Antes que nada me gustaría agradecer a mi profesor de ciberseguridad por haberme dejado realizar esta práctica en clase, dado que así me la he tomado más en serio y me he puesto las pilas para sacar el proyecto adelante.

Ahora sí, una vez concluidos los agradecimientos, vamos a poner en práctica este despliegue de honeypot «casero» donde vamos a poder comprobar que las herramientas con las que cuenta Debian de forma nativa nos permiten levantar un honeypot SSH sin que nos haga falta clonar interminables repositorios de GitHub que, al mismo tiempo que nos permiten montar la herramienta, pueden tener vulnerabilidades que los atacantes pueden utilizar para realizar un ataque vertical. Vamos allá:

INTRODUCCIÓN

Un honeypot es una herramienta de ciberseguridad diseñada para actuar como un señuelo o trampa, simulando ser un sistema, servicio o recurso legítimo de una red. Su objetivo principal es atraer a atacantes, registrar sus actividades y aprender sobre sus métodos, motivaciones y herramientas. Esto permite mejorar la seguridad de los sistemas reales al identificar vulnerabilidades y posibles amenazas.

Las características principales de un honeypot son:

- **Simulación de sistemas reales:** Emula servidores, aplicaciones o servicios para parecer un objetivo atractivo para los atacantes.
- **Aislamiento controlado:** Está diseñado para contener el daño y proteger el entorno real, de modo que los atacantes interactúen únicamente con el honeypot.
- **Registro y análisis:** Monitoriza y registra todos los intentos de acceso, comandos ejecutados y tráfico generado para su análisis posterior.
- **Uso específico:** Se puede configurar para diferentes propósitos, como analizar malware, identificar ataques de fuerza bruta o comprender las técnicas de los atacantes.

OBJETIVOS

En este mini-proyecto vamos a poner en marcha un honeypot SSH casero valiéndonos únicamente de las herramientas disponibles en los repos de Debian, siempre y cuando la herramienta no sea en sí misma un honeypot, claro está. Los objetivos a cumplir son:

- Capturar todos los intentos de autenticación en el servidor SSH (Fecha, IP, Usuario y Contraseña)
- Capturar todas las interacciones con la CLI, tanto los comandos que ingresa el atacante, como la salida que ve en la terminal al momento de ejecutarlos.
- Enviar toda la actividad anterior a un servidor SIEM para su análisis en tiempo real (o posterior).

VENTAJAS

- No es necesario cambiar el puerto por defecto del servidor SSH ni usar forwarding mediante IPTables o NFTables, dado que todo se realiza sobre el propio servidor SSH oficial del dispositivo.
- Extremadamente difícil para un atacante identificar que está dentro de un honeypot.
- No hay necesidad de preocuparse de que el servicio del honeypot se inicie o no, porque todo el sistema es un honeypot y el sistema siempre se inicia.

REQUISITOS

- Un ordenador, máquina virtual o contenedor LXC con Debian.
- Una conexión a internet activa a Internet.
- Control sobre el router que tiene la conexión activa a internet (dado que hay que realizar el reenvío de puertos).

CONSIDERACIONES PREVIAS

SOBRE LA INSTALACIÓN EN CONTENEDORES LXC DE PROXMOX

Si el honeypot se implementa en un contenedor LXC de Proxmox, debemos realizar toda la instalación con el usuario root. Nunca instalaremos el paquete su, ya que un contenedor Debian de Proxmox no trae activado el usuario su y es una característica que no nos interesa para el honeypot.

SOBRE LA INSTALACIÓN EN BAREMETAL O MÁQUINAS VIRTUALES

Si utilizamos la imagen **netinst** de Debian para la instalación, el paquete **sudo** estará instalado, pero el usuario por defecto (ID 1000) no tendrá permisos sudo. En este caso, eliminaremos ese usuario y desinstalaremos el paquete sudo, ejecutando como root:

```
deluser nombre_usuario
apt remove sudo -y
```

Si utilizamos el instalador gráfico (Calamares) de Debian, el usuario por defecto (ID 1000) tendrá permisos sudo. En este caso:

1. Establecemos una contraseña para el usuario root:
   `passwd root`
2. Iniciamos sesión como root y eliminamos el usuario por defecto (ID 1000):
   `deluser --remove-home nombre_usuario`
3. Desinstalamos el paquete sudo:
   `apt remove sudo -y`

DESPLIEGUE

.

