Universitat Oberta de Catalunya

uoc.edu

Industrial Control Systems and IoT Botnets

Javier Jesús Rejón Orellana
Master Interuniversitario en Seguridad de la Información y de las Telecomunicaciones
Seguridad en la Internet de las cosas

Consultor: Carlos Hernández Gañán
Responsable de la asignatura: Víctor Garcia Font

3/6/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Industrial Control Systems and IoT Botnets*
Nombre del autor:	*Javier Jesús Rejón Orellana*
Nombre del consultor/a:	*Carlos Hernández Gañán*
Nombre del PRA:	*Máster en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Fecha de entrega:	06/2019
Titulación:	*Máster en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*M1.848 TFM-Seguridad en la Internet de las cosas*
Idioma del trabajo:	*Español*
Palabras clave	*Honeypot, IoT, industrial*

Resumen

En este trabajo se realiza un estudio de las amenazas actuales en el entorno de redes de control industrial. Estas amenazas se refieren a ciberataques e intentos de penetración o alteraciones en el funcionamiento de sistemas industriales en producción.

Por un lado se pone de manifiesto la creciente vulnerabilidad de los actuales sistemas industriales, debido una mayor conectividad e integración en los sistemas de información empresariales.

Por otro lado, para ayudar en la detección de estos intentos de intrusión, estudiamos la estrategia denominada “honeypot” y en qué se basa su funcionamiento. Verificamos cómo las herramientas honeypot ayudan en la mejora de la seguridad también en redes de control industrial ICS (Industrial Control System).

Fruto de este estudio, se propone el diseño e implementación de un “honeypot”, señuelo específico, cuyo fin es el poder detectar este tipo de ataques. El producto de software obtenido supone un método de detección de ataques en sistemas en producción.

Nuestro diseño de honeypot hace uso de paquetes de software disponibles bajo licencia libre. Se realiza una implementación modular, flexible y fácilmente mantenible.

Finalmente efectuamos la validación del honeypot bajo estudio mediante una serie de pruebas locales y su exposición directa a Internet. Una vez probado el funcionamiento se verifica su usabilidad en entornos de producción.

Abstract

In this work a study of the current threats in the environment of industrial control networks is carried out. These threats refer to cyber attacks and attempts to penetrate or change the functioning of industrial systems in production.

On the one hand it highlights the growing vulnerability of current industrial systems, due to greater connectivity and integration in business information systems.

On the other hand, to help in the detection of these intrusion attempts, we study the strategy called "honeypot" and on what its operation is based. We verify how the honeypot tools help in the improvement of security also in Industrial Control System networks ICS

As a result of this study, we propose the design and implementation of a "honeypot", a specific decoy, whose purpose is to detect this type of attack. The software product obtained supposes a method of detection of attacks in systems in production.

Our honeyport design makes use of software packages available under a free license. A modular, flexible and easily maintainable implementation is carried out.

Finally, we carried out the validation of the honeypot under study through a series of local tests and its direct exposure to the Internet. Once the operation is tested, its usability is verified in production environments.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación

En este Trabajo se desarrolla un estudio sobre la problemática en la seguridad de las redes de control industrial (ICS) y el diseño e implementación de un “honeypot” que permita la detección de ataques contra dispositivos de control industrial, considerados elementos IoT y posibles víctimas de botnets.

En los últimos años se está produciendo un auge importante en la utilización del tipo de dispositivo denominados IoT. La penetración de Internet en herramientas y aparatos con usos diferentes al ordenador personal es fruto de la reducción de costes en su producción, en parte motivado por el crecimiento de la industria asiática y la globalización en el mercado electrónico.

Arrastrados por estas sinergias, los entornos industriales empiezan a ser considerados redes de componentes IoT, cada vez más conectados y cada vez más expuestos. En este contexto los sistemas de control industrial se convierten en importantes objetivos con intereses diferenciados para el atacante. Debido a ello, existe una necesidad creciente de protección ante estas nuevas amenazas que no existían hace unos años.

Para protegerse de las amenazas es necesario detectarlas primero. Esta detección es la necesidad a solucionar antes de poder protegerse de manera efectiva de un ataque hacia los dispositivos de control industrial. En la actualidad existe un desarrollo importante en el mundo de la ciberseguridad en entornos empresariales y gubernamentales. Este no es el caso de las redes ICS que empresas y fabricantes explotan en sus negocios. En este trabajo se aporta un primer nivel de detección de ataques cuyo destino es específicamente dispositivos IoT desplegados en entornos de control industrial.

1.2 Objetivos

El objetivo principal de este trabajo es entender las redes de control industrial, su funcionamiento, sus condiciones especiales de trabajo. Analizar sus debilidades y su evolución, los actuales retos que plantea el incremento en la conectividad de estas redes especiales y los peligros que ello supone.

Se quiere realizar un análisis del funcionamiento de la estrategia honeypot, y su particularización para el caso concreto de entornos ICS. Se propone un diseño e implementación de un Honeypot específico para redes ICS.

Estos objetivos se concretan en el siguiente listado:

- Profundizar los conocimientos sobre el concepto de Internet of Things.
- Tener una visión clara sobre las vulnerabilidades de los componentes IoT y los riesgos que implica.
- Tener una visión general sobre los dispositivos de control industrial y las redes SCADA.
- Determinar las principales vulnerabilidades en las redes de control ICS.
- Entender el concepto de “honeypot”, sus tipos, ventajas e inconvenientes.
- Establecer requisitos que tendrá que tener el honeypot.
- Obtener un análisis de alto nivel del software en función de los requisitos y de la funcionalidad que se le quiere dar.
- Basándonos en el análisis del punto anterior, obtener el diseño de la aplicación a implementar basándolos en componentes de software existentes.
- Implementar la aplicación en función del diseño obtenido.
- El sistema generará los registros de ataques recibidos identificando el origen y tipología.
- Generar los juegos de pruebas adecuadas para garantizar el correcto funcionamiento de la aplicación.

1.3 Enfoque y método seguido

Un software tipo honeypot se aprovecha de determinados paquetes y utilidades existentes en repositorios libremente disponibles. Actualmente es posible implementar un honeypot sin realizar un proyecto software desde cero, ya que determinados tipos de ataques serán detectados por paquetes ya disponibles y ampliamente utilizados.

La versatilidad en la utilización de paquetes de detección a disposición del usuario, nos lleva a realizar un trabajo de integración en la aplicación, más que un desarrollo al uso. Esta estrategia proporciona varias ventajas:

- Reducción en el tiempo de implementación.
- Garantía del uso de paquetes de madurez contrastada.
- Disponibilidad de documentación y “know-how” existente.
- Mejora de la escalabilidad.
- Flexibilidad de la estructura del aplicativo y facilidad de cambio.

1.4 Planificación

En primer lugar se presentan las bases teóricas necesarias para la realización del software honeypot que se pretende. Para ello se cuenta con recursos documentales reseñados en la bibliografía más abajo, así como numerosos artículos, blogs y otras fuentes de información localizadas en Internet. Además, se utilizarán recursos físicos mínimos, como un ordenador suficientemente potente, virtualizando el sistema operativo anfitrión del software.

Tarea 1:

Temporización: 2 semanas.

Descripción: Realización del presente plan de trabajo y planificación del proyecto.

Objetivos: Determinar las necesidades del TFM y realizar la planificación temporal en su ejecución.

Hitos: Documento de planificación del TFM.

Tarea 2:

Temporización: 4 semanas.

Descripción: Análisis teórico sobre redes IoT y redes de control industrial (ICS) y sus vulnerabilidades.

Objetivos: Profundizar los conocimientos sobre el concepto de Internet of Things. Tener una visión clara sobre las vulnerabilidades de los componentes IoT y los riesgos que implica. Tener una visión general sobre los dispositivos de control industrial y las redes SCADA. Determinar las principales vulnerabilidades en las redes de control ICS. Entender el concepto de “honeypot”, sus tipos, ventajas e inconvenientes.

Hitos: Redacción de la base teórica necesaria para el consiguiente diseño de honeypot especializado en redes ICS. Primera versión de desarrollo de la memoria TFM.

Tarea 3:

Temporización: 4 semanas.

Descripción: Diseño del Honeypot para detección de ataques específicos en entornos ICS.

Objetivos: Establecer requisitos que tendrá que tener el honeypot. Obtener un análisis de alto nivel del software en función de los requisitos

y de la funcionalidad que se le quiere dar. Basándonos en el análisis del punto anterior, obtener el diseño de la aplicación a implementar basándolos en componentes de software existentes. Efectuar un análisis sobre la transformación de datos capturados en información sobre el atacante.

Hitos: Obtener el análisis y el diseño de la aplicación honeypot documentando las decisiones tomadas.

Tarea 4:

Temporización: 5 semanas.

Descripción: Implementación y pruebas del Honeypot.

Objetivos: Implementar la aplicación en función del diseño obtenido. Generar los juegos de pruebas adecuadas para garantizar el correcto funcionamiento de la aplicación. Memoria de proyecto.

Hitos: Obtención de un producto ejecutable y funcional. Documentar la instalación de la aplicación. Documentar el uso de las diferentes opciones de configuración que pueda tener. Documentación de la aplicación. Documentación sobre los juegos de pruebas.

Tarea 5:

Temporización: 1 semana.

Descripción: Presentación.

Objetivos: Síntesis de la memoria tomada durante el proyecto. Crear una presentación en vídeo de la aplicación creada.

Hitos: Memoria del proyecto. Presentación de la aplicación.

Tarea 6:

Temporización: 1 semana.

Descripción: Defensa del TFM.

Objetivos: Superación en el defesa del proyecto TFM.

Hitos: Defensa del TFM.

Esta planificación queda reflejada en el siguiente diagrama:

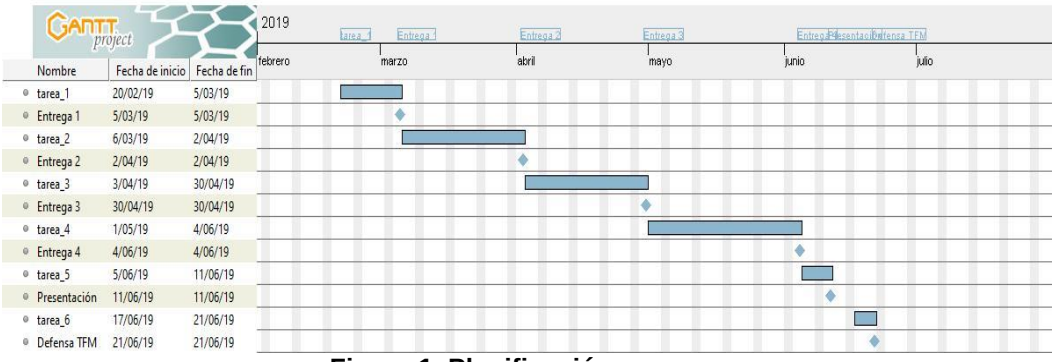


Figura 1. Planificación.

1.5 Breve sumario de productos obtenidos

El producto de este trabajo lo conforman por un lado una memoria y por otro un aplicativo de software, denominado en el argot informático como “honeypot”, especializado en la detección de ataques conocidos a redes de control industrial.

La memoria incluye tanto la visión general del estado del arte en cuanto a qué elementos hay involucrados en los ciberataques a redes industriales, como las consideraciones en el diseño de un sistema honeypot orientado el entorno industrial.

1.6 Breve descripción de capítulos

En el capítulo 1 se plantean los objetivos y la planificación de este trabajo.

En el capítulo 2 se realiza una introducción al concepto de IoT, así como a los peligros emergentes que supone. Se hará un breve repaso a los conceptos fundamentales de control industrial que son objetos de ataques interesados.

En el capítulo 3 se aborda la estrategia honeypot como método de detección de ataques a la integridad de cualquier infraestructura informática. Se definirán los tipos de honeypot y su idoneidad en entornos industriales. Se efectúa también un diseño general del aplicativo que queremos implementar.

En el capítulo 4 se realiza el diseño de un honeypot concretando secciones técnicas específicas a nuestras necesidades. Se determinan los paquetes de software específicos así como la estrategia de utilización.

En el capítulo 5 se especifica la integración de la aplicación como tal, sus componentes y las configuraciones necesarias. Se realizarán

también algunas pruebas simulando ataques que nos permitan asegurar el funcionamiento correcto de la aplicación.

En el capítulo 6 se presentas los formatos de log obtenidos y cómo se identifican los datos principales: IP origen de la conexión y servicio accedido.

Finalmente en los anexos hemos incluido información práctica de instalación de algunos componentes.

2. Redes IoT y redes de control industrial (ICS). Vulnerabilidades.

2.1 Introducción al concepto IoT. Evolución y vulnerabilidades

Internet of Thing (IoT) es el concepto emergente que ha surgido en los últimos años cuando Internet ha llegado a dispositivos hasta hace poco impensables. Ya no sólo nos conectamos a Internet con ordenadores personales sino que toda una gama de pequeños dispositivos se conectan también con diversos propósitos. Por un lado podemos diferenciar dispositivos domésticos o cotidianos, como televisores, cámaras IP, frigoríficos, aspiradoras, pequeños routers y en general dispositivos de domótica en el hogar. Pero por otro lado debemos pensar en dispositivos que controlan o recogen información en grúas, cadenas de montajes, plantas de procesado, válvulas, actuadores electromecánicos, sensores de todo tipo, etc.

En realidad el concepto de IoT no es nuevo. Ya en 2009 Kevin Ashton publicó un artículo en el que se defendían las bondades de podría tener una tostadora conectada a Internet [2]. A partir de ahí el impulso ha sido importante y se calcula que para alrededor de 2021 existirán unos 50.000 millones de dispositivos conectados a Internet. El mercado IoT alcanzará 250 mil millones de dólares [5].

El problema de los dispositivos IoT es que la mayoría nunca han sido realmente pesados para estar conectados en redes vulnerables. Además en muchos casos suelen ser dispositivos de pequeño tamaño donde se ha añadido la electrónica mínima para realizar su función. Esta electrónica no ha sido diseñada pensando en la ciberseguridad, no proporciona los mecanismos necesarios que los hagan robustos frente a ataques clásicos. Hablamos de passwords débiles, carencia de firewall, vulnerabilidades en el software que ejecutan, desprotección a ataques DoS, etc [6].

Muchos dispositivos IoT son diseñados para recibir una configuración básica inicial. Posteriormente son olvidados mientras no presenten deficiencias de funcionamiento y no reciben actualización alguna.

Como vemos, las IoT pueden adolecer de multitud de vulnerabilidades. Por extensión se aplica también al llamado el Internet de las cosas del mundo industrial, el IIoT (Industrial Internet of Thing). IIoT se refiere a los dispositivos IoT especialmente diseñados para los entornos industriales más modernos. Es la llamada Industria 4.0. Estos dispositivos suelen ser más robustos y son objetos de ataques específicos, como el robo de información o lo que es más grave alterar las funciones de control sobre la maquinaria.

2.2 Las botnets y su relación con las IoT

El concepto de botnet es en realidad anterior al de IoT. Las primeras botnets desplegadas surgen a finales de los ochenta y están relacionadas con el uso del protocolo IRC. El principal peligro de un dispositivo IoT es el secuestro de sus funciones para ser utilizado como un robot dentro de una red botnet. Una vez controlado el IoT mediante ciertas manipulaciones tras un ciberataque, será utilizado remotamente para ataques combinados junto con otros robots, robo de información, etc [7].

Mediante el secuestro lógico del dispositivo IoT se crean canales de comunicación para su control que permitirán manipular el sistema atacado en cualquier momento. En general el objetivo del atacante es lograr suficientes privilegios en la víctima para establecer los canales de control pertinentes y orquestar un modelo distribuido donde los sistemas comprometidos ejecuten ataques sincronizados, entre otras acciones.

Para desplegar una botnet se distinguen varias etapas:

1. Búsqueda e identificación de los objetivos.

2. Explotación y acceso no autorizado.

3. Infección y toma de control.

4. Apertura de los canales de comunicación y ejecución de tareas.

Nuestro honeypot deberá ser capaz de detectar cada uno de estas situaciones, identificar el origen y registrar un log.

Existen muchas botnets que han sido significativamente destacadas. En los últimos años quizás las más destacadas han sido Mirai, Satori y HNS (Hide 'N Seek IoT).

Estudiemos un poco el caso de Mirai, quizás uno de los más importantes hasta el momento, pues afectó a importantes multinacionales como Spotify, Twitter, Netflix y Amazon. Mirai [3] realiza una exploración de amplio alcance de direcciones IP en la red con la intención de localizar dispositivos IoT cuyas credenciales de acceso son vulnerables. Hablamos de parejas de usuario/password fáciles de quebrantar mediante ataques de fuerza bruta tipo diccionario. Una vez se obtiene el acceso al dispositivo IoT se descarga un paquete de software con el código de comando y control correspondiente. Cada dispositivo comprometido realizará esta misma función de infección en el intento de ampliar la botnet con más dispositivos.

Veamos un ejemplo más detenidamente. El objetivo principal de Mirai era realizar ataques DDoS [3]. Para ello ejecuta inundaciones HTTP, escondiéndose bajo los siguientes agentes de usuario:

```
Mozilla / 5.0 (Windows NT 10.0; WOW64) AppleWebKit / 537.36
(KHTML, como Gecko) Chrome / 51.0.2704.103 Safari / 537.36
Mozilla / 5.0 (Windows NT 10.0; WOW64) AppleWebKit / 537.36
(KHTML, como Gecko) Chrome / 52.0.2743.116 Safari / 537.36
Mozilla / 5.0 (Windows NT 6.1; WOW64) AppleWebKit / 537.36
(KHTML, como Gecko) Chrome / 51.0.2704.103 Safari / 537.36
Mozilla / 5.0 (Windows NT 6.1; WOW64) AppleWebKit / 537.36
(KHTML, como Gecko) Chrome / 52.0.2743.116 Safari / 537.36
Mozilla / 5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit /
601.7.7 (KHTML, como Gecko) Versión / 9.1.2 Safari / 601.7.7
```

Además, Miari efectúa ataques a nivel IP y nivel 4 OSI mediante inundaciones GRE IP, SYN y ACK, así como ataques de inundación DNS entre otros.

Mirai posee características interesantes, como la capacidad de derivación para evitar soluciones de seguridad:

```
#define TABLE_ATK_DOSARREST 45 // "servidor: dosarrest"
#define TABLE_ATK_CLOUDFLARE_NGINX 46 // "servidor: cloudflare-
nginx"

if (util_stristr (generic_memes, ret, table_retrieve_val
(TABLE_ATK_CLOUDFLARE_NGINX, NULL))! = -1)
                        conn-> protection_type =
HTTP_PROT_CLOUDFLARE;

if (util_stristr (generic_memes, ret, table_retrieve_val
(TABLE_ATK_DOSARREST, NULL))! = -1)
                        conn-> protection_type =
HTTP_PROT_DOSARREST;
```

Además el malware incorpora una lista blanca para evitar el ataque a ciertos blancos gubernamentales, así como rutinas para intentar erradicar secuestros de botnets competidoras o eliminar procesos botnet existentes:

```
#DEFINE TABLE_MEM_QBOT // INFORME% S:% S
#DEFINE TABLE_MEM_QBOT2 // HTTPFLOOD
#DEFINE TABLE_MEM_QBOT3 // LOLNOGTFO
#DEFINE TABLE_MEM_UPX // \ X58 \ X4D \ X4E \ X4E \ X43 \ X50 \
X46 \ X22
#DEFINE TABLE_MEM_ZOLLARD // ZOLLARD
```

```
searching for .anime process
       table_unlock_val(TABLE_KILLER_ANIME);
                // If path contains ".anime" kill.
                if (util_stristr(realpath, rp_len - 1,
table_retrieve_val(TABLE_KILLER_ANIME, NULL)) != -1)
                {
                    unlink(realpath);
                    kill(pid, 9);
```

```
                }
table_lock_val(TABLE_KILLER_ANIME);
```

El malware Mirai ha sido estudiado profusamente tras su gran impacto en 2016 y la liberación de su código fuente. Podríamos analizar sus mecanismos internos de funcionamiento. No obstante lo que nos interesa especialmente es poder detectarlo. Para ello se dispone de los hash correspondiente a las diversas arquitecturas en las que intenta penetrar:

```
MD5 ( 'mirai.arm') = b98bc6ab2ed13028cd5178c422ec8dda
MD5 ( 'mirai.arm7') = 33987c397cefc41ce5e269ad9543af4c
MD5 ( 'mirai.mips') = 8e36a1fb6f6f718ec0b621a639437d8b
MD5 ( 'mirai.ppc') = e08befb4d791e8b9218020292b2fecad
MD5 ( 'mirai.sh4') = 030159a814a533f30a3e17fe757586e6
MD5 ('mirai.sparc') = ac61ba163bffc0ec94944bb7b7bb1fcc
MD5 ('mirai.x86') = 6b7b6ee71c8338c030997d902a2fa593
```

La detección de estos hashes es una de las estrategias de localización por un honeypot. No obstante esta técnica, actualmente básica, se combinará con otras estrategias más modernas, como la detección por características en lugar de por firmas.

2.3 Introducción al control industrial (ICS) Industrial Control System

La automatización industrial implica el uso de elementos electromecánicos, electrónicos y computacionales con el objeto de proporcionar procesos automáticos en mayor o menor medida. Aunque su uso principal es en fábricas, control de maquinaria, líneas de montaje y plantas de producción, la automatización se extiende a importantes sectores estratégicos [4].

Los sistemas actuales utilizan básicamente PLCs (Programmable Logic Controller) o sistemas similares con diferentes arquitecturas pero con la misma funcionalidad de lógica de control programable.

Un esquema típico de aplicación podría ser el siguiente:

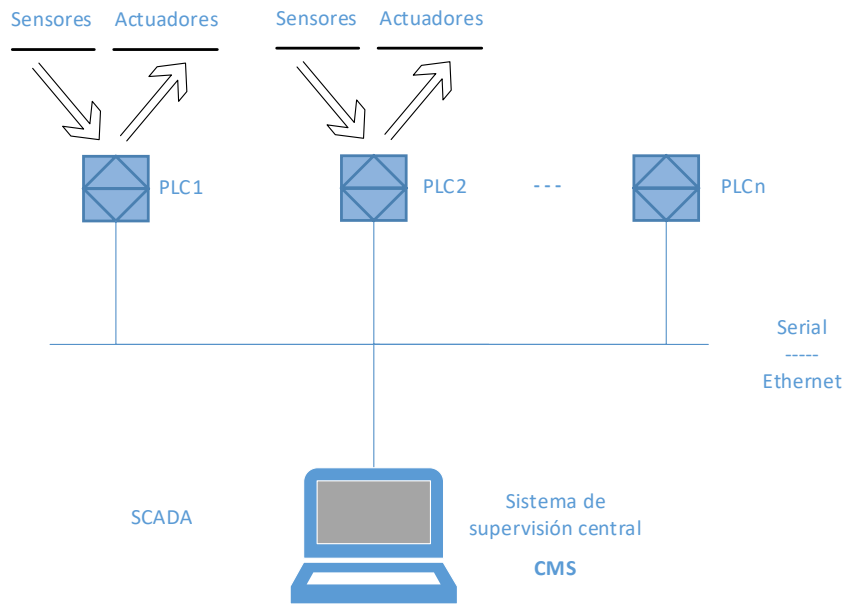


Figura 2. Escenario.

Uno o varios PLCs gestionan actuadores mediante la información proporcionada por los sensores existentes y la programación definida en el software de control que ejecuta.

Existen varios tipos de PLCs así como protocolos de comunicaciones entre los sensores/actuadores y los propios PLCs. Estos protocolos pueden ser entre otros MPI, Modbus, Profinet, Canbus, Modnet, Profibus, etc.

Este concepto de automatización en el que varios PLCs controlan coordinadamente un proceso o parte de él, es lo que se conoce como sistema de control distribuido. En el sistema distribuido los distintos controladores trabajan con una base de datos común de señales. Normalmente existirá un sistema de supervisión central o CMS ejecutando algún software HMI. A todo este conjunto se le denomina sistema SCADA.

Tradicionalmente los sistemas de control distribuidos operaban de manera independiente al resto de la ofimática de la organización. Evidentemente pronto surgió la conveniencia del acceso a datos en la red de producción para incorporarlos a bases de datos pertenecientes a software ERP o similar, utilizados para la gestión empresarial y el análisis de la eficiencia en los procesos productivos. Todo ello para toma de decisiones estratégicas abaladas.

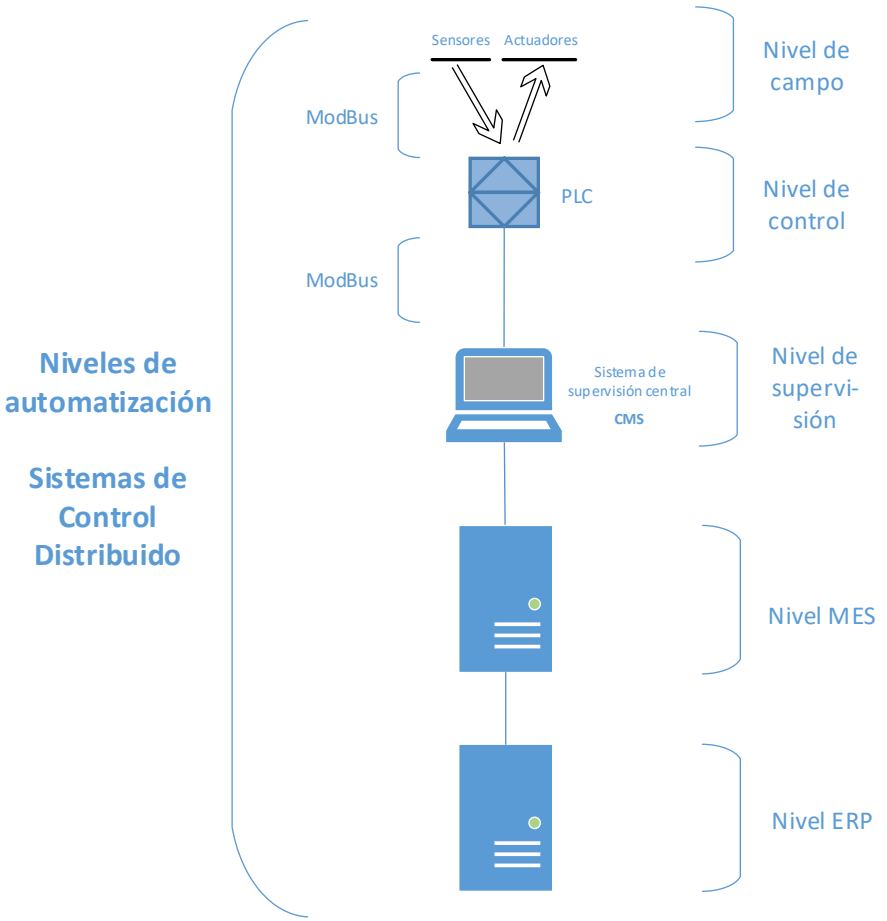


Figura 3. Sistema SCADA.

A partir de la proliferación de este tipo de escenarios, el acceso a Internet desde las redes de producción o a través de las redes IT de la organización se ha convertido en algo habitual. Esto supone un peligro emergente por las nuevas vías de comunicación con las redes de control industrial que antes no existían.

2.4 Vulnerabilidades en las redes de control industrial

En los últimos años ha aumentado notablemente los ataques a redes ICS y existen casos notables ampliamente difundidos. Algunas fuentes de información son organismos como Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), NVD/CVE, SCADA Strangelove, o Siemens Product CERT.

Las vulnerabilidades detectadas son catalogadas según las Common Vulnerability Scoring System (CVSS) versiones 2 y 3. El número de vulnerabilidades específicas ICS continúa creciendo cada año. Así por ejemplo en 2015 ya se contabilizaban 189, casi el 50% de ellas críticas.

Las amenazas presentes en redes ICS siguen un modelo similar a las presentes en el mundo IoT [10]:

Nivel	Amenaza
Application Level	Data Leakage
	DoS Attacks
	Malicious Code Injection
Transportation Level	Routing Attacks
	DoS Attacks
	Data Transit Attacks
Perception Level	Physical Attacks
	DoS Attacks
	Routing Attacks (e.g. in WSN, RSN)
	Data Transit Attacks (in WSN or RSN)

Mirai fue un malware destinado a infectar dispositivos IoT con el objeto de incorporarlos a una botnet y realizar ataques DDoS contra diversos objetivos. Tuvo un gran impacto en el mundo empresarial a nivel mundial. Sin embargo Mirai no fue diseñado para infectar redes industriales específicamente, aunque causara importantes daños en sistemas expuestos en empresas de carácter industrial.

Debemos destacar algunos ejemplos más como BlackEnergy o Stuxnet.

BlackEnergy es un malware que ejecuta ataques de denegación de servicios además de espionaje y destrucción de información. En 2015 afectó a los sistemas SCADA de la empresa de distribución eléctrica de Ucrania para entorpecer la reactivación del sistema eléctrico tras provocar un apagón sin precedentes. BlackEnergy se propaga mediante phishing en emails en donde de descarga un fichero Excel con macros que infectan el equipo en la red de destino.

Otro ejemplo importante es Stuxnet. Éste es un gusano informático que ataca a sistemas SCADA y se piensa que fue el causante de importantes daños tras ciertos ciberataques contra centrales nucleares de Irán. Stuxnet ataca específicamente a dispositivos PLCs a través de equipos Windows infectados que buscan cadenas del protocolo S7. Se trata de un protocolo muy popular de PLCs Siemens. Stuxnet se compone de un gusano, un archivo de enlace y un rootkit. Su introducción en el sistema se realizó a través de un dispositivo USB infectado y se propagó a través de la red.

Stuxnet introduce el rootkit infectado en el PLC con S7, modifica el código y le da instrucciones al PLC mientras retorna un bucle de valores normales del sistema de operaciones a los usuarios. Esto se efectúa a través de una plataforma de gestión industrial bajo Windows denominada WinCC, interceptando las comunicaciones entre el software WinCC y los PLCs.

Stuxnet constituye un gran ejemplo de ataque a redes ICS específicamente. Su estudio en profundidad excede el objetivo de este trabajo, pero supuso un punto de inflexión en la irrupción en este nuevo tipo de amenazas y representa el comienzo de una mayor preocupación por la protección de los entornos ICS.

3. La estrategia honeypot. Consideraciones de diseño general

3.1 Introducción al concepto honeypot

Un honeypot es un software que se ejecuta en un hardware estratégicamente ubicado en la red. Este software es un señuelo diseñado y desarrollado específicamente para ser atacado. El honeypot recibe el ataque en caso de que llegue a él. Esta es la estrategia en la que un servidor simula una serie de puertos abiertos y hace que su respuesta parezca vulnerable.

La idea proviene de la organización sin ánimo de lucro denominada The Money Project creada por Lance Spitzne [1]. El principal objetivo de esta estrategia es la detección del ataque en caso de que llegue a penetrar en el sistema. Esta detección evidenciará no sólo el hecho en sí sino algunos datos del ataque, origen, tipo y posibles daños. La información ayudará a reforzar las medidas de seguridad convenientemente. Así pues el honeypot es una herramienta de investigación.

En general los atacantes utilizan herramientas de software en sus procedimientos de sondeo e intrusión. Dichas herramientas dejan ciertos rastros o bien sus movimientos pueden ser detectados en tiempo real debido a las técnicas que se utilizan. El honeypot utilizará otras técnicas y herramientas paralelas para la detección en tiempo real de la actividad del ataque, algunas de ellas pueden llegar a ser muy complejas como la detección mediante deep autoencoders [9].

Varios equipos honeypot pueden formar una red, denominada honeynet, en la que diversos equipos con distintos sistemas operativos se especializan en detectar diferentes tipos de ataques.

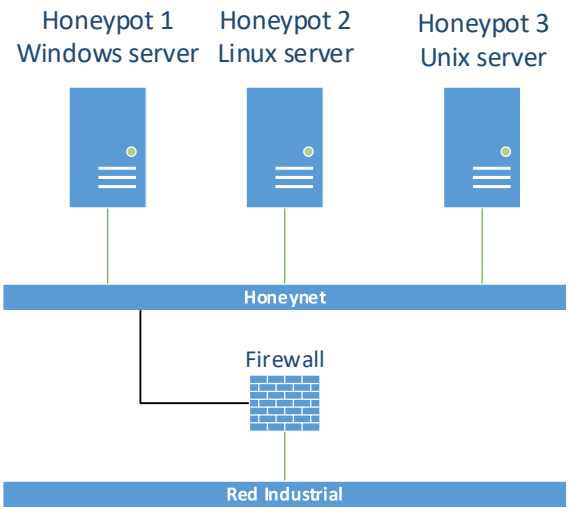


Figura 4. HoneyNet.

Si bien la instalación de una honeynet con diferentes arquitecturas es la solución más completa, su mantenibilidad y actualización será cada vez más difícil y adsorberá cada vez más recursos al tiempo que se incrementa el volumen y complejidad de las amenazas. Por ello en este trabajo nos decantamos por un honeypot de máquina única, fácil de respaldar y lo más polivalente posible.

El honeypot recolectará la mayor cantidad de información disponible. Para ello lo mejor es operar por niveles o capas. Por ejemplo un capturador de tramas nos permitirá detectar las firmas conocidas así como un IDS operará en los niveles de red y de transporte.

3.2 Tipos de honeypot

Los honeypot se pueden clasificar de varias maneras. Las más importantes son atendiendo a un finalidad o según su nivel de interacción. Según la finalidad podemos distinguir entre implementaciones de honeypots para entornos de producción y para investigación. Los primeros se implementan de manera paralela a las redes de datos y monitorizan las actividades de la red en todo momento. Proporcionan protección mediante la detección, prevención y respuesta al ataque. Las implementaciones para investigación por su parte constituyen recursos demostrativos y se centran en el estudio de amenazas de todo tipo y patrones de ataque. Con este tipo es posible determinar tendencias en las actividades intrusivas o prevención anticipada.

El honeypot de este trabajo constituye una implementación pensada para entornos de producción. La idea es que la información que proporcione sea de utilidad para la mejora continua de la infraestructura IT/OT industrial.

En cuanto la clasificación según el nivel de interacción, ésta define el rango de posibles ataques que se le permite tener al atacante, así como el rango de vulnerabilidades que permitiremos que se exploren. Se distinguen dos opciones principales:

1. Honeypot de baja interacción.

Trabajan simulando sistemas operativos y servicios abiertos. La actividad del atacante se encuentra limitada al entorno de emulación. Esto implica en general un bajo riesgo y mayor simplicidad ya que los servicios emulados limitan el riesgo de penetración al no tener acceso al sistema operativo real. Además los honeypot de baja interacción son más fácilmente mantenibles.

Los honeypot de baja interacción capturan datos de ataques preestablecidos por lo que su capacidad de registro está limitada.

Además los servicios simulados que se brindan al atacante están limitados a un cierto nivel de profundidad o límite operacional. En esencia, el honeypot simulará servicios abiertos, habitualmente abrirá determinados puertos TCP y UDP que no pertenecen a servicios reales pero que dará respuestas al atacante como si los fuera.

Esto último se puede conseguir de varias maneras. Así por ejemplo podríamos utilizar el servicio netcat (nc) de Linux para simular este comportamiento. Si quisiéramos simular un servicio HTTP abriríamos con netcat el puerto 80 para escuchar peticiones. Cuando se recibe una conexión ejecutaríamos un script que despliega el contenido de una página web falsa, de señuelo. A partir de ahí mediante respuestas a determinadas acciones del atacante podríamos simular un comportamiento determinado. He aquí un honeypot rudimentario.

En realidad existen ya un buen número de herramientas que sirven para hacer esto mismo. Así pues el diseño e implementación de un honeypot a medida se trata más bien de un trabajo de integración de componentes disponibles, su configuración y su puesta en común.

2. Honeypot de alta interacción.

Bajo este tipo de implementación encontramos sistemas operativos reales instalados en hardware real ejecutando servicios reales. Ello permite capturar una gran cantidad de información cuyo análisis facilita estudiar el modus operandi del atacante permitiendo estudiar nuevas técnicas de ataque. Esta estrategia no da por hecho nada sobre el posible comportamiento del atacante, brindándole un entorno abierto para capturar toda su actividad.

Sin duda un buen uso de los honeypots de alta interacción es la integración de honeynets para el estudio en profundidad de nuevos ataques y vulnerabilidades existentes. Estas honeynets simulan una red de producción real.

La potencia de captura del honeypot de alta interacción lo convierte en un riesgo mayor para la infraestructura IT que lo aloja. Principalmente puede suponer una vía de ataque hacia los demás servidores no honeypot de la infraestructura facilitando un ataque desde dentro.

En nuestro caso, el honeypot que pretendemos implementar, trabajará en una red industrial donde lo que queremos no es el análisis de los tipos de ataques ni el estudio amplio de los mismos, sino la detección de dichos ataques con el objeto de la mejora en los sistemas de defensa de la red industrial. Así pues nuestro diseño se basa en un honeypot de baja interacción.

3.3 Topologías de interconexión

La instalación de un honeypot en la infraestructura de producción se puede hacer de varias maneras. Existen dos modelos principalmente, tal como se muestran en los dos gráficos siguientes.

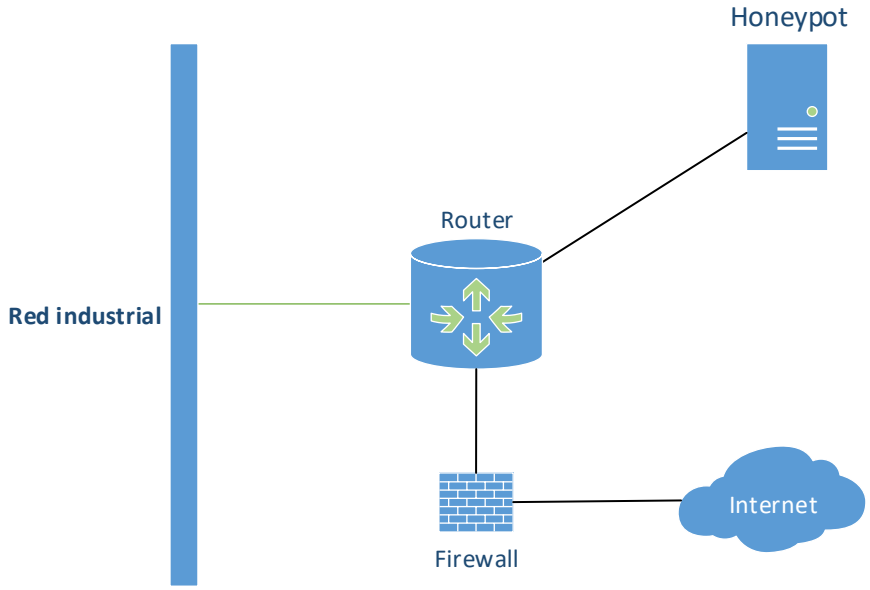


Figura 5. Conexión honeypot tipo I.

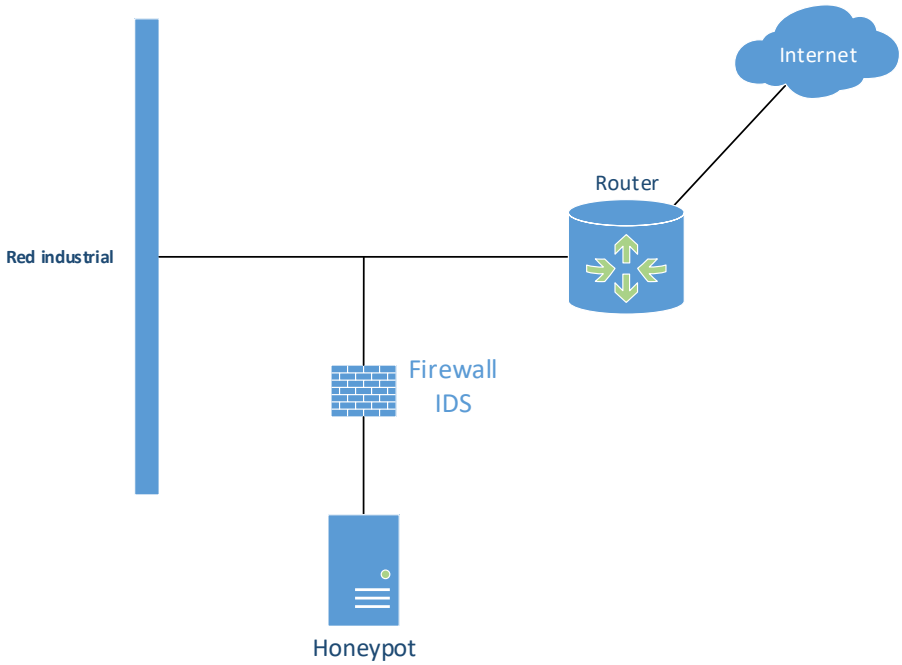


Figura 6. Conexión honeypot tipo II.

Esta última arquitectura es más fácil de implementar, más difícil de detectar y tiene un mantenimiento más seguro. El firewall/IDS materializa

el concepto de Honeynet gateway que antes se implementaban por separado. Este último caso representa el escenario utilizado en este trabajo, en el que integraremos el firewall y el IDS en la arquitectura del honeypot.

4. Diseño del Honeypot para entornos ICS

4.1 Descripción

El objetivo que nos hemos encomendado consiste en integrar un sistema honeypot sencillo capaz de detectar ataques específicos en redes de control industrial. Además de eso deberá detectar cualquier tipo de ataque considerado genérico y también otros propios de diversos dispositivos IoT.

Como se ha comentado anteriormente, nuestro honeypot se ejecutará bajo una máquina virtual, pues se trata de un honeypot de baja interacción. Existen varias opciones para ello. En nuestro caso hemos optado por VMware Workstation Player la cual es gratuita para uso personal y académico. El Workstation Player permite crear y ejecutar máquinas virtuales de todo tipo. Es una plataforma potente y estable la cual ha alcanzado un importante grado de madurez. Como el resto de componentes VMware, permite soportar una amplísima gama de dispositivos virtuales, aspecto en el que otras soluciones open source están menos desarrolladas.

El sistema honeypot constará de varias herramientas. Principalmente de tres: un firewall, un IDS y un emulador de servicios debidamente configurado. Ese último paquete es el honeypot propiamente dicho y podemos encontrar varias soluciones ya implementadas, tanto comerciales como Open Source. La siguiente tabla muestra algunas soluciones generales de baja interacción:

Nombre	**SO**	**Licencia**
SPECTER	Windows	Comercial
HONEYD	Linux	Open Source
KFSENSOR	Windows	Comercial
BackOfficer Friendly (BOF)	Windows	Free
Deception Toolkit (DTK)	Linux	Open Source
PatriotBox	Windows	Comercial

Además de esto existen soluciones de honeypots disponibles para IoT, pero ninguna de ellas por sí sola cumple con los requerimientos suficientes para ser considerados soluciones más o menos efectivas en el entorno industrial.

Por otra parte existen algunas implementaciones destinadas al entorno industrial que se exponen a continuación:

IoTPOT: Simula el Servicio Telnet en varios dispositivos IoT. Es software libre y consiste en un frontend de baja interacción. [16].

Conpot: Se trata de un honeypot open source orientado a protocolos.

IoTCandyJar: Se trata de un honeypot adaptativo que simula varios protocolos y arquitecturas. Es un honeypot potente y sofisticado [15].

GridPot: Se trata de un honeypot open source para la simulación de redes eléctricas.

GasPot: Es un simulador del dispositivo Gaurdian AST del fabricante Veeder Root el cual es un tipo de medidor común en el sector del petróleo y gas. Especialmente usado en tanques de gasolineras.

SCADA-honeynet: Desarrollado por DigitalBond se trata de un conjunto de dos equipos virtualizados, aunque también permite la utilización de un dispositivo físico. Una de las máquinas tiene el sistema de monitorización de la red, WallEye, y firmas de QuickDraw para IDS, que veremos más adelante en nuestra configuración de IDS. La segunda simula un PLC con servicios expuestos para atraer atacantes.

SCADA HoneyNet Project. Se trata de un proyecto que utiliza Honeyd para la simulación de un array de PLCs virtuales.

Tras analizar las ventajas e inconvenientes de cada una de ellas, decidimos utilizar en este proyecto el paquete Honeyd. Efectivamente Honeyd no es un paquete destinado al entorno industrial, ni siquiera a elementos IoT. Sin embargo posee una gran polivalencia permitiendo simular las respuestas de un amplio espectro de dispositivos de todo tipo.

Honeyd es potente, flexible y Open Source. Es un software ya veterano y ha sido mejorado ampliamente pues su aparición data de 2002. Está diseñado para plataformas Linux, otra de las características deseadas en este trabajo, y ejecuta servicios arbitrarios ampliamente configurables. Honeyd puede monitorizar millones de direcciones IP así como direcciones utilizadas para IP stack spoofing. Además es capaz de simular cientos de sistemas operativos.

Por todo ello, Honeyd, pese a llevar cierto tiempo sin actualizaciones recientes, se configura como una opción sumamente operativa, junto con los módulos firewall e IDS.

Efectivamente, otro elemento importante es la incorporación de un IDS [8] a nuestro diseño. El IDS, como sistema de detección de intrusión, recogerá toda la información entrante de aspecto inusual y la registrará, no limitando los procesos de conexión hacia el Honeyd. El IDS complementa al servicio Honeyd detectando intentos de ataques que podemos considerar más genéricos. Al utilizar Honeyd estamos

presuponiendo un sistema Linux. Snort será por lo tanto la opción elegida como IDS, por su potencia, flexibilidad, estandarización y documentación.

El sistema honeypot se conecta habitualmente a un firewall de la infraestructura formando una red DMZ. Dicho firewall es importante porque es el primer eslabón de monitorización y control de las conexiones entrantes. Probablemente no podamos gestionar dicho componente dentro de una red industrial real en producción.

Es necesario pues disponer de un servicio de firewall intrínseco para que el honeypot se encuentre de manera efectiva en una red DMZ, aunque no se conecte directamente a un firewall de la red. Nuestro sistema honeypot dispondrá de iptables como firewall de entrada estándar. Poco hay que decir de iptables, es la solución estándar y potente para sistemas Linux ampliamente utilizada.

4.2 Topología

Nuestro sistema honeypot se diseña para poder ser conectado directamente a la red ICS.

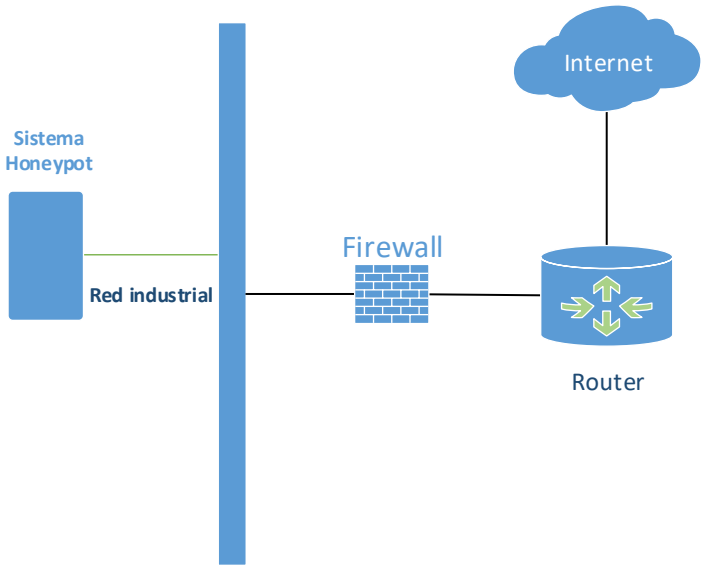


Figura 7. Topología de conexión.

Es posible la conexión a la infraestructura a través de un firewall externo como ya se ha comentado. No obstante nuestro diseño garantizará la protección del resto de la red aunque se inserte directamente en la red de control industrial sin ningún otro elemento intermedio.

4.3 Arquitectura

Según lo visto hasta ahora, debemos hablar más que de un honeypot, de un **sistema honeypot**. Nuestro sistema honeypot dispondrá por tanto de un firewall, un IDS y un servicio de honeypot, todo ello instalado en una máquina virtual y bajo un sistema anfitrión o host real. Esta arquitectura se refleja en la siguiente gráfica:

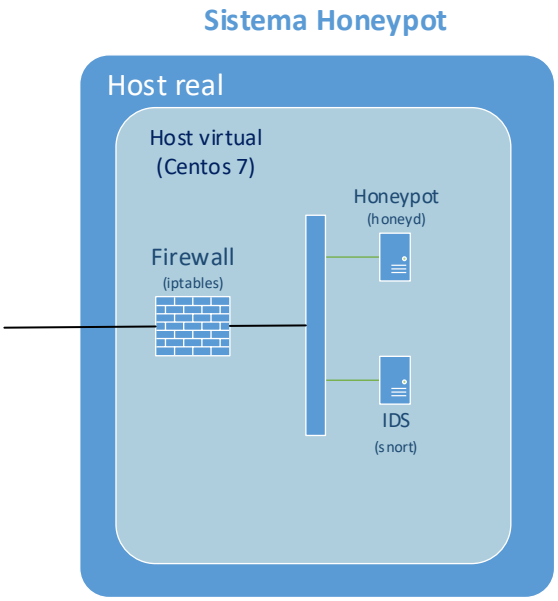


Figura 8. Arquitectura sistema Honeypot.

Definimos así un sistema autocontenido en el que todos los módulos necesarios para la detección se encuentran enclaustrados en la máquina virtual. Cualquier tipo de ataque hacia el honeypot debe quedar limitado sólo a él y no deberá salir de la máquina real hacia la red de producción.

Para entender el funcionamiento de la arquitectura detallaremos a continuación cada uno de los bloques.

4.3.1 Firewall

El firewall del sistema honeypot es el primer componente que se encontrará un atacante que acceda al interfaz de red del equipo que aloja el sistema. El firewall permitirá seleccionar qué tipo de tráfico se quiere examinar y alertará en el momento de un ataque.

Pero el objetivo del firewall no es tanto bloquear la entrada de conexiones sino más bien la salida desde el sistema hacia el exterior. Si el honeypot acaba siendo comprometido por algún tipo de ataque, el firewall evitará realizar conexiones o sondeos hacia el resto de la red industrial. Es decir el firewall protegerá a la red industrial de ataques desde dentro de la propia red provenientes del honeypot.

4.3.2 Sistema IDS

La función principal del IDS es detectar tipos de ataques genéricos que no implican en muchos casos ningún intento de vulneración en puertos TCP o UDP asociados a servicios determinados. Efectivamente los servicios simulados del honeypot no están hechos para enfrentarse a un ataque tipo IP spoofing por ejemplo. Necesitamos un componente para la detección de una ristra importante de ataques a nivel de red y enlace de datos. Dicho elemento es un IDS. Además el IDS permitirá la detección en base a firmas almacenadas, identificando y alertando de estos ataques bien conocidos.

4.3.3 Servicios Honeypot

El servicio honeypot corresponderá al demonio que recibe un intento de conexión y penetración indebido, tras pasar por el firewall y el IDS. Este intento de conexión aparenta ser legítimo merced a que el servidor está esperando conexiones con normalidad y proporciona unas respuestas coherentes con el servicio abierto. Es un servicio operativo y por lo tanto un intento de conexión ordinario no será frenado ni por el firewall ni detectado como amenaza por el IDS.

Los servicios habituales disponibles en dispositivos IoT son principalmente HTTP, telnet, SSH, FTP y en menor medida SNMP. Ahora bien, los protocolos industriales transportados a través de redes Ethernet también implican servicios específicos de control industrial y están expuestos a nuevos tipos de ataques. Como ya se ha comentado nuestro objetivo es detectar este tipo de intrusiones basados en estos protocolos industriales que funcionan bajo Ethernet. Nuestro sistema honeypot está basado en Ethernet, y es a través de Ethernet por donde los hackers intentarán penetrar en los sistemas industriales. Esto es evidente, la penetración en segmentos de control industrial basados en otras tecnologías de operación a bajo nivel y más tradicionales en el sector, como los segmentos tipo bus, son más difíciles y requieren en la mayoría de los casos una presencia física en el medio.

Estamos hablando de protocolos de comunicación industrial de tipo bus de campo como: Interbus, Profibus, CANBus, DeviceNet, Fieldbus, Lonworks, Modbus y Bitbus entre otros. Estos protocolos operan en segmentos restringidos y por su diseño y operación pueden resultar muy complejos de vulnerar, requiriendo acceso físico al bus en la instalación.

Sin embargo, a nivel de célula y especialmente a nivel de planta, empezó a ser habitual utilizar Ethernet para la interconexión de PLCs, sistemas HMI y robots entre otros. Esto ha sido así por su popularidad y avances en el mundo empresarial y su bajo costo en comparación con otro tipo de despliegues. Hablamos ahora de protocolos como EtherCAT, EtherNet/IP, PROFINET, POWERLINK, SERCOS III, CC-Link IE, y Modbus-TCP.

Como ya se ha expuesto, el hecho de monitorizar y permitir el análisis de datos procedentes de sistemas de control en los niveles de planta, significa que estos equipos también se extienden en la dirección hacia las redes IT o redes ofimáticas y sus conexiones a Internet. Esto aumenta enormemente la exposición de las redes de control industrial a intrusiones y amenazas.

En este trabajo proponemos cubrir un entorno simulado con los siguientes protocolos: telnet, FTP, HTTP, Modbus y Step7 (S7) como punto de partida. Además queremos simular el comportamiento de PLCs. Todo ello se puede hacer mediante una configuración particularizada de Honeyd.

Como ataques genéricos pretendemos entre otras cosas capturar algunos tipos de ataque de fuerza bruta sobre SSH, actividad maliciosa sobre HTTP y transferencias de ficheros ilícitos sobre FTP.

4.4 Topología simulada

Mediante la flexibilidad y potencia de los componentes de software seleccionados, es posible recrear casi cualquier entorno de casi cualquier tipo que nos podamos imaginar. Sólo es cuestión de adecuar la programación de los sistemas ya disponibles o crear nuestros propios scripts a tal efecto.

En nuestro caso vamos a simular una red industrial típica compuesta por dos PLCs de fabricantes y protocolos diferentes, un equipo Windows como CMS y un router. Todo ello conectado en un mismo segmento de red Ethernet.

Dicha topología lógica se representa a continuación, incluyendo los protocolos seleccionados y el direccionamiento IP:

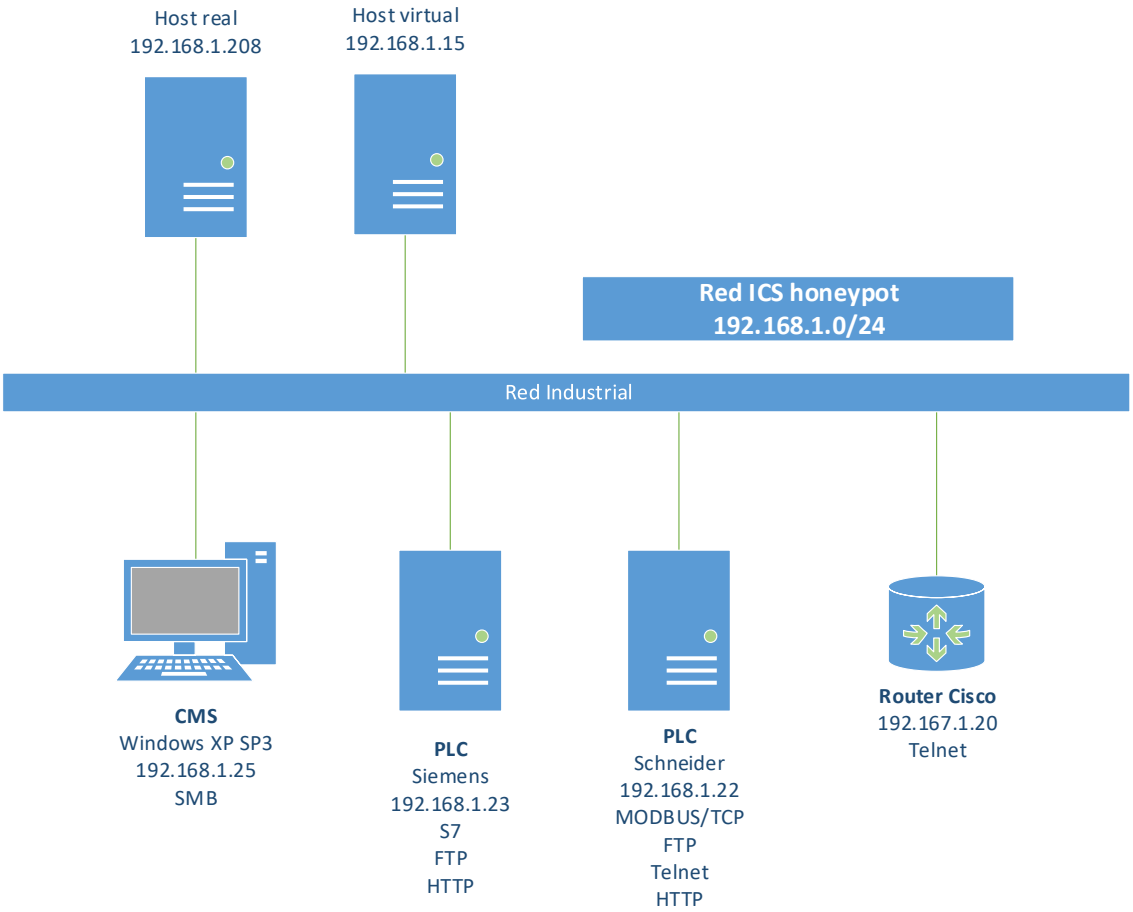


Figura 9. Topología honeypot ICS simulada.

4.5 Software

El software utilizado en la implementación de nuestro sistema queda resumido en la siguiente tabla.

Nro.	Software	Especificación
1	VMware Workstation Player 15.0.4	1.3GHz or faster core speed 2GB RAM minimum/ 4GB RAM or more recommended
2	CentOS-7-x86_64	RAM = 1 Gb HD = 10 Gb
3	iptables	V1.4.21
4	Snort	V.2.9.12
5	Honeyd	1.6d
6	Perl	V5.16.3
7	Python	V.2.7.5

Estas versiones han sido probadas en su conjunto pero otras combinaciones deberían dar un resultado adecuado en la mayoría de los casos.

5. Integración y pruebas del Honeypot.

El sistema honeypot que pretendemos implementar no es un producto como tal, sino una arquitectura que operará como un conjunto.

5.1 Máquina anfitrión.SO y características.

Para nuestras pruebas, la implementación del sistema honeypot se realiza bajo VMware Workstation 15 ejecutado en un sistema operativo real Windows 10. Las características del hardware utilizado son las siguientes:

Versión:	10.0.17134 compilación 17134
Fabricante del SO:	Microsoft Corporation
Nombre del sistema:	Asus
Fabricante del sistema:	ASUSTeK COMPUTER INC.
Modelo del sistema:	K55VD
Tipo de sistema:	PC basado en x64
SKU del sistema:	ASUS-NotebookSKU
Procesador:	Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz, 2301 Mhz, 4 procesadores principales, 8 procesadores lógicos
Versión y fecha de BIOS:	American Megatrends Inc. K55VD.304, 24/05/2012
Versión de SMBIOS:	2.7
Versión de controladora:	255.255
Modo de BIOS:	Heredado
Fabricante placa base:	ASUSTeK COMPUTER INC.
Rol de plataforma:	Móvil
Directorio de Windows	C:\WINDOWS
Directorio del sistema	C:\WINDOWS\system32
Dispositivo de arranque	\Device\HarddiskVolume1
Configuración regional	España
Capa de abstracción hd:	Versión = "10.0.17134.619"
Memoria física instal·lada:	8,00 GB
Memoria física total:	7,89 GB
Memoria física disponible:	959 MB
Memoria virtual total	13,6 GB
Memoria virtual disponible	5,08 GB
Espacio de paginación:	5,75 GB
Archivo de paginación:	C:\pagefile.sys
Protección DMA kernel:	Desactivada
Hyper-V - Extensiones de modo de monitor VM:	Sí
Hyper-V - Extensiones de traducción dirección 2º nivel:	Sí
Hyper-V - Virtualización habilitada en Firmware:	Sí
Hyper-V -Protección de ejecución de datos:	Sí

En cuanto al sistema operativo que ejecuta el sistema simulado, hemos optado por un Centos 7 en su última versión disponible 7.6. A partir de la versión 7, Centos sólo está oficialmente disponible en una arquitectura de 64 bits.

5.2 Instalación de Centos 7

La instalación de Centos 7.6 no tiene nada especial. Simplemente descargamos la imagen del DVD de uno de los mirrors disponibles en:

http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1810.iso

En el proceso de configuración inicial seleccionamos Servidor con GUI dentro del Entorno Base. Esto instalará un gnome suficiente para administrarlo más fácilmente. Posteriormente será posible instalar cualquier paquete de los repositorios CentOS configurados por defecto para la distro.

5.3 Instalación de iptables, Snort y Honeyd

5.3.1 iptables

En Centos 7 se ha sustituido iptables por el nuevo demonio llamado firewalld. Es probable que se vaya extendiendo el uso de firewalld, hoy presente en unas pocas distribuciones, con el respaldo de Red Hat. No obstante nosotros vamos a prescindir de él y restauraremos el servicio iptables.

Realmente firewalld no sustituye al backend iptables, sino al Servicio. El servicio firewalld gestiona al backend iptables, junto con otros componentes.

Realizamos las siguientes operaciones:

Deshabilitando el servicio firewalld:

```
# systemctl stop firewalld
# systemctl mask firewalld
```

Chequeamos el status:

```
# systemctl status firewalld
```

Instalar y habilitar el servicio iptables:

```
# yum install iptables-services
# systemctl enable iptables
```

```
# systemctl start iptables
```

Chequear el servicio iptables:

```
# systemctl status iptables
```

A partir de aquí podemos usar iptables igual que hacíamos en Centos 6.

5.3.2 Snort

Para la instalación de Snort acudimos a la última versión disponible en su web oficial https://www.snort.org/.

Snort utiliza unas librerías llamadas Data Acquisition library (DAQ) que es preciso instalar también.

```
# yum install https://www.snort.org/downloads/snort/daq-2.0.6-
1.f21.x86_64.rpm

# yum install https://www.snort.org/downloads/snort/snort-
2.9.12-1.centos7.x86_64.rpm
```

Una vez hecho esto, podemos tener problemas con la librería libdnet, la cual se haya instalada pero no es reconocida por Snort por una discrepancia en la nomenclatura. Debemos realizar lo siguiente:

```
# cd /usr/lib64
# ln -s libdnet.so.1.0.1 libdnet.1
```

Ahora desactivamos el inicio automático de Stort ya que queremos controlarlo manualmente:

```
# systemctl stop snortd
# systemctl disable snortd
```

5.3.3 Honeyd

La instalación de Honeyd deberá ser hecha descargándonos las fuentes, instalando las dependencias oportunas y compilando.

1. Descarga de Honeyd.

La última versión disponible se encuentra en el repositorio GitHub y la descargamos de la siguiente manera:

```
# git clone https://github.com/DataSoft/Honeyd.git
```

Se trata de la versión 1.6d la cual es compatible con las librerías estándar de Centos 7.

2. Instalación de dependencias.

Honeyd depende de las siguientes librerías:

- libevent - event notification
- libdnet - packet creation
- libpcap - packet sniffing
- libpcre - perl regular expression library

En adición a nuestra instalación de Centos 7, debemos instalar lo siguiente:

```
# yum install libevent-devel libpcap-devel libedit-devel
libdnet-devel zlib-devel
```

Además para la ejecución de la simulación snmp necesitaremos algunos módudos perl adicionales:

```
# yum install perl-Convert-BER.noarch perl-CPAN.noarch
perl-SNMP_Session.noarch
```

3. Compilación e instalación de Honeyd.

Una vez instaladas las dependencias anteriores la compilación se ejecuta sin problemas:

```
# ./autogen.sh
# ./configure
# make
# make install
```

Verificamos la ejecución del binario:

```
# ./honeyd -V
Honeyd V1.6d Copyright (c) 2002-2007 Niels Provos
honeyd[23128]: started with -V
Honeyd Version 1.6d
```

5.4 Configuración del firewall

Nuestros objetivos para iptables son:

- Permitir acceso sólo a conexiones hacia servicios determinados, bloqueando el resto.
- Permitir tráfico saliente sólo de conexiones establecidas previamente.
- Registrar en un log los paquetes bloqueados.
- Proteger contra ataques SYN flood.
- Bloquear paquetes con flags TCP falsos.

Vamos a configurar iptables para que no permita conexiones salientes del equipo pero sí entrantes. De hecho vamos a permitir conexiones

entrantes hacia algunos puertos bien conocidos, los cuales serán gestionados por el honeypot.

Como sabemos hay dos maneras de configurar iptables, en línea de comandos o editando directamente el fichero:

```
/etc/sysconfig/iptables
```

Otra opción es usar alguna utilidad gráfica como system-config-firewall. No obstante la nomenclatura usada en las reglas es la comúnmente utilizada.

Dividiremos las configuraciones en bloques funcionales. Esta configuración se incluye como anexo al final debido a su longitud.

Otro punto importante es la configuración de logs en iptables. Para registrar en un log la actividad de iptables utilizamos el modificador `-j LOG`. Podemos definir los niveles de registro mediante la opción `--log-level <nivel>`, en el que <nivel> puede ser uno de los identificadores normales a nivel syslog: emerg, alert, crit, error, warning, notice, info o debug. Además existen opciones adicionales como `--log-prefix <prefix>` que permite especificar un prefijo personalizable a los mensajes.

En concreto nos interesa registrar los paquetes bloqueados por la cadena INPUT, para lo cual introducimos la siguiente regla:

```
iptables -A INPUT -j LOG --log-level info  --log-prefix
"IPTABLES-DROP: "
```

Con esta regla se registran los mensajes en journald y se envían a rsyslog por defecto y serán almacenados en el fichero regular /var/log/messages.

Por otra parte los mensajes se registran en journald como mensajes del kernel, por lo que también se puede hacer una consulta de mensajes del kernel de manera continua:

```
journalctl –k -f
```

Ya tenemos integrado un log de iptables con rsyslog. Ahora vamos a derivar este tráfico escogido a un fichero particular. Para ello primero creamos el archivo destino del log:

```
# touch /var/log/iptables.log
```

Ahora editamos el fichero /etc/rsyslog.conf y localizamos la sección que comienza por #### RULES ####. Agregamos:

```
:msg, startswith, "IPTABLES" -/var/log/iptables.log
& ~
```

La primera línea envía el registro con la coincidencia "IPTABLES" y la segunda línea “& ~” hace que rsyslog descarte esos mensajes para que no se repita el registro en los dos sitios.)

Con todo esto hemos conseguido que iptables registre en el log los paquetes bloqueado solamente. Efectivamente los intentos de conexiones permitidas las registraremos en el honeypot.

5.5 Configuración de Snort

Nuestros objetivos con Snort son los siguientes:

- Detectar ataques de tipo genérico
- Detectar algunos ataques específicos IoT basados en firmas.

La herramienta Snort posee cuatro modos de trabajar:

1. Sniffer de paquetes.
2. Packet logger, almacenando los paquetes capturados para un análisis posterior.
4. Network Intrusion Detection, como motor de comparación que intenta reconocer diferentes tipos de ataques de red.
5. Modo inline, actuando como un IPS (Intrusion Prevention System).

Realmente podemos considerar un IPS como un IDS que bloquea el tráfico detectado como atacante.

En nuestro sistema será configurado como un IDS. Snort se configura mediante un fichero en /etc/snort/snort.conf.

Un parámetro importante es el llamado RULE_PATH y especificado como:

```
var RULE_PATH /etc/snort/rules
```

Esta línea indica el path donde se ubican ficheros con las reglas utilizadas por Snort. Estos ficheros son llamados mediante sentencias *include* normalmente situadas al final del fichero snort.conf.

Snort proporciona una serie de reglas estándar de detección, que no están presentes inicialmente en nuestra instalación. Las descargamos de:

```
https://www.snort.org/downloads/community/community-rules.tar.gz
```

Descomprimimos el fichero tar y copiamos community-rules en su sitio:

```
tar -xzvf community-rules.tar.gz
cp community.rules ../
```

En el fichero /etc/snort/snort.conf colocaremos la siguiente línea al final:

```
include $RULE_PATH/community.rules
```

Además de todo lo anterior, podemos ver que en el fichero /etc/snort/snort.conf existen llamadas a multitud de ficheros de reglas que no existen en el equipo. Estas reglas están preconfiguradas y contienen detección de anomalías genéricas pero necesitan ser descargadas aparte. Para ello:

```
# yum install
https://forensics.cert.org/centos/cert/7/x86_64//snort-sample-
rules-2.9.12-1.el7.noarch.rpm
```

Ahora podemos ver en /etc/snort/rules todos los ficheros de reglas que inicialmente no se incluían en la instalación.

Con todo esto ya tenemos un Snort preparado para la detección de los ataques más habituales.

Existen auténticas librerías de reglas de detección estándar, como ya hemos visto, y se suele utilizar el fichero llamado local.rules para las reglas personalizadas que incluyamos.

Dicho fichero ya es reconocido por snort.conf, el cual debe contener la línea:

```
include $RULE_PATH/local.rules
```

Además desactivamos el uso de reglas dinámicas, que no usamos en nuestra configuración, para evitar ciertos warnings relacionados con la localización de esas librerías:

```
#dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

Otro cambio a realizar es el siguiente. Sustituir

```
#var RULE_PATH /etc/snort/rules
#var SO_RULE_PATH ../so_rules
#var PREPROC_RULE_PATH ../preproc_rules

#var WHITE_LIST_PATH ../rules
#var BLACK_LIST_PATH ../rules
```

Por:

```
var RULE_PATH rules
var SO_RULE_PATH so_rules
var PREPROC_RULE_PATH preproc_rules

var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
```

Una vez tenemos el IDS configurado para detectar ataques genéricos vamos a incluir firmas específicas destinadas a objetivos IoT de carácter industrial. Este es un punto de configuración crucial en el sistema honeypot que proponemos.

Snort forma parte del conjunto detectando anomalías e intentos de intrusión enfocados a entornos industriales. Para lograr ello importaremos las firmas especializadas orientadas a protocolos industriales, como MODBUS o S7. Estas firmas estás disponibles en varias fuentes. Especialmente importaremos las reglas disponibles en:

```
https://github.com/digitalbond/Quickdraw-Snort
```

Quickdraw es un proyecto de investigación para crear una aplicación de generador de registro de eventos de seguridad pasiva para dispositivos IoT o dispositivos de campo que carecen de capacidades de registro de seguridad.

Importaremos el fichero all-quickdraw.rules en /etc/snort/rules/ e incluiremos la siguiente entrada en /etc/snort/snort.conf:

```
include $RULE_PATH/all-quickdraw.rules
```

Además de eso algunas variables deberán ser definidas en snort.conf para nuestra red local en concreto.

Otra fuente de reglas SCADA son las generadas por Lenny Hansson a partir de capturas de laboratorio. Este fichero se puede obtener en:

```
https://networkforensic.dk/SNORT/SCADA.zip
```

Descargaremos este fichero y lo cargamos en Snort de la misma manera que hemos hecho antes.

El estudio y desarrollo de las reglas Snort para los protocolos industriales puede ser complejo en algunos casos. Existen varios estudios e iniciativas como por ejemplo el trabajo de Hao y otros [12] sobre el protocolo NP3.

En cuanto a los logs, Snort los guarda en /var/log/snort con la siguiente estructura:

- alert: contiene metadatos de alerta en formato de texto
- snort.log.########## - archivos PCAP de los paquetes que activaron la alerta

Por tanto el archivo /var/log/snort/alert será nuestro fichero de log de referencia para la salida de datos de Snort.

5.6 Configuración de Honeyd

Los ficheros de configuración de Honeyd se encuentran en

```
/usr/share/honeyd
```

Aquí están disponibles algunos ficheros de referencia que deberemos utilizar. Veamos algo más sobre estos ficheros:

`config.sample`: es un fichero de configuración de ejemplo. En él se definen los equipos que se van a emular.

`nmap.assoc`: *personalities* o lista de fingerprints que se pueden emular y son detectados por nmap.

`nmap-os-db`: base de datos de fingerprintring OS nmap de 2ª generación.

`nmap-mac-prefixes`: identificadores (OUIs) de fabricantes por el comienzo de la dirección MAC.

`p0f`: archivo de firmas para escaneo pasivo.

`xprobe2.conf`: fichero de configuración/fingerprints para XPROBE2.

Estos ficheros darán una respuesta controlada cuando un atacante trata de averiguar el sistema operativo mediante escaneos activos y pasivos por las herramientas indicadas.

El formato de ejecución de Honeyd con las opciones más importantes es el siguiente:

```
Honeyd [-dP] [-l logfile] [-i interface] [-p personalities]

[-x xprobe] [-a assoc] [-f config] [net…]
```

Las opciones disponibles más importantes son:

`-d`

Ejecuta Honeyd sin demonizar y habilita los mensajes de depuración detallados.

`-p`

Se ejecuta en modo de sondeo (polling) en lugar de usar pcap para el registro.

`-l logfile`

Es el archivo en el que se escriben los registros de Honeyd.

`-i interface`

Interfaz es el número de la interfaz de red.

`-p file`

Personalities permite a Honeyd leer huellas digitales (fingerprints). Las huellas dactilares son archivos nmap que están diseñados para definir el comportamiento de la pila TCP/IP simulada. Hay dos archivos incorporados en nmap: nmap.assoc y nmap.print.

`-x file`

Permite especificar un archivo de fingerprints del estilo xprobe. Un archivo xprobe controla cómo responde Honeyd cuando alguien usa una herramienta de huellas dactilares ICMP en su contra.

`-a assocfile`

Permite especificar un archivo de asociación que asocia las huellas digitales de estilo nmap con las huellas digitales de estilo xprobe.

`-m`

Permite especificar la localización para el fichero que contiene los prefijos MAC que se utilizan con nmap.

`-f config`

Le dice a Honeyd que lea una configuración de un archivo específico. Un archivo de configuración le dice a Honeyd qué scripts ejecutar.

`net`

No es realmente un parámetro, sino un marcador de posición para una dirección IP o un rango de direcciones IP. La opción permite especificar un rango IP que Honeyd emulará. Si no se especifica una dirección o un rango de direcciones, Honeyd reclamará cualquier dirección IP para la cual detecte tráfico.

Después de instalar Honeyd, debemos definir las direcciones IP en las que el software escuchará, los sistemas operativos virtuales, la emulación de huellas dactilares (fingerprint) que Honeyd utilizará, los puertos IP que imitará el software y las opciones de registro que se utilizarán. Estas opciones se definen como parámetros en línea de comandos o en el archivo de configuración.

En el archivo de configuración de Honeyd se identifican las plantillas de hosts virtuales, las direcciones IP vinculadas a esas plantillas y las "*personalities*" de sistemas operativos y puertos correspondientes.

Los *templates* o plantillas, son el método que usa Honeyd para realizar el seguimiento del entorno del sistema operativo virtual. En cada template se define una IP única asociada a la personality y los puertos usados por el SO virtual entre otros parámetros.

Se debe especificar una plantilla predeterminada que se usa cuando no se aplica ninguna otra.

Veamos algunos parámetros importantes a incluir:

```
create <templatename>
```

Define una plantilla o template para un SO simulado.

Para enlazar una dirección IP con un template en concreto utilizamos la declaración:

```
bind <ipaddress> <templatename>
```

Normalmente estas direcciones estarán en el rango real de la máquina que ejecuta el Honeypot, pero no tiene que ser necesariamente así. En caso de usar un rango diferente se deben configurar las rutas correspondientes en la infraestructura para que sean accesibles al atacante.

Honeyd utiliza el concepto de *personality* de un sistema operativo. Una personality define el stack IP de un SO. Es decir la respuesta particular y ligeramente diferente que un SO proporciona.

Las personalities utilizan las bases de datos de huellas dactilares, o fingerprints, de nmap y xprobe2, utilidades éstas bien conocidas para el sondeo y utilizadas habitualmente por los atacantes. Para ello se utilizan los siguientes ficheros:

Nmap.prints (actualmente sustituido por nmap-os-db) contiene las respuestas del stack IP para las diferentes personalities.

Nmap-assoc compara los fingerprints de Xprobe2 con Nmap para mantener la consistencia entre las respuestas de ambas herramientas.

Para conocer qué sistemas operativos podemos incluir como personality debemos consultar en el fichero nmap-os-db los fingerprints disponibles. Por otra parte Honeyd usa el archivo xprobe2.conf para simular la sección ICMP de las pilas del sistema operativo.

A la hora de configurar un sistema operativo (personality) debemos escribirlo exactamente igual al que aparece en nmap-os-db.

La nomenclatura que utilizamos para definir un sistema operativo es la siguiente:

```
set <templatename> personality "<personalityname>"
```

Con todo ello para crear un template default como ejemplo hacemos:

```
annotate "Windows NT 4.0 Server SP5-SP6"
create default
set default personality "Windows NT 4.0 Server SP5-SP6"
bind 192.168.150.15 default
```

Por último debemos asignar los puertos al sistema operativo simulado. La sintaxis se muestra a continuación:

```
set default default tcp action open
set default default udp action reset
```

Estas dos sentencias indican a Honeyd cómo tratar intentos de conexión si no se ha definido para un puerto específico.

Otros ejemplos de asignaciones pueden ser los siguientes:

```
add default tcp port 21 open
add default tcp port 137 open
add default tcp port 139 block
```

Ahora bien ¿qué respuesta da Honeyd a nivel de aplicación? Si se simula por ejemplo un servidor web se necesitará emular las salidas de dicho servidor tras una conexión HTTP, es decir, se necesitará servir algún tipo de página web. Para ello Honeyd utiliza dos posibilidades.

1. Derivar la conexión a un tercero, a un proxy, para que sea él el que proporcione el contenido del servicio. Un proxy es una máquina externa configurada para aceptar las redirecciones de Honeyd para servicios especificados. Esto permite dar al atacante una impresión más real al actuar sobre un servidor real y no una simulación limitada.

Un ejemplo de configuración de la opción proxy web es la siguiente:

```
add default tcp port 80 proxy 192.168.1.50:80
```

2. Utilizar scripts que proporcionan servicios emulados.

En Honeyd podemos simular servicios que responden en diferentes puertos mediante scripts de servicio que interactúan con Honeyd. Parte de la potencia de Honeyd como honeypot se debe a esta capacidad. Podemos usar varios lenguajes como Python, Perl, C, C++ o incluso programar en Shell script nuestros propios scripts.

Honeyd proporciona una serie de scripts por defecto que cubren de una manera general la mayoría de servicios bien conocidos. Asimismo están disponibles en Internet scripts más o menos especializados de diferentes fuentes.

Para añadir un script a un puerto determinado dentro del archivo de configuración utilizamos una nomenclatura como el siguiente ejemplo:

```
add default udp port 21 "sh scripts\ftp.sh”
```

Es posible probar muchos de estos scripts desde el interfaz de comandos antes de incorporarlos a Honeyd. Por ejemplo:

```
# cd /usr/share/honeyd/scripts
# ./ftp.sh
```

En este trabajo creamos un fichero de configuración personalizado y lo colocamos en /etc:

```
/usr/share/honeyd/honeyd.conf
```

Ejecutaremos el demonio de la siguiente manera

```
# honeyd -d -p nmap-os-db -l /var/log/honeyd.log -i ens33 -f
./honeyd.conf 192.168.1.0/24 -u 0 -g 0 --disable-webserver
```

Por otra parte nuestra red de servicios honeypot se compondrá de los siguientes elementos con los protocolos y puertos correspondientes como ya han sido definidos:

- Un router Cisco. Daremos algunas respuestas locales de un supuesto router de acceso a la red ICS.
    - Telnet, puerto TCP 23.
- PCL SCHNEIDER. Para la simulación de un PLC Schneider vamos a utilizar los scripts disponibles en el SCADA HoneyNet Project [17]. En concreto se simulan los siguientes protocolos:
    - Telnet, puerto TCP 23. Se simula el servidor telnet de un Schneider TSX ETY5101.
    - FTP, puerto TCP 21y 21. Si bien las respuestas frente al login serán como las del modelo anterior, no se podrá hacer login en ningún momento.
    - HTTP, puerto TCP 80. Se simula un frontend básico propio del TSX ETY5101.
    - MODBUS, puerto TCP 502. El script proporcionado simula una serie de funciones, todas ellas presentes en el TSX ETY5101:
        - 1 - Read Coil Status
        - 2 - Read Input Status

- 3 - Read Holding Registers
- 4 - Read Input Registers
- 5 - Write Coil
- 6 - Write Single Register
- 15 - Force Multiple Coils
- 16 - Write Multiple Registers

- PLC SIEMENS. Al igual que el caso anterior, utilizaremos los scripts ya disponibles para la simulación de los siguientes servicios:

    - FTP, puertos TCP 20 y 21. Se simula el servidor FTP del modelo CP 343-1 IT. No es posible hacer login en la simulación.
    - HTTP, puerto TCP 80. Se simula un frontend básico propio del CP 343-1 IT.
    - S7, puerto TCP 102. Se proporciona una simulación básica del servidor S7 interno del CP 343-1 IT.

- Equipo CMS Windows XP. En la red ICS debe haber algún equipo CMS de control y monitorización de la infraestructura de señales. Normalmente son sistemas Windows. Debido a la longevidad de las instalaciones industriales, se encuentran operativos sistemas Windows muy antiguos como XP. Simulamos aquí un sistema XP SP3 con los siguientes servicios habituales en los CMS:

    - SMB, puerto TCP 139 y 445, UDP 137 y 138.
    - HTTP puerto 80.

    En cuanto a las direcciones MAC asignadas a cada dispositivo, debemos configurar un prefijo de cada una acorde al fabricante del mismo o al chipset de red que utiliza. Para ello seleccionamos en el fichero nmap-mac-prefixes los primeros 3 bytes que mejor corresponda.

    Finalmente en lo referente a los fingerprints utilizados y que se especifican como *personality* en Honeyd, se encuentran disponibles en el fichero nmap-os-db y no se requiere un recreación de los mismo.

    Con todo ello, para lograr este comportamiento generamos el siguiente contenido en el fichero de configuración honeyd.conf:

```
### Cisco router
create router
set router personality "Cisco 1700 router (IOS 12.0)"
set router ethernet "00:02:4b:bb:00:25"
set router default tcp action closed
add router tcp port 23 "/usr/bin/perl scripts/routerCisco/router-
telnet.pl"
set router uptime 1327650
bind 192.168.1.20 router

#### PLC SIEMENS
create plcSiemens
set plcSiemens ethernet "00:1f:f8:bb:00:23"
set plcSiemens personality "Siemens Simatic 300 programmable
logic controller"
set plcSiemens default tcp action closed
add      plcSiemens      tcp      port      80      "/usr/bin/perl
scripts/plcSiemens/honeyd-http-siemens.py"
add      plcSiemens      tcp      port      20      "/usr/bin/perl
scripts/plcSiemens/honeyd-ftp-siemens.py"
add      plcSiemens      tcp      port      21      "/usr/bin/perl
scripts/plcSiemens/honeyd-ftp-siemens.py"
#add plcSiemens udp port 161 "perl scripts/plcSiemens/fake-
snmp.pl              public              private              --
config=/usr/share/honeyd/scripts/plcSiemens/"
add      plcSiemens      tcp      port      102      "/usr/bin/perl
scripts/plcSiemens/honeyd-s7.py"
set plcSiemens uptime 3892650
bind 192.168.1.23 plcSiemens

#### PLC SCNEIDER
create plcSchneider
set plcSchneider ethernet "00:11:00:bb:00:22"
set plcSchneider personality "Schneider Electric TSX ETY
programmable logic controller"
set plcSchneider default tcp action closed
add      plcSchneider      tcp      port      80      "/usr/bin/perl
scripts/plcSchneider/honeyd-http-schneider.py"
add      plcSchneider      tcp      port      20      "/usr/bin/perl
scripts/plcSchneider/honeyd-ftp-schneider.py"
add      plcSchneider      tcp      port      21      "/usr/bin/perl
scripts/plcSchneider/honeyd-ftp-schneider.py"
add      plcSchneider      tcp      port      23      "/usr/bin/perl
scripts/plcSchneider/honeyd-telnet-schneider.py"
add      plcSchneider      tcp      port      502      "/usr/bin/perl
scripts/plcSchneider/honeyd-modbus.py"
set plcSchneider uptime 38926650
bind 192.168.1.22 plcSchneider

#### WinXPComputer
create WinXPComputer
set WinXPComputer personality "Microsoft Windows XP SP3"
set WinXPComputer ethernet "00:0c:f1:bb:00:25"
set WinXPComputer default tcp action closed
set WinXPComputer default udp action closed
add WinXPComputer tcp port 139 open
add WinXPComputer tcp port 137 open
add WinXPComputer udp port 137 open
add WinXPComputer udp port 135 open
add WinXPComputer tcp port 80 "sh scripts/win32/web.sh"
set WinXPComputer uptime 5981234
bind 192.168.1.25 WinXPComputer
```

6. Tratamiento de los datos

La centralización de la información puede permitir observar en tiempo real la actividad de un atacante. Esto es importante porque se estima que el daño producido por un atacante durante 30 minutos puede necesitar entre 30 y 40 horas entender qué es lo que ha pasado.

Los tres módulos que utiliza el sistema honeypot, iptables, snort y honeyd, generan sus respectivos logs en los siguientes ficheros:

```
/var/log/iptables.log
/var/log/snort/alert
/var/log/honeyd.log
```

Cada una uno de ellos posee una estructura de diferentes campos de texto en un orden específico. Si bien existe cierta estandarización a nivel de algunos tipos de aplicaciones como servidores web (Common Log Format) lo cierto es que no hay un consenso común para cualquier tipo de aplicaciones.

En iptables el formato es el utilizado en rsyslog, Snort genera un formato particular y por último Honeyd genera un formato más escueto.

Lo importante para nosotros será identificar especialmente el origen de la conexión en los diferentes logs. Veamos cómo se localiza dicha información, origen, destino y puerto, en estos logs:

```
==> /var/log/iptables.log <==
May 30 21:27:47 localhost kernel: IPTABLES-DROP: IN=ens33 OUT=
MAC=ff:ff:ff:ff:ff:ff:dc:85:de:01:8d:8e:08:00   SRC=192.168.1.208
DST=192.168.1.255  LEN=78  TOS=0x00  PREC=0x00  TTL=128  ID=4368
PROTO=UDP SPT=137 DPT=137 LEN=58

==> /var/log/snort/alert <==
[**] [1:100001002:1] NF - SCAN PING NMAP [**]
[Classification: Detection of a Network Scan] [Priority: 3]
05/30-21:28:01.726698 192.168.1.208 -> 192.168.1.23
ICMP TTL:52 TOS:0x0 ID:27813 IpLen:20 DgmLen:28
Type:8  Code:0  ID:44360   Seq:0  ECHO
[Xref => http://networkforensic.dk]

==> /var/log/honeyd.log <==
2019-05-31-06:41:38.9273   tcp(6)   S   134.209.219.144   39604
192.168.1.22 23
2019-05-31-06:42:38.9312   tcp(6)   E   134.209.219.144   39604
192.168.1.22 23: 0 0
```

Una vez identificada la dirección IP origen de la conexión será, sencillo su geolocalización física.

Además de esto, debemos integrar la información que proporcionan estos ficheros para una visualización sencilla en la que se pueda identificar el tipo de conexión o ataque y el origen del mismo.

Para poder ver en tiempo real los logs integrados generados por nuestros tres archivos de logs podemos hacer:

```
# tail -f /var/log/iptables.log /var/log/snort/alert
/var/log/honeyd.log
```

Este último comando nos permitirá hacer un seguimiento en tiempo real de los tres ficheros al mismo tiempo.

El siguiente paso en el tratamiento de los log generados, es procesarlos mediante alguna aplicación que permita una visualización gráfica más amigable y sencilla. Este trabajo de ampliación se inicia en el anexo 2 mediante la instalación de ELK (Elascticsearch, Logstash, Kibana).

7. Pruebas del sistema

Para probar nuestro sistema Honeypot seguiremos dos estrategias. Por un lado simularemos algunos ataques de sondeo y estableceremos algunas conexiones desde un equipo externo observando el registro generado y verificando la detección del honeypot.

Por otro lado expondremos nuestro Honeypot directamente a Internet con puertos abiertos durante un cierto periodo de tiempo y observaremos las detecciones que se consiguen. Para ello abriremos todos los puertos del simulador de PLC Siemens desde exterior y también el puerto modbus y telnet del PLC Schneider.

7.1 Pruebas locales

En primer lugar realizamos una prueba de sondeo de puertos sobre la red ICS desde un equipo que accede al mismo segmento de red. Para ello utilizaremos una distribución de Kali Linux.

7.1.1 Prueba 1. Sondeo con nmap

Realizamos escaneo en busca de puertos abiertos en la red industrial simulada. Escogemos sólo el rango de direcciones válidas para minimizar las salidas innecesarias.

```
root@kali:~# nmap -v -O  192.168.1.20-25
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-30 14:03 EDT
Initiating Ping Scan at 14:03
Scanning 6 hosts [4 ports/host]
Completed Ping Scan at 14:03, 0.10s elapsed (6 total hosts)
Initiating Parallel DNS resolution of 6 hosts. at 14:03
Completed Parallel DNS resolution of 6 hosts. at 14:03, 0.06s elapsed
Initiating SYN Stealth Scan at 14:03
Scanning 6 hosts [1000 ports/host]
Discovered open port 21/tcp on 192.168.1.22
Discovered open port 21/tcp on 192.168.1.23
Discovered open port 80/tcp on 192.168.1.23
Discovered open port 80/tcp on 192.168.1.22
Discovered open port 139/tcp on 192.168.1.25
Discovered open port 80/tcp on 192.168.1.25
Discovered open port 23/tcp on 192.168.1.22
Discovered open port 23/tcp on 192.168.1.20
Discovered open port 20/tcp on 192.168.1.23
Discovered open port 20/tcp on 192.168.1.22
Initiating OS detection (try #1) against 6 hosts
Retrying OS detection (try #2) against 4 hosts
Nmap scan report for 192.168.1.20
Host is up (0.11s latency).
Not shown: 998 closed ports
PORT    STATE    SERVICE
23/tcp  open     telnet
514/tcp filtered shell
```

```
Aggressive OS guesses: Actiontec MI424WR-GEN3I WAP (99%), DD-WRT v24-sp2
(Linux 2.4.37) (97%), Linux 4.4 (97%), Microsoft Windows XP SP3 or
Windows 7 or Windows Server 2012 (96%), Linux 3.2 (96%), Microsoft
Windows XP SP3 (96%), BlueArc Titan 2100 NAS device (91%)
No exact OS matches for host (test conditions non-ideal).
TCP Sequence Prediction: Difficulty=252 (Good luck!)
IP ID Sequence Generation: Incrementing by 2

Nmap scan report for 192.168.1.21
Host is up (0.00033s latency).
Not shown: 991 filtered ports
PORT      STATE  SERVICE
981/tcp   closed unknown
1034/tcp  closed zincite-a
1138/tcp  closed encrypted_admin
1687/tcp  closed nsjtp-ctrl
2043/tcp  closed isis-bcast
5903/tcp  closed vnc-3
6100/tcp  closed synchronet-db
10003/tcp closed documentum_s
14441/tcp closed unknown
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: specialized|general purpose|printer
Running (JUST GUESSING): Lancom LCOS 8.X (90%), Linux 2.6.X (87%), IBM
embedded (86%)
OS    CPE:    cpe:/o:lancom:lcos:8.00    cpe:/o:linux:linux_kernel:2.6.38
cpe:/h:ibm:infoprint_1754
Aggressive OS guesses: Lancom LCOS 8.00 (90%), Linux 2.6.38 (87%), IBM
InfoPrint 1754 printer (86%)
No exact OS matches for host (test conditions non-ideal).

Nmap scan report for 192.168.1.22
Host is up (0.11s latency).
Not shown: 985 closed ports
PORT      STATE    SERVICE
20/tcp    open     ftp-data
21/tcp    open     ftp
23/tcp    open     telnet
30/tcp    filtered unknown
80/tcp    open     http
125/tcp   filtered locus-map
514/tcp   filtered shell
1100/tcp  filtered mctp
2100/tcp  filtered amiganetfs
2401/tcp  filtered cvspserver
5405/tcp  filtered pcduo
9010/tcp  filtered sdr
9500/tcp  filtered ismserver
44176/tcp filtered unknown
49157/tcp filtered unknown
Aggressive OS guesses: Actiontec MI424WR-GEN3I WAP (99%), DD-WRT v24-sp2
(Linux 2.4.37) (98%), Linux 3.2 (98%), Microsoft Windows XP SP3 or
Windows 7 or Windows Server 2012 (96%), Linux 4.4 (96%), Microsoft
Windows XP SP3 (96%), BlueArc Titan 2100 NAS device (91%)
No exact OS matches for host (test conditions non-ideal).
TCP Sequence Prediction: Difficulty=260 (Good luck!)
IP ID Sequence Generation: Incrementing by 2

Nmap scan report for 192.168.1.23
Host is up (0.30s latency).
Not shown: 984 closed ports
PORT      STATE    SERVICE
19/tcp    filtered chargen
20/tcp    open     ftp-data
21/tcp    open     ftp
80/tcp    open     http
425/tcp   filtered icad-el
```

```
514/tcp   filtered shell
1001/tcp  filtered webpush
1007/tcp  filtered unknown
1984/tcp  filtered bigbrother
2638/tcp  filtered sybase
5000/tcp  filtered upnp
5414/tcp  filtered statusd
6580/tcp  filtered parsec-master
6666/tcp  filtered irc
20005/tcp filtered btx
56738/tcp filtered unknown
Device type: general purpose
Running: Microsoft Windows XP|7|2012
OS   CPE:   cpe:/o:microsoft:windows_xp::sp3   cpe:/o:microsoft:windows_7
cpe:/o:microsoft:windows_server_2012
OS details: Microsoft Windows XP SP3, Microsoft Windows XP SP3 or
Windows 7 or Windows Server 2012
TCP Sequence Prediction: Difficulty=254 (Good luck!)
IP ID Sequence Generation: Incremental

Nmap scan report for 192.168.1.24
Host is up (0.00030s latency).
Not shown: 992 filtered ports
PORT      STATE  SERVICE
981/tcp   closed unknown
1034/tcp  closed zincite-a
1138/tcp  closed encrypted_admin
1687/tcp  closed nsjtp-ctrl
2043/tcp  closed isis-bcast
5903/tcp  closed vnc-3
6100/tcp  closed synchronet-db
10003/tcp closed documentum_s
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: specialized|general purpose|printer
Running (JUST GUESSING): Lancom LCOS 8.X (90%), Linux 2.6.X (87%), IBM
embedded (86%)
OS    CPE:    cpe:/o:lancom:lcos:8.00    cpe:/o:linux:linux_kernel:2.6.38
cpe:/h:ibm:infoprint_1754
Aggressive OS guesses: Lancom LCOS 8.00 (90%), Linux 2.6.38 (87%), IBM
InfoPrint 1754 printer (86%)
No exact OS matches for host (test conditions non-ideal).

Nmap scan report for 192.168.1.25
Host is up (0.31s latency).
Not shown: 997 closed ports
PORT    STATE    SERVICE
80/tcp  open     http
139/tcp open     netbios-ssn
514/tcp filtered shell
Device type: general purpose
Running: Microsoft Windows XP|7|2012
OS   CPE:   cpe:/o:microsoft:windows_xp::sp3   cpe:/o:microsoft:windows_7
cpe:/o:microsoft:windows_server_2012
OS details: Microsoft Windows XP SP3, Microsoft Windows XP SP3 or
Windows 7 or Windows Server 2012
TCP Sequence Prediction: Difficulty=251 (Good luck!)
IP ID Sequence Generation: Incrementing by 2

Read data files from: /usr/bin/../share/nmap
OS detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 6 IP addresses (6 hosts up) scanned in 343.34 seconds
           Raw packets sent: 8942 (411.324KB) | Rcvd: 8456 (339.936KB)
root@kali:~#
```

Observamos que la detección de puertos es correcta menos los puertos de los PLCs que simulan MODBUS y S7. Estos dos puertos no se detectan como abiertos aunque lo están, por lo que esta inconsistencia podemos atribuirlo a un defecto de nmap. También vemos que la detección de los diversos sistemas operativos no es concluyente, aunque sí se asignan posibles firmas de algunos dispositivos IOT. Esto es algo bastante habitual debido a cierta ambigüedad de muchos fingerprints.

Veamos ahora los registros generados en nuestro honeypot. Para esta labor hemos ejecutado en paralelo el siguiente comando que nos centraliza la visualización de los diferentes ficheros en tiempo real y genera un solo fichero de logs.

```
# tail -f /var/log/iptables.log /var/log/snort/alert
/var/log/honeyd.log |tee test_nmap01.log
```

El log generado durante esta prueba de nmap alcanza un tamaño de más de 15.000 líneas, habiendo durado el escaneo unos siete minutos. Presentamos a continuación sólo algunos resultados extraídos del log de nuestro honeypot.

Lo primero que vemos es la detección de diversos escaneos de red tipo ping por parte de Snort:

```
[**] [1:100001002:1] NF - SCAN PING NMAP [**]
[Classification: Detection of a Network Scan] [Priority: 3]
05/30-21:28:01.728419 192.168.1.208 -> 192.168.1.20
ICMP TTL:39 TOS:0x0 ID:28096 IpLen:20 DgmLen:28
Type:8  Code:0  ID:25434   Seq:0  ECHO
[Xref => http://networkforensic.dk]
```

Honeyd detecta intentos de conexión recurrentes en la detección de puertos abiertos:

```
==> /var/log/honeyd.log <==
2019-05-30-21:28:02.3968 tcp(6) - 192.168.1.208 11629 192.168.1.22 22:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.3973 tcp(6) - 192.168.1.208 11630 192.168.1.23 22:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.3982 tcp(6) - 192.168.1.208 11632 192.168.1.25 22:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.3984 tcp(6) - 192.168.1.208 11633 192.168.1.20 22:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.3986 tcp(6) - 192.168.1.208 11636 192.168.1.23 25:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.3995 tcp(6) - 192.168.1.208 11635 192.168.1.22 25:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.4962 tcp(6) - 192.168.1.208 11638 192.168.1.22 139:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.7367 tcp(6) - 192.168.1.208 11627 192.168.1.25 443:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.8982 tcp(6) - 192.168.1.208 11629 192.168.1.22 22:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.8993 tcp(6) - 192.168.1.208 11633 192.168.1.20 22:
52 S [Windows 2000 RFC1323]
2019-05-30-21:28:02.8999 tcp(6) - 192.168.1.208 11630 192.168.1.23 22:
52 S [Windows 2000 RFC1323]
```

```
2019-05-30-21:28:02.9003 tcp(6) - 192.168.1.208 11632 192.168.1.25 22:
52 S [Windows 2000 RFC1323]
```

Al mismo tiempo Snort detecta intentos de apertura HTTP sospechosos y Honeyd también lo registra:

```
==> /var/log/honeyd.log <==
2019-05-30-21:28:04.3511    tcp(6)    E    192.168.1.208    11795
192.168.1.25 80: 0 371

==> /var/log/snort/alert <==
[**] [1:100000:1] HTTP Web Viewing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/30-21:28:04.351084 192.168.1.208:11795 -> 192.168.1.25:80
TCP TTL:128 TOS:0x0 ID:27044 IpLen:20 DgmLen:40 DF
***A*R** Seq: 0xB005D264  Ack: 0xEF513FD1  Win: 0x0  TcpLen: 20
```

Otros tipos de detecciones sospechosas por parte de Snort:

```
==> /var/log/snort/alert <==
[**] [1:100001003:1] NF - SCAN Potential VNC Scan 5800-5820 [**]
[Classification: Detection of a Network Scan] [Priority: 3]
05/30-21:28:04.748770 192.168.1.208:11860 -> 192.168.1.22:5810
TCP TTL:128 TOS:0x0 ID:4249 IpLen:20 DgmLen:52 DF
******S* Seq: 0x83DFF276  Ack: 0x0  Win: 0xFAF0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK
[Xref => http://networkforensic.dk]
```

Detección de escaneo basado en SSH:

```
==> /var/log/snort/alert <==
[**] [1:100001008:1] NF - SCAN Potential SSH Scan [**]
[Classification: Detection of a Network Scan] [Priority: 3]
05/30-21:30:29.620539 192.168.1.208:15912 -> 192.168.1.22:22
TCP TTL:128 TOS:0x0 ID:5438 IpLen:20 DgmLen:52 DF
******S* Seq: 0x5D5A8702  Ack: 0x0  Win: 0xFAF0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK
[Xref => http://networkforensic.dk]
```

Otros tipos de detecciones sospechosas:

```
==> /var/log/snort/alert <==
[**] [1:100012001:1] NF - DHCP Request - SCADA networks do not
use DHCP [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/30-21:31:08.855340 0.0.0.0:68 -> 255.255.255.255:67
UDP TTL:64 TOS:0x10 ID:0 IpLen:20 DgmLen:334 DF
Len: 306
[Xref => http://networkforensic.dk]

==> /var/log/snort/alert <==
[**] [1:100011001:1] NF - ICMP Payload to big for normal use -
Covert Channel [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/30-21:33:40.398126 192.168.1.208 -> 192.168.1.20
ICMP TTL:40 TOS:0x0 ID:31349 IpLen:20 DgmLen:148
Type:8  Code:9  ID:58131   Seq:295  ECHO
[Xref => http://networkforensic.dk]
[**] [1:100001001:1] NF - ICMP Echo Reply undefined code [**]
```

```
[Classification: Detection of a Network Scan] [Priority: 3]
05/30-21:33:40.427064 192.168.1.23 -> 192.168.1.208
ICMP TTL:64 TOS:0x0 ID:51452 IpLen:20 DgmLen:178
Type:0  Code:0  ID:58132  Seq:296  ECHO REPLY
[Xref => http://networkforensic.dk]

[**] [1:100001005:1] NF - SCAN NMAP OS Detection Probe [**]
[Classification: Detection of a Network Scan] [Priority: 3]
05/30-21:33:40.450037 192.168.1.208:59920 -> 192.168.1.20:39738
UDP TTL:53 TOS:0x0 ID:31351 IpLen:20 DgmLen:328
Len: 300
[Xref => http://networkforensic.dk]
```

7.1.2 Prueba 2. Apertura de conexiones

Verificamos la apertura de los servicios simulados y observamos cómo lo registra nuestro honeypot.

1. Apertura de conexiones HTTP

Desde el host real (192.168.1.108) abrimos la página del PLC Siemens simulado.

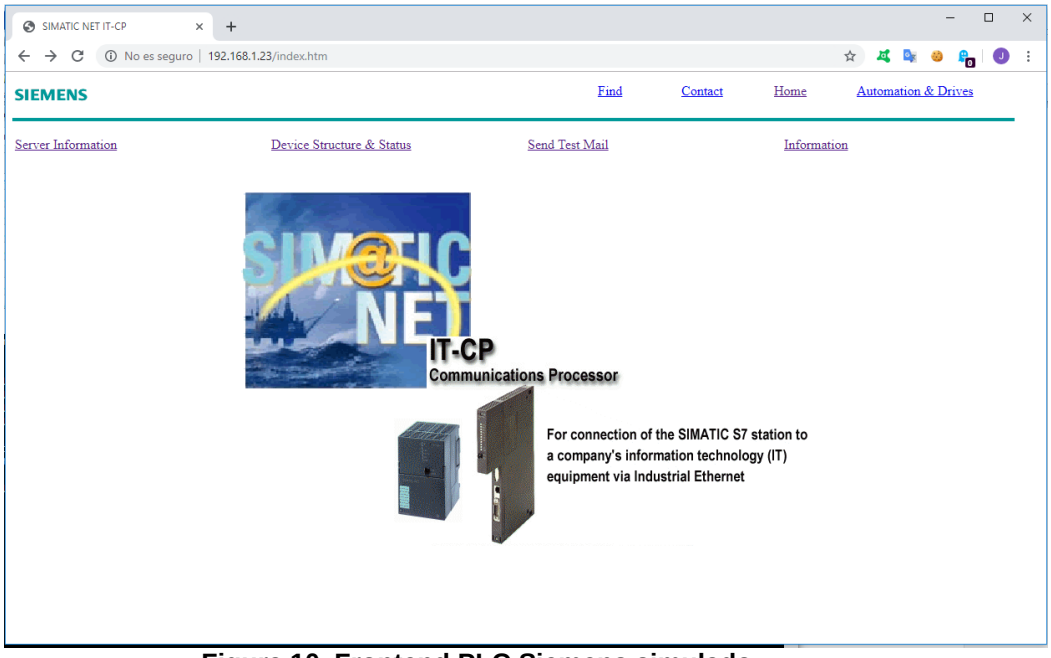


Figura 10. Frontend PLC Siemens simulado

Del mismo modo abrimos la página del PLC Schneider:

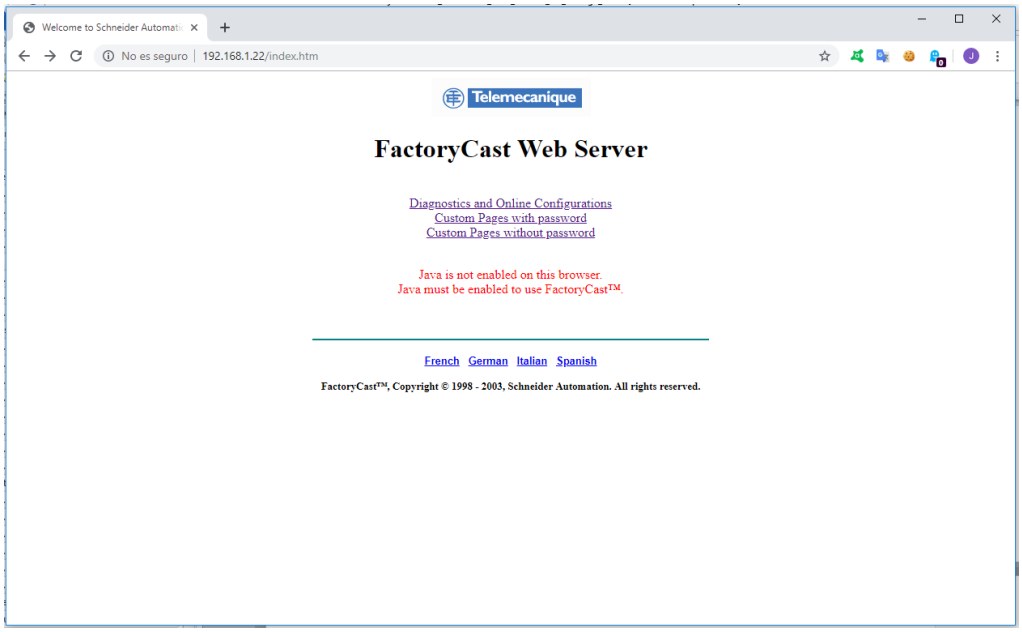


Figura 11. Frontend PLC Schneider simulado

Simplemente hemos abierto desde otro equipo en la red sus dos páginas principales en el navegador (cierta navegación interior está también diseñada). Los logs generados incluyen el siguiente tipo de registros:

```
==> /var/log/snort/alert <==

[**] [1:100000:1] HTTP Web Viewing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/01-16:07:18.438217 192.168.1.208:22579 -> 192.168.1.23:80
TCP TTL:128 TOS:0x0 ID:32331 IpLen:20 DgmLen:52 DF
******S* Seq: 0x12ACCA81  Ack: 0x0  Win: 0xFAF0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK

[**] [1:100000:1] HTTP Web Viewing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/01-16:07:21.437669 192.168.1.208:22578 -> 192.168.1.23:80
TCP TTL:128 TOS:0x0 ID:32338 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xE18BE5F6  Ack: 0xD866D207  Win: 0xFF70  TcpLen:
20

==> /var/log/honeyd.log <==
2019-06-01-16:07:18.8992    tcp(6)    E    192.168.1.208    22579
192.168.1.23 80: 440 221

==> /var/log/honeyd.log <==
2019-06-01-16:07:25.1664    tcp(6)    E    192.168.1.208    22578
192.168.1.23 80: 491 820

==> /var/log/honeyd.log <==
2019-06-01-16:07:25.5580    tcp(6)    S    192.168.1.208    22586
192.168.1.23 80 [Windows 2000 RFC1323]

[**] [1:100000:1] HTTP Web Viewing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/01-16:09:18.269913 192.168.1.208:22605 -> 192.168.1.22:80
TCP TTL:128 TOS:0x0 ID:3945 IpLen:20 DgmLen:40 DF
```

```
***A**** Seq: 0xA2B2A5D4  Ack: 0x5AC14FAC  Win: 0xFF70  TcpLen:
20

==> /var/log/honeyd.log <==
2019-06-01-16:09:18.2702    tcp(6)    S    192.168.1.208    22606
192.168.1.22 80 [Windows 2000 RFC1323]

2019-06-01-16:09:18.5875    tcp(6)    E    192.168.1.208    22605
192.168.1.22 80: 320 2414
```

2. Apertura de conexiones telnet y FTP

Realizamos varios intentos de conexión telnet y ftp contra los dos PLCs y observamos el registro.

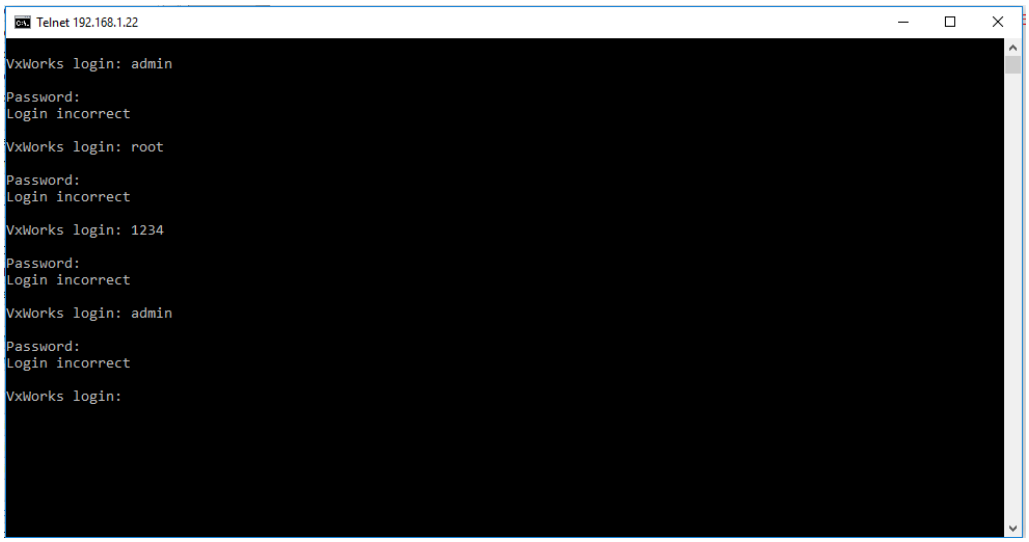


Figura 12. Conexión telnet al PLC Schneider simulado

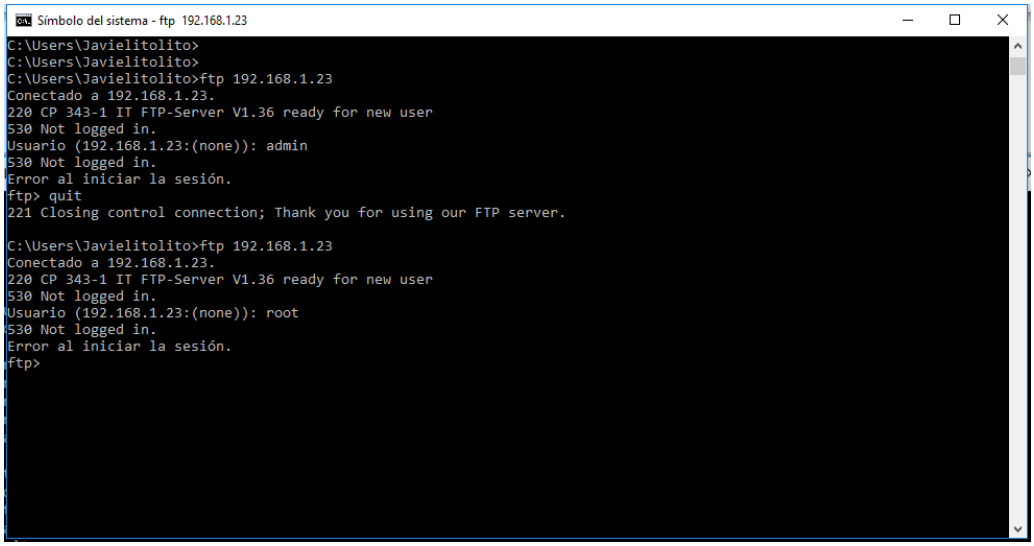


Figura 13. Conexión FTP al PLC Siemens simulado

```
==> /var/log/honeyd.log <==
2019-06-01-16:51:14.5734    tcp(6)    S    192.168.1.208    22911
192.168.1.22 23 [Windows 2000 RFC1323]

[**] [1:100007005:1] NF - TELNET - Bad Login [**]
[Classification: An Attempted Login Using a Suspicious Username
was Detected] [Priority: 2]
06/01-16:51:25.457744 192.168.1.22:23 -> 192.168.1.208:22911
TCP TTL:64 TOS:0x0 ID:11567 IpLen:20 DgmLen:76
***A**** Seq: 0x67447172  Ack: 0xB009E469  Win: 0x3E80  TcpLen:
20
[Xref => http://networkforensic.dk]

==> /var/log/honeyd.log <==
2019-06-01-16:52:48.0820    tcp(6)    -    192.168.1.208    22918
192.168.1.23 23: 52 S [Windows 2000 RFC1323]
2019-06-01-16:52:48.5830    tcp(6)    -    192.168.1.208    22918
192.168.1.23 23: 52 S [Windows 2000 RFC1323]
2019-06-01-16:52:49.0847    tcp(6)    -    192.168.1.208    22918
192.168.1.23 23: 52 S [Windows 2000 RFC1323]
2019-06-01-16:53:05.3388    tcp(6)    S    192.168.1.208    22923
192.168.1.23 21 [Windows 2000 RFC1323]

[**] [1:110000003:1] NF - FTP - 530 Not logged in [**]
[Classification: An Attempted Login Using a Suspicious Username
was Detected] [Priority: 2]
06/01-16:53:05.502792 192.168.1.23:21 -> 192.168.1.208:22923
TCP TTL:64 TOS:0x0 ID:679 IpLen:20 DgmLen:60
***A**** Seq: 0x8A0504D8  Ack: 0x70634F05  Win: 0x3E80  TcpLen:
20
[Xref => http://networkforensic.dk]
```

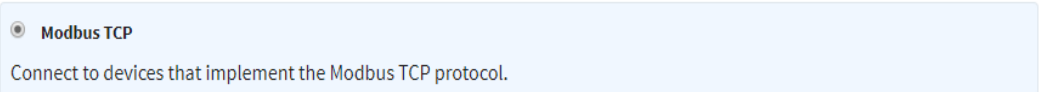
3. Acceso a los PLCs.

Debemos verificar el funcionamiento de la simulación de PLCs con MODBUS/TCP y S7. Para ello podemos utilizar varias alternativas de software como Simply Modbus, Modbus Poll, algún paquete de SIMATIC, TIA portal, Ignition o algún otro.

En nuestro caso vamos a utilizar una plataforma Ignition de Inductive Automation para conectarlo a los PLCs simulados y verificar cómo se registran estas conexiones. Ignition es descargable gratuitamente y su licencia trial es completamente funcional. Es una herramienta sencilla y polivalente para nuestra prueba.

Configuramos Ignition previo a efectuar las conexiones a los PLCs y registrarlas. Para ello nos logamos en el Ignition Gateway y configuramos dos dispositivos en la sección OPC UA Device Connections.

El primero será un dispositivo Modbus TCP:



con las siguientes características de conexión:

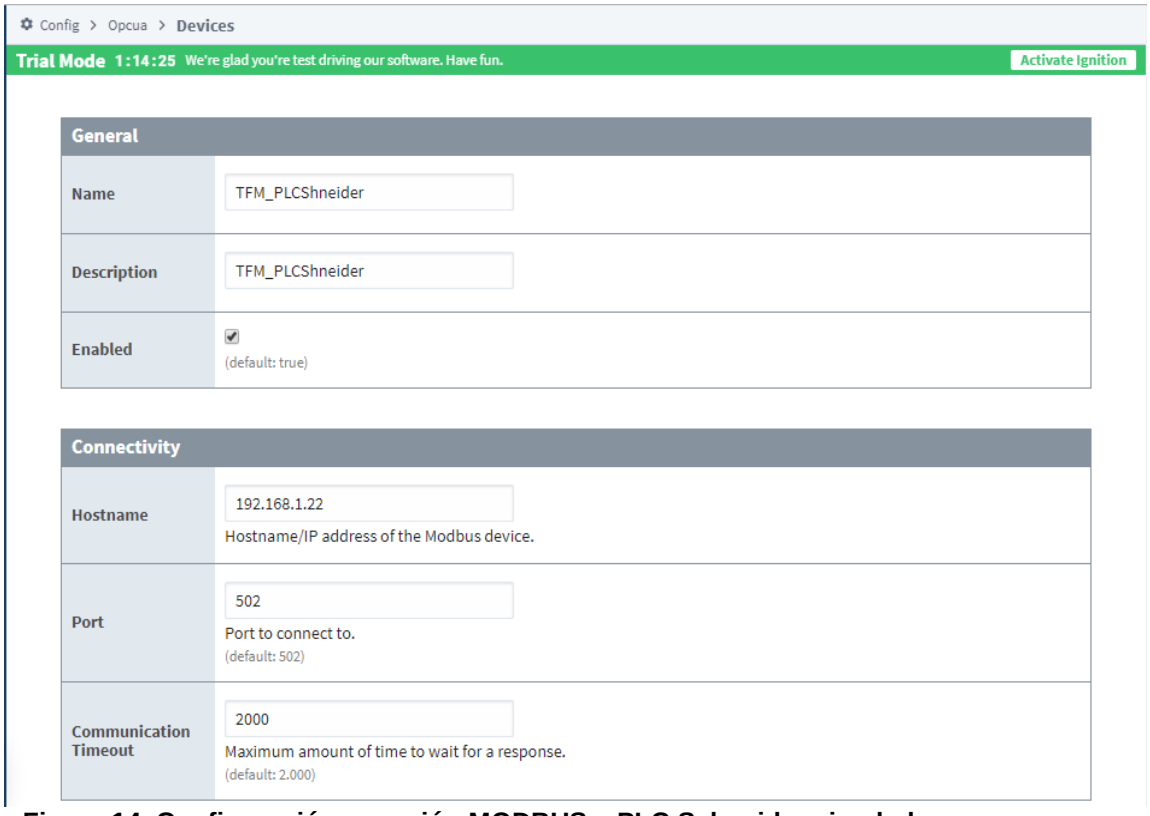
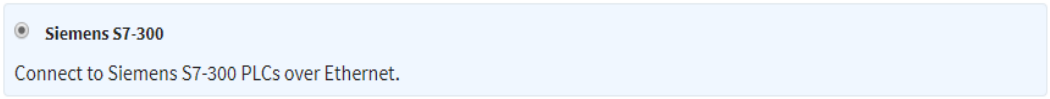


Figura 14. Configuración conexión MODBUS a PLC Schneider simulado

El segundo constituye la conexión al PLC Siemens como PLC de la serie S7-300:



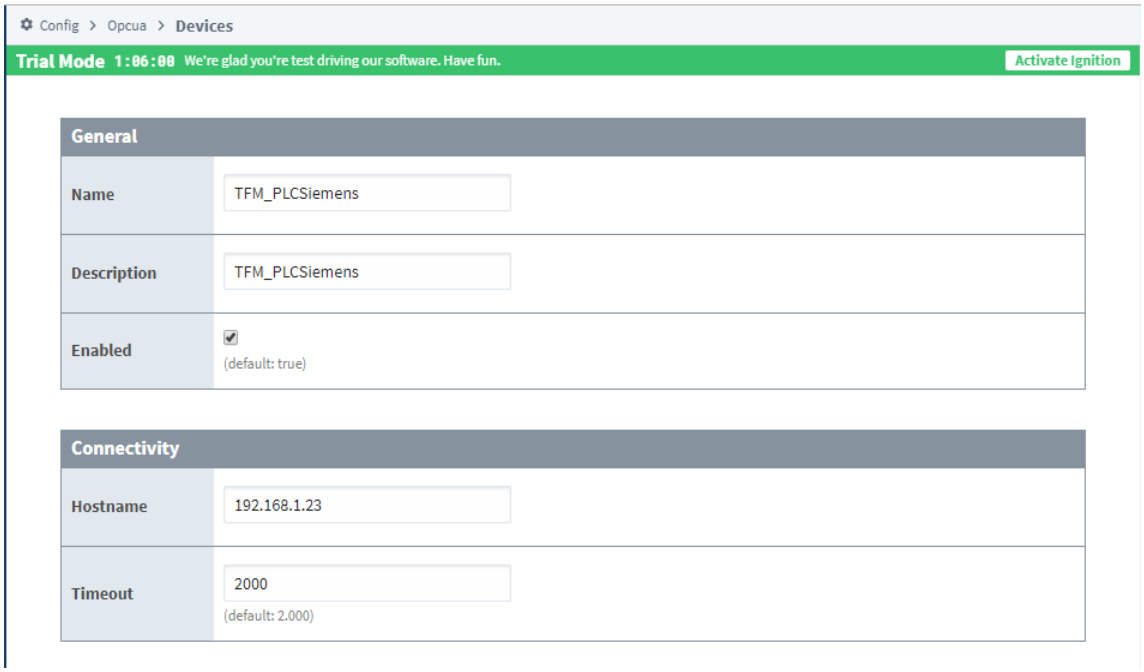


Figura 15. Configuración conexión S7 a PLC Siemens simulado

El aspecto de las conexiones a estos PLCs es el siguiente:

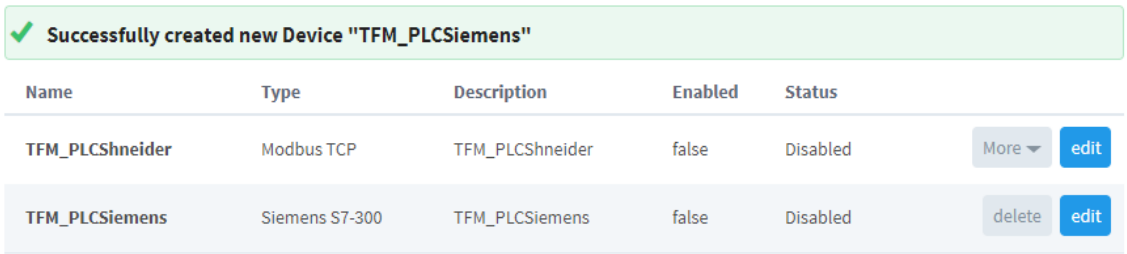


Figura 16. Estados previo a conexiones a PLCs

Ahora iniciamos el simulador y habilitamos estos dos dispositivos en Ignition verificando su conexión a la plataforma:

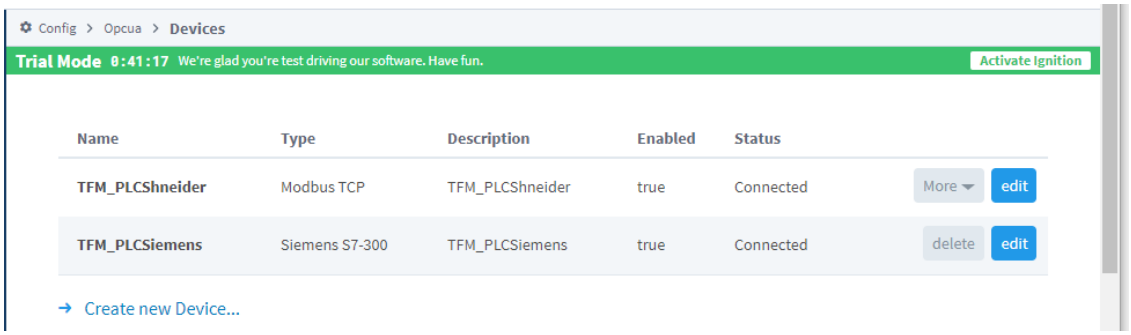


Figura 17. Verificación estados tras conexión a PLCs simulados

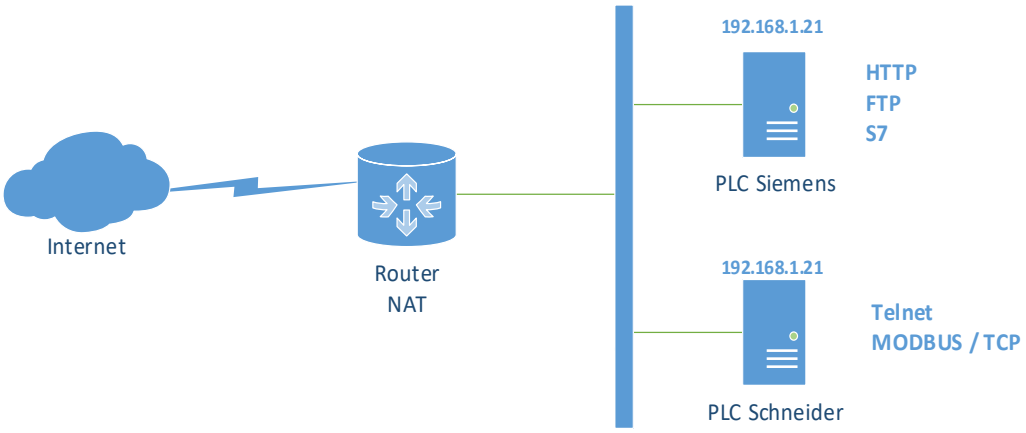
Por último verificamos que estas conexiones generan los siguientes registros en los logs del honeypot:

```
==> /var/log/honeyd.log <==
2019-06-01-19:22:52.8119 honeyd log started ------
2019-06-01-19:23:23.6161    tcp(6)    S    192.168.1.208    24095
192.168.1.22 502 [Windows 2000 RFC1323]

2019-06-01-19:23:34.6157    tcp(6)    S    192.168.1.208    24103
192.168.1.23 102 [Windows 2000 RFC1323]

[**] [1:120001001:1] NF - SCADA S7 - Connection established [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/01-19:23:35.523447 192.168.1.208:24103 -> 192.168.1.23:102
TCP TTL:128 TOS:0x0 ID:32409 IpLen:20 DgmLen:65 DF
***AP*** Seq: 0x6AF0862A  Ack: 0xF594EAC2  Win: 0xFF5A  TcpLen:
20
[Xref => http://networkforensic.dk]
```

7.2 Exposición a Internet

Como colofón a estos resultados, decidimos exponer los PLCs simulados directamente a Internet y registrar aquellas conexiones que se pudieran presentar. Para ello configuramos en un router de acceso a Internet el mapeo de puertos correspondientes a los servicios indicados.

El esquema de conexión que utilizamos es el siguiente:



Antes de hacer la prueba, vaciamos los archivos de log para partir de un escenario limpio de conexiones anteriores. Con ello generamos tres archivos de log que aglutinamos en uno solo mediante la instrucción ya vista:

```
# tail -f /var/log/iptables.log /var/log/snort/alert
/var/log/honeyd.log |tee test_Inet01.log
```

Esta instrucción se ejecuta desde el comienzo de la prueba y permite visualizar al mismo tiempo los tres ficheros log y grabarlos en uno solo.

Los logs generados podrán ser procesados posteriormente con alguna aplicación automatizada. Ello permitirá generar presentaciones más amigables, gráficos, etc. con utilidades como ELK.

Realizamos la exposición a Internet durante algo más de diez horas.

Analizando el log generado, desechando el contenido sin interés e identificamos las direcciones IP origen de las conexiones que se van registrando, así como el servicio al que se conectan. Tabulamos estas entradas en una tabla resumen y realizamos algunos conteos significativos. Además de esto, utilizamos algunos servicios de geolocalización disponibles en Internet (por ejemplo https://www.iplocation.net/) e identificamos el país origen de la conexión.

Obtenemos algunos interesantes resultados:

- No pasan ni 5 minutos desde que se expone el sistema a Internet y se empiezan a recibir intentos de conexión a los servicios abiertos.
- En total se registran 123 conexiones en algo más de 10 horas.
- De estas conexiones, 100 de ellas son desde IP diferentes.
- En total se identifican 29 países desde donde se realiza el acceso.
- Nuestro resultado confirma el hecho general de que la inmensa mayoría de intentos de penetración se efectúan desde Estados Unidos y China.
- A estos dos países le siguen Taiwán, Brasil e India en menor medida.
- De las 123 conexiones sólo una corresponde a un protocolo industrial. Se trata de MODBUS y se realiza desde Lituania.
- La inmensa mayorías (más del 80%) de conexiones se realizan hacia el servicio telnet. Se puede deducir aquí el gran interés existente por el acceso a un interfaz de comandos en el equipo.

Presentamos a continuación a modo de muestra algunos de los registros obtenidos. Así por ejemplo, Snort muestra algunas advertencias respecto a ciertas peticiones web:

```
==> /var/log/snort/alert <==
[**] [1:100000:1] HTTP Web Viewing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/31-06:38:47.209674 93.159.184.160:476 -> 192.168.1.23:80
TCP TTL:243 TOS:0x0 ID:8278 IpLen:20 DgmLen:40 DF
******S* Seq: 0x4E422A38  Ack: 0x0  Win: 0x3908  TcpLen: 20

==> /var/log/snort/alert <==
[**] [1:100000:1] HTTP Web Viewing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/31-06:49:19.897177 172.245.90.241:41323 -> 192.168.1.23:80
TCP TTL:239 TOS:0x0 ID:54321 IpLen:20 DgmLen:40
******S* Seq: 0xE051BCF3  Ack: 0x0  Win: 0xFFFF  TcpLen: 20
```

```
==> /var/log/snort/alert <==
[**] [1:100000:1] HTTP Web Viewing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/31-06:58:07.346118 112.125.92.74:1734 -> 192.168.1.23:80
TCP TTL:42 TOS:0x0 ID:11910 IpLen:20 DgmLen:64 DF
******S* Seq: 0x76778DA9  Ack: 0x0  Win: 0xFFFF  TcpLen: 44
TCP Options (9) => MSS: 1460 NOP WS: 0 NOP NOP TS: 0 0 NOP NOP
SackOK
```

El registro de los accesos de Honeyd a los servicios simulados presenta en siguiente patrón:

```
==> /var/log/honeyd.log <==
2019-05-31-09:14:32.1063    tcp(6)    S    217.216.139.56    17879
192.168.1.22 23

2019-05-31-09:15:32.1097    tcp(6)    E    217.216.139.56    17879
192.168.1.22 23: 0 0

019-05-31-09:52:22.7125    tcp(6)    S    5.188.210.139    44783
192.168.1.23 80


2019-05-31-09:53:22.7203    tcp(6)    E    5.188.210.139    44783
192.168.1.23 80: 0 0

2019-05-31-07:17:45.7128    tcp(6)    S    125.62.196.93    38984
192.168.1.22 23

2019-05-31-07:18:45.7277    tcp(6)    E    125.62.196.93    38984
192.168.1.22 23: 0 0

2019-05-31-16:04:43.2933    tcp(6)    S    184.105.247.203    41207
192.168.1.23 21

2019-05-31-16:05:43.3058    tcp(6)    E    184.105.247.203    41207
192.168.1.23 21: 0 0

019-05-31-08:54:24.3304    tcp(6)    S    185.254.122.35    55537
192.168.1.22 502

2019-05-31-08:55:24.3388    tcp(6)    E    185.254.122.35    55537
192.168.1.22 502: 0 0
```

Por último presentamos la tabla resultado una vez organizado por países de origen y el resumen de resultados:

IP Origen	Localización origen	IP PLC:puerto	Servicio
177.154.226.154	Brazil	192.168.1.22 23	Telnet
201.42.73.178	Brazil	192.168.1.23 80	HTTP
201.68.246.82	Brazil	192.168.1.22 23	Telnet
177.129.157.130	Brazil	192.168.1.22 23	Telnet
143.255.0.225	Brazil	192.168.1.22 23	Telnet
189.69.201.121	Brazil	192.168.1.22 23	Telnet

201.13.102.119	Brazil	192.168.1.22 23	Telnet
41.216.147.53	Burkina Faso	192.168.1.22 23	Telnet
144.217.156.104	Canada	192.168.1.22:23	Telnet
159.203.6.102	Canada	192.168.1.22 23	Telnet
54.39.149.21	Canada	192.168.1.22 23	Telnet
113.116.17.95	China	192.168.1.22 23	Telnet
122.226.184.226	China	192.168.1.22 23	Telnet
110.249.212.46	China	192.168.1.23 80	HTTP
112.125.92.74	China	192.168.1.23:80	HTTP
116.8.112.27	China	192.168.1.22 23	Telnet
118.113.17.7	China	192.168.1.22 23	Telnet
124.133.28.82	China	192.168.1.23 80	HTTP
125.69.67.24	China	192.168.1.22 23	Telnet
223.97.187.73	China	192.168.1.22:23	Telnet
42.237.24.44	China	192.168.1.22 23	Telnet
31.31.230.22	Czech Republic	192.168.1.22 23	Telnet
5.135.209.161	France	192.168.1.22 23	Telnet
137.74.154.198	France	192.168.1.22 23	Telnet
151.80.152.189	France	192.168.1.22 23	Telnet
2.2.2.1	France	192.168.1.23 80	HTTP
37.49.225.32	Iceland	192.168.1.22 23	Telnet
37.49.225.32	Iceland	192.168.1.22 23	Telnet
122.165.154.179	India	192.168.1.22:23	Telnet
103.50.7.123	India	192.168.1.23 80	HTTP
104.211.207.95	India	192.168.1.22 23	Telnet
123.63.238.185	India	192.168.1.23 80	HTTP
125.62.196.93	India	192.168.1.22:23	Telnet
68.183.89.236	India	192.168.1.22 23	Telnet
5.200.194.244	Iran	192.168.1.22 23	Telnet
77.42.75.163	Iran	192.168.1.22 23	Telnet
95.38.209.90	Iran	192.168.1.22 23	Telnet
62.0.72.247	Israel	192.168.1.22 23	Telnet
195.223.37.217	Italy	192.168.1.22 23	Telnet
212.131.20.1	Italy	192.168.1.22 23	Telnet
149.200.223.113	Jordan	192.168.1.22 23	Telnet
185.254.122.35	Lithuania	192.168.1.22 502	MODBUS
189.154.192.23	Mexico	192.168.1.22 23	Telnet

187.133.117.117	Mexico	192.168.1.22:23	Telnet
187.155.27.75	Mexico	192.168.1.23 80	HTTP
185.244.25.219	Netherlands	192.168.1.22 23	Telnet
185.244.25.97	Netherlands	192.168.1.22 23	Telnet
185.244.25.97	Netherlands	192.168.1.22 23	Telnet
93.159.184.160	Poland	192.168.1.23:80	HTTP
159.205.157.197	Poland	192.168.1.22 23	Telnet
2.83.241.241	Portugal	192.168.1.22 23	Telnet
86.126.104.197	Romania	192.168.1.22 23	Telnet
5.53.17.125	Russia	192.168.1.22 23	Telnet
5.188.210.139	Russia	192.168.1.23 80	Telnet
81.22.45.100	Russia	192.168.1.23 80	HTTP
81.22.45.228	Russia	192.168.1.22 23	Telnet
128.199.252.238	Singapore	192.168.1.22 23	Telnet
217.216.46.9	Spain	192.168.1.22:23	Telnet
217.216.139.56	Spain	192.168.1.22:23	Telnet
85.53.88.157	Spain	192.168.1.23 80	HTTP
188.150.34.131	Sweden	192.168.1.22 23	Telnet
1.161.123.182	Taiwan	192.168.1.22 23	Telnet
61.219.175.191	Taiwan	192.168.1.22 23	Telnet
115.165.216.112	Taiwan	192.168.1.22 23	Telnet
1.160.63.143	Taiwan	192.168.1.22 23	Telnet
1.172.73.115	Taiwan	192.168.1.22 23	Telnet
111.250.83.166	Taiwan	192.168.1.22 23	Telnet
114.35.156.211	Taiwan	192.168.1.22 23	Telnet
114.44.37.198	Taiwan	192.168.1.22 23	Telnet
196.61.8.122	Tanzania	192.168.1.22 23	Telnet
119.42.121.159	Thailand	192.168.1.23 80	HTTP
85.105.194.228	Turkey	192.168.1.22 23	Telnet
188.119.58.223	Turkey	192.168.1.22 23	Telnet
178.128.39.134	United Kingdom	192.168.1.22 23	Telnet
178.62.126.131	United Kingdom	192.168.1.22 23	Telnet
178.128.160.41	United Kingdom	192.168.1.22 23	Telnet
68.183.103.167	United States	192.168.1.22 23	Telnet
72.176.195.115	United States	192.168.1.22 23	Telnet
134.209.219.144	United States	192.168.1.22:23	Telnet
184.105.247.203	United States	192.168.1.23 21	FTP

104.248.186.55	United States	192.168.1.22 23	Telnet
134.209.43.118	United States	192.168.1.22 23	Telnet
138.197.196.19	United States	192.168.1.22 23	Telnet
157.230.147.34	United States	192.168.1.22 23	Telnet
159.89.181.210	United States	192.168.1.22:23	Telnet
172.245.90.241	United States	192.168.1.23:80	HTTP
178.128.157.15	United States	192.168.1.22 23	Telnet
196.52.43.62	United States	192.168.1.23 80	HTTP
206.189.227.73	United States	192.168.1.22 23	Telnet
206.189.232.13	United States	192.168.1.22 23	Telnet
45.79.106.170	United States	192.168.1.23 21	FTP
50.74.242.22	United States	192.168.1.22 23	Telnet
66.240.205.34	United States	192.168.1.23 80	HTTP
74.82.47.40	United States	192.168.1.23 80	HTTP
74.82.47.44	United States	192.168.1.22 23	Telnet
35.137.139.215	United States	192.168.1.22 23	Telnet
216.243.31.2	United States	192.168.1.23 80	HTTP
1.54.103.102	Viet Nam	192.168.1.22:23	Telnet
1.55.72.234	Viet Nam	192.168.1.22 23	Telnet
113.22.249.160	Viet Nam	192.168.1.22 23	Telnet

Total conexiones	**123**
Total IPs diferentes	**100**
Total paises	**29**
Conexiones Telnet	**80**
Conexiones HTTP	**17**
Conexiones FTP	**2**
Conexiones MODBUS	**1**

8. Conclusiones

Los sistemas industriales con cada vez más vulnerables merced a un aumento significativo en la conectividad de estas redes hasta entonces restringidas.

Se demuestra que la estrategia honeypot es un método efectivo para la detección de intentos de intrusión también en redes industriales. Ha quedado patente que una exposición directa a Internet supone un enorme peligro para cualquier dispositivo y especialmente para componentes IoT, generalmente carentes de protecciones especiales.

Por otra parte los honeypots representan diferentes alternativas de diseño e implementación. Las arquitecturas pueden ser muy complejas, lejos de la idea de un simple equipo de monitoreo conectado a la red.

Debido al interés creciente por la vulneración de los dispositivos IoT y especialmente los elementos de redes industriales, se intuye la necesidad de que los IoT industriales deberían ser certificados en el ámbito de la seguridad [11]

9. Glosario

Botnet	Red de equipos infectados (robots) controlados remotamente por un atacante.
CMS	Central Monitoring Sustem. Sistema de supervisión central
CRM	Customer Relationship Managemente
DMZ	DeMilitarized Zone. Red desminitarizada
FTP	File Transfer Protocol. Protocolo de transferencia de ficheros.
HMI	Human Machine Interface. Interfaz hombre-máquina
Honeypot	Tarro de miel, referido a señuelo
Honeynet	Red de varios honeypots
HTTP	Hypertext Transfer Protocol
ICS	Industrial Control System. Sistema de contrl industrial.
IDS	Intrusion Detection System. Sistema de detección de intrusos.
IoT	Internet of Things
IP	Internet Protocol. Referido a una dirección IP
MES	Manufacturing Execution System
PLC	Programable Logic Controller
SCADA	Supervisory Control and Data Acquisition. Sistema de control y adquisición de datos
SO	Sistema operativo
TCP	Transmission Control Protocol
TFM	Trabajo Final de Máster
UDP	User Datagram Protocol
VM	Virtual Machine. Máquina virtual

10. Bibliografía

[1] Lance Spitzner. Honeypots tracking hackers. Addison-Wesley. ISBN 0-321-10895-7. 2002.

[2] Kevin Ashton, "That 'Internet of Things' Thing", RFID Journal, 22 June 2009.

[3] Ben Herzberg, Dima Bekerman, Igal Zeifman. Breaking Down Mirai: An IoT DDoS Botnet Analysis. Disponible en https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html; accedido el 11/03/2019.

[4] Li Da Xu, Wu He, and Shancang Li. Internet of Things in Industries: A Survey. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 10, NO. 4, NOVEMBER 2014.

[5] Louis Columbus. IoT Market Predicted To Double By 2021, Reaching $520B. Disponible en https://www.forbes.com/sites/louiscolumbus/2018/08/16/iot-market-predicted-to-double-by-2021-reaching-520b/#583034fc1f94. Accedido el 13/03/2019.

[6] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17, NO. 3, THIRD QUARTER 2015.

[7] Elisa Bertino, Nayeem Isla. Botnets and Internet of Things Security. PUBLISHED BY THE IEEE COMPUTER SOCIETY. 2017;50;2;10.1109/MC.2017.62

[8] Weizhi Meng. Intrusion Detection in the Era of IoT: Building Trust via Traffic Filtering and Sampling. PUBLISHED BY THE IEEE COMPUTER SOCIETY. 2018;51;7;10.1109/MC.2018.3011034.

[9] Yair Meidan; Michael Bohadana; Yael Mathov; Yisroel Mirsky; Asaf Shabtai; Dominik Breitenbacher; Yuval Elovici. N-BaIoT--Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE PERVASIVE COMPUTING, VOL. 13, NO. 9, JULY-SEPTEMBER 2018.

[10] Mario FRUSTACI, Pasquale PACE, Gianluca ALOI. Securing the IoT world: issues and perspectives. IEEE 978-1-5386-3070-9. 2017.

[11] Gianmarco Baldini; Antonio Skarmeta; Elizabeta Fourneret; Ricardo Neisse; Bruno Legeard; Franck Le Gall. Security certification and

labelling in Internet of Things. IEEE 3rd World Forum on Internet of Things (WF-IoT);10.1109/WF-IoT.7845514. 2016.

[12] Butti, Laurent; Veysset, Franck. Honeypot Technology: Principles and Applications. Disponible en: https://www.first.org/resources/papers/conference2006/veysset-franck-slides.pdf. Accedido el 25/03/2019.

[13] Hao Li ; Guangjie Liu ; Weiwei Jiang ; Yuewei Dai: Designing snort rules to detect abnormal DNP3 network data. IEEE DOI: 10.1109/ICCAIS.2015.7338690.

[14] A. Acien, A. Nieto, G. Fernandez, and J. Lopez, “A comprehensive methodology for deploying IoT honeypots”, 15th International Conference on
Trust, Privacy and Security in Digital Business (TrustBus 2018) vol. LNCS 11033, pp. 229243, 2018

[15] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia, Xin Ouyang. IoTCandyJar: Towards an Intelligent-Interaction Honeypot for IoT Devices. Disponible en https://www.blackhat.com/docs/us-17/thursday/us-17-Luo-Iotcandyjar-Towards-An-Intelligent-Interaction-Honeypot-For-IoT-Devices-wp.pdf. Accedido el 27/04/2019.

[16] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Christian Rossow. IoTPOT: A novel honeypot for revealing current IoT threats. Journal of Information Processing Vol.24 No.3 522–533 (May 2016)

[17] SCADA HoneyNet Project
(https://sourceforge.net/projects/scadahoneynet/)

11. Anexo 1. Configuración completa de reglas iptables.

A continuación se expone el listado completo de las reglas iptables utilizadas en el honeypot desarrollado en este trabajo.

```
# Eliminar reglas anteriores
iptables --flush
iptables -t nat --flush
iptables --delete-chain
iptables -t nat --delete-chain

# Permitimos conexiones locales
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Configuracón de logs para drops de entradas
iptables -N LOGGING
iptables -A INPUT -j LOGGING
 #iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTABLES-
DROP: " --log-level info
iptables -A LOGGING -j LOG --log-prefix "IPTABLES-DROP: " --log-level info
iptables -A LOGGING -j DROP

# Establecer políticas por defecto
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Permitir conexiondes salientes solo de conexiones esteblecidas previamente
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Permitir conexiones entrantes en puertos escogidos.
#TFP
iptables -A INPUT -p tcp --dport 20 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --dport 21 -m state --state NEW,ESTABLISHED -j ACCEPT
#SSH
iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
#Telnet
iptables -A INPUT -p tcp --dport 23 -m state --state NEW,ESTABLISHED -j ACCEPT
#SMTP
iptables -A INPUT -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
#WHOIS
iptables -A INPUT -p tcp --dport 43 -m state --state NEW,ESTABLISHED -j ACCEPT
#DNS
iptables -A INPUT -p tcp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
#TFTP
iptables -A INPUT -p udp --dport 69 -m state --state NEW,ESTABLISHED -j ACCEPT
#HTTP
iptables -A INPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p udp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
#POP2/3
iptables -A INPUT -p tcp --dport 109 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A INPUT -p tcp --dport 110 -m state --state NEW,ESTABLISHED -j
ACCEPT
#RPC
iptables -A INPUT -p tcp --dport 111 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A INPUT -p udp --dport 111 -m state --state NEW,ESTABLISHED -j
ACCEPT
```

```
#SFTP
iptables -A INPUT -p udp --dport 115 -m state --state NEW,ESTABLISHED -j
ACCEPT
#NNTP
iptables -A INPUT -p tcp --dport 119 -m state --state NEW,ESTABLISHED -j
ACCEPT
#NTP
iptables -A INPUT -p udp --dport 123 -m state --state NEW,ESTABLISHED -j
ACCEPT
#NETBIOS NS
iptables -A INPUT -p tcp --dport 137 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A INPUT -p udp --dport 137 -m state --state NEW,ESTABLISHED -j
ACCEPT
#NETBIOS Datagram
iptables -A INPUT -p udp --dport 138 -m state --state NEW,ESTABLISHED -j
ACCEPT
#NETBIOS Session
iptables -A INPUT -p tcp --dport 139 -m state --state NEW,ESTABLISHED -j
ACCEPT
#IMAP
iptables -A INPUT -p tcp --dport 143 -m state --state NEW,ESTABLISHED -j
ACCEPT
#SQL
iptables -A INPUT -p tcp --dport 156 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A INPUT -p udp --dport 156 -m state --state NEW,ESTABLISHED -j
ACCEPT
#SNMP
iptables -A INPUT -p udp --dport 161 -m state --state NEW,ESTABLISHED -j
ACCEPT
#IMAP V3
iptables -A INPUT -p tcp --dport 220 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A INPUT -p udp --dport 220 -m state --state NEW,ESTABLISHED -j
ACCEPT


############################################################
## A continuación se configuran algunas protecciones extandar

# Protección contra ataque DDoS SYN flood
## Se limita a 500 conexiones/segundo, normalmene inferior a lo permitido en
el kernel.
iptables -N syn_flood
iptables -A INPUT -p tcp --syn -j syn_flood
iptables -A syn_flood -m limit --limit 500/s --limit-burst 2000 -j RETURN
iptables -A syn_flood -j DROP

# Bloquear paquetes con flags TCP falsos
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG
NONE -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,FIN FIN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL ALL -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL FIN,PSH,URG -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL SYN,FIN,PSH,URG -j
DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j
DROP
```

```
# Bloquear Ataques Enmascaramiento
#iptables -A INPUT -p icmp -m icmp --icmp-type address-mask-request -j DROP
#iptables -A INPUT -p icmp -m icmp --icmp-type timestamp-request -j DROP
#iptables -A INPUT -p icmp -m icmp -m limit --limit 1/second -j ACCEPT

#Descartar Paquetes Inválidos
iptables -A INPUT -m state --state INVALID -j DROP
iptables -A FORWARD -m state --state INVALID -j DROP
iptables -A OUTPUT -m state --state INVALID -j DROP

#Descartar paquetes RST Excesivos para Evitar Ataques Enmascarados
iptables -A INPUT -p tcp -m tcp --tcp-flags RST RST -m limit --limit 2/second
--limit-burst 2 -j ACCEPT
```

12. Anexo 2. Instalación de ELK.

En este trabajo hemos hecho un análisis de los logs en bruto. Para el tratamiento de los ficheros de logs dispersos existen varias alternativas que aportan una visión más amigable y sencilla de los datos. Una de las más completas es la combinación de tres herramientas complementarias: Elasticsearch, Logstash y Kibana. A este conjunto se le denomina ELK.

Como ampliación a este trabajo, se propone aquí la instalación de ELK para un posterior proyecto de configuración particularizada para nuestro honeypot, el cual puede ser más o menos complejo. ELK es potente y versátil. Permite recoger logs desde diferentes fuentes, procesarlos y generar vistas personalizadas con los datos. A continuación se indican los pasos para una primera instalación de ELK.

Deberemos disponer de Java en el caso de no estar instalado:

```
# yum install java-1.8.0-openjdk-devel
```

Bajar la clave pública de Elastic:

```
# rpm --import https://artifacts.elastic.co/GPG-KEY-
elasticsearch
```

Editar:

```
# Nano  /etc/yum.repos.d/elasticsearch.repo
```

Instalar Elasticsearch:

```
# yum install elasticsearch
```

Para configurar elasticsearch, editar el fichero:

```
#nano  /etc/elasticsearch/elasticsearch.yml
```

Localizar la última línea con network.host

```
network.host: localhost
```

Descomentar también

```
http.port: 9200 (en realidad es el puerto por defecto)
```

Configuramos el inicio de elasticsearch:

```
# systemctl start elasticsearch
```

```
# systemctl enable elasticsearch
```

Para probar la instalación de elasticsearch podemos hacer:

```
# curl -X GET "localhost:9200"
```

Ahora instalamos Kibana:

```
# yum install kibana
```

Configuramos el inicio de Kibana:

```
# systemctl enable kibana

# systemctl start kibana
```

Kibana por defecto está configurado para ser accesible sólo desde el localhost. Para que sea accesible desde cualquier equipo podemos instalar un reverse proxy para permitir conexiones externas, como Nginx. Esta configuración queda fuera del alcance de este trabajo.

Para verificar la instalación podemos acceder a Kibana:

```
http://localhost:5601/status
```

Ahora instaremos Logstash:

```
# yum install logstash
```

Una vez instalado, la configuración de Logstash se define en una serie de ficheros de tipo json que se encuentran en

```
/etc/logstash/conf.d/.
```

Para configurarlo crearemos el fichero:

```
/etc/logstash/conf.d/syslog-logstash.conf
```

El contenido será el siguiente:

```
#Secccion INPUT
input {
  beats {
    port => 5044
#    ssl => true
#    ssl_certificate => "/etc/pki/tls/certs/logstash-beats.crt"
#    ssl_key => "/etc/pki/tls/private/logstash-beats.key"
  }
}
#Secccion FILTER
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{SYSLOGHOST:syslog_hostname}
```

```
%{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match  =>  [  "syslog_timestamp",  "MMM    d  HH:mm:ss",  "MMM  dd
HH:mm:ss" ]
    }
  }
}

#Secccion OUTPUT
output {
  elasticsearch {
    hosts => ["localhost:9200"]
    sniffing => true
    manage_template => false
    index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    document_type => "%{[@metadata][type]}"
  }
}
```

Ahora habilitamos el servicio:

```
# systemctl start logstash
# systemctl enable logstash
```

Para recuperar datos desde varias fuentes de logs debemos instalar algún tipo de beat que se encargue de esto y lo canalice hacia Logstash. Para ello utilizaremos Filebeat.

```
# yum install filebeat
```

Ahora editamos el fichero de configuración:

```
 /etc/filebeat/filebeat.yml
```

Por defecto tendrá como activo el envío hacia Elitesearch. Debemos comentar las líneas:

```
#output.elasticsearch:
  # Array of hosts to connect to.
#  hosts: ["localhost:9200"]
```

y descomentar las líneas:

```
output.logstash:
  # The Logstash hosts
   hosts: ["localhost:5044"]
```

Ahora debemos configurar las fuentes de datos, el origen de la información, es decir, nuestros tres ficheros de logs. Por defecto viene configurado para capturar todo lo que hay en /var/log/*.log.

Comentamos la línea "- /var/log/*.log" e incluimos:

```
  enabled: true

- /var/log/iptables.log
- /var/log/snort/alert
- /var/log/honeyd.log
```

Una vez hecho esto podemos chequear la configuración con:

```
# filebeat test config -e
```

Por ultimo habilitamos el servicio:

```
# systemctl start filebeat
# systemctl enable filebeat
```

Una vez que se arranca todo, y se empiezan a generar logs, podemos consultar en Elasticsearch qué información existe asociada al índice filebeat-*:

```
# curl -XGET 'http://172.17.0.2:9200/filebeat-*/_search?pretty'
```

Ahora debemos realizar alguna acción en el frontend de Kibana. Los datos provenientes de Filebeat y deben ser detectados:

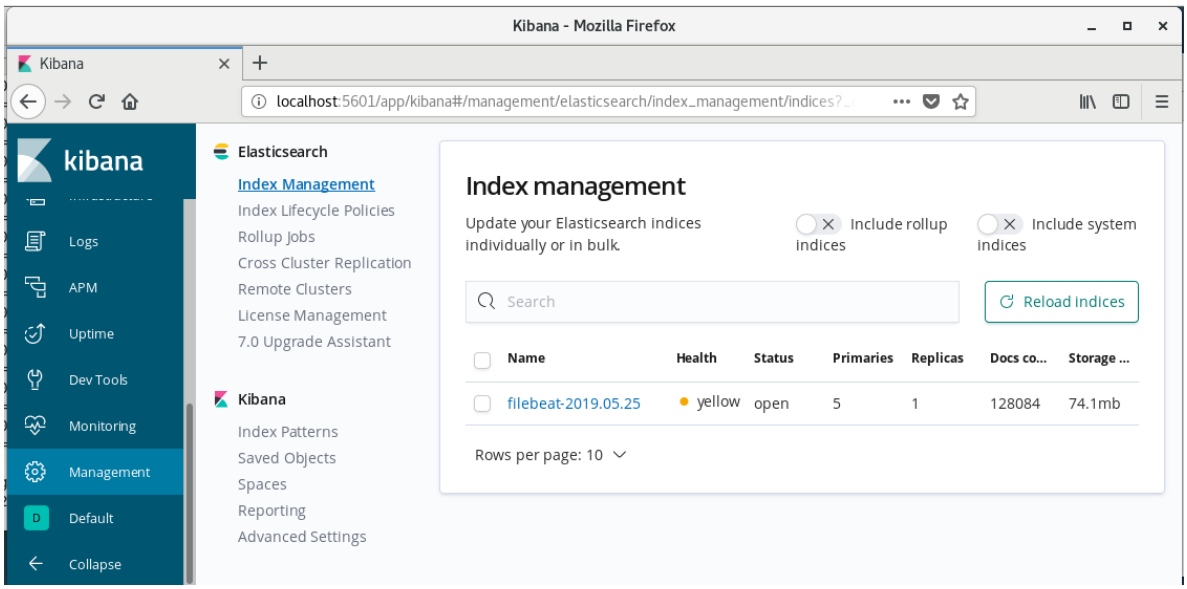


Figura 18. Frontend Kibana.

Filebeat crea índices bajo el patrón filebeat-[fecha]. Debemos especificar este patrón en Kibana:

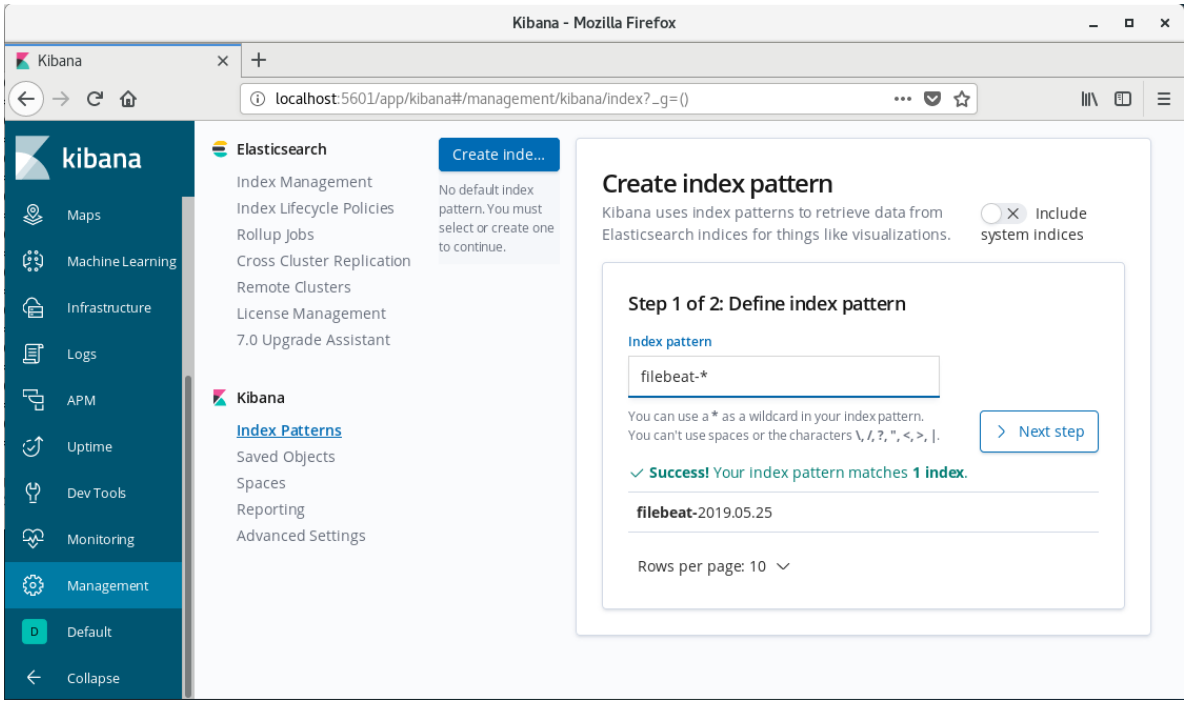


Figura 19. Creación de índices en Kibana

Elegimos inicialmente el filtro de campo por defecto:

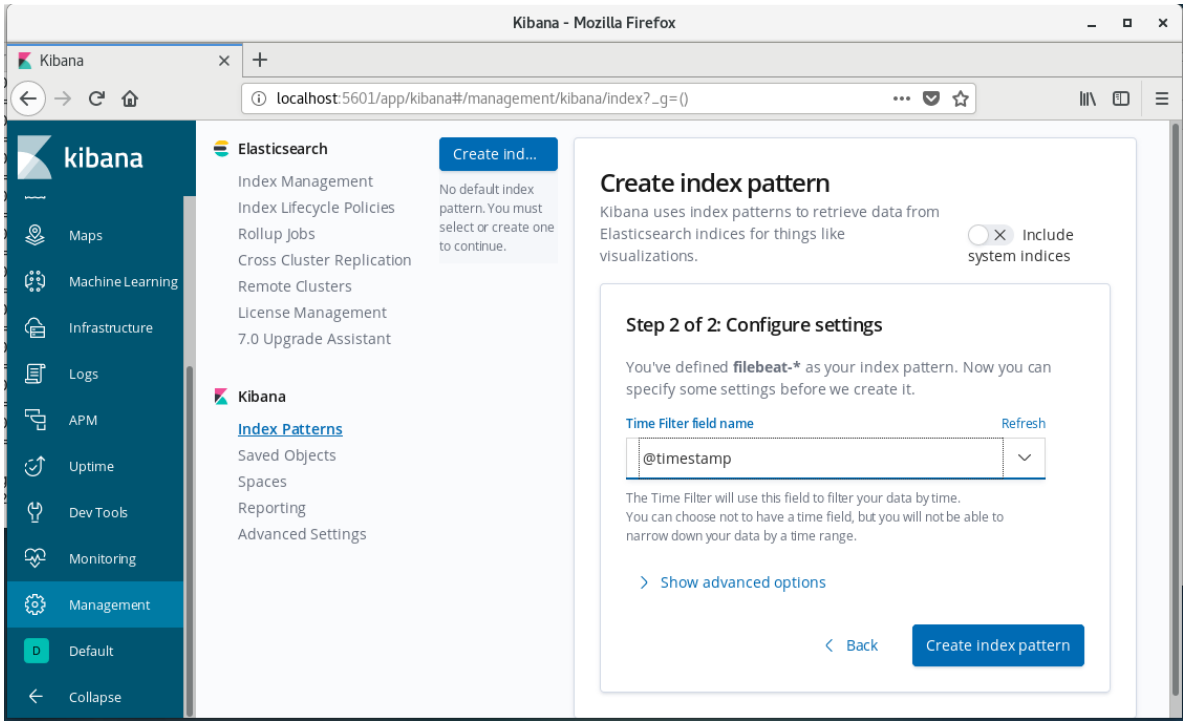


Figura 20. Agregación de un campo de datos en Kibana.

Con todo ello ya podremos ver alguna visualización:

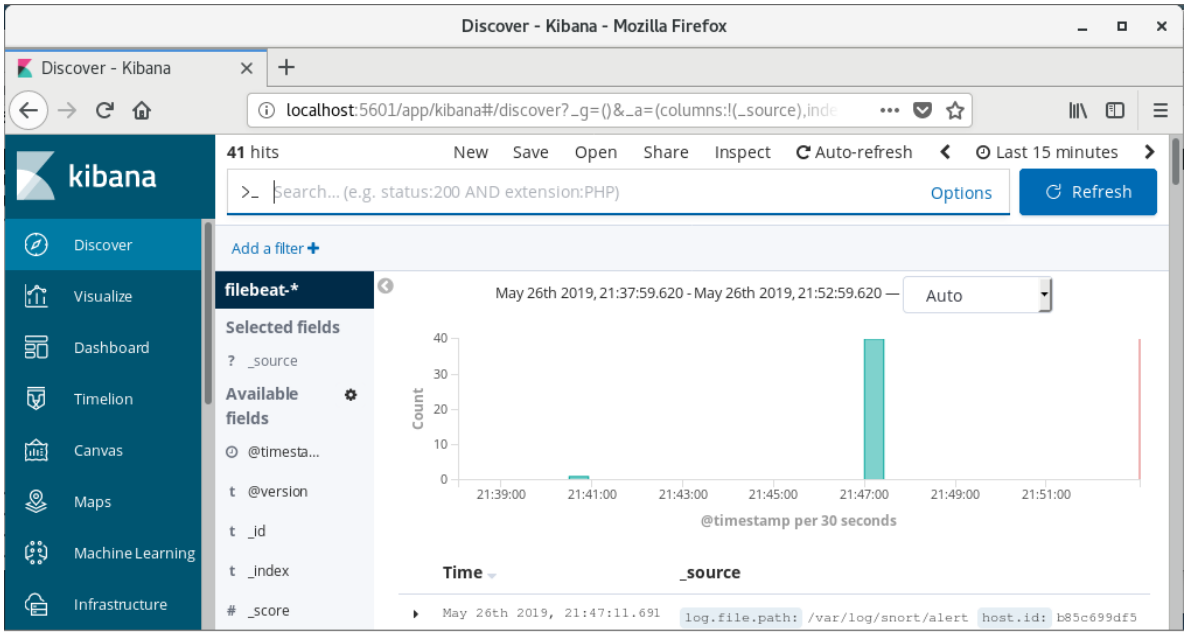


Figura 21. Ejemplo de representación gráfica en Kibana

Finalmente podemos eliminar si es preciso todos los datos cargados por Filebeat en Elasticsearch. Para ello eliminamos toda la información asociada a los índices con el patrón filebeat-*:

```
# curl -XDELETE 'http://172.17.0.2:9200/filebeat-*'
```

Como vemos, hemos obtenido una representación gráfica pero sin información de interés. Aún se requiere un trabajo de formateo y adecuación de los logs definiendo los campo de los mismos que nos interesa almacenar y el formato unificado con el que poder posteriormente generar gráficas interesantes. Dicho trabajo se deja como ampliación.