, se ha instalado un servidor Debian 10 en una máquina virtual y sobre ella se ha clonado el repositorio e inicializado la instalación, una vez se instaló Debian se utilizaron los siguientes comandos.

```
$ git clone https://github.com/telekom-security/tpotce
$ cd tpotce/iso/installer/
$ cp tpot.conf.dist tpot.conf
$ sudo ./install.sh --type=auto --conf=tpot.conf
```

Después de haber realizado la instalación con la creación de usuarios *kevin:kevin* (la instalación duró aproximadamente una hora) se procedió a acceder a la Web UI de monitoreo, esta vez de manera correcta.

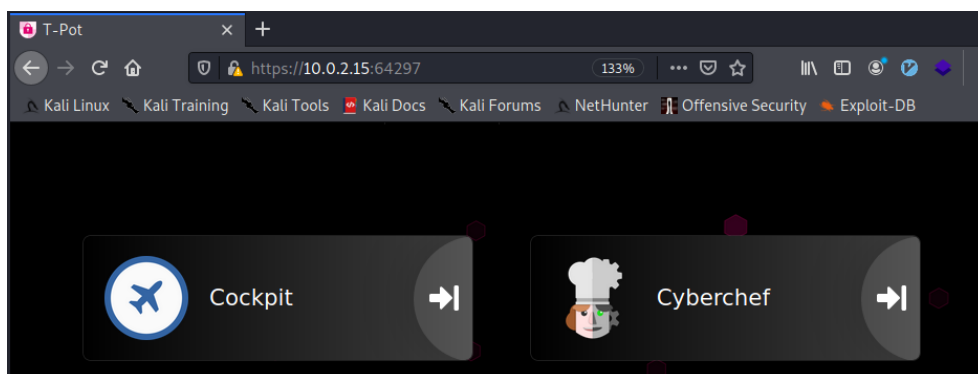


Figura 13: tpot web

3.3. Pruebas honeypot

Además para poner a prueba el sistema se realizó un escaneo de todos los puertos (`$ nmap -p- 10.0.2.15`), se puede observar en la figura 14 que el sistema nos devolvió absolutamente todos los puertos abiertos.

```
62074/tcp open  unknown
62075/tcp open  unknown
62076/tcp open  unknown
62077/tcp open  unknown
62078/tcp open  iphone-sync
62079/tcp open  unknown
62080/tcp open  unknown
62081/tcp open  unknown
62082/tcp open  unknown
```

Figura 14: Todos los puertos abiertos

Además, escaneando los 1000 puertos más importantes, sin añadir parámetros, nos aparece que están abiertos, como es de esperar, los puertos ftp, ssh, telnet, smtp, etc.

```
(kali㉿kali)-[~]
$ nmap 10.0.2.15
1 x
Starting Nmap 7.91 ( https://nmap.org )
:58 EST
Nmap scan report for 10.0.2.15 (10.0.2.15)
Host is up (0.067s latency).
Not shown: 142 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
42/tcp    open  nameserver
80/tcp    open  http
81/tcp    open  hosts2-ns
110/tcp   open  pop3
135/tcp   open  msrpc
143/tcp   open  imap
```

Figura 15: Escaneo de puertos más importantes

Y entrando legítimamente desde la máquina virtual podemos observar en

la imagen 16, el sistema operativo legítimo donde se ha instalado T-Pot es un Debian 10 Buster.

```
[kevin@weirdbronchitis:~]$ whoami
kevin
[kevin@weirdbronchitis:~]$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
[kevin@weirdbronchitis:~]$
```

Figura 16: El sistema operativo es un Debian 10

Prueba SSH

Cuando accedemos por SSH, podemos acceder al sistema introduciendo credenciales, siempre la primera contraseña que introduces es la válida, en este caso *kevin:a*. Si se introduce otro usuario no podremos acceder, sin embargo, cuando conseguimos entrar al sistema, podemos observar como aparece en la imagen 17 que el sistema nos intenta engañar poniendo como nombre del sistema *ubuntu*.

```
(kali㉿kali)-[~]
└─ssh kevin@10.0.2.15 -p 22

Password:

The programs included with the De
ware;
the exact distribution terms for
individual files in /usr/share/dc

Debian GNU/Linux comes with ABSOL
permitted by applicable law.
kevin@ubuntu:~$ whoami
kevin
```

Figura 17: Acceso concedido ingresando cualquier contraseña

En la sección de Kibana de **Cowrie**, podemos ver estadísticas acerca de los intentos de conexión hacia el servidor, en la imagen 18 observamos los usuarios y contraseñas utilizadas en el ataque.

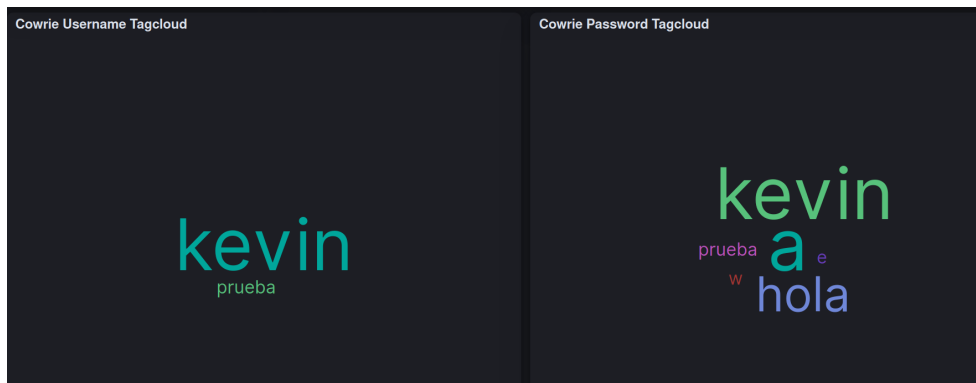


Figura 18: Usuarios y contraseñas más probadas

Información del sistema

El siguiente día se ingresó con el usuario root probamos a mostrar el fichero `/etc/shadow` donde se muestran los usuarios y las contraseñas hasheadas del sistema, en este caso nos muestra información falsa para engañar, como son usuarios y contraseñas inexistentes.

```
root@ubuntu:~# cat /etc/shadow
root:$6$4aOmWdpJ$/kyPOik9rR0kSLyABIYNXgg/UqLWX3c1eIao
:
daemon*:15800:0:99999:7:::
bin*:15800:0:99999:7:::
sys*:15800:0:99999:7:::
sync*:15800:0:99999:7:::
games*:15800:0:99999:7:::
man*:15800:0:99999:7:::
lp*:15800:0:99999:7:::
mail*:15800:0:99999:7:::
news*:15800:0:99999:7:::
uucp*:15800:0:99999:7:::
proxy*:15800:0:99999:7:::
www-data*:15800:0:99999:7:::
backup*:15800:0:99999:7:::
list*:15800:0:99999:7:::
irc*:15800:0:99999:7:::
gnats*:15800:0:99999:7:::
nobody*:15800:0:99999:7:::
libuuid!:15800:0:99999:7:::
sshd*:15800:0:99999:7:::
phil:$6$ErqInBoz$FibX212AFnHMvyZdWW87bq5Cm3214CoffqFu
```

Figura 19: Fichero `/etc/shadow` falso

Vamos a realizar un ataque por fuerza bruta con Hydra, para ello se ha

```
$ hydra -l root -P rockyou-15.txt 10.0.2.15 -t 4 ssh
$ hydra -l kevin -P rockyou-15.txt 10.0.2.15 -t 4 ssh
$ hydra -l prueba -P rockyou-15.txt 10.0.2.15 -t 4 ssh
```

[illegible]

Denegación de servicio

El comando que se ha utilizado es el siguiente

Que desglosado realiza las siguientes acciones:

- -c 10000
Número de paquetes que se van a enviar
- -d 120
Tamaño del dato
- -S
SYN tcp activado

12

- -w 64
Tamaño de la ventana TCP
- -p 21
Puerto destino
- --flood
Envía los paquetes lo más rápido posible
- --rand-source
Habilita el modo aleatorio de direcciones origen

Como se aleatoriza la dirección IP origen que se envía aparece como si se estuviese mandando de distintos países, en la figura 21 se puede observar como Kibana ordena estas direcciones IP por países y establece que la mayoría de ataques vienen de Estados Unidos.

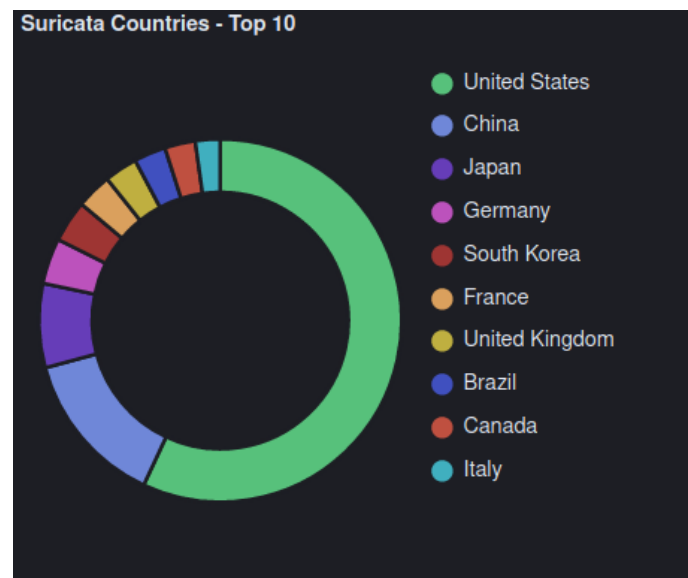


Figura 21: Ranking de países de donde viene el ataque

Asimismo, también es posible plasmar en un mapa los dispositivos de donde se han enviado las peticiones, como se muestra en la imagen 22

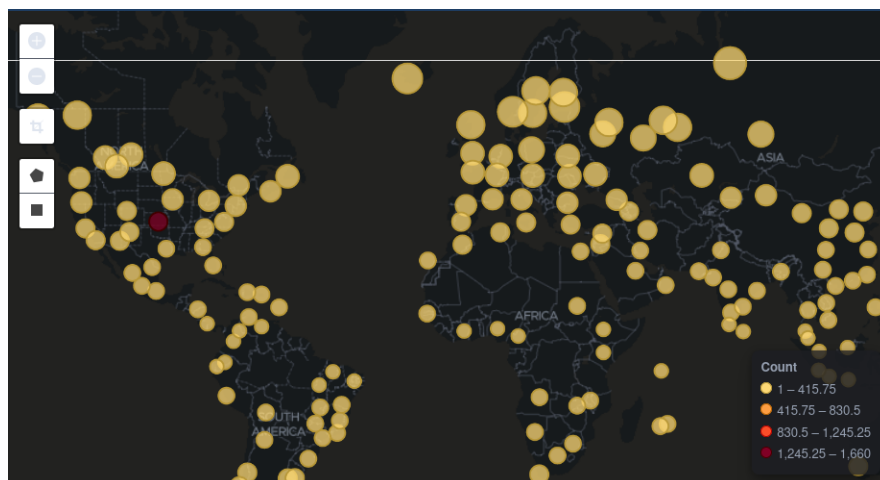


Figura 22: Mapa global de ataque

Kibana también ofrece un módulo en el que asocia las direcciones IP con las compañías de telecomunicaciones, en este caso en la figura 23 se muestra que direcciones IP asociadas a la compañía Chinanet han atacado el servidor Debian con T-pot mayoritariamente.

AS ↕	ASN ↕	CNT ↕
4134	Chinanet	134
7922	Comcast Cable Communications, LLC	102
7018	AT&T Services, Inc.	98
4837	CNCGROUP China169 Backbone	81
721	DoD Network Information Center	72
3356	Level 3 Communications, Inc.	55
4766	Korea Telecom	50
3320	Deutsche Telekom AG	48
4713	NTT Communications Corporation	42
701	MCI Communications Services, Inc. d/b/a Verizon Business	41

Figura 23: Compañías de telecomunicaciones

Metasploit

Para este apartado vamos a utilizar el módulo **Diaonea**. La intención de diaonea es atrapar el malware que explota las vulnerabilidades por los servicios

que ofrece la red, es decir, el objetivo final es obtener una copia del malware. Para este ejemplo vamos a lanzar el mismo ataque de las transparencias, utilizando metasploit y el ataque `admin/smb/check_dir_file`², que comprueba la existencia de un archivo en un red de hosts SMB.

Como podemos observar en la figura 24 buscamos el archivo ‘Windows’, y el Honeypot nos devuelve que el archivo se encuentra en el sistema, lo cual no es así ya que ni existe y ni si quiera es un sistema Windows.

```
msf6 auxiliary(admin/smb/check_dir_file) > exploit

[+] 10.0.2.15:445      - File FOUND: \\10.0.2.15\C$\Windows
[*] 10.0.2.15:445      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/smb/check_dir_file) > set rpath prueba_kevin
rpath => prueba_kevin
msf6 auxiliary(admin/smb/check_dir_file) > exploit

[+] 10.0.2.15:445      - File FOUND: \\10.0.2.15\C$\prueba_kevin
[*] 10.0.2.15:445      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/smb/check_dir_file) > 
```

Figura 24: Ataque lanzado con Metasploit

Desde la parte del módulo **Diaonea** podemos observar que el sistema recoge información acerca del tipo de ataque, a que puerto ha atacado, sobre que protocolo ataca, el número de IPs y las IPs que han atacado, en este caso el Kali Linux de manera local.

4 Conclusiones

Después de haber instalado y configurado el honeypot endlessh y T-pot que incluye todo tipo de módulos y herramientas, podemos concluir que actualmente estas instalaciones y configuraciones no son difíciles de implementar en sistemas, dependiendo de lo que se va a proteger necesitaremos más almacenamiento o menos.

Los Honeypots de baja interacción no me han llamado la atención, sin embargo los de alta interacción me ha parecido una buena idea, para poder conocer además como actúa el atacante, que contraseñas utiliza y una vez dentro del sistema que comandos ejecuta.

La integración que posee T-pot con Kibana me parece una idea espectacular para poder gestionar y visualizar de una mejor manera, sin embargo y desde mi punto de vista he echado en falta una manera de instalar que te pregunte

²https://www.rapid7.com/db/modules/auxiliary/admin/smb/check_dir_file/

desde el comienzo que módulos quieres instalar, para no tener que configurarlo después ni tener todos los puertos abiertos de primeras.