_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DESARROLLO DE UN HONEYPOT PARA LA MONITORIZACIÓN Y PREVENCIÓN DE ATAQUES

Sergio Espí Luis.

Tutor: José Enrique López Patiño.

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación.

Curso 2016-17

Valencia, 3 de julio de 2017

Resumen

En el presente proyecto vamos a implementar dos Honeypots, Kippo y Dionaea, sobre la plataforma portátil Raspberry Pi 3, con la intención de recopilar información relativa al atacante para que posteriormente nos permita prevenir un ataque desde la misma ubicación. Utilizando dichos Honeypots emularemos tres servicios, SSH, FTP y HTTP. Estos servicios se encontraran desplegados en un entorno controlado, que solo permite a un atacante interactuar con el sistema trampa. Una vez desplegados los Honeypots y obtenido la información que nos interesa, principalmente direcciones IP, procederemos a desarrollar unos scripts que nos permitan ejecutar comandos de bloqueo en Iptables, un Firewall de Linux. Esto impedirá que sea posible volver a realizar un ataque desde la misma dirección IP. A su vez, también desarrollaremos un script en python que almacene todas las direcciones IP atacantes, y que nos permita comprobar desde otro sistema si una dirección IP a priori atacante se encuentra en nuestra lista o no.

Resum

En el present projecte implementarem dos Honeypots, Kippo i Dionaea, sobre la plataforma portàtil Raspberry Pi 3, amb la intenció de recopilar informació relativa a l'atacant per a què posteriorment ens permeta previndre un atac des de la mateixa ubicació. Utilitzant aquestos Honeypots emularem tres servicis, SSH, FTP i HTTP. Estos servicis els trobarem desplegats en un entorn controlat, que només permet a un atacant interactuar amb el sistema trampa. Una vegada desplegats els Honeypots i obtinguda la informació que ens interessa, principalment direccions IP, procedirem a desarrotllar uns scripts que ens permeten executar comandos de bloqueig en Iptables, un Firewall de Linux. Açò impedirà que siga possible tornar a realitzar un atac des de la mateixa direcció IP. Al seu torn, també desenrotllarem un script en python que emmagatzeme totes les direccions IP atacants, i que ens permeta comprovar des d'un altre sistema si una direcció IP a priori atacant està en la nostra llista o no.

Abstract

In this project we are going to implement two Honeypots, Kippo and Dionaea, on the Raspberry Pi 3 portable platform, with the intention to collect information about the attacker in order to allow us to prevent an attack from the same location. Using these Honeypots we emulate three services, SSH, FTP, and HTTP. These services will be deployed in a controlled environment, which only allows an attacker to interact with the system trap. Once deployed Honeypots and obtained the information that interests us, mainly IP addresses, we will proceed to develop some scripts that allow us to execute commands on Iptables, a Linux Firewall. This will prevent that it is possible to carry out an attack from the same IP address. At the same time, we will also develop a script in Python that store all IP addresses attackers, and to allow us to check from another system if an IP address attacker is in our list or not.

Agradecimientos

Sin duda, una de las partes más importantes de un proyecto son las personas que lo hacen posible, aunque sería injusto no mirar hacia atrás y agradecer también el esfuerzo de todas las personas que han hecho posible llegar hasta aquí.

Por ello quiero agradecer a toda mi familia así como a mis amigos el apoyo que me han brindado en los momentos más duros. Por supuesto, no es posible olvidarse de las personas que han hecho posible este proyecto, una de ellas ha sido mi tutor, José Enrique López Patiño, que me ha ayudado siempre que lo he necesitado, aunque sin duda Joan Soriano Aguilar ha sido un pilar fundamental en este proyecto y sin su ayuda todo esto no hubiera sido posible.

Índice

Capítulo 1.

Introducción

1.1 Motivaciones

La seguridad informática abarca gran cantidad de campos, por lo que antes de empezar este proyecto nos preguntarnos sobre que ámbito profundizar. En este instante nos encontramos con una herramienta perfecta para analizar gran parte de la actividad ilícita que se comete en internet, un Honeypot.

Los ataques cibernéticos existen desde hace muchos años aunque hoy en día están cobrando mayor importancia dada su repercusión. Recientemente se descubrió Mirai, un malware con el que se consiguieron controlar millones de dispositivos, principalmente cámaras IP. Dichos equipos fueron adheridos a una botnet que posteriormente se utilizó para dejar sin servicio a uno de los proveedores de DNS más importantes de EEUU, lo que inmediatamente provoco la caída de numerosos servicios, como Facebook o Twitter. Como Mirai, recientemente se difundió WannaCry, un Ramsonware que infectó y cifró gran cantidad de equipos a compañías importantes como Telefónica, pidiendo posteriormente un rescate monetario para recuperar la información del equipo, cosa que rara vez ocurre.

Tal y como veremos más adelante, una idea errónea acerca de los Honeypots es creer que son dispositivos diseñados para atraer a los atacantes. Creo que esto no es cierto, ya que los servicios emulados por Honeypots no se anuncian en la red para que cualquiera los ataque, sino que los mismos atacantes los encuentran por sus propios medios y con relativa facilidad, como ya hemos visto empleando Mirai o WannaCry. Todo esto hace pensar que los Honeypots simplemente están diseñados para capturar su actividad.

Otro gran error es pensar que tu equipo no es importante para los atacantes por su poco valor, pero todo el mundo es un blanco, ya que los atacantes tienen como objetivo atacar al mayor número de equipos posibles con el mínimo esfuerzo requerido, pudiendo usarlos en su beneficio para, por ejemplo, atacar otro sistema.

Como vemos, un Honeypot es muy diferente de la mayoría de los mecanismos tradicionales de seguridad, ya que su valor radica en ser atacado, si el sistema nunca es atacado, entonces tiene poco o ningún valor. Sin embargo, hay que evitar que resulte demasiado obvio que se trata de un sistema trampa, ya que de ser así cualquier atacante descubrirá y evitara todo contacto con él. Por supuesto, hay que tener en cuenta que permitir que un atacante consiga el control del sistema entraña numerosos peligros, por lo que habrá que asegurar que el atacante no pueda hacerse con el control total del sistema ni destruir datos. Esto significa que al designar cualquier recurso como Honeypot, nuestras expectativas y metas se centran en que el sistema sea atacado y potencialmente explotado. Esto es exactamente lo contrario que ocurre en la mayoría de los sistemas, en los que no se desea que se produzca un ataque. Por ello, hay que tener en cuenta que no se ha diseñado para recibir tráfico legítimo por lo que cualquier persona o recurso que se comunique con él debe ser considerado como malicioso.

Después de conocer todo esto pasamos a planificar donde, como y con qué finalidad desplegamos el Honeypot, poniendo especial interés en lo que pretendemos extraer de ellos, ya que nuestro objetivo principal no se centra en los Honeypots en sí, sino más bien en extraer de ellos información relevante acerca del atacante. Esta información nos permitirá prevenir posteriores ataques en un futuro, ya que los Honeypots pueden servir tanto para la detección de atacantes internos como externos a la red donde están alojados, y se debe tener siempre en cuenta la posibilidad de establecer Honeypots internos para la detección de estos atacantes, que a veces pueden llegar a ser más peligrosos que los externos, incluso sin darse cuenta.

Para conseguirlo emplearemos la Raspberry Pi 3. La hemos elegido dadas sus reducidas dimensiones y la posibilidad de ser portable así como la potencia de su sistema operativo Raspbian Jessie, una distribución Linux basada en Debian. También deberemos tener en cuenta que los Honeypots deben integrarse con el resto de herramientas del sistema, de tal manera que sea posible asegurar que no interfieran con las medidas de seguridad que puedan existir en la red, como por ejemplo el Firewall. En última instancia, para prevenir futuros ataques en él sistema real, utilizaremos Iptables, un Firewall de Linux, para bloquear las direcciones IP que consigamos extraer del Honeypot.

1.2 Objetivos

Teniendo claro el objetivo principal del proyecto procedemos a decidir qué servicios desplegaremos, que datos pretendemos obtener y que se espera conseguir a posteriori.

Llegados a este punto, y después de estudiar gran cantidad de posibilidades, vamos a centrarnos en los Honeypots que implementaremos para este proyecto, siendo estos Kippo y Dionaea. Básicamente, el primero de ellos se centra en el emular el servicio SSH, mientras que el otro se centra en emular servicios FTP y HTTP. Implementaremos estos servicios dado que pretendemos simular que nuestro equipo es un Linux Server, por lo que emular servicios propios de otros sistemas operativos podría alertar al atacante, lo que provocaría pérdida de información. Si esto sucediese no conseguiríamos extraer la información que necesitamos y por consiguiente no podríamos cumplir nuestro objetivo principal, prevenir futuros ataques.

Para conseguir nuestro objetivo centraremos nuestra atención en la prevención de dichos ataques, monitorizando toda la actividad relativa a dichos Honeypots, obteniendo archivos con grandes cantidades de datos acerca de los atacantes, entre los que se encontraran sus direcciones IPs así como sus métodos de ataque. Esto nos permitirá posteriormente bloquear dichas IPs, impidiendo que vuelvan a realizar un ataque desde la misma ubicación. Para conseguir esto, desplegaremos dos Honeypots, simulando tres servicios funcionando en un sistema aparentemente real, con la particularidad de que estará apartado del sistema operativo de la Raspberry, Raspbian, y que los datos expuestos en este software virtual, con los que el atacante interactuara, carecen de importancia. Esto provocará que el atacante crea que ha conseguido acceso a un sistema real, pero no será así ya que nosotros monitorizaremos toda su sesión y recopilaremos datos que nos permitirán bloquearlo mediante Iptables, un potente Firewall de Linux, y que no pueda volver a tener acceso al sistema posteriormente.

Llegados a este punto, estudiaremos la viabilidad de desarrollar una herramienta que permita, tanto a los propios sistemas reales de la organización como a cualquier usuario externo, consultar si una dirección IP se encuentra en nuestra lista de direcciones atacantes.

1.3 Fases y estructura del proyecto

Como podremos ver más adelante, el proyecto está dividido fundamentalmente en cuatro partes.

La primera de ellas consta del Capítulo 1 y 2, en esta fase empezamos a indagar en el mundo de los Honeypots apoyándonos en grandes proyectos como lo es The Honeynet Project o los estudios realizados por Lance Spitzner. Es por ello que nos centraremos en explicar qué es un Honeypot y su funcionamiento, así como sus ventajas y desventajas, ubicación, clasificación, tipos, etc. En esta parte también explicaremos el porqué de este proyecto así como que se pretende conseguir con él.

Después de conocer más información acerca de esta herramienta, nos disponemos a buscar información acerca de cómo implementarla así como los recursos y componentes que necesitamos.

En segundo lugar tenemos el Capítulo 3 y el Anexo I, que contienen toda la fase relativa al análisis de recursos y componentes, que derivan en el posterior desarrollo e implementación de todas las herramientas que componen el proyecto.

Seguidamente, en el Capítulo 4 podremos observar los resultados extraídos de los Honeypots, así como de las otras herramientas implementadas.

Para finalizar expondremos los usos futuros a los que puede aplicarse la idea principal de la que parte este proyecto.

Capítulo 2.

Introducción a los Honeypots

2.1 Definición

No es posible referirse a un término correctamente sin previamente hablar acerca de su historia, el término Honeypot o “tarro de miel” fue acuñado durante la Guerra Fría para designar una técnica de espionaje, definiéndose en el campo de la seguridad informática como un recurso que no se ha diseñado para recibir tráfico legítimo, sino para aparentar servicios, sistemas operativos o incluso redes enteras, que puedan resultar atractivos para un eventual atacante y que presenten vulnerabilidades fáciles de explotar, alejando al intruso de los recursos reales a la vez que se monitoriza toda esa actividad malintencionada.

Desde hace años, la falta de una definición clara y ampliamente aceptada de un Honeypot es una de las principales razones por las que la comunidad de expertos en seguridad ha tardado tanto en adoptarlos. Y es que, cada uno tiene su propia definición de un Honeypot, esto crea una gran confusión y falta de comunicación, que derivan en una falta de implementación de estos sistemas. Algunos piensan que un Honeypot es una herramienta para el engaño, mientras que otros consideran que es un arma para atraer a los piratas informáticos, y otros creen que es simplemente otra herramienta de detección de intrusos. Por otro lado, algunos creen que un Honeypot debe emular vulnerabilidades, mientras que otros lo ven simplemente como una cárcel. Estos diferentes puntos de vista han causado grandes discrepancias acerca de lo que es un Honeypot y, por lo tanto su valor. Para los propósitos de este trabajo, vamos a definir un Honeypot como un recurso de seguridad cuyo valor radica en ser sondeado, agredido o comprometido.

Esto significa que al designar cualquier recurso como Honeypot, nuestras expectativas y metas se centran en que el sistema sea sondeado, atacado y potencialmente explotado. No importa cuál sea el recurso (en nuestro caso serán scripts de funcionamiento de los servicios emulados junto con un entorno cerrado), lo que importa es que el valor del recurso radica en ser atacado. Si el sistema nunca es atacado, entonces tiene poco o ningún valor. Esto es exactamente lo contrario que ocurre en la mayoría de los sistemas, en los que no se desea que se produzca un ataque. Como es evidente a partir de esta definición, los Honeypots son diferentes de la mayoría de las herramientas de seguridad, ya que ofrecen un gran abanico de posibilidades, aunque siempre hay que tener en cuenta que no se ha diseñado para recibir tráfico legítimo por lo que cualquier persona o recurso que se comunique con él debe ser considerado como malicioso.

2.2 Funcionamiento

Un Honeypot puede ser tan simple como un host que ejecuta un software, un script… con la intención de analizar el tráfico que entra y sale del host, escuchando en cualquier número de puerto. También puede ser una compleja red de ordenadores totalmente operativa o por otro lado una Honeynet en la que se simularía una red aparentemente operativa pero en la que sus host serían virtuales.

Una idea errónea acerca de los Honeypots es creer que son dispositivos diseñados para atraer a los atacantes. Creo que esto no es cierto, ya que los Honeypots no se anuncian para que cualquiera los ataque, sino que los mismos atacantes los encuentran por sus propios medios y con relativa facilidad. Por ello, pienso que los Honeypots simplemente están diseñados para capturar su actividad. Otro gran error es pensar que tu equipo no es importante para los atacantes por su poco valor, pero todo el mundo es un blanco, ya que los atacantes tienen como objetivo atacar al mayor número de equipos posibles con el mínimo esfuerzo requerido, pudiendo usarlos en su beneficio para, por ejemplo, atacar otro sistema.

Por todo lo mencionado anteriormente, podemos afirmar un Honeypot debe cumplir una serie de funciones principales, entre las que se encuentran:

-Desviar la atención del atacante para salvar al sistema principal y disuadirle de seguir adelante con la intrusión o ganar un tiempo muy valioso que nos permita reaccionar y tomar las medidas oportunas para frenar el ataque.

-Capturar nuevos tipos de malware para su posterior estudio.

- Poder obtener una base de datos con direcciones IP de atacantes y métodos de ataque desconocidos.

- Poder conocer nuevas vulnerabilidades y de esta manera poder aplicar medidas para que no afecten a la seguridad de nuestros sistemas.

En este trabajo nos centraremos en obtener las direcciones IP de los atacantes, para que posteriormente puedan ser bloqueadas.

2.3 Clasificación

La clasificación de los Honeypots debe dividirse según su ambiente de implementación y según su nivel de interacción.

2.3.1 Honeypots según su ambiente de Implementación

Dependiendo del ambiente de implementación de los Honeypot, podemos definir dos tipos de Honeypots: para la Producción y para la Investigación.

2.3.1.1. Honeypots para la Producción

Son aquellos que se utilizan para proteger a las organizaciones en ambientes reales. Están sujetas a ataques constantes las 24 horas del día y durante los 7 días de la semana. Se les concede cada vez más importancia debido a las herramientas de detección que pueden proporcionar y por la forma en cómo pueden complementar la protección en la red y en los hosts. Por ello se considera que añaden valor a la seguridad de una organización específica y ayudar a mitigar el riesgo, sin embargo nos da menos información sobre los ataques o los atacantes que los Honeypots desarrollados para la investigación. Podremos conocer qué sistemas utilizan los atacantes o lo que exploit lanzan, pero lo más probable es que no vamos a poder aprender cómo se comunican entre sí o cómo se desarrollan sus herramientas.

2.3.1.2. Honeypots para la Investigación

Este tipo de Honeypots no son implementados con la finalidad de proteger a alguna organización, sino que constituyen recursos educativos de naturaleza demostrativa y de investigación, cuyo fin se centra en el estudio de patrones de ataque y amenazas de todo tipo. Gran parte de la atención actual se centra en este tipo de Honeypots, que son utilizados para recolectar información sobre las acciones de los atacantes. Esta información nos permite comprender mejor quiénes son nuestras amenazas y cómo funcionan. Teniendo este conocimiento podemos protegernos mejor de ellas.

Un ejemplo referente a este tipo de Honeypots lo podemos encontrar en una organización llamada The Honeynet Project, en la que se realizan investigaciones sobre seguridad utilizando Honeypots para recolectar información sobre todo tipo de ataques en la red.

2.3.2. Honeypots según su Nivel de Interacción

Este tipo de clasificación se basa en el nivel de libertad que tiene el atacante al estar en contacto con el Honeypot, consiste en la interacción que existe entre estos dos, mientras más tiempo de interacción exista mayor cantidad de información podemos extraer del ataque.

2.3.2.1 Honeypots de Baja Interacción

Normalmente estos Honeypots emulan servicios y sistemas operativos y la actividad del atacante se encuentra limitada a dicho nivel de emulación. La ventaja de un Honeypot de baja interacción radica en su simplicidad, ya que estos tienden a ser fáciles de utilizar y mantener con un riesgo prácticamente nulo. Por ejemplo, un servicio FTP emulado, es el que está escuchando en el puerto 21, y que probablemente estará emulando algún login FTP o probablemente soportará algunos comandos de FTP adicionales, pero no es un riesgo para la seguridad, ya que lo más probable es que no esté ligado a ningún servidor FTP real.

Por lo general, el proceso de implementación de un Honeypot de baja interacción se basa en una instalación de un software de emulación de sistema operativo. Los servicios emulados mitigan el riesgo de penetración en el sistema, conteniendo la actividad del atacante que nunca tiene acceso al sistema operativo real donde podría atacar o dañar otros sistemas.

La principal desventaja de este tipo de Honeypots, radica en que registran únicamente una información limitada, ya que están diseñados para capturar una actividad predeterminada. Por lo tanto, es relativamente sencillo para un intruso hábil detectar un Honeypot de baja interacción con el paso del tiempo. En nuestro caso implementaremos dos Honeypots de baja interacción, Kippo y Dionaea, y emularemos un servicio SSH utilizando Kippo, y un servicio FTP, HTTP, TFTP y HTTPS utilizando Dionaea, pero de esto hablaremos más adelante.

2.3.2.2 Honeypots de Alta Interacción

Implementar este tipo de Honeypots implica cierta complejidad, ya que hace necesaria la utilización de sistemas operativos y aplicaciones reales montados sobre hardware real, sin la utilización de software de emulación e involucrando aplicaciones reales que se ejecutan de manera normal. Por ejemplo, si se desea implementar un Honeypot sobre un servidor Linux que ejecute un servidor FTP, se tendrá que construir un verdadero sistema Linux y montar un verdadero servidor FTP.

En lo referente a las ventajas de este tipo de solución, podemos citar dos: Por un lado, se posee la capacidad de capturar grandes cantidades de información referentes al modo de operación de los atacantes debido a que los intrusos se encuentran frente a un sistema real. De esta forma, se está en posibilidad de estudiar todas sus actividades.

Sin embargo esta última ventaja, también cuenta con un inconveniente, ya que incrementa el riesgo de que los atacantes puedan utilizar este sistema para atacar a los dispositivos de dicha red que no forman parte de estos Honeypots. Por lo tanto, se requiere la implementación de una tecnología adicional que prevenga al atacante a dañar a otros sistemas que no sea el propio Honeypot. Uno de los mejores ejemplos de un Honeypot de alta interacción son las Honeynets.

2.4 Ubicación

Para que una trampa sea eficaz debe ser desplegada en la ubicación correcta. La decisión de dónde desplegar el Honeypot depende de lo que se quiera conseguir, teniendo en cuenta que tienen un carácter pasivo y no tendrán valor alguno si el atacante no puede acceder a ellos. Una ubicación de difícil acceso quitara valor al Honeypot, dado que será más complicado obtener datos. Por otro lado, si su ubicación es demasiado obvia cualquier atacante descubrirá y evitara todo contacto con el Honeypot.

Debe tenerse en cuenta que los Honeypots deben integrarse con el resto de herramientas del sistema, de tal manera que sea posible asegurar que no interfieran con las medidas de seguridad que puedan existir en la red, como por ejemplo el Firewall, IDS, etc.

Los Honeypots pueden servir tanto para la detección de atacantes internos como externos a la red donde están alojados, y se debe tener siempre en cuenta la posibilidad de establecer Honeypots internos para la detección de estos atacantes internos a la red.

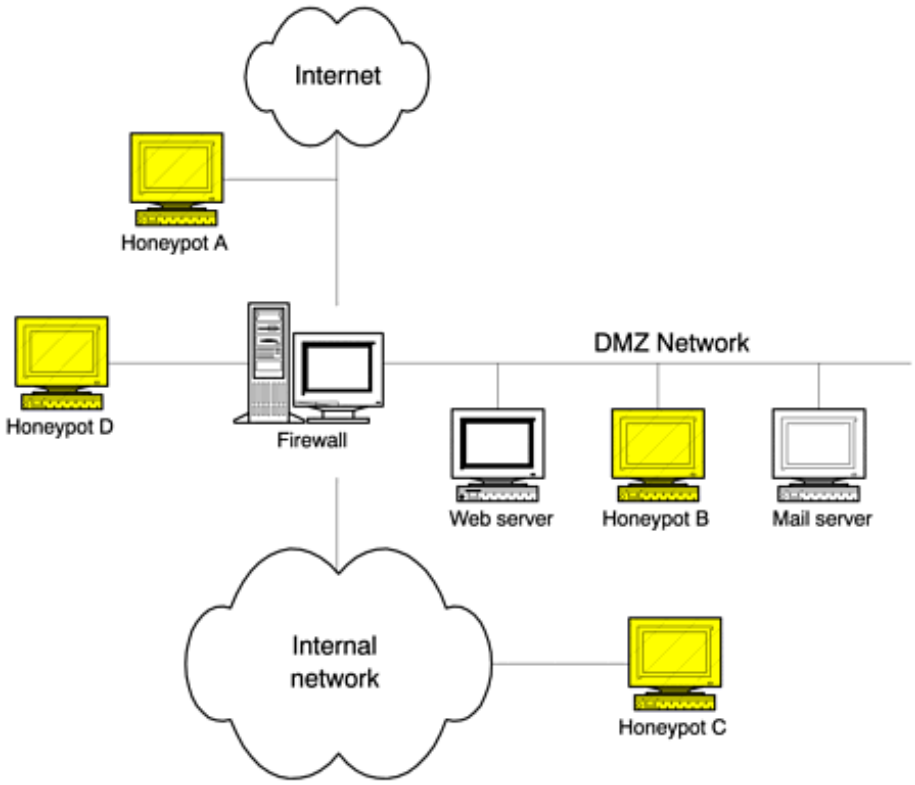


Figura 1. Posibles ubicaciones del Honeypot.

En la imagen se pueden observar las diferentes ubicaciones en la cuales se puede desplegar un Honeypot. Las ubicaciones en las que tienen mayor valor son la zona DMZ y en la red interna, Honeypots B y C en la figura, pero esto nos centraremos más a continuación. En nuestro caso los Honeypots estarán ubicados en la red interna, concretamente en la dirección 192.168.1.111. Nuestros Honeypots tienen la intención de engañar a los atacantes que consiguen penetrar con éxito el servidor de seguridad. Estos Honeypots confundirían atacantes, desperdiciando su tiempo y recursos.

2.4.1. Antes del Firewall

Este tipo de localización permitirá evitar el incremento del riesgo innato a la situación del Honeypot, ya que al encontrarse fuera de la zona protegida por el firewall puede ser atacado sin ningún tipo de peligro para el resto de la red en la que encuentra. Esta situación evitará otro tipo de alarmas de otros sistemas de seguridad, por ejemplo IDS, al recibir los ataques.

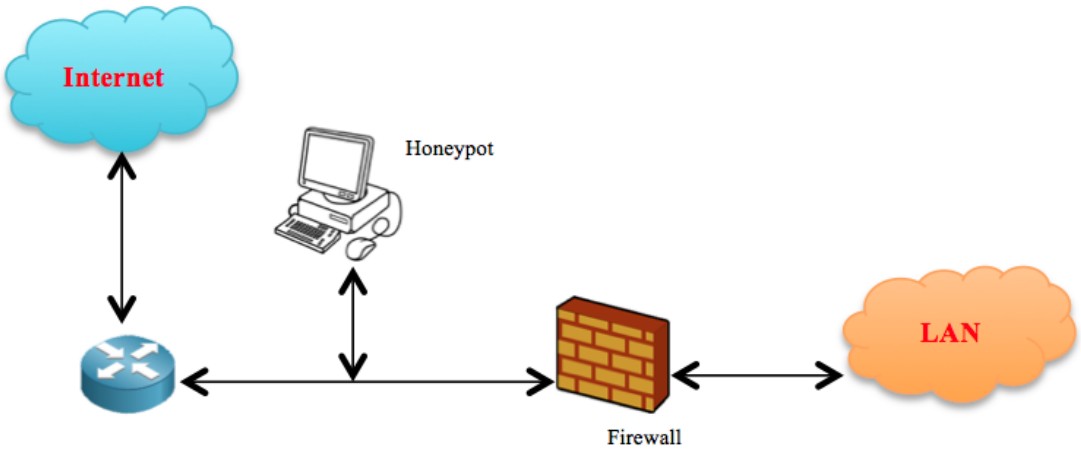


Figura 2. Honeypot antes del firewall.

En este escenario, cualquier atacante externo lo primero que se va a encontrar es el Honeypot y esto generará un gran consumo de ancho de banda y espacio en los ficheros ".log". Por otro lado, esta ubicación evita la detección de los atacantes internos. Pero es interesante a la hora de recibir todos los ataques que puede recibir una red, sin que estos sean frenados por un Firewall. Sin embargo existe el peligro de generar un tráfico excesivo, debido precisamente a la facilidad que ofrece el Honeypot para ser atacado.

2.4.2. Detrás del Firewall

En este escenario, el Honeypot queda afectado por las reglas de filtrado del Firewall. Por un lado se tiene que modificar las reglas para permitir algún tipo de acceso al Honeypot por posibles atacantes externos, pero por otro lado al introducir un elemento potencialmente peligroso dentro de la red interna, se puede permitir a un atacante que gane el acceso al Honeypot y a la red. En nuestro caso, lo que haremos para permitir dicho acceso, será redirigir las conexiones que se hagan a la red intentando conectar con un determinado número de puerto a la Raspberry. Cuando esto ocurra el atacante conseguirá acceso al Honeypot y empezara la captura de datos.

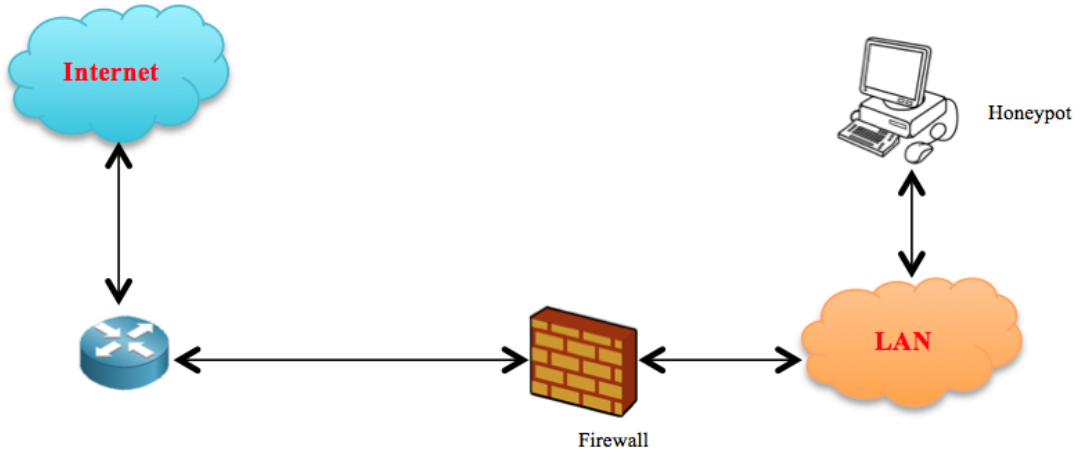


Figura 3. Honeypot detrás del firewall, en la red interna

Este tipo de ubicación, permite la detección de atacantes internos así como Firewalls mal configurados, máquinas infectadas por gusanos, virus… Sin embargo, se generan una gran cantidad de alertas de seguridad provocadas por los sistemas de detección de intrusos tales como Firewalls, IDS… Al recibir ataques, el Honeypot se ve en la necesidad de asegurar al resto de nuestra red contra el mismo Honeypot mediante el uso de Firewall extra, dentro de la propia red.

No obstante, hay varias circunstancias que obligan la utilización de esta arquitectura, como por ejemplo la detección de atacantes internos o la imposibilidad de utilizar una dirección IP externa para el Honeypot.

2.4.3. En una Zona Desmilitarizada (DMZ)

Esta ubicación permite por un lado juntar a los servidores de producción con el Honeypot y por el otro controlar el peligro que añade su uso, ya que tiene un firewall que lo aísla del resto de la red local.

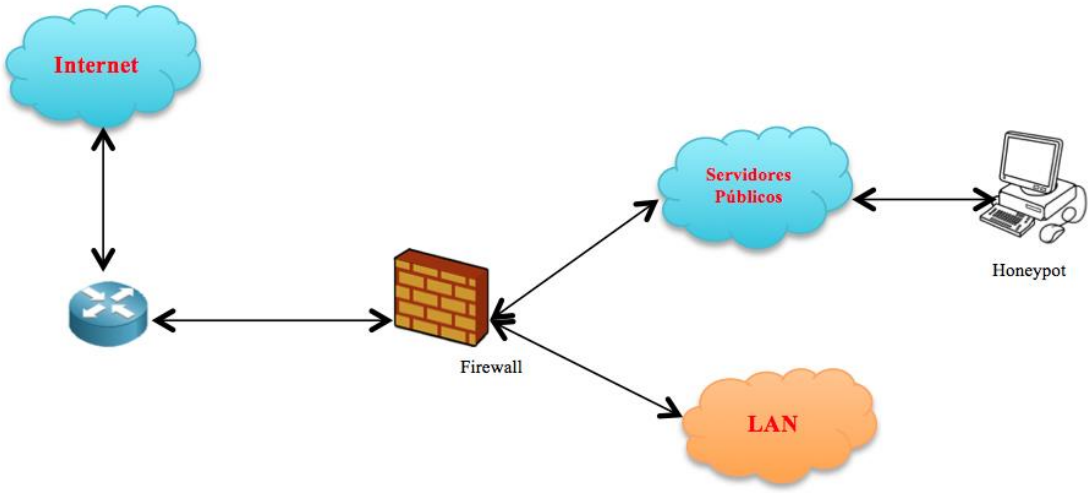


Figura 4. Honeypot en la zona DMZ.

Este tipo de arquitectura nos permite tener la posibilidad de detectar ataques externos e internos con una simple reconfiguración del Firewall, ya que se encuentra en una zona de acceso público.

Por otro lado, con este tipo de ubicación se eliminan las alertas producidas por los sistemas de detección de intrusos, IDS, y el peligro que supone a la red al no estar en contacto directo con los ataques.

No obstante la detección de atacantes internos se ve algo debilitada, ya que al no compartir el mismo segmento de la red LAN, un atacante local no accederá al Honeypot a menos que este desee hacerlo y por lo tanto quede detectado por el Honeypot.

2.5 Ventajas y desventajas

Ahora que hemos definido más concretamente que son los Honeypots y cómo funcionan, podemos tratar de establecer su valor. Tal y como se mencionó anteriormente, a diferencia de mecanismos como cortafuegos y sistemas de detección de intrusos, un Honeypot no aborda un problema específico. En cambio, es una herramienta que contribuye a la arquitectura general de la seguridad. El valor de los Honeypots y los problemas que ayudan a resolver dependen de la forma de desplegarlos, su ubicación y su objetivo.

Como hemos visto los Honeypots tienen muchas ventajas, lo que no quita que también tengan algunas desventajas. Ahora vamos a examinar esas ventajas y desventajas más de cerca, empezando por sus ventajas.

- Los Honeypots son ordenadores a los que ningún usuario debe acceder. Permitiendo de esta forma, relevar cualquier tipo de acceso del atacante o una configuración errónea del sistema, sin llegar a tener prácticamente falsos positivos.

- Se necesitan recursos mínimos, ya que a diferencia de otro tipo de sistemas de seguridad, sus requisitos son mínimos, prácticamente no consume ni ancho de banda ni memoria. No necesita complejas arquitecturas o un gran número de ordenadores centralizados, cualquier ordenador conectado a la red puede realizar el trabajo de un Honeypot.

- Es un tipo de sistema que sirve tanto para atacantes internos como externos. Su objetivo es el de pasar de manera desapercibida en una red como una máquina más.

- Generan un volumen pequeño de datos siempre que estén desplegados en la ubicación correcta y al contrario que los demás sistemas de seguridad, como Firewall o IDS, que generan un gran volumen de datos conteniendo incluso información que no es necesaria, mientras que los Honeypots generan información con muy pocos datos pero de muy alto valor.

Por otro lado, como todo tipo de sistema tiene también una serie de desventajas, las cuales son:

- Son elementos totalmente pasivos. De esta forma, si no reciben ningún ataque no sirven para gran cosa, por ello siempre tienen que estar en el lugar que les corresponde.

- Son fuentes potenciales de riesgo para nuestra red. Debido a la atracción que ejercen sobre los atacantes, de tal manera que si no medimos su alcance y lo convertimos en un entorno controlado puede ser utilizado como fuente para ataques a otras redes o incluso a la propia red.

- Tienen una visión limitada, ya que solo pueden rastrear y capturar cierta actividad.

Debido a estos inconvenientes, podemos afirmar que los Honeypots no sustituyen a ningún mecanismo de seguridad, sino que trabajan conjuntamente permitiendo mejorar su perímetro de seguridad.

2.6 Honeypots empleados

Llegados a este punto, y después de estudiar gran cantidad de posibilidades, vamos a centrarnos en los Honeypots que implementaremos para este proyecto, siendo estos Kippo y Dionaea. Básicamente, el primero de ellos se centra en el emular el servicio SSH, mientras que el otro se centra en emular servicios ftp y http. Ahora vamos a centrarnos en explicar un poquito más en cada uno de ellos.

- Kippo:

Es uno de los Honeypots de baja interacción más utilizados, ya que su nivel de análisis y detección es bastante potente, tal y como explicaremos más adelante. Kippo es un Honeypot que emula un servicio SSH, detecta intrusiones, y ataques de fuerza bruta efectuados sobre la red en la que este implementado. Hay que tener en cuenta que la gran mayoría de ataques producidos en una red son realizados una vez el atacante ha tomado el control, y por lo tanto es interesante implementar un Honeypot que nos detecte este tipo de ataques y tengamos la posibilidad de mitigar el ataque desde la raíz. Más adelante, nos centraremos en explicar el funcionamiento y las características de dicho Honeypot, así como su instalación paso a paso.

- Dionaea:

Dionaea es un Honeypot de baja interacción realmente interesante, ya que su principal objetivo es la captura y análisis de muestras de malware. Es capaz de desplegar varios tipos de servicios y esperar a que los atacantes intenten hacerse con el control de dicho servicio por medio de peticiones maliciosas y el envío de payloads. De entre dichos servicios, en este proyecto solo implementaremos un servicio ftp y un servicio http, ya que de no ser así el atacante podría llegar a pensar que no se trata de un sistema real y borrar su rastro, por lo que el Honeypot no tendría ningún valor. Más adelante, nos centraremos en explicar el funcionamiento y las características de dicho Honeypot, así como su instalación paso a paso.

Capítulo 3.

Diseño e Implementación

3.1 Introducción

Tal y como hemos mencionado en el capítulo anterior, el valor de un Honeypot radica en ser atacado, si el sistema nunca es atacado, entonces tiene poco o ningún valor. Sin embargo, hay que evitar que resulte demasiado obvio que se trata de un sistema trampa, ya que de ser así cualquier atacante descubrirá y evitara todo contacto con el Honeypot. Por supuesto, hay que tener en cuenta que permitir que un atacante consiga el control del sistema entraña numerosos peligros, por lo que habrá que asegurar que el atacante no pueda hacerse con el control total del sistema ni destruir datos.

Para ello centraremos nuestra atención en la prevención de dichos ataques, teniendo como objetivo monitorizar toda la actividad relativa a dichos Honeypots, obteniendo las IPs atacantes así como sus métodos de ataque. Esto nos permitirá posteriormente bloquear dichas IPs, impidiendo que vuelvan a realizar un ataque desde la misma ubicación, pero en esto nos centraremos más adelante. Para conseguir esto, desplegaremos dos Honeypots, simulando tres servicios funcionando en un sistema aparentemente real, con la particularidad de que estará apartado del sistema operativo de la Raspberry, Raspbian, y que los datos expuestos en este software virtual, con los que el atacante interactuara, carecen de importancia. Esto provocará que el atacante crea que ha conseguido acceso a un sistema real, pero no será así ya que nosotros monitorizaremos toda su sesión y recopilaremos datos que nos permitirán bloquearlo y que no pueda volver a tener acceso posteriormente.

También es cierto que los Honeypots que implementaremos son de baja interacción, por lo que no recopilaremos gran cantidad de datos del atacante, pero si los suficientes para conseguir nuestro objetivo y que no pueda volver a tener acceso al sistema. Para ello, vamos a proceder a desplegar ciertas herramientas y en los siguientes apartados explicaremos cada una de ellas con detalle.

3.2 Análisis de objetivos

En primer lugar, empezamos por preguntarnos sobre que ámbito de la seguridad profundizar cuando nos encontramos con una herramienta perfecta para analizar la actividad ilícita que se comete en internet, y después de esto pasamos a planificar donde, como y con qué finalidad desplegamos el Honeypot.

Teniendo claro el objetivo del proyecto procedemos a decidir qué servicios desplegaremos, sobre qué sistema lo haremos y que datos que se pretenden recoger. Dado que el objetivo es conseguir un sistema que simplemente pretende recoger información de muy bajo nivel, sobre todo direcciones IP, implementaremos Honeypots de baja interacción que nos permitirán conocer dicha información sin ningún tipo de problema, evitando poner al sistema principal en riesgo. A su vez, dado que serán desplegados en una Raspberry pi 3, tampoco sería viable

implementar toda una Honeynet con muchos host virtuales y emulando gran cantidad de servicios, ya que no dispondríamos de suficiente capacidad de computación. Llegados a este punto, deberemos ver cuáles son los Honeypots que mejor se adaptan a nuestras necesidades, así como donde debemos ubicarlos.

Después de estudiar gran cantidad de posibilidades, nos decantamos por desplegar Kippo y Dionaea por su capacidad de análisis, síntesis y obtención de datos, así como su sencillez y a la vez eficaz forma de trabajar.

Podríamos haber implementado más Honeypots, y en consecuencia emulado más servicios, pero tal y como comentamos en el capítulo anterior un atacante con cierta experiencia se daría cuenta rápidamente que se trata de una trampa, ya que no existiría concordancia entre los equipos que contiene el sistema con los servicios que emula. Tampoco lo hemos hecho por una cuestión de recursos del sistema, dado que al aumentar el número de Honeypots se debería aumentar tanto su capacidad de computación como su memoria.

Ahora habrá que tener en cuenta los recursos que necesitaremos para llevar a cabo el proyecto.

3.3 Análisis de recursos y componentes

En este punto, necesitamos definir la arquitectura sobre la que desplegaremos nuestros Honeypots, siendo la mejor opción la plataforma portátil Raspberry pi 3. Nos hemos decidido a usar la Raspberry dadas sus reducidas dimensiones y la posibilidad de ser portable así como la potencia de su sistema operativo Raspbian Jessie, una distribución Linux basada en Debian.

En la imagen que aparece a continuación podemos observar la arquitectura de la Raspberry, pero hay que tener en cuenta que al no tener ningún sistema de refrigeración, el chip donde lleva la CPU, la GPU y la RAM y el chip que controla los puertos USB se calientan, siendo necesario añadir dos disipadores de calor, ya que si no podríamos provocar el sobrecalentamiento y la posterior rotura del equipo.

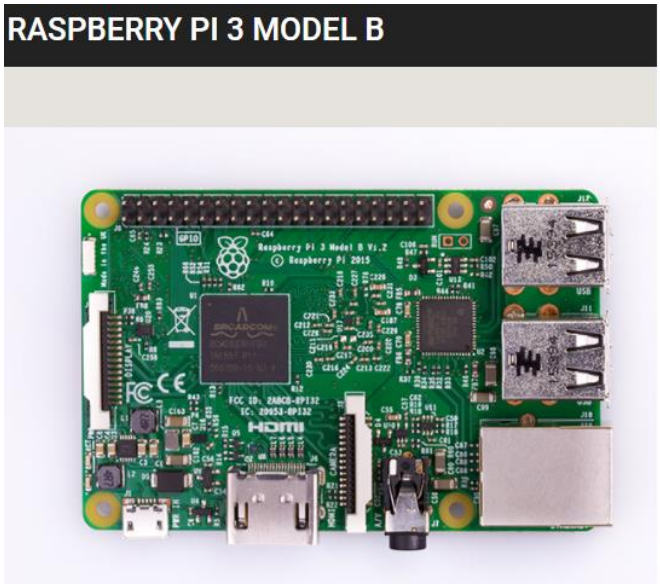


Figura 5. Arquitectura Raspberry Pi 3.

Los componentes adicionales para llevar a cabo el proyecto estarán relacionados con la parte software, pero hablaremos de ellos más adelante.

3.4 Fase de Diseño

Llegados a este punto podemos afirmar que los componentes necesarios para desplegar los Honeypots y llevar a cabo el objetivo del proyecto son:

- Un equipo real
- Una Raspberry pi 3
- Honeypot Kippo
- Honeypot Dionaea
- Sistemas de visualización de contenidos
- Script para obtener las direcciones IP y bloquear su acceso a posteriori

A continuación se explicara cada uno de dichos componentes.

- Equipo real: Se trata de un PC del que nos hemos valido para configurar el punto fuerte del proyecto, la Raspberry, para ello hemos utilizado herramientas como Putty, para conectarnos por SSH a la Raspberry, en la dirección 192.168.1.111 de la red interna, donde se encuentra ubicada.
  También hemos utilizado VNC, teniendo en este equipo instalado el VNC Viewer y en la Raspberry un VNC Server, que nos permitirá visualizar el escritorio del equipo remoto sin la necesidad de tenerlo conectado a una pantalla. Esto será posible desde dentro de la propia red, aunque también desde fuera, ya que abrimos en el router los puertos del 5901 al 5909. Tener esta última herramienta nos ha facilitado mucho el despliegue de todas las herramientas, ya que es mucho más cómodo que usando simplemente el Putty.
  A su vez, también hemos usado este equipo para instalar la imagen de Raspbian Jessie, utilizando para ello el Win32Disk Imager.

- Raspberry pi 3 Model B: Esta Raspberry Pi es la tercera generación de estos mini PC, y ha incorporado mejoras notables que proporcionan un rendimiento diez veces superior al de la Raspberry original. Podemos destacar mejoras tales como el procesador Broadcom quad-core ARM Cortex A53 a 1.2 GHz con una arquitectura de 64 bits, 802.11n Wireless LAN y Bluetooth 4.1. También ha mantenido las características de su antecesor, 1 Gb de RAM así como 4 puertos USB, HDMI, puerto Ethernet, salida de audio Jack y por supuesto lector de tarjetas micro sd. Esta tarjeta contendrá su disco duro y su disco de arranque, y permitirá que sea posible arrancar el dispositivo desde la imagen que contiene. En un principio partimos de una micro sd de 8 Gb, pero después de tener algunos problemas de espacio que no permitían el correcto funcionamiento del equipo nos vimos obligados a clonar dicha tarjeta a una de mayor tamaño, de 16 Gb. En lo referente a la alimentación del equipo, será alimentado por un micro USB de 5V y 2500 mA. Como comentamos anteriormente, su tamaño es muy reducido,

concretamente de 86 x 56 x20 mm, así como su precio, 35$, aunque incorporando la caja y algunos complementos sube un poco más.

- Honeypot Kippo: Kippo es un Honeypot de baja interacción que emula servicios de Secure Shell, SSH. Su función principal consiste en recopilar información acerca de los ataques de inicio de sesión utilizando fuerza bruta contra el servicio y sesión SSH. Cuenta con una serie de características que hacen que sea el Honeypot más utilizado de este tipo. Algunas de ellas consisten en almacenar todos los archivos que se han descargado durante una sesión SSH. A su vez también almacena información acerca del comportamiento del atacante en el sistema operativo, intentos de sesión con user y pass, comandos que utiliza el atacante, huellas digitales, etc.
  Hay que tener en cuenta que la gran mayoría de ataques producidos en una red son realizados una vez el atacante ha tomado el control, y por lo tanto es interesante implementar un Honeypot que nos detecte este tipo de ataques y tengamos la posibilidad de mitigar el ataque desde la raíz.

- Honeypot Dionaea: Tal y como hemos comentado en capítulos anteriores, Dionaea es un Honeypot de baja interacción que ofrece una gran variedad de servicios. Es capaz de desplegar varios tipos de servicios y esperar a que los atacantes intenten hacerse con el control de dicho servicio por medio de peticiones maliciosas. De entre dichos servicios, en este proyecto implementaremos un servicio ftp y un servicio http, ya que de no ser así el atacante podría llegar a pensar que no se trata de un sistema real y borrar su rastro, por lo que el Honeypot no tendría ningún valor.

- Sistemas de visualización de contenidos: Para visualizar correctamente los datos recogidos por cada uno de los Honeypots desplegados se ha implementado, para cada uno de ellos, su parte gráfica. Dicha parte permite visualizar de una manera más interactiva los logs, direcciones IP, etc, ya que puede resultar un poco engorroso visualizar todos estos datos en los propios archivos en los que el Honeypot los almacena.

- Script para bloquear las direcciones IP: Después de recoger todos los datos referentes al atacante necesitaremos cumplir con nuestro objetivo y bloquear su dirección para que no le sea posible volver a atacarnos. Es por ello que hemos desarrollado un script que nos permitirá bloquear dichas direcciones.

3.5 Fase de implementación

En este apartado se explicara el proceso de implementación de todas las herramientas usadas en el proyecto.

Para empezar, hay que destacar que la implementación se ha llevado a cabo en la red doméstica, por lo que hemos configurado el router para que sea posible acceder a la Raspberry desde fuera

de la red local. Para no tener problemas con la dirección IP de la Raspberry le hemos asignado una dirección IP estática, la 192.168.1.111. A su vez, también hemos redirigido cualquier conexión que se realice a nuestra red, 84.126.16.45, a la Raspberry en determinado número de puerto, tal y como se puede ver en la imagen.

En la imagen podemos ver que hemos redirigido cualquier conexión al puerto 22, en este puerto se encontrara corriendo el Honeypot Kippo, emulando un servicio SSH. Por ello hemos tenido que cambiar el verdadero servicio SSH al puerto 5555, para que así podamos conectarnos al equipo por SSH, ya que de intentar conectar en el puerto 22 nos encontraremos el Honeypot.

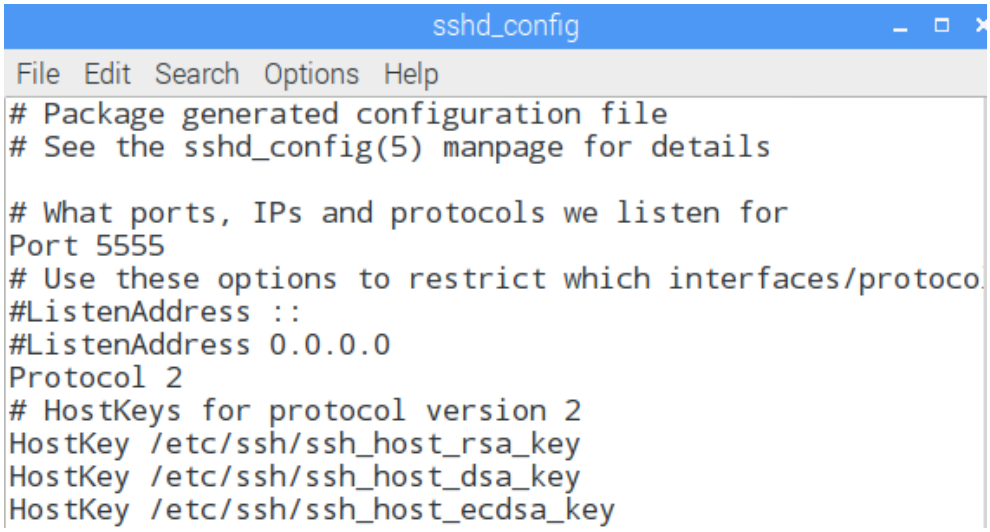


Figura 6. Cambio de puerto SSH.

También hemos redirigido cualquier conexión del puerto 5901 al 5909, esto es para poder utilizar el VNC Viewer desde fuera de la red local, ya que no siempre trabajamos en el mismo sitio donde se encuentra el equipo.

A su vez, hemos redirigido puerto 80 y el 443 para emular el servicio http y https de Dionaea, y el puerto 8000 para DionaeaFR. Así como el puerto 21 para ftp, el puerto 69 para tftp. Para que funcionen todos estos servicios habrá que abrir también el puerto 3306, correspondiente al MySQL.

Dirección IP externa: 84.126.16.45

Reenvío de puertos

Addr IP local	Externo		Interno		Protocolo	Activado	Eliminar
	Puerto inicial	Puerto final	Puerto inicial	Puerto final			
192.168.1.111	22	23	22	23	TCP	☑	☐
192.168.1.111	5555	5555	5555	5555	TCP	☑	☐
192.168.1.111	1	10	1	10	Ambos	☑	☐
192.168.1.111	5901	5909	5901	5909	TCP	☑	☐
192.168.1.111	80	80	80	80	TCP	☑	☐
192.168.1.111	443	443	443	443	TCP	☑	☐
192.168.1.111	20	21	20	21	TCP	☑	☐
192.168.1.111	69	69	69	69	UDP	☑	☐
192.168.1.111	3306	3306	3306	3306	Ambos	☑	☐
192.168.1.111	8000	8000	8000	8000	TCP	☑	☐

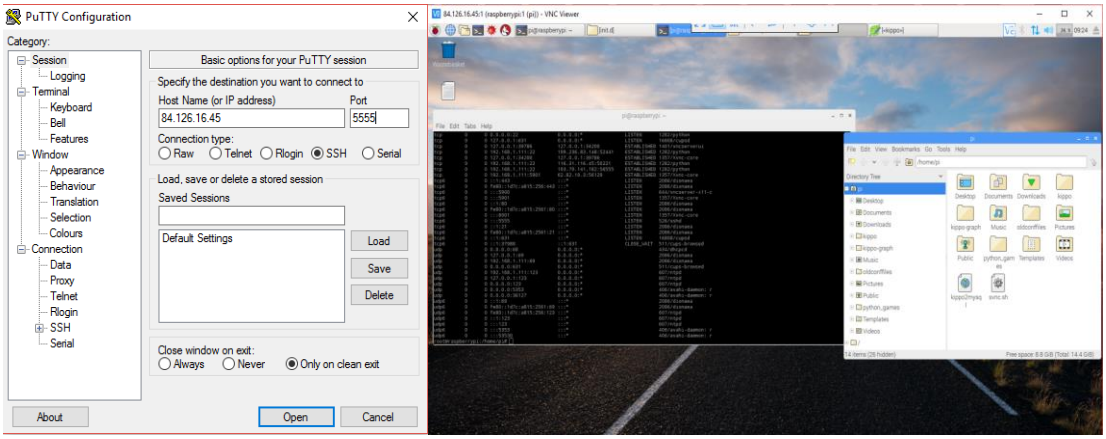
Figura 7. Puertos redirigidos en el router.

Podemos ver la coincidencia entre los puertos abiertos en el router y los puertos abiertos en la Raspberry.

```
root@raspberrypi:/home/pi# netstat -putan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:443           0.0.0.0:*               LISTEN      2086/dionaea
tcp        0      0 192.168.1.111:443       0.0.0.0:*               LISTEN      2086/dionaea
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN      30142/python
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      1025/mysqld
tcp        0      0 0.0.0.0:5900            0.0.0.0:*               LISTEN      22975/vncserver-x11
tcp        0      0 0.0.0.0:5901            0.0.0.0:*               LISTEN      1357/Xvnc-core
tcp        0      0 127.0.0.1:80            0.0.0.0:*               LISTEN      2086/dionaea
tcp        0      0 192.168.1.111:80        0.0.0.0:*               LISTEN      2086/dionaea
tcp        0      0 0.0.0.0:6001            0.0.0.0:*               LISTEN      1357/Xvnc-core
tcp        0      0 0.0.0.0:5555            0.0.0.0:*               LISTEN      526/sshd
tcp        0      0 127.0.0.1:21            0.0.0.0:*               LISTEN      2086/dionaea
tcp        0      0 192.168.1.111:21        0.0.0.0:*               LISTEN      2086/dionaea
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1282/python
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      23504/cupsd
tcp        0      0 192.168.1.111:22        93.56.12.164:55048      ESTABLISHED 1282/python
tcp        0    179 192.168.1.111:5901      158.42.174.183:25244    ESTABLISHED 1357/Xvnc-core
tcp        0      0 192.168.1.111:22        181.23.43.73:44601      ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:22        95.250.16.137:58367     ESTABLISHED 1282/python
tcp        0      0 127.0.0.1:39786         127.0.0.1:34200         ESTABLISHED 1401/vncserverui
tcp        0      0 192.168.1.111:5555      158.42.174.183:25229    ESTABLISHED 29876/sshd: pi [pri
tcp        0      0 192.168.1.111:36178     151.101.132.133:80      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:21        46.133.16.151:56345     ESTABLISHED 2086/dionaea
tcp        0      1 192.168.1.111:60930     84.126.16.45:8000       SYN_SENT    30258/libpepflashpl
tcp        0      0 127.0.0.1:34200         127.0.0.1:39786         ESTABLISHED 1357/Xvnc-core
tcp        0      0 192.168.1.111:22        117.21.191.219:3779     ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:36166     151.101.132.133:80      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:22        59.45.175.24:46392      ESTABLISHED 1282/python
tcp        0      1 192.168.1.111:60932     84.126.16.45:8000       SYN_SENT    30258/libpepflashpl
tcp        0      0 192.168.1.111:36170     151.101.132.133:80      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:36168     151.101.132.133:80      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:80        192.168.1.1:44806       TIME_WAIT   -
tcp        0      0 192.168.1.111:33110     216.58.210.163:443      ESTABLISHED 30258/libpepflashpl
tcp        0      1 192.168.1.111:60938     84.126.16.45:8000       SYN_SENT    30258/libpepflashpl
tcp        0      0 192.168.1.111:22        83.218.36.226:39720     ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:49406     216.58.213.170:443      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:22        81.138.11.109:55171     ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:54158     62.42.250.55:443        ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:59962     216.58.214.174:443      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:35314     216.58.211.212:443      ESTABLISHED 30258/libpepflashpl
tcp      336      0 192.168.1.111:36238     216.58.214.163:443      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:22        60.191.29.20:13586      ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:52834     216.58.208.202:443      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:22        31.207.47.55:38852      ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:36164     151.101.132.133:80      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:36176     151.101.132.133:80      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:36150     216.58.204.110:443      ESTABLISHED 30258/libpepflashpl
tcp        0      0 192.168.1.111:22        185.38.148.238:48269    ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:22        103.79.141.182:56555    ESTABLISHED 1282/python
tcp        0      0 192.168.1.111:80        192.168.1.1:44804       TIME_WAIT   -
tcp        0      0 192.168.1.111:22        59.45.175.24:58406      ESTABLISHED 1282/python
tcp6       0      0 ::1:443                 :::*                    LISTEN      2086/dionaea
tcp6       0      0 fe80::1d7c:a815:256:443 :::*                    LISTEN      2086/dionaea
tcp6       0      0 :::5900                 :::*                    LISTEN      22975/vncserver-x11
tcp6       0      0 :::5901                 :::*                    LISTEN      1357/Xvnc-core
tcp6       0      0 ::1:80                  :::*                    LISTEN      2086/dionaea
tcp6       0      0 fe80::1d7c:a815:2561:80 :::*                    LISTEN      2086/dionaea
tcp6       0      0 :::6001                 :::*                    LISTEN      1357/Xvnc-core
tcp6       0      0 :::5555                 :::*                    LISTEN      526/sshd
tcp6       0      0 ::1:21                  :::*                    LISTEN      2086/dionaea
tcp6       0      0 fe80::1d7c:a815:2561:21 :::*                    LISTEN      2086/dionaea
tcp6       0      0 ::1:631                 :::*                    LISTEN      23504/cupsd
tcp6       1      0 ::1:37988               ::1:631                 CLOSE_WAIT  511/cups-browsed
```

Figura 8. Conexiones activas Raspberry.

Tal y como hemos comentado en apartados anteriores, es posible acceder a la Raspberry mediante SSH, en nuestro caso accedíamos con Putty, y también mediante VNC sería posible ver de forma gráfica el escritorio remoto. A mi gusto, VNC tiene grandes ventajas ya que en muchas ocasiones es muy útil ver el escritorio aunque simplemente con Putty podríamos haberlo hecho todo. En las siguientes se pueden ver los dos programas empleados.



Figuras 9. Herramientas para interactuar con la Raspberry.

3.5.1Kippo y Kippo-graph

El primer Honeypot implementado en nuestro sistema ha sido Kippo, el cual hemos mencionado anteriormente en el apartado 3.4. La razón principal por la que hemos implementado dicho Honeypot es la facilidad con la que almacena todos los intentos de autentificación en el puerto 22 del sistema. Por lo que si se desea utilizar el servicio SSH hay que redirigirlo al puerto 5555, tal y como comentábamos en el apartado anterior.

Para conocer lo referente a la instalación y la posterior puesta a punto de Kippo es conveniente dirigirse al Anexo I, en el cual se explican todos los comandos y los scripts de arranque que se han empleado. En dicho Anexo también se encuentra explicado el proceso llevado a cabo para instalar Kippo-graph.

3.5.2 Dionaea y DionaeaFR

Como hemos comentado anteriormente, después de instalar Kippo procedemos a instalar Dionaea y posteriormente su parte gráfica, DionaeaFR. La razón principal por la que hemos implementado este Honeypot es su versatilidad a la hora de emular diferentes tipos de servicios.

La parte relativa a la instalación y a la configuración de Dionaea y DionaeaFR se puede encontrar explicada paso a paso en el Anexo I.

Hay que tener en cuenta que deberemos apagar el servidor apache antes de enchufar DionaeaFR, ya que de no ser así no conseguiremos poner en marcha dicha parte gráfica. Para apagar el apache deberemos usar /etc/init.d/apache2 stop.

3.5.3 Bloqueo IP

Tal y como llevamos comentando a lo largo del proyecto, nuestro objetivo final siempre ha estado centrado en prevenir los posibles ataques que pudiese realizar un atacante sobre nuestra propia red. Para conseguirlo, hemos utilizado todo lo mencionado anteriormente, siempre teniendo en cuenta lo que pretendemos conseguir, sus direcciones IP así como su modo de llevar a cabo el ataque. Una vez obtenemos esto, empleando un sistema trampa que no pone en peligro nuestro sistema real, podemos evitar que el atacante vuelva a atacarnos bloqueando su dirección IP en Iptables, un potente firewall integrado en el Kernel de Linux. Para hacer este proceso más efectivo lo hemos automatizado mediante scripts, que leen del archivo (.log), todos los datos recabados por los Honeypots. Poco después, programamos Crontab, un programador de tareas en Linux, para que ejecute cada 5 minutos un archivo que utiliza dichas direcciones IP junto a una instrucción de bloqueo en el Firewall. Todo el proceso realizado desde que el atacante se conecta a nuestro sistema trampa, hasta que obtenemos su dirección IP y después la bloqueamos está explicado más detalladamente en el Anexo I.

3.5.4 Consulta IP

No cabe duda de que si pretendemos utilizar todo lo desarrollado anteriormente y conseguir proteger nuestros sistemas correctamente, no deberemos ubicar la Raspberry en la red interna, tal y como se encuentra en estos momentos, sino que deberemos ubicarla antes del firewall principal de la organización, para que así sea posible recibir ataques en los sistemas trampa, y sean bloqueados ya en este primer Firewall, sin que tengan la posibilidad de colarse por el Firewall principal.

Pero, teniendo en cuenta que un atacante con grandes conocimientos podría llegar a penetrar en la red de la organización, implementaremos en el sistema real otra herramienta que, combinada con esta, hará posible consultar cualquier dirección IP sospechosa en el archivo de IPs atacantes, export_data.txt, tal y como mencionábamos en el Anexo I. A su vez, también podremos valernos tanto de los scripts de bloqueo como del archivo de IPs maliciosas para bloquearlas en el Firewall principal, reduciendo notablemente el riesgo de sufrir un ataque. Aunque siempre hay que tener en cuenta que, tal y como decía Gene Spafford, la única posibilidad de no sufrir ningún ataque es cuando el sistema se encuentra apagado, dentro de un bloque de hormigón, protegido dentro de una habitación sellada y rodeada de guardias armados, por lo que no podemos asegurar que algún atacante no dedique tiempo a estudiar nuestra organización y consiga colarse. Lo que si podemos asegurar es que intentaremos detenerlo, teniendo en cuenta que puede cometer algún pequeño error y ser detectado por nuestros honeypots.

Si este fuera el caso y algún atacante lograse adueñarse del sistema trampa, podría introducirse en nuestro sistema real, suponiendo que lo logra y también suponiendo que alguno de los sistemas de monitorización de la empresa capte su dirección IP, podríamos hacer uso de las herramientas que llevamos explicando a lo largo del proyecto junto con otra que nos permita comprobar que efectivamente dicho atacante había intentado penetrar en nuestro sistema trampa, y que dadas las circunstancias, lo ha conseguido. Esto nos permitirá poder recabar alguna información que pueda haber dejado en los sistemas trampa acerca de su metodología.

Para ello hemos implementado un servidor y un cliente TCP, que se encargan de comprobar si la dirección IP introducida por el cliente coincide con alguna de las direcciones IP que nos habían atacado anteriormente. Una vez hecho esto, de coincidir con alguna dirección IP de nuestro sistema trampa, podremos buscar información relativa al atacante pero de no ser así, sabemos que probablemente nos enfrentamos a un individuo que ha estudiado muy bien nuestra infraestructura. Todo el proceso estará explicado más detalladamente en el Anexo I.

Capítulo 4.

Análisis de resultados

4.1 Resultados obtenidos con Kippo

Tal y como comentamos en el Anexo I, debe haber algún problema relativo a la compatibilidad de alguna librería de Raspbian Jessie con alguna dependencia de Kippo-graph, ya que toda la configuración se ha realizado correctamente, varias veces y utilizando diferentes versiones, y no debería presentar ningún fallo, pero presenta errores en el momento de generar los gráficos. Esto no repercute directamente en el objetivo del proyecto, ya que las direcciones IP, comandos, etc. se encuentran almacenados en los archivos .log de Kippo, aunque si nos impide mostrar esos datos de una forma mucho más clara, sintetizada y directa.

De todos modos mostraremos algunos de los datos recabados por Kippo, almacenados en sus archivos .log, similares a un contenedor Big Data.

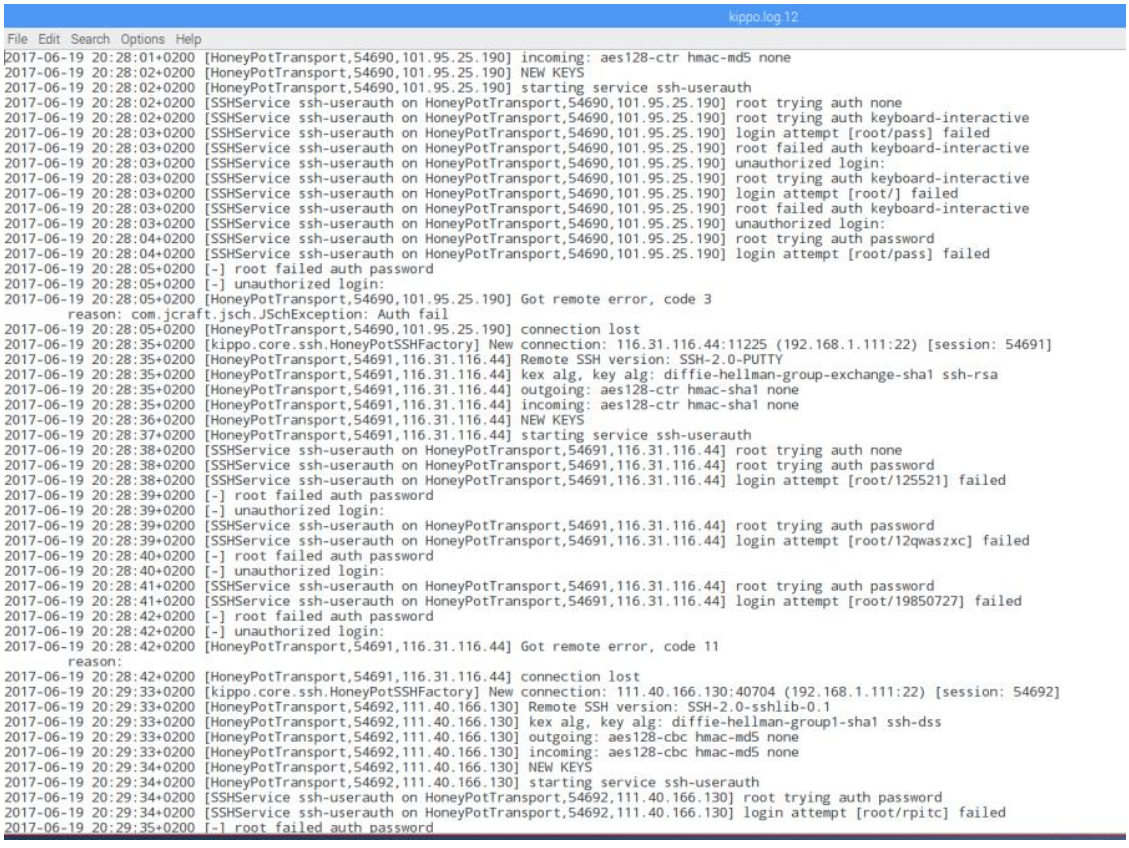


Figura 10. Archivo .log Kippo

Después, hemos ejecutado algunos comandos en el terminal de la Raspberry, que nos permitirán extraer información como el número de conexiones o el número de intentos de login, una conexión supone tres intentos de login. A su vez, mediante el script de bloqueo, hemos extraído las direcciones IP y las hemos guardado en un archivo .txt. Por último hemos ejecutado el

comando unique, que nos permite saber el número de direcciones IP diferentes desde las que se ha ejecutado un ataque.

```
root@raspberrypi:/home/pi/kippo/log# cat kippo.log* | grep login | wc -l
468498
root@raspberrypi:/home/pi/kippo/log# cat kippo.log* | grep login | sort | uniq |
 wc -l
456143
root@raspberrypi:/home/pi/kippo/log#
```

Figura 10a. Información Kippo.

4.2 Resultados obtenidos con Dionaea

Estos almacenes de datos mencionados anteriormente en Kippo también existirían al utilizar Dionaea. Se pueden encontrar dentro de la carpeta Bistreams en Dionaea, pero en este caso la parte gráfica que se encarga de extraer los datos de bistreams, DionaeaFR, si funciona sin ningún tipo de problema, por lo que mostraremos dichos datos de forma mucho más clara y directa.

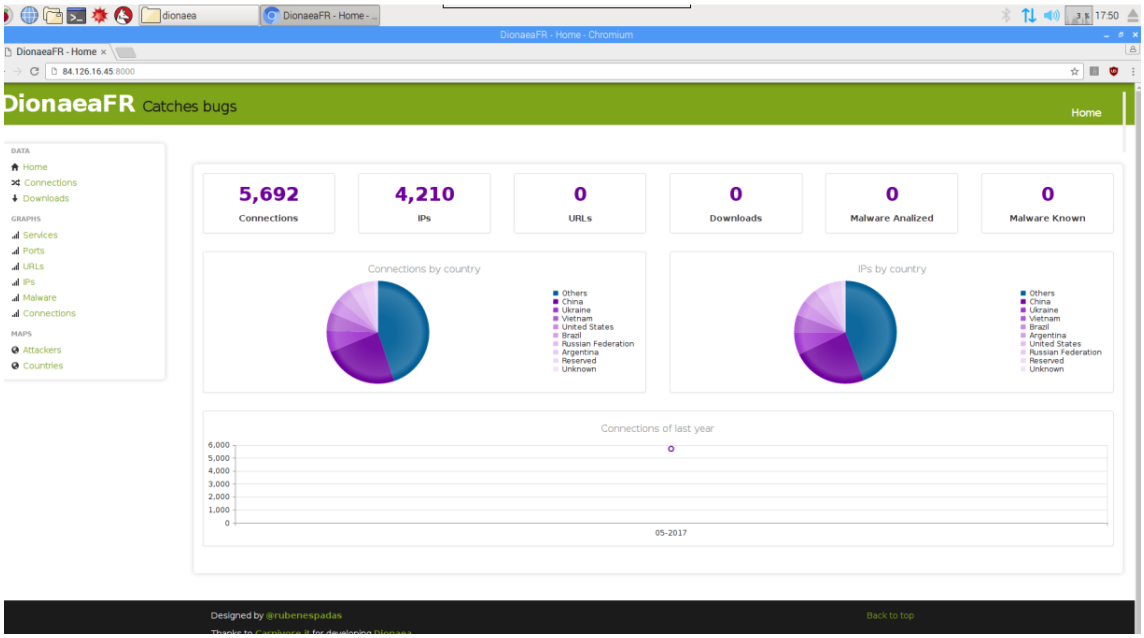


Figura 11. Home DionaeaFR.

En esta imagen podemos ver la página principal de DionaeaFR, en la que nos muestra tanto en número de conexiones que ha recibido, 5692 conexiones, así como el número de direcciones IP, 4210 IPs. Estos dos campos serán los más importantes de esta imagen para nosotros. De tener activado el servicio SMB, los atacantes hubiesen dejado malware, pero no tenemos activado el servicio, con lo que no tendremos malware. No hemos activado dicho servicio, ya que pertenece a otro sistema operativo, Windows, por lo que al tratarse de un Linux Server un atacante podría intuir que se trata de una trampa.

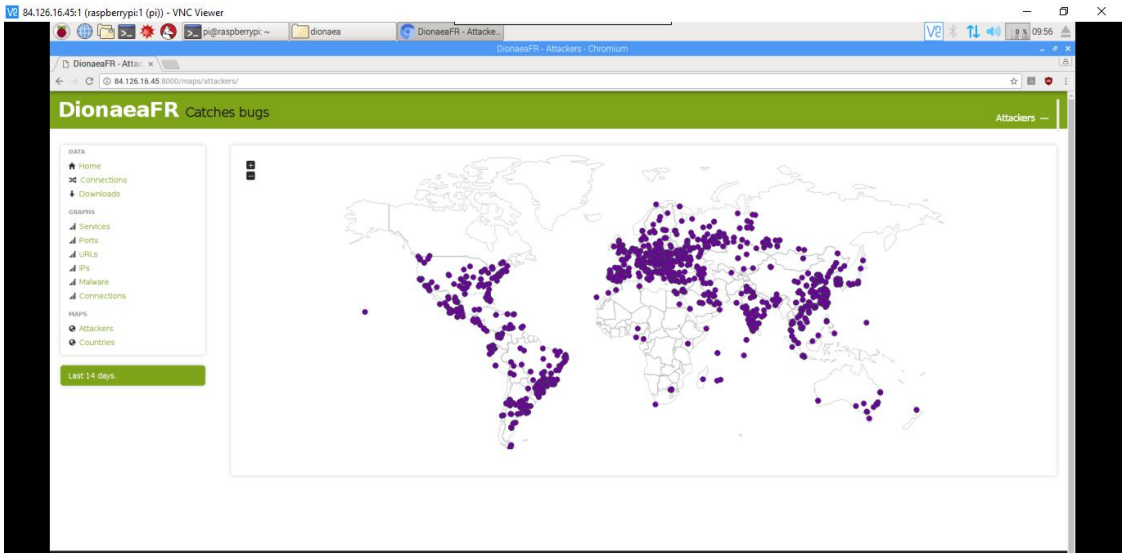


Figura 12. Ataques clasificados por geo-localización.

En esta imagen podemos la ubicación desde la que se han realizado los ataques, en este caso las zonas desde las que hemos recibido más ataques son China, Irán, Europa, Rusia y América.

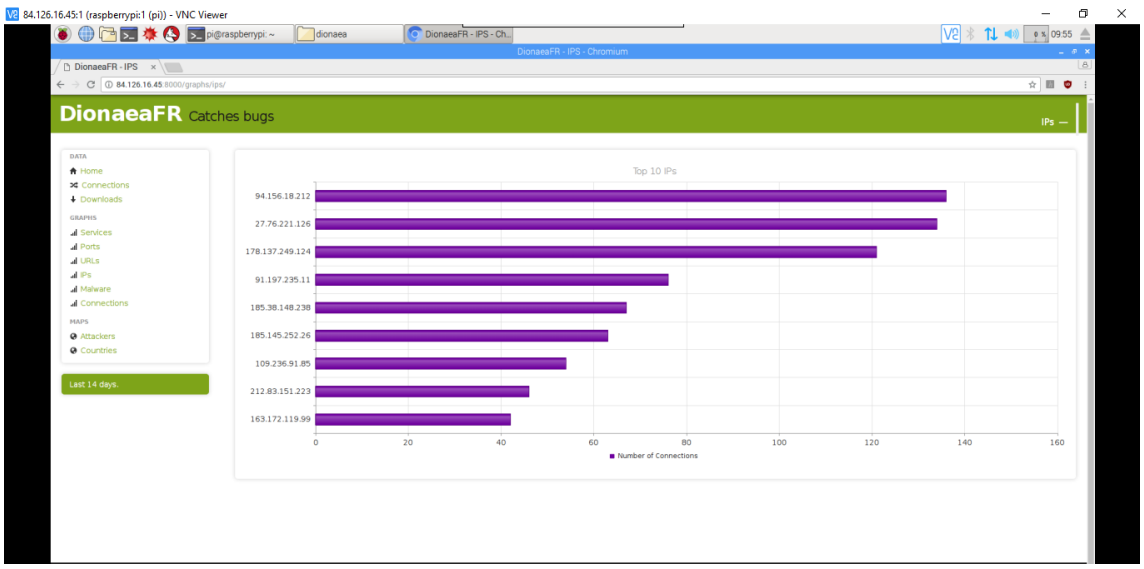


Figura 13. Ataques clasificados por IPs origen.

A su vez, también podemos observar cuales han sido las direcciones IP desde las que hemos recibido más ataques. En primer lugar tendríamos a 94.156.18.212 con casi 140 conexiones. Le sigue de cerca 27.76.221.126 con algo menos de 140 conexiones. Tal y como hemos mencionado anteriormente, con cada conexión el atacante dispone de tres intentos de login. Y eso teniendo en cuenta que esta imagen se tomó apenas una semana después de haber desplegado Dionaea, con lo que ahora el número sería mucho mayor.

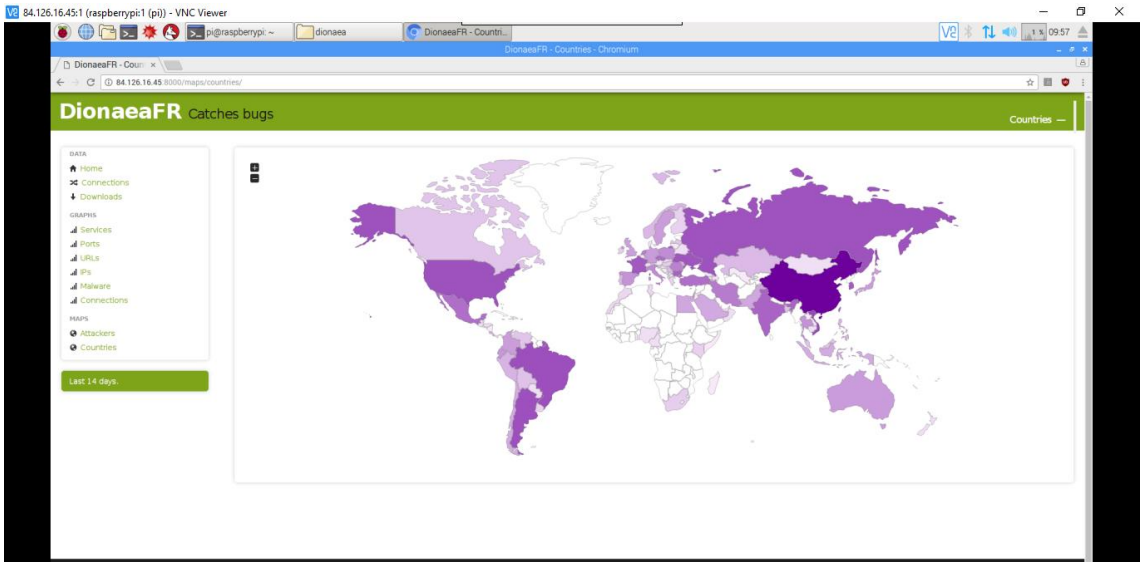


Figura 14. Ataques clasificados por países.

En esta imagen se puede ver mejor definido que antes, los países desde los que más ataques se han recibido, entre los que más destacan está China en cabeza, seguida de EEUU, Rusia, América latina y algo menos en Europa. En la imagen anterior Europa tenía muchos más atacantes concentrados, por lo que es posible que oculten su verdadera dirección IP a través de algún proxy o herramienta similar.

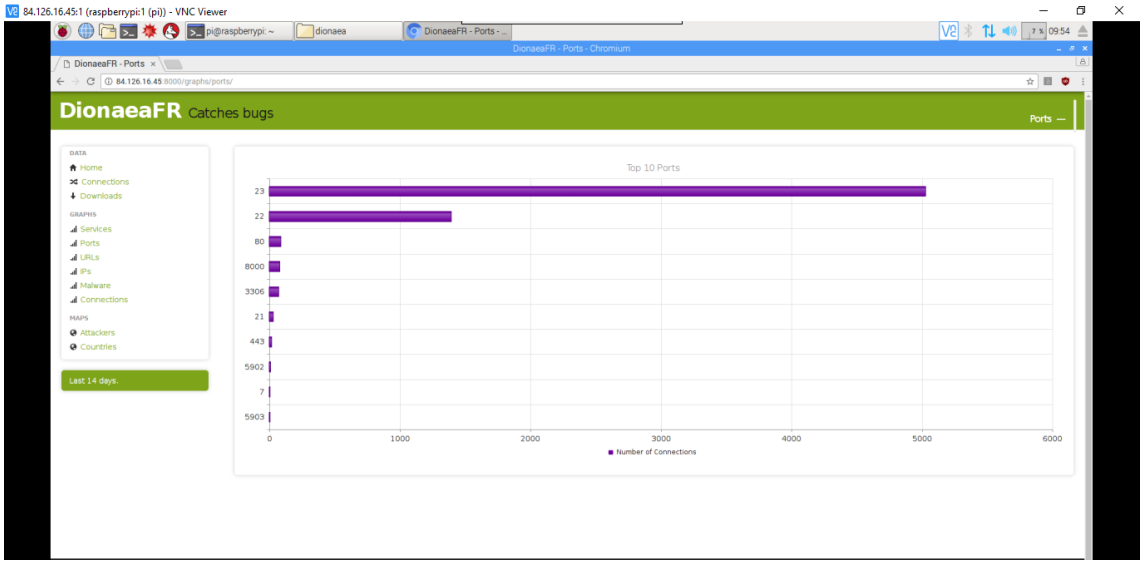


Figura 15. Ataques clasificados por puertos.

En estas imágenes podemos ver los ataques recibidos clasificados por puertos y por servicios, donde podemos observar que aunque no tenemos establecidos en Dionaea ni el puerto 22 ni el 23, correspondientes al ssh y al telnet, la mayoría de ataques se centran en estos puertos. Esto se debe a que el número de ataques que reciben estos servicios son muchísimo mayores que las de cualquier otro tipo de servicio, ya que una vez conseguido el control del equipo tienen todo lo que necesitan. También se reciben ataques correspondientes a servicios como http o ftp, pero como decimos en mucha menor medida que en los otros servicios.

Podríamos haber recibido mayor cantidad de ataques ubicando la Raspberry en un lugar de más fácil acceso, pero para nuestro propósito es suficiente con los datos que hemos recabado.

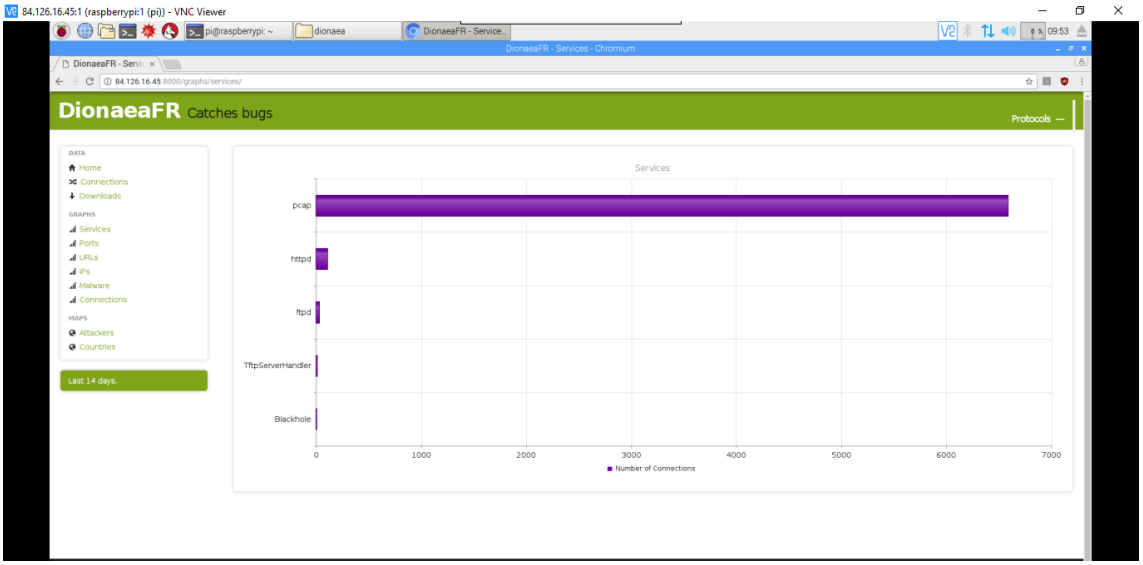


Figura 16. Ataques clasificados por servicios.

4.3 Resultados Bloqueo IP

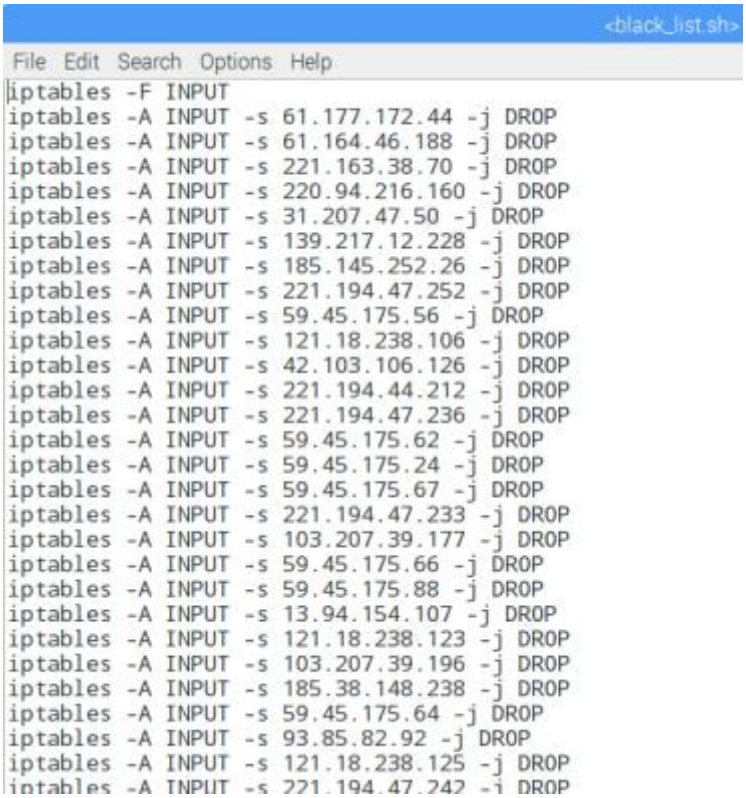


Figura 17. Archivo ejecutable con instrucciones de bloqueo.

Tal y como comentamos en el Anexo I, después de obtener los datos que necesitamos mediante los Honeypots, vamos a extraer las direcciones IP de los atacantes, introduciéndolas de forma automática en el archivo ejecutable que hemos mostrado arriba, y que está programado para ejecutarse cada 5 minutos y que se actualicen dichas direcciones.

Cada vez que se ejecute este archivo se ejecutarán las instrucciones que contiene, en este caso el comando Iptables, que bloqueara dichas direcciones en el Firewall.

Como podemos observar en la siguiente imagen, al introducir el comando iptables –L –n podremos ver las direcciones que están bloqueadas, y que en consecuencia no podrán volver a enviar ningún paquete hacia ningún punto de nuestra red.

```
root@raspberrypi:/home/pi# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  61.177.172.44        0.0.0.0/0
DROP       all  --  61.164.46.188        0.0.0.0/0
DROP       all  --  221.163.38.70        0.0.0.0/0
DROP       all  --  220.94.216.160       0.0.0.0/0
DROP       all  --  31.207.47.50         0.0.0.0/0
DROP       all  --  139.217.12.228       0.0.0.0/0
DROP       all  --  185.145.252.26       0.0.0.0/0
DROP       all  --  221.194.47.252       0.0.0.0/0
DROP       all  --  59.45.175.56         0.0.0.0/0
DROP       all  --  121.18.238.106       0.0.0.0/0
DROP       all  --  42.103.106.126       0.0.0.0/0
DROP       all  --  221.194.44.212       0.0.0.0/0
DROP       all  --  221.194.47.236       0.0.0.0/0
DROP       all  --  59.45.175.62         0.0.0.0/0
DROP       all  --  59.45.175.24         0.0.0.0/0
DROP       all  --  59.45.175.67         0.0.0.0/0
DROP       all  --  221.194.47.233       0.0.0.0/0
DROP       all  --  103.207.39.177       0.0.0.0/0
DROP       all  --  59.45.175.66         0.0.0.0/0
DROP       all  --  59.45.175.88         0.0.0.0/0
DROP       all  --  13.94.154.107        0.0.0.0/0
DROP       all  --  121.18.238.123       0.0.0.0/0
DROP       all  --  103.207.39.196       0.0.0.0/0
DROP       all  --  185.38.148.238       0.0.0.0/0
DROP       all  --  59.45.175.64         0.0.0.0/0
DROP       all  --  93.85.82.92          0.0.0.0/0
DROP       all  --  121.18.238.125       0.0.0.0/0
DROP       all  --  221.194.47.242       0.0.0.0/0
DROP       all  --  94.156.18.212        0.0.0.0/0
DROP       all  --  27.76.221.126        0.0.0.0/0
DROP       all  --  178.137.249.124      0.0.0.0/0
DROP       all  --  91.197.235.11        0.0.0.0/0
DROP       all  --  185.38.148.238       0.0.0.0/0
DROP       all  --  185.145.252.26       0.0.0.0/0
DROP       all  --  109.236.91.85        0.0.0.0/0
DROP       all  --  212.83.151.223       0.0.0.0/0
```

Figura 18. Direcciones bloqueadas en el Firewall mediante comandos Iptables.

Todas estas direcciones IP las almacenaremos en un archivo de texto, export_data, para conseguir ampliar el alcance de nuestra herramienta, tal y como hemos mencionado en el capítulo anterior.

4.4 Resultados Consulta IP

Tal y como comentamos en el capítulo anterior, un vez tenemos las direcciones IP almacenadas en export_data.txt, procedemos a implementar el servidor y el cliente TCP, que permitirán consultar desde cualquier sistema si una dirección IP se encuentra en nuestra lista.

Para ello deberemos ejecutar el servidor cada vez que queramos realizar una consulta, ya que de no ser así tendremos problemas con el chequeo de la dirección IP. Por otro lado, también deberemos ejecutar el cliente, poniendo como argumento la dirección IP a consultar. Tal y como podemos ver en la imagen, después de realizar la comprobación dicho script nos devolverá un "0" si la dirección no se encuentra en la lista, de lo contrario nos devolverá un "1".

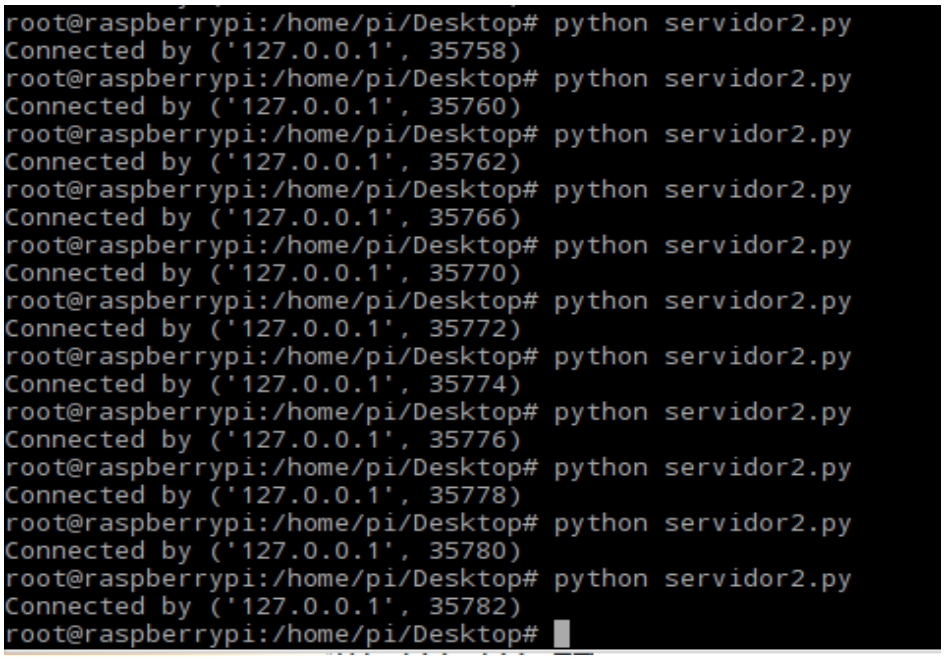
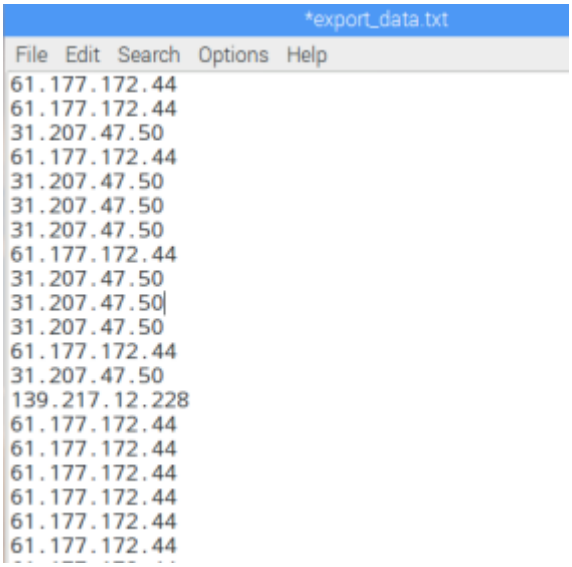


Figura 19. Servidor TCP

```
root@raspberrypi:/home/pi/Desktop# python cliente.py 139.217.12.229
Received '0'
0
root@raspberrypi:/home/pi/Desktop# python cliente.py 139.217.12.228
Received '1'
1
root@raspberrypi:/home/pi/Desktop# python cliente.py 61.177.172.44
Received '1'
1
root@raspberrypi:/home/pi/Desktop# python cliente.py 61.177.172.45
Received '0'
0
root@raspberrypi:/home/pi/Desktop# python cliente.py 31.207.47.50
Received '1'
1
root@raspberrypi:/home/pi/Desktop# python cliente.py 31.207.47.51
Received '0'
0
root@raspberrypi:/home/pi/Desktop# python cliente.py 103.207.39.177
Received '1'
1
```

Figura 20. Cliente TCP



Tal y como podemos comprobar comparando la ejecución del cliente y el archivo que contiene nuestra lista de direcciones IP, podemos asegurar que dicha consulta se realiza correctamente, dando la posibilidad al afectado de comprobar si las direcciones IP sospechosas están o no en nuestra lista.

Capítulo 5.

Conclusiones y futuras vías de trabajo

En este capítulo hablaremos acerca de las conclusiones extraídas después de realizar este proyecto, así como de las posibles futuras vías de trabajo que podrían plantearse. Para ello, pensamos que la mejor forma de extraer conclusiones es comprobar si se han conseguido los objetivos que nos marcamos en un principio.

Desde el inicio del proyecto nos marcamos tres objetivos, el primero de ellos consistía en implementar dos Honeypots que emularan tres servicios propios de un Linux Server. Con esto pretendíamos conseguir recabar información de posibles atacantes y utilizar las herramientas gráficas de los Honeypots para mostrar algunas estadísticas que nos den más información acerca del atacante, así como de sus conocimientos y motivaciones. Dicho objetivo se ha cumplido tal y como hemos podido comprobar a lo largo del proyecto. Debemos destacar que gran cantidad de los ataques son automatizados, lo que nos indica que se trata de ataques masivos, indiscriminados y que utilizan fuerza bruta, siendo mucho más habituales los ataques destinados al puerto 22, correspondiente al servicio de SSH, dado que darían al atacante el control total del equipo.

A continuación, una vez recopilada dicha información, fundamentalmente direcciones IP, nos marcamos el objetivo de prevenir los ataques de la siguiente forma, utilizar dicha información para bloquear sus direcciones IP y que no sea posible realizar otra conexión desde la misma dirección. En el capítulo 4, en el apartado en el que exponemos los resultados es posible comprobar que esto también se cumple. Esto permitirá que cada atacante que intente colarse en el sistema expuesto, el sistema trampa, vea su dirección sea bloqueada, con lo que será prácticamente imposible que llegue al sistema real.

Llegados a este punto observamos que podíamos ampliar el alcance de nuestra herramienta, permitiendo que los propios sistemas de la organización o incluso cualquier usuario externo que disponga de los scripts así como del archivo de direcciones IP maliciosas pueda comprobar si una dirección potencialmente sospechosa se encuentra en nuestra lista. Para ello, hemos implementado unos scripts que permiten realizar dicha comprobación. Tanto en el capítulo 3 como en el Anexo I, y posteriormente en el capítulo 4 se ha hablado sobre esto, mostrando tanto su implementación como los resultados, por lo que este objetivo que nos marcamos una vez arrancado el proyecto también lo daríamos por cumplido.

Implementar las dos últimas herramientas mencionadas en la misma red puede reportar enormes beneficios para cualquier sistema, ya que le permitiría bloquear al intruso no solo en el primer cortafuegos, sino en toda la red.

Tal y como hemos estado viendo a lo largo del proyecto las posibilidades de los Honeypots son muy amplias, por lo que hemos tenido que focalizar nuestra atención en aquello que realmente nos interesaba. Por ello, es muy probable que en un futuro indaguemos más acerca de este tema pudiendo estudiar algún campo más relacionado con este concepto, haciendo especial hincapié en el servicio SSH. Este servicio es, de lejos, el servicio más atacado, pudiendo registrar hasta 800.000 ataques en un mes.

También sería interesante introducir mejoras en las herramientas desarrolladas, incluso publicarlas en Github para que cualquier usuario pueda utilizarlas y aportar opiniones y mejoras.

Aunque, tal y como mencionamos en el capítulo 2, sigue habiendo algunos problemas de concepto y mucha controversia acerca del valor de los Honeypots , pero simplemente es, como venimos diciendo, un simple problema de concepto, ya que no tiene ninguna implicación directa en el resultado que obtendremos. Es por ello que después de haber estado unos meses investigando acerca del tema y de haberlos utilizado es difícil creer como esta herramienta no está más extendida y presente en muchos más sistemas.

Por todo ello esperemos que teniendo en cuenta la eficacia de las herramientas, en un futuro el concepto de honeypot se extienda más en el mundo de la seguridad, evitando discrepancias sobre sus usos y utilizándolo para lo que realmente funciona, capturar la actividad de los Crackers en internet.

A su vez, sería interesante que en un futuro se pudiesen implementar estas herramientas en una empresa con sistemas reales, aunque habría que cambiar la ubicación de la Raspberry así como algunas de sus funcionalidades, dificultando la detección por parte de los atacantes y consiguiendo extraer mayor cantidad de información, como por ejemplo el análisis de malware, etc.

Bibliografía

[1]The Honeynet Project http://www.honeynet.org/project

[2] Spitzner, Lance. Honeypots: Tracking Hackers, 2002.

[3] The Cuckoo's Egg. Stoll, Clifford. 2005.

[4]Blog de Shinayoshi http://www.shinayoshi.net/post/2016/07/09/raspbian-jessie-setup/

[5] RaspberryPi.org https://www.raspberrypi.org/documentation/linux/

[6]Configuración Dionaea https://dionaea.readthedocs.io/en/latest/configuration.html

[7]Repositorio Dionaea https://github.com/DinoTools/dionaea

[8]Blog S2 https://www.securityartwork.es/2014/03/05/honeydrive-episode-i/

[9]Documentación Dionaea https://thehackerway.com/2015/03/26/honeypots-parte-2-introduccion-a-dionaea/

[10] Documentación DionaeaFR http://kinomakino.blogspot.com.es/2017/03/honeypots-xvii-dionaea-con-interface.html

[11]Configuración kippo https://thelosingedgeblog.wordpress.com/2016/02/15/kippo-kali-pi/

[12]Documentación Kippo https://thehackerway.com/2015/03/24/honeypots-parte-1-kippo/

[13] Repositorio Kippo https://github.com/desaster/kippo

[14]Configuración Kippo-graph http://bruteforcelab.com/kippo-graph

[15] Repositorio Kippo-graph https://github.com/ikoniaris/kippo-graph

[16]Configuración Kippo-graph http://www.shinayoshi.net/post/2016/08/16/install-kippo-graph/

[17] Fundamentos teóricos https://es.wikipedia.org/wiki/Honeypot

[18] Fundamentos teóricos http://www.egov.ufsc.br/portal/sites/default/files/honeypots_monitorizando_a_los_atacantes.pdf

[19]Bloqueo IP: Iptables https://wiki.archlinux.org/index.php/Iptables_(Espa%C3%B1ol)

[20]Netfilter:Iptables https://es.wikipedia.org/wiki/Netfilter

[21]Información Bloqueo IP http://blog.ls20.com/securing-your-server-using-ipset-and-dynamic-blocklists/

[22]Información Bloqueo IP https://lawsonry.com/2014/01/quickly-block-traffic-with-ipset-and-iptables/

[23]Información Conexiones activas https://es.wikipedia.org/wiki/Netstat

ANEXO I

Scripts de Instalación

- Instalación Kippo:

Primero conseguimos privilegios de usuario root con:

sudo su

Instalamos dependencias necesarias:

apt-get install python-dev openssl python-openssl python-pyasn1 python-twisted python-mysqldb mysql-server

apt-get install subversión

apt-get install authbind

Añadimos usuario Kippo:

adduser Kippo #password raspot

Le damos todos los privilegios

visudo #Añadimos -> Kippo ALL=(ALL:ALL) ALL

sudo touch /etc/authbind/byport/22

sudo chown Kippo:Kippo /etc/authbind/byport/22

sudo chmod 777 /etc/authbind/byport/22

Cambiamos el nombre del archivo de configuración

cd Kippo

mv Kippo.cfg.dist Kippo.cfg

Creación de la Base de Datos Mysql

mysql -u root -p

create database Kippo;

Damos permisos

GRANT ALL ON Kippo.* TO Kippo@localhost IDENTIFIED BY 'root'

mysql -u Kippo –p

use Kippo;

source ./doc/sql/mysql.sql;

sudo nano Kippo.cfg

#Añadimos las siguientes lineas

ssh_port = 22

hostname = root@webserver

[database_mysql]

host = localhost

database = Kippo

username = Kippo

password = root

port = 3306

sudo nano start.sh

Ahora borramos la línea twistd –y Kippo.tac -l log/Kippo.log --pidfile Kippo.pid

##Y ponemos esta:

authbind --deep twistd -y Kippo.tac -l log/Kippo.log --pidfile #Kippo.pid

Ahora ya lo tenemos todo configurado, vamos a arrancar Kippo

sudo su

cd /home/pi/Kippo

./start.sh

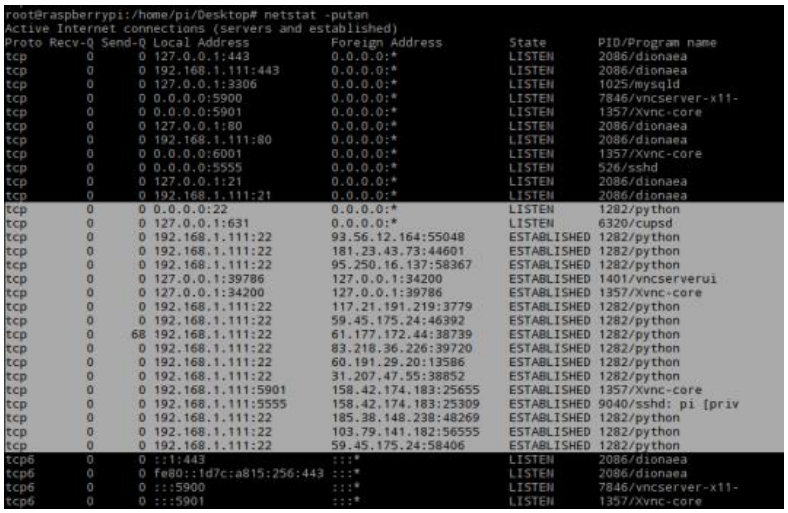


Figura 21. Netstat Kippo.

Tal y como podemos ver en la imagen, ya tenemos emulado un servicio SSH en el puerto 22 e incluso podemos ver algún intruso como por ejemplo 93.56.12.164 y todos los que le siguen. Indagando un poco nos hemos dado cuenta que dicha dirección IP está asociada a una cámara de vigilancia, por lo que es probable que dicho equipo haya sido atacado por un malware, por ejemplo recientemente se dio a conocer Mirai, un malware que entre otras cosas aprovechaba ciertas vulnerabilidades para hacerse con el control de las cámaras y usarlas para realizar ataques desde una botnet. Puede que sea el caso o puede que no, pero lo cierto es que ya tenemos algunos intrusos. Por otra parte, también se puede ver en la imagen como estamos conectados al verdadero servicio SSH en el puerto 5555.

Después de esto, vamos a añadir un script en /etc/init.d para que cada vez que se encienda la Raspberry, también lo haga Kippo. Este script le indicara donde se encuentra el archivo de arranque de Kippo.

#!/bin/bash

/etc/start_Kippo.sh

- Instalación Kippo-graph:

Primero conseguimos privilegios de usuario root con:

sudo su

Instalamos dependencias necesarias:

apt-get update && apt-get install -y libapache2-mod-php5 php5-mysql php5-gd php5-curl

Reiniciamos el servidor apache

/etc/init.d/apache2 restart

Obtenemos el Kippo-graph

wget http://bruteforcelab.com/wp-content/uploads/Kippo-graph-0.9.3.tar.gz

Movemos la carpeta al directorio desde el que apache muestra la web

mv Kippo-graph-0.9.3.tar.gz /var/www/html

cd /var/www/html

Descomprimimos el archivo

tar zxvf Kippo-graph-0.9.3.tar.gz

mv Kippo-graph-0.9.3 Kippo-graph

cd Kippo-graph

Damos permisos

chmod 777 generated-graphs

cp config.php.dist config.php

nano config.php

Modificamos los siguientes parámetros

define('DB_HOST','localhost');

define('DB_USER','Kippo');

define('DB_PASS','root');

define('DB_NAME','Kippo');

define('DB_PORT','3306');

Reiniciamos el servidor apache

sudo service apache2 restart

Para ver los gráficos poner en el navegador http://localhost/Kippo-graph y saldrá

Figura 22. Kippo-graph

Después de probar muchas versiones de la herramienta y de muchísimos intentos intentando que se generaran los gráficos no ha sido posible conseguirlo. Debe haber algún problema con la base de datos ajeno a nuestra configuración, más bien debe ser algún problema relativo a la compatibilidad de alguna librería de Raspbian Jessie con alguna dependencia de Kippo-graph o algo de esta índole, ya que toda la configuración se ha realizado correctamente, varias veces, y no debería presentar ningún fallo.

Aunque esto no repercute directamente en el objetivo del proyecto, ya que las direcciones IP, comandos, etc. se encuentran almacenados en los archivos .log de Kippo, si nos impide mostrar esos datos de una forma mucho más clara, sintetizada y directa.

- Instalación Dionaea:

Primero conseguimos privilegios de usuario root con:

sudo su

Instalamos dependencias necesarias:

apt-get install software-properties-common

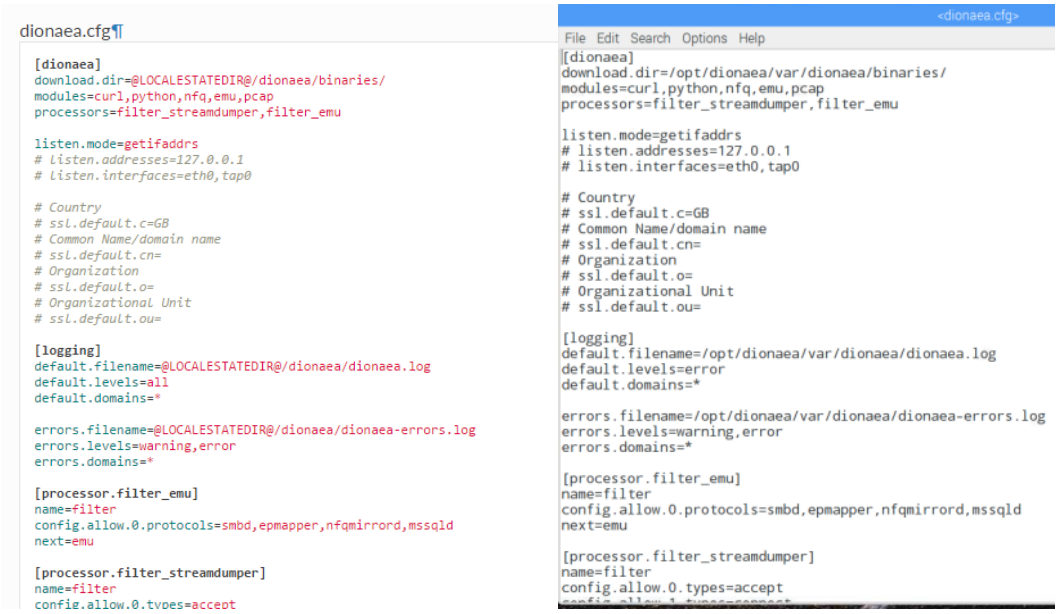
add-apt-repository ppa:honeynet/nightly

apt-get update

cd /opt

git clone https://github.com/DinoTools/Dionaea.git

Editamos el archivo de configuración de Dionaea cambiando la parte de los logs, ya que de no ser así, almacenara demasiados logs, provocando una sobrecarga de memoria.



Figuras 23 y 24. Archivos configuración Dionaea

Tal y como se puede ver en la imagen, hemos cambiado el default.levels=all por default.levels=error.

Ahora podemos arrancar el Honeypot

sudo service Dionaea start

cd /opt/Dionaea/etc/Dionaea/services-enabled

Aquí encontraremos todos los servicios disponibles por Dionaea, simplemente borramos aquellos servicios que no queremos implementar y dejamos los que sí. En la siguiente imagen podremos ver todos los servicios que puede emular Dionaea y también los que finalmente hemos emulado.

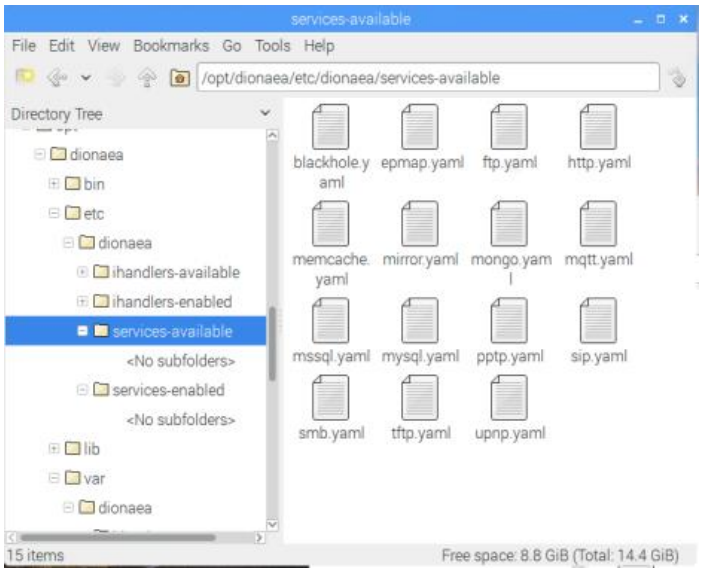


Figura 25. Servicios disponibles Dionaea.

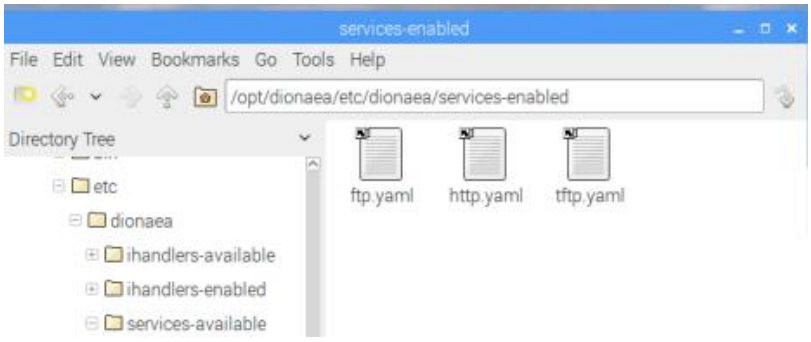


Figura 26. Servicios implementados Dionaea.

Estos últimos tres servicios son los que hemos implementado, FTP, TFTP, HTTP/HTTPS. Como hemos comentado en otros apartados, hemos emulado solo esos servicios ya que tratamos de hacer ver al atacante que se trata de un sistema Linux real. Si por el contrario desplegásemos más servicios el atacante podría percatarse de que se trata de una trampa.

Ahora ya podemos enchufar Dionaea empleando /opt/Dionaea/bin/Dionaea –D. Vamos a comprobar si hemos recibido la visita de algún atacante, y bingo, aquí está.

Figura 27. Almacén de datos de Dionaea.

Aunque a primera vista pueda parecer ilegible, esto nos da información de los flujos entrantes y salientes, versión de HTML, tipo de navegador utilizado (User-Agent), direcciones IP, etc.

A continuación mostraremos estos mismos datos pero de una forma mucho más visual, mediante DionaeaFR.

- Instalación DionaeaFR:

Primero conseguimos privilegios de usuario root con:

sudo su

Instalamos dependencias necesarias:

apt-get install python-pIP python-netaddr unzIP

pIP install Django==1.8.0

pIP install pygeoIP
pIP install django-pagination
pIP install django-tables2
pIP install django-compressor
pIP install django-htmlmin

cd /opt/
wget https://github.com/benjiec/django-tables2-simplefilter/archive/master.zIP
unzIP master.zIP
mv django-tables2-simplefilter-master/ django-tables2-simplefilter/
cd django-tables2-simplefilter/
python setup.py install

git clone https://github.com/bro/pysubnettree.git
cd pysubnettree/

python setup.py install

cd /opt/
wget http://nodejs.org/dist/v0.8.16/node-v0.8.16.tar.gz
tar xzvf node-v0.8.16.tar.gz

cd node-v0.8.16
./configure
make

make install

npm install -g less npm install -g promise

cd /opt/
wget https://github.com/RootingPuntoEs/DionaeaFR/archive/DionaeaFR.zIP
unzIP DionaeaFR.zIP
mv DionaeaFR-master/ DionaeaFR

cd /opt/
wget http://geolite.maxmind.com/download/geoIP/database/GeoLiteCity.dat.gz
wget http://geolite.maxmind.com/download/geoIP/database/GeoLiteCountry/GeoIP.dat.gz
gunzIP GeoLiteCity.dat.gz
gunzIP GeoIP.dat.gz
mv GeoIP.dat DionaeaFR/DionaeaFR/static
mv GeoLiteCity.dat DionaeaFR/DionaeaFR/static

cp /opt/DionaeaFR/DionaeaFR/settings.py.dist /opt/DionaeaFR/DionaeaFR/settings.py

nano /opt/DionaeaFR/DionaeaFR/settings.py

Aquí debemos indicar la base de datos que usa Dionaea, en nuestro caso es:
/opt/Dionaea/var/Dionaea/Dionaea.sqlite

mkdir /var/run/DionaeaFR
cd /opt/DionaeaFR/
python manage.py collectstatic

En este momento elegir "yes"

Ahora ya podemos encender DioneaFR empleando

python manage.py runserver 0.0.0.0:8000

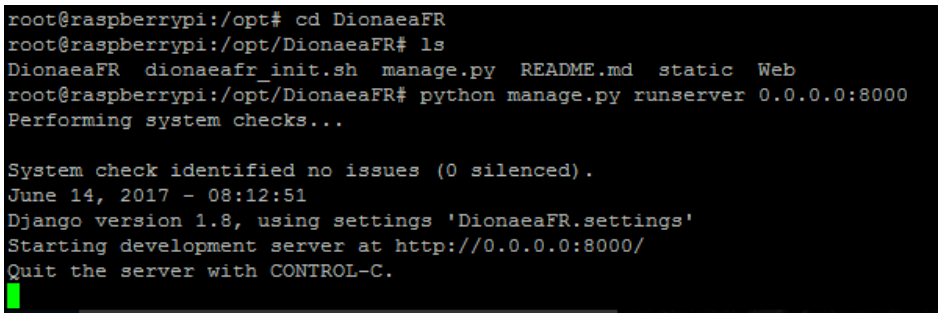


Figura 28. DionaeaFR Server.

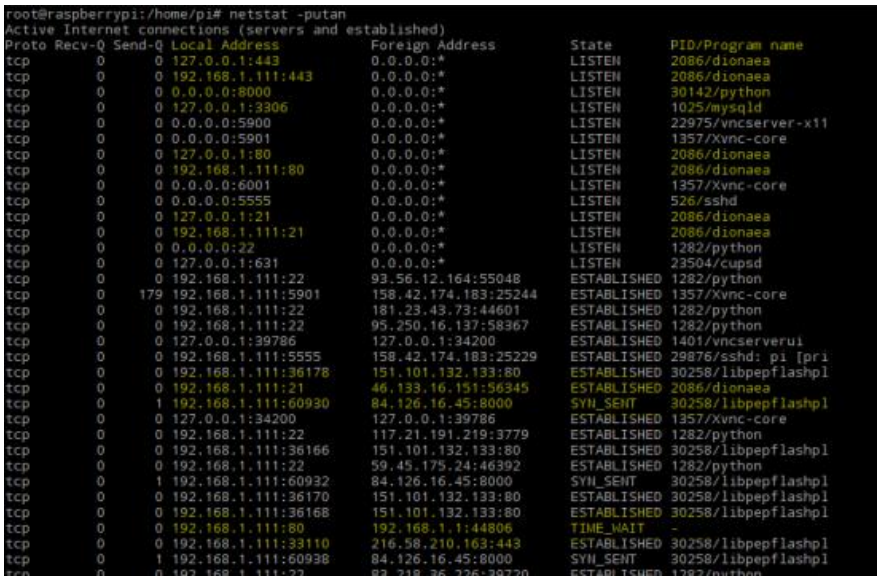


Figura 29. Netstat Dionaea y DionaeaFR.

Después de haber realizado toda esta configuración, tanto del Honeypot como de su parte gráfica, podemos comprobar empleando netstat, un comando que permite visualizar todas las conexiones de un equipo, así como su número de puerto, nombre del programa… que todos los servicios están siendo emulados correctamente, e incluso tenemos algún intruso. Tal y como se puede ver en la imagen estamos emulando un servicio HTTP y un servicio FTP. También se observa que tenemos un intruso atacando el servicio FTP, incluso se puede ver que en nuestro equipo tenemos abierta la conexión al puerto 80 de Dionaea, así como al puerto 8000, de DionaeaFR.

Para acceder a DionaeaFR deberemos poner en el navegador http://84.126.16.45:8000, y veremos algo así:

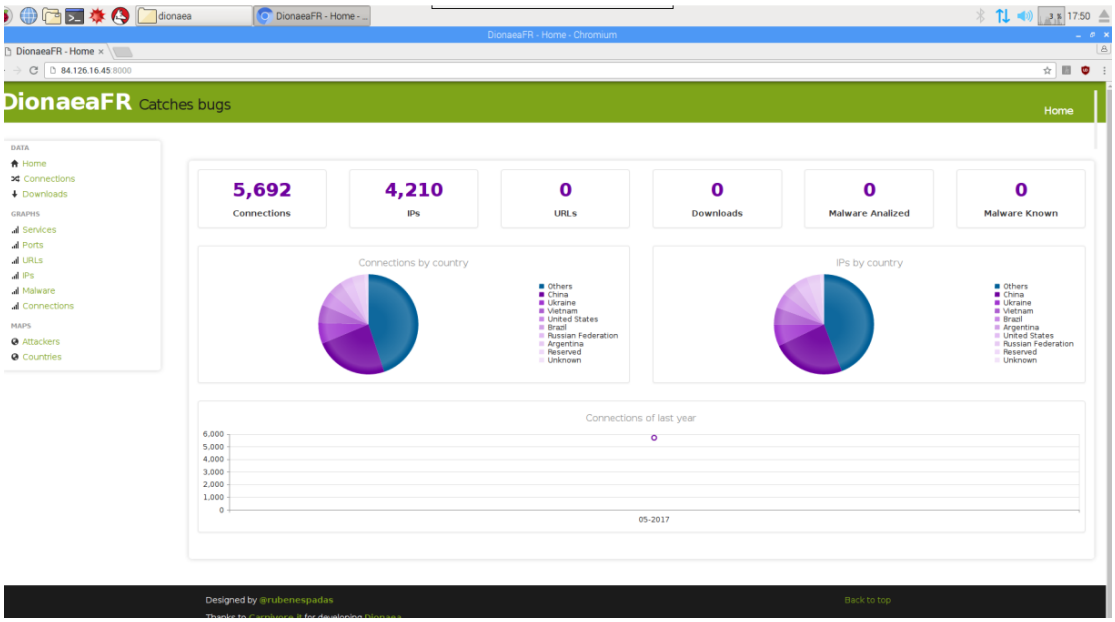


Figura 30. Home DionaeaFR.

En la imagen vemos la página de inicio de DionaeaFR, en la que se muestran el número de conexiones que ha recibido el Honeypot, así como las direcciones IP referentes a dichas conexiones, país de procedencia del ataque, puertos y servicios explotados, etc. Sobre todo esto mencionado anteriormente daremos más datos en el capítulo 4.

- Bloqueo IP

Tal y como hemos comentado, una vez tenemos almacenados todos los datos en el archivo .log, este archivo sería similar a un contenedor Big Data, necesitaremos extraer la información que nos es útil, en este caso necesitamos la dirección IP. Para extraer estos datos, hemos creado un script en python que nos permitirá leer del .log, así como extraer la dirección IP y guardar todas estas direcciones en un archivo de texto, export_data.txt, y que utilizaremos posteriormente. A su vez, nos permitirá almacenar dichas direcciones en un archivo ejecutable, black_list.sh, con una particularidad, se guardara la dirección IP insertada en una instrucción de bloqueo referente a Iptables, un potente Firewall de Linux. Inmediatamente programamos Crontab, un programador de tareas en Linux, para que ejecute black_list.sh cada 5 minutos.

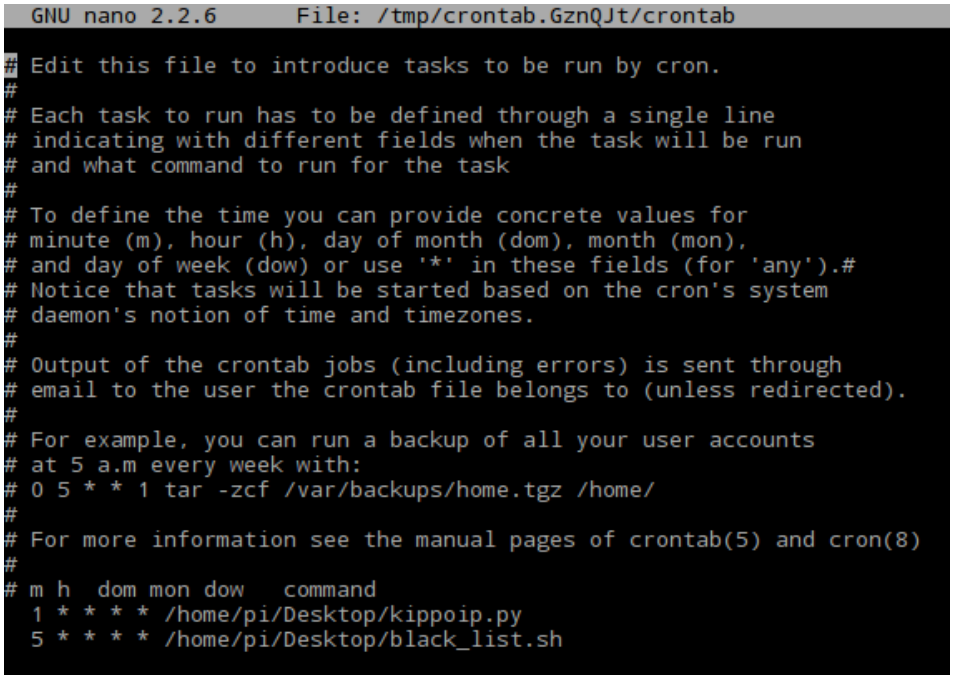


Figura 31. Archivo de ejecución crontab

Esto permitirá que dicho Firewall bloquee las direcciones atacantes cada poco tiempo, lo que mitigara el riesgo de sufrir un ataque en los sistemas reales, y eliminara la posibilidad de volver a sufrir un ataque desde la misma dirección IP tanto en los sistemas trampa como en los sistemas reales.

Los scripts que permitirán todo esto, se pueden observar a continuación.

kippoip.py - /home/pi/Desktop/kippoip.py (3.4.2)

File Edit Format Run Options Windows Help

```
#!/usr/bin/env python
import re

RUTA_LOG_KIPPO = "/home/pi/kippo/log/kippo.log"


def main():

    f = open(RUTA_LOG_KIPPO, "r")
    for line in f:
        if "New connection" in line:
          ip1 = line.split(" ")[5]
          ip2 = ip1.split(":")[0]
          f1 = open("export_data.txt", "a")
          f1.write(str(ip2)+"\n")
          check_ip(ip2)

def check_ip(ip):
        flag = 0
        f2 = open("black_list.sh","r")
        for line in f2:
                if ip in line:
                        flag = 1
        if flag == 0:
                f3 = open("black_list.sh","a")
                f3.write("iptables -A INPUT -s "+ip+" -j DROP \n")



if __name__ == "__main__":
  main()
```

Figura 32. Script bloqueo IP.

<black_list.sh>

File Edit Search Options Help

```
iptables -F INPUT
iptables -A INPUT -s 61.177.172.44 -j DROP
iptables -A INPUT -s 61.164.46.188 -j DROP
iptables -A INPUT -s 221.163.38.70 -j DROP
iptables -A INPUT -s 220.94.216.160 -j DROP
iptables -A INPUT -s 31.207.47.50 -j DROP
iptables -A INPUT -s 139.217.12.228 -j DROP
iptables -A INPUT -s 185.145.252.26 -j DROP
iptables -A INPUT -s 221.194.47.252 -j DROP
iptables -A INPUT -s 59.45.175.56 -j DROP
iptables -A INPUT -s 121.18.238.106 -j DROP
iptables -A INPUT -s 42.103.106.126 -j DROP
iptables -A INPUT -s 221.194.44.212 -j DROP
iptables -A INPUT -s 221.194.47.236 -j DROP
iptables -A INPUT -s 59.45.175.62 -j DROP
iptables -A INPUT -s 59.45.175.24 -j DROP
iptables -A INPUT -s 59.45.175.67 -j DROP
iptables -A INPUT -s 221.194.47.233 -j DROP
iptables -A INPUT -s 103.207.39.177 -j DROP
iptables -A INPUT -s 59.45.175.66 -j DROP
iptables -A INPUT -s 59.45.175.88 -j DROP
iptables -A INPUT -s 13.94.154.107 -j DROP
iptables -A INPUT -s 121.18.238.123 -j DROP
iptables -A INPUT -s 103.207.39.196 -j DROP
iptables -A INPUT -s 185.38.148.238 -j DROP
iptables -A INPUT -s 59.45.175.64 -j DROP
iptables -A INPUT -s 93.85.82.92 -j DROP
iptables -A INPUT -s 121.18.238.125 -j DROP
iptables -A INPUT -s 221.194.47.242 -j DROP
```

Figura 33. Bloqueo en el Firewall.

Figura 34. Script de bloqueo Dionaea.

- Consulta IP

Tal y como hemos comentado anteriormente, observamos que podíamos ampliar el alcance de nuestra herramienta y por ello vamos a recopilar una lista de direcciones IP maliciosas, haciendo posible que un usuario externo o incluso los propios sistemas reales de la organización consulten si una dirección se encuentra en dicha lista o no. Dicha consulta se hará de forma automática, el usuario simplemente introducirá la dirección IP y un script desarrollado en python se encargara de comprobar si dicha dirección está o no en la lista. Para comprobar las direcciones que introduce el cliente se valdrá del archivo export_data.txt, que contiene las direcciones IP atacantes, extraídas del archivo .log.

En las siguientes imágenes podemos ver los scripts que implementan tanto el servidor como el cliente TCP.

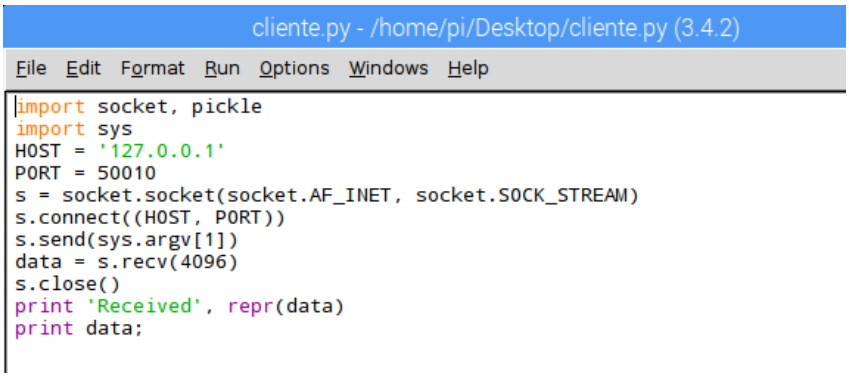


Figura 35. Script Cliente TCP.

servidor2.py - /home/pi/Desktop/servidor2.py (3.4.2)

File Edit Format Run Options Windows Help

```
import socket, pickle;
HOST = '127.0.0.1';
PORT = 50010;
s= socket.socket(socket.AF_INET, socket.SOCK_STREAM);
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1);
s.bind(('',PORT));
s.listen(1);

#Nombre del fichero de lectura
file = "/home/pi/Desktop/export_data.txt"

conn, addr = s.accept();
print 'Connected by' , addr;
flag = 0
while 1:
        data = conn.recv(4096);
        if not data: break;
        f = open(file, "r")
        for line in f:
                if data in line:
                        flag = 1
        if flag == 1:
                data1 = "1"
        else:
                data1 = "0"
        flag = 0
        conn.send(data1);
        break;
conn.close();
s.shutdown(0);
s.close();
```

Figura 36. Script Servidor TCP.

Después de haber desarrollado está herramienta, creo que podría ser muy beneficioso combinarla con los scripts de bloqueo. Ya que permitiría a los sistemas reales de cualquier organización bloquear a posibles atacantes antes de que logren introducirse en el sistema. Teniendo en cuenta que para ello habría que cambiar la ubicación de la Raspberry e introducir algunas mejoras para que sea más difícil su detección.