BASTIONADO DEL HONEYPOT

Para asegurarnos de que el honeypot no podrá ingresar a nuestra red, implementaremos unas reglas en el hook ingress usando NFTables. Para ello, primero instalamos el paquete ejecutando como root:

```
apt-get -y update
apt-get -y install nftables
```

...aunque, dependiendo de la versión de Debian en la que estemos desplegando el honeypot, nftables seguramente ya esté instalado.

Una vez instalado, vamos a crear el script de reglas para impedir tráfico a la subred. Primero creamos el script, ejecutando como root:

`touch /root/nftables.rules`

...y luego lo populamos con el siguiente texto:

```
table netdev TablaIngressEth0 {

  chain CadenaIngressEth0 {
    type filter hook ingress device eth0 priority -500; policy accept;

    ip daddr 191.168.1.1    counter accept
    ip daddr 191.168.1.10   counter accept
    ip daddr 191.168.1.0/24 counter drop

  }

}
```

En el texto de arriba, 192.168.1.1 es la pata LAN de nuestro router, 192.168.1.10 es el servidor SIEM y el /24 es toda la subred de casa.

Luego de guardar el archivo lo cargamos en **/etc/nftables.conf** agregando a dicho archivo la línea:

```
include "/root/nftables.rules"
```

...justo debajo de:

```
flush ruleset
```

El resto del archivo podemos borrarlo. Así, una vez realizado esos cambios, podemos aplicarlos, ejecutando como root:

`nft --file /etc/nftables.conf`

...pudiendo visualizar las reglas activas mediante la ejecución de:

`nft list ruleset`

A partir del siguiente reinicio, siempre que se inicie el servidor debian con el honeypot se cargarán automáticamente las reglas, pero los atacantes nunca sabrán cuales son dado que están en un archivo al que no podrán acceder sin ser root.

CAPTURA DE CREDENCIALES SSH

Para registrar las contraseñas introducidas por los atacantes cuando intentan iniciar sesión mediante SSH vamos a utilizar la combinación de PAM con la llamada a un script de bash. Entonces, primero creamos la carpeta donde irá el script, ejecutando como root:

`mkdir -p /root/honeypot/scripts/`

...y luego creamos el archivo **/root/honeypot/scripts/RegistrarCredencialesSSH.sh** con este contenido:

```
#!/bin/bash
read PASSWORD
cFechaDeEjec=$(date +a%Ym%md%d@%T.%4N)
echo "$cFechaDeEjec sship='"$PAM_RHOST"' sshuser='"$PAM_USER"' sshpass='"$PASSWORD"'" >>
/root/honeypot/logs/CredencialesSSH.log
```

Nos aseguramos de que el archivo tenga permisos de ejecución:

`chmod +x /root/honeypot/scripts/RegistrarCredencialesSSH.sh`

Creamos la carpeta y el archivo de log, para que esté listo cuando se empiece a escribir en él:

```
mkdir -p /root/honeypot/logs/CredencialesSSH.log
touch    /root/honeypot/logs/CredencialesSSH.log
```

Ahora, para registrar las contraseñas introducidas utilizaremos el módulo **pam_exec.so** que, con la opción **expose_authtok** nos permitirá leer las contraseñas ingresadas desde la autenticación PAM. Para ello, editamos el archivo **/etc/pam.d/sshd** y añadimos la siguiente línea al inicio:

```
auth required pam_exec.so expose_authtok /root/honeypot/scripts/RegistrarCredencialesSSH.sh
```

Después de hacer esa modificación, veremos que tras reiniciar el servicio con:

`systemctl restart ssh`

...cada vez que intentemos acceder por ssh se registrarán en el archivo **/root/honeypot/logs/CredencialesSSH.log**, la fecha, la IP de origen, el usuario y la contraseñas con las que los atacantes intentan un logueo exitoso.

ENVÍO DE LOGS AL SIEM

Suponiendo que como SIEM estamos utilizando la combinación de Loki y Grafana, deberemos instalar Promtail en el honeypot. Promtail es el forwarder oficial del proyecto Grafana/Loki y nos permitirá múltiples formas de guardar los logs. Para ello, ejecutamos como root lo siguiente:

```
apt-get -y install curl unzip
curl -L https://github.com/grafana/loki/releases/download/v3.3.0/promtail-linux-amd64.zip -o /tmp/promtail.zip
cd /tmp/
unzip promtail.zip
cp -fv /tmp/promtail-linux-amd64 /usr/bin/
```

Lo siguiente es crear el archivo de configuración de promtail, que contendrá la configuración de como enviar las contraseñas capturadas al SIEM. Primero creamos la carpeta, ejecutando como root:

`mkdir -p /etc/promtail`

...y luego hacemos un toch a **/etc/promtail/promtail-config.yaml** y lo populamos con el siguiente texto:

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0
  log_level: info

positions:
  filename: /var/lib/promtail/positions.yaml

clients:
  - url: http://10.5.20.3:3100/loki/api/v1/push

scrape_configs:
  - job_name: honeypot_logs
    static_configs:
      - targets:
          - localhost
        labels:
          job: fuerzabrutassh
          host: servidorhp
          __path__: /root/honeypot/logs/CredencialesSSH.log
```

Antes de crear el servicio de systemd deberemos crear la carpeta /var/run/promtail, para que el proceso no de fallo. Lo hacemos ejecutando como root:

```
mkdir -p /var/lib/promtail/
```

Ahora sí, creamos el servicio de systemd, «tocando» **/etc/systemd/system/promtail.service** y populándolo con el siguiente texto:

```
[Unit]
Description=Promtail Service
After=network.target

[Service]
Type=simple
ExecStart=/usr/bin/promtail-linux-amd64 --config.file /etc/promtail/promtail-config.yaml
Restart=always
RestartSec=5
User=root
Group=root

[Install]
WantedBy=multi-user.target
```

También es posible loguear a un archivo directamente:

```
ExecStart=/usr/bin/promtail-linux-amd64 --config.file /etc/promtail/promtail-config.yaml --log-
output=file=/var/log/promtail/promtail.log
```

Activamos e iniciamos el servicio:

```
systemctl enable promtail.service --now
```

Pero en este caso, no nos interesa que los atacantes puedan tener un log dentro del propio host del honeypot al que puedan echar un vistazo...

Finalmente, y dado que ya no nos hacen falta, desinstalamos los paquetes curl y unzip:

```
apt-get -y autoremove curl unzip
```

PUNTO DE CONTROL INTERMEDIO

En el punto en el que estamos, si hemos realizado todo bien, estaremos capturando las credenciales con las que se intenta acceder al servicio SSH y éstas se estarán enviando al SIEM para su análisis y creación de dashboards y alertas. Lo siguiente a desplegar es la herramienta para capturar todo lo que hace el atacante en la terminal.

HERRAMIENTA DE CAPTURA DE TERMINAL

.

Para que cualquier usuario que inicie sesión pueda escribir en la carpeta **/var/run/tlog** hacemos que el propietario de la carpeta sea el root y asignamos permisos 1777. De esta forma, usando el bit pegajoso (1) cualquier usuario podrá escribir en la carpeta pero ningún usuario podrá modificar los archivos que tengan un propietario distinto al suyo propio.

```
mkdir -p /var/run/tlog/
chown root:root /var/run/tlog/
```

```
chmod 1777 /var/run/tlog/
```

Creamos el archivo de configuración:

```
echo '{'                            > /etc/tlog/tlog-rec-session.conf
echo '  "shell" : "/bin/bash",'    >> /etc/tlog/tlog-rec-session.conf
echo '  "notice" : "",'            >> /etc/tlog/tlog-rec-session.conf
echo '  "log": {'                  >> /etc/tlog/tlog-rec-session.conf
echo '    "input" : true,'         >> /etc/tlog/tlog-rec-session.conf
echo '    "output" : true,'        >> /etc/tlog/tlog-rec-session.conf
echo '    "window" : true'         >> /etc/tlog/tlog-rec-session.conf
echo '  },'                        >> /etc/tlog/tlog-rec-session.conf
echo '  "limit": {'                >> /etc/tlog/tlog-rec-session.conf
echo '    "action" : "pass"'       >> /etc/tlog/tlog-rec-session.conf
echo '  },'                        >> /etc/tlog/tlog-rec-session.conf
echo '  "file": {'                 >> /etc/tlog/tlog-rec-session.conf
echo '    "path" : "/var/tmp/ses"' >> /etc/tlog/tlog-rec-session.conf
echo '  },'                        >> /etc/tlog/tlog-rec-session.conf
echo '  "writer" : "file"'         >> /etc/tlog/tlog-rec-session.conf
echo '}'                           >> /etc/tlog/tlog-rec-session.conf
```

Hay una forma más compleja de configurar tlog utilizando sssd, en la que podremos personalizar que tipo de usuarios o grupos capturar o excluir de la captura, agregando un archivo de configuración **/etc/sssd/conf.d/sssd-session-recording.conf** con diferentes configuraciones dentro. Pero para la tarea de este honeypot autogestionado no nos atañe, dado que en este sólo habilitaremos un usuario para loquearse y no nos interesa llenar el honeypot de diferentes servicios. Cuanto más limpio, mejor.

SCRIPT PARA PARSEAR LAS SESIONES GRABADAS

Una forma más ordenada de tener los datos es tener cada sesión parseada hacia un archivo json independiente. Para ello vamos a crear un script que se ejecutará cada vez que una sesión SSH se cierre y que recorrerá el archivo de log completo extrayendo todos los datos de cada sesión para acabar juntándolos en un nuevo archivo que se identificará con el nombre de la sesión y la extensión .json. El script **/root/honeypot/scripts/ExportarSesiones.sh** contendrá este texto:

```
#!/bin/bash
vArchivoLog="/var/tmp/ses"
vCarpetaSesiones="/root/honeypot/logs/sesiones"
aIDs=()

while IFS= read -r vID; do
  aIDs+=("$vID")
done < <(cat "$vArchivoLog" | jq -r '.rec' | sort | uniq)

for vIDenArray in "${aIDs[@]}"; do
  jq -c 'select(.rec == "'$vIDenArray'")' $vArchivoLog > $vCarpetaSesiones/$vIDenArray.json
done
```

Ahora debemos conseguir que, cada vez que se cierre una sesión, todas las sesiones guardadas en el archivo /var/tmp/ses se guardarán individualmente, con su ID correspondiente, y en formato json. Lo hacemos editando el archivo **/etc/pam.d/sshd** y agregando al final la línea:

```
session optional pam_exec.so /root/honeypot/scripts/ExportarSesiones.sh
```

MEDIDAS DE SEGURIDAD

Hay que tomar algunas medidas de seguridad para mitigar las acciones que los usuarios maliciosos realizen para tirar el servidor una vez que accedan a él. Por ejemplo, limitar el número de procesos máximos que puedan abrir. Si no lo hacemos, nos exponemos a un ataque de tipo bash fork bomb que, además de probablemente tirarles la conexión a sí mismos, impida que otros atacantes accedan al honeypot. Para ello creamos el archivo /etc/security/limits.d/20-nproc.conf y metemos dentro la siguiente directiva:

```
admin hard nproc 100
```

Eso limitará a 100 la cantidad máxima de procesos que el usuario admin pueda crear.

Para un límite global más agresivo, podemos editar **/etc/systemd/system.conf** y **/etc/systemd/user.conf** y tocar el parámetro DefaultLimitNPROC.

DIFICULTAR AÚN MÁS LA DETECCIÓN DEL HONEYPOT

En este punto del despliegue el honeypot es completamente funcional, pero hay algunos cambios que podemos hacer en el sistema para dificultar más aún que el atacante detecte que está dentro de un honeypot

CAMBIAR LA TERMINAL EN EL ARCHIVO PASSWD

Movemos el archivo **/usr/bin/tlog-rec-session** a **/usr/bin/shell** y modificamos el archivo **/etc/passwd** indicando que el nuevo archivo sea la terminal que ejecutará al iniciar exitosamente la sesión:

```
mv /bin/tlog-rec-session /bin/shell
sed -i -e 's|/bin/tlog-rec-session|/bin/shell|g' /etc/passwd
```

MOVER LOS BINARIOS DE TLOG AL BIN DEL ROOT

Movemos los binarios sobrantes de tlog a la carpeta bin del root, para esconderlos de la mirada del atacante:

```
mkdir -p             /root/honeypot/bin/
mv -v /bin/tlog-play /root/honeypot/bin/
mv -v /bin/tlog-rec  /root/honeypot/bin/
```

EVITAR EL ACCESO A LOS PROCESOS DEL ROOT

Para evitar que un usuario que no es root vea los procesos del root, tendremos que hacer que el kernel no monte automáticamente la carpeta /proc y deberemos montarla nosotros mismos, manualmente, en el archivo /etc/fstab. Lo hacemos agregando la siguiente línea:

```
proc /proc proc defaults,hidepid=2 0 0
```

hidepid=2: Oculta completamente los procesos de otros usuarios, incluido el comando ejecutado.