Universitat Oberta de Catalunya

uoc.edu

Desarrollo de honeypot para entornos ICS con Docker

Rafael Ruiz López

Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones

Seguridad en el internet de las cosas

Helena Rifà Pous

Carlos Hernández Gañán

04/06/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo de un honeypot para entornos ICS con Docker*
Nombre del autor:	*Rafael Ruiz López*
Nombre del consultor/a:	Helena Rifà Pous
Nombre del PRA:	Carlos Hernández Gañán
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	Máster Universitario 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 aula 2*
Idioma del trabajo:	*Español*
Palabras clave	*Máximo 3 palabras clave, validadas por el director del trabajo (dadas por los estudiantes o en base a listados, tesauros, etc.)*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Investigación sobre los dispositivos ICS, sus comunicaciones, y el desarrollo de un sistema de honeypot para detectar ataques estos dispositivos, así como analizar la incidencia y los tipos de los mismos.

Para este desarrollo se ha utilizado Docker, así como otras tecnologías open source como Kibana, ElasticSearch y logstash.

Abstract (in English, 250 words or less):

ICS devices and communication analisys, through a honeypot system development. This system will detect atacks to this devices and offer data to analice what kind of attacks and the severity.

Docker software has been used for this development, and other open source tecnologies like kibana, elasticsearch or logstash.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente el uso de dispositivos IoT en todos los ámbitos esta muy extendido, y la previsión futura es que la tendencia continúe. Estos dispositivos se utilizan con una amplia gama de fines, desde sistemas de entretenimiento en el ámbito del hogar (Alexa, Google Home, domótica), sistemas de vigilancia (cámaras de seguridad, alarmas conectadas), y en el ámbito industrial, se utilizan en con diferentes objetivos, como la monitorización de activos, SCADA, PLCs, etc. En este TFM se abordará la creación de un honeypot orientado a la detección de ataques informáticos a sistemas IoT, en concreto, a dispositivos utilizados en sistemas de control industriales (ICS).

1.2 Objetivos del Trabajo

El objetivo de este TFM es estudiar la seguridad de los sistemas de control industriales que hacen uso de dispositivos IoT, analizando cuales son las posibles amenazas a estos sistemas mediante el desarrollo de un honeypot que permita capturar esos ataques.

- Estudio del estado actual que incluya:
  - Cuales son los dispositivos IoT usados en ICS potencialmente vulnerables
    Con el objetivo de seleccionar cuales son los mejores dispositivos a emular en el honeypot para tener mas probabilidad de tener ataques.
  - Cuales son las vulnerabilidades mas relevantes en función de criticidad, frecuencia y fecha de descubrimiento
    Con el objetivo de seleccionar cuales son las vulnerabilidades mas apropiadas para el desarrollo del honeypot y tener mas probabilidad de detección de posibles ataques. Para esto, se realizará un análisis de los firmwares / softwares mas utilizados en estos dispositivos, así como las versiones de sus componentes para descubrir posibles vulnerabilidades.
  - Determinar los atacantes mas comunes
    Con el objetivo de conocer como funcionan y organizan los atacantes mas comunes, como las botnets, o que tipos de ramsonwares se utilizan para intentar obtener el control de estos dispositivos.
- Creación de un honeypot funcional que permita la detección de ataques que permita:
  - Configuración para emular diferentes dispositivos
    Se desarrollará una solución que permita configurar la emulación de diferentes dispositivos IoT usados en ICS.
  - Configuración para emular diferentes vulnerabilidades
    La solución desarrollada, además de emular diferentes dispositivos, permitirá emular las diferentes vulnerabilidades de cada uno de los dispositivos seleccionados.
  - Escalabilidad
    La solución desarrollada permitirá la puesta en funcionamiento de múltiples instancias con diferentes configuraciones, ampliando las posibilidades de obtener datos de diferentes ataques simultáneamente.
  - Fácil mantenimiento
    Debe ser sencillo iniciar y parar el sistema, lanzar nuevas instancias, y recoger la información obtenidas a partir de estas instancias.

1.3 Enfoque y método seguido

Para la obtención de la información se utilizarán fuentes contrastadas sobre dispositivos utilizados en ICS, cuales son los principales protocolos que utilizan, cuales son sus vulnerabilidades y se obtendrá información sobre estas vulnerabilidades también en web contrastadas para este fin como: cvedetails.com, vuldb.com, etc.

Para el desarrollo de la infraestructura y del software se utilizará la metodología waterfall debido a que el desarrollo lo realizará una única persona y se conocen los requisitos con antelación.

Por tanto, este desarrollo tendrá las siguientes fases:

1. Análisis: Durante este proceso se analizarán cuales son los requerimientos específicos del software y la infraestructura.
2. Diseño del sistema: Durante esta fase del proceso, se designarán cuales son las tecnologías a utilizar en este proyecto, así como cuales son las configuraciones adecuadas para el mismo.
3. Diseño del programa: Durante esta fase, se establecerá cual es la estructura del desarrollo software, y como quedarán organizados los diferentes algoritmos que se utilicen.
4. Configuración del sistema y codificación: Durante esta fase, se realizarán todas las configuraciones necesarias para para poner en funcionamiento el entorno de ejecución de la aplicación y se codificará la aplicación previamente diseñada.
5. Pruebas: Durante esta fase se probará que todo lo realizado previamente en las diferentes fases de desarrollo e implementación funciona correctamente.

1.4 Planificación del Trabajo

	TAREA	FECHA DE INICIO	FECHA DE FIN	EMPIEZA EL DIA	DURACION* (DIAS LABORABLES)
DOCUMENTACION					
	Investigar Dispositivos IoT	8/3/19	9/3/19	0	1
	Documentar Dispositivos IoT	8/3/19	9/3/19	0	1
	Investigar Software Dispositivos IoT	8/3/19	9/3/19	0	1
	Documentar Software Dispositivos IoT	8/3/19	9/3/19	0	1
	Investigar Vulnerabilidades IoT	9/3/19	10/3/19	1	1
	Documentar Vulnerabilidades IoT	9/3/19	10/3/19	1	1
	Investigar Ataques IoT	9/3/19	10/3/19	1	1
	Documentar Ataques IoT	9/3/19	10/3/19	1	1
DESARROLLO					
	Selección tecnologías de virtualización	11/3/19	12/3/19	3	1
	Selección entorno de desarrollo	11/3/19	12/3/19	3	1
	Configuración entorno de desarrollo	14/3/19	17/3/19	6	3
	Configuración sistema de virtualización	14/3/19	25/3/19	6	11
	Selección sistema de detección y registro	11/3/19	12/3/19	3	1
	Configuración sistema de detección y registro	25/3/19	27/3/19	17	2
	Diseño del software del sistema	26/3/19	30/3/19	18	4

Implementación del software del sistema	30/3/19	13/4/19	22	14
Realización de pruebas del sistema	13/4/19	14/4/19	36	1
Documentación análisis del sistema	11/3/19	14/4/19	3	34
Documentación diseño del sistema	11/3/19	14/4/19	3	34
Doc. configuración y desarrollo del sistema	11/3/19	14/4/19	3	34
Documentación pruebas del sistema	11/3/19	14/4/19	3	34

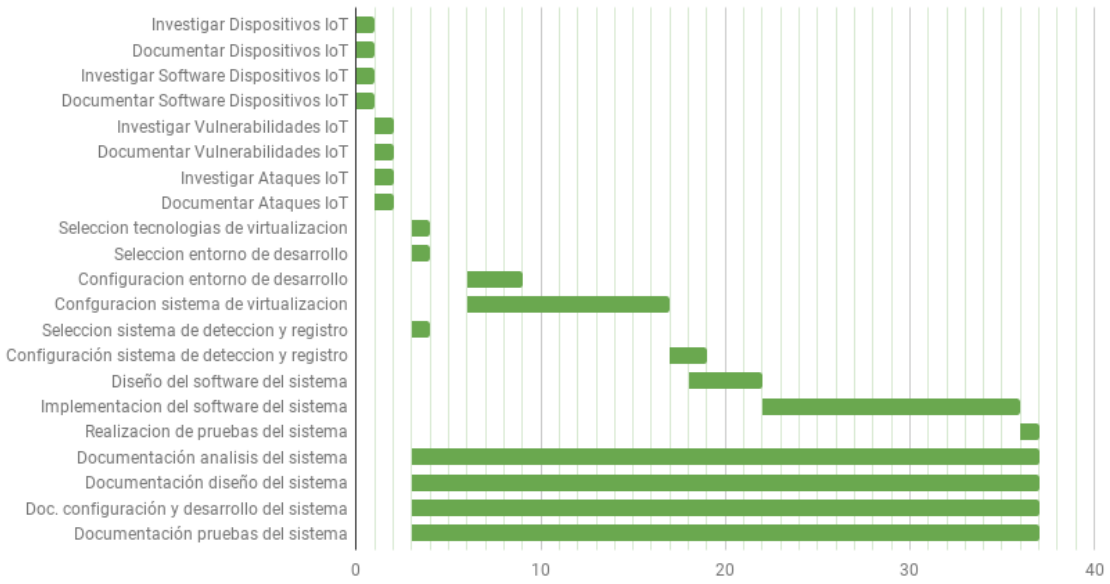

Ilustración 1 - Gantt

1.5 Breve sumario de productos obtenidos

Se ha desarrollado un sistema basado en docker, que permite emular dispositivos ICS, o al menos algunas de sus características, y también permite registrar interacciones entre los mismos, así como con terceros dispositivos.
Se han configurado aplicaciones que permiten el registro y análisis de los logs generados así como del trafico generado en la red.

Con esta información podríamos establecer si estamos siendo objetivo de un ataque sin exponer nuestros dispositivos ICS reales.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos se profundizará en la metodología, el análisis de los dispositivos ICS objetivo, en el análisis de requisitos, el diseño del sistema, y la infraestructura a utilizar y posteriormente se entrará en detalle sobre la implementación del sistema.

2. Resto de capítulos

2.1 Metodología

La metodología a seguir para el desarrollo del proyecto será la siguiente:

Para la obtención de la información se utilizarán fuentes contrastadas sobre dispositivos utilizados en ICS, cuales son los principales protocolos que utilizan, cuales son sus vulnerabilidades y se obtendrá información sobre estas vulnerabilidades también en web contrastadas para este fin como: cvedetails.com, vuldb.com, etc.

Para el desarrollo de la infraestructura y del software se utilizará la metodología waterfall debido a que el desarrollo lo realizará una única persona y se conocen los requisitos con antelación.

Por tanto, este desarrollo tendrá las siguientes fases:

1. Análisis: Durante este proceso se analizarán cuales son los requerimientos específicos del software y la infraestructura.
2. Diseño del sistema: Durante esta fase del proceso, se designarán cuales son las tecnologías a utilizar en este proyecto, así como cuales son las configuraciones adecuadas para el mismo.
3. Diseño del programa: Durante esta fase, se establecerá cual es la estructura del desarrollo software, y como quedarán organizados los diferentes algoritmos que se utilicen.
4. Configuración del sistema y codificación: Durante esta fase, se realizarán todas las configuraciones necesarias para para poner en funcionamiento el entorno de ejecución de la aplicación y se codificará la aplicación previamente diseñada.
5. Pruebas: Durante esta fase se probará que todo lo realizado previamente en las diferentes fases de desarrollo e implementación funciona correctamente.

2.2 Análisis de los dispositivos ICS objetivo

2.2.1 Introducción

IoT es la interconexión a través de internet de dispositivos de uso cotidiano, permitiéndoles enviar y recibir información. ICS es un término genérico para describir la integración de hardware y software con el objetivo de ofrecer soporte a infraestructura crítica.

El uso de dispositivos conectados (IoT) se está generalizando en la sociedad actual en todos los ámbitos de esta. En los hogares cada vez es mas común disponer de televisores, cámaras de seguridad, video consolas y otros dispositivos de entretenimiento conectados a internet.

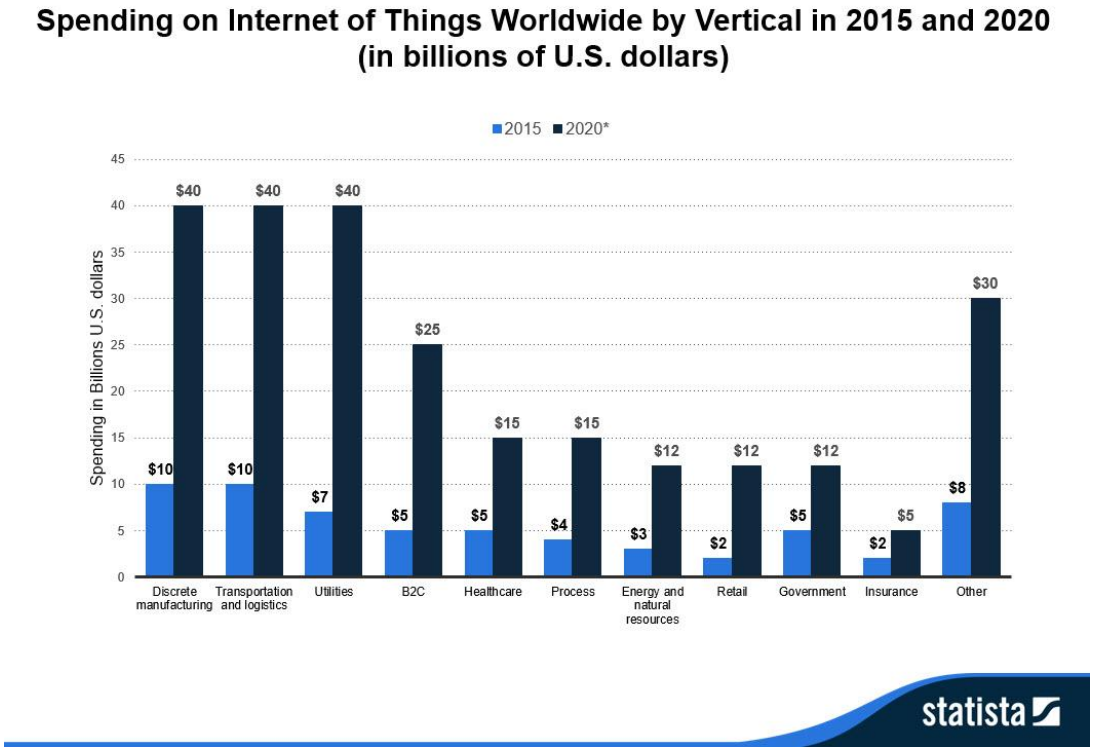

Ilustración 2 - Gasto en IoT mundial en 2015 y 2020

Pero este cambio de paradigma en los dispositivos tecnológicos no solo es perceptible en los hogares, sino cada vez más, en entornos ICS, donde es posible observar la presencia de dispositivos IoT con diferentes fines. SCADAs, sistemas de video vigilancia, sistemas de control de acceso, maquinaría de producción, etc., cada vez es mas común que estos dispositivos estén conectados a internet, para su monitorización, su uso o gestión. Estos sistemas conectados facilitan la centralización de la información y la realización de informes y estadísticas gracias a esta conexión a internet, incrementando la eficacia y la eficiencia de los procesos. Sin embargo,

exponer estos dispositivos, en algunos casos críticos, a una conexión a internet puede convertirlos en sistemas inseguros.

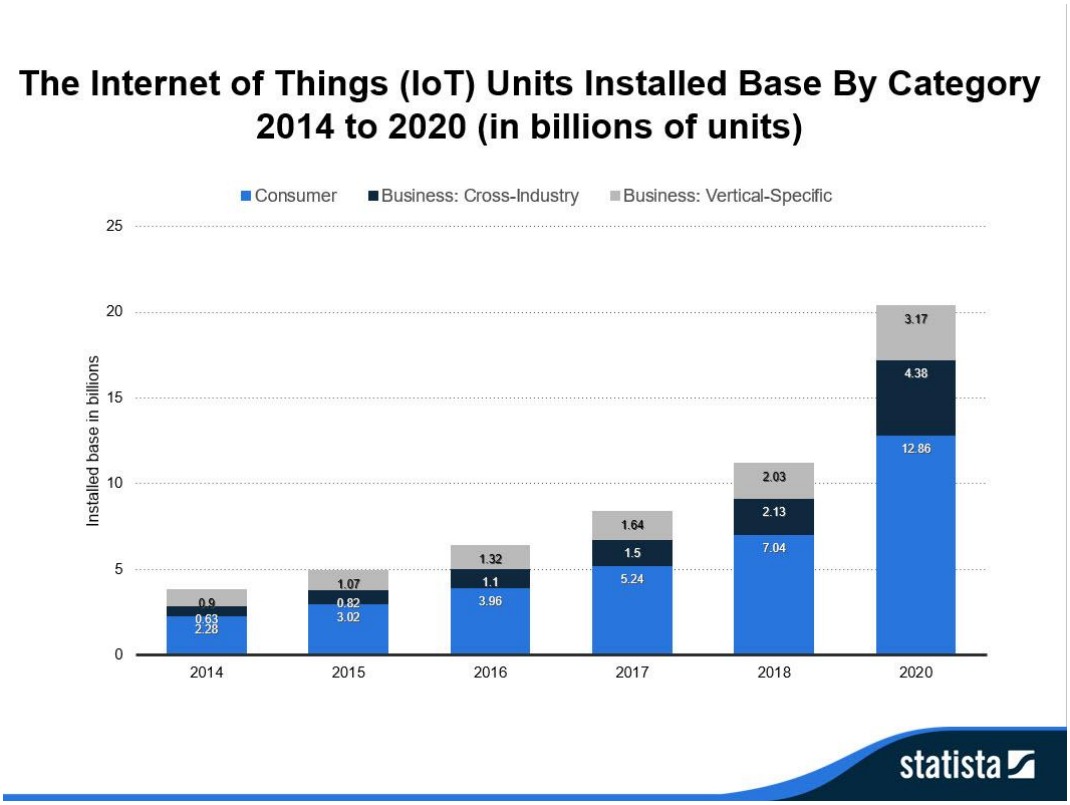

Ilustración 3 - Dispositivos IoT instadas por categoría

Una de las grandes diferencias entre los dispositivos IoT en el hogar y los dispositivos IoT en ICS es la criticidad de estos. En el caso del hogar, un ataque puede provocar pérdida de información personal, o que utilicen nuestros dispositivos para otros fines como ataques DDoS o procesamiento con diferentes fines. Sin embargo, un ataque en un dispositivo IoT en ICS, puede provocar cuantiosas perdidas económicas para la empresa o incluso para la región en la que se encuentre, poniendo en peligro la seguridad económica del entorno. También puede ocasionar malfuncionamiento que ponga en riesgo la vida de operarios y técnicos relacionados con esos dispositivos.

2.2.2 Dispositivos IoT en ICS

Las tecnologías ICS incluyen entre otras, el control y adquisición de datos (SCADA), sistemas de control distribuidos (DCS), los sistemas de automatización y control industrial (IACS), los sistemas de control lógico programable (PLCs), los sistemas de control de automatismos programable (PACs), las unidades de terminal remota (RTUs), los dispositivos electrónicos inteligentes (IEDs) y sensores.

El ecosistema tecnológico asociado a estos sistemas es muy heterogéneo, con multitud de proveedores diferentes, y diferentes tecnologías tanto hardware como

software para estos sistemas. Muchos fabricantes ofrecen soluciones especificas para diferentes sectores tecnológicos: Gestión eléctrica, gestión del agua, gestión de producción etc.

Algunos de los principales proveedores de este tipo de soluciones son Siemens, Schneider, ABB, Osisoft, Emerson, Omron, Mitsubishi, Rockwell automation, Honeywell, General Electric o Yokogawa. Como se ha comentado previamente, las soluciones son muy heterogéneas ya que se aplican a un entorno en concreto, siendo diferentes en cada uno de los entornos que se quieran controlar o monitorizar. Las soluciones proporcionadas pueden disponer de algunas de las tecnologías mencionadas previamente o disponer de todas, en función de las necesidades planteadas por el cliente para su proceso.

2.2.2.1 Sistema de Supervisión, Control y Adquisición de Datos (SCADA)

Por norma general un SCADA es un software que se permite supervisar y controlar múltiples procesos industriales a distancia, a través de la obtención de datos de diferentes fuentes, y el análisis de eventos. También suele permitir consultar históricos de las variables medidas. Este software suele instalarse en un hardware estándar, conectado a la red local o a internet, que permita el acceso de los usuarios a través de una interfaz hombre – máquina (HMI).

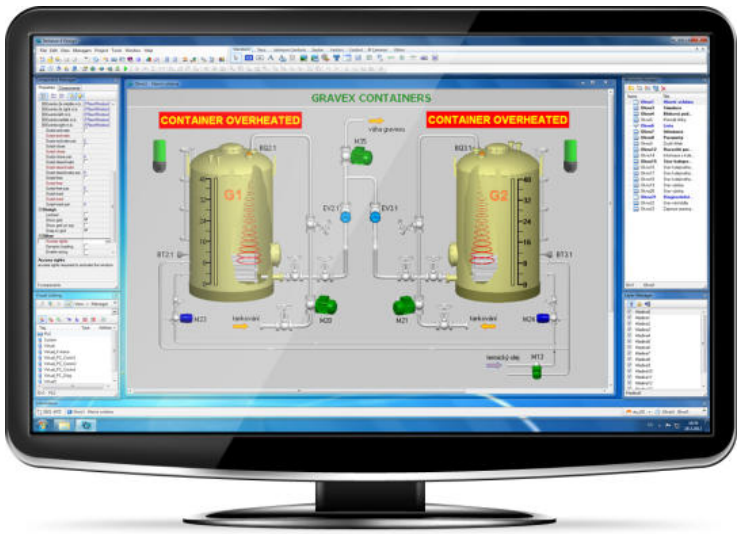

Ilustración 4 - SCADA HMI

2.2.2.2 Sistemas de control distribuidos (DCS)

Los DCS son sistemas de control orientados a procesos industriales complejos. Los DCS actúan sobre un único proceso, a mas bajo nivel que los SCADA. La funcionalidad ofrecida y funcionamiento es similar al de los SCADA, sin embargo, su orientación es a mas bajo nivel, y mientras un DCS controla un proceso, un SCADA

puede obtener la información de múltiples DCS para tener un control mas global de los procesos de la industria. Al igual que los SCADA, el hardware utilizado para estos sistemas suelen ser servidores estándar conectados a una red local que permita su control desde un HMI, y la adquisición de datos de los diferentes sensores.

2.2.2.3 Sistemas de control lógico programable (PLCs)

Un control lógico programable o PLC es un computador dedicado a la automatización de procesos electromecánicos. Los PLC están diseñados para gestionar múltiples entradas y salidas, por lo que utilizan un hardware especifico, con múltiples entradas y salidas de diferentes tipos. Por otra parte, los PLC están pensados para actuar en tiempo real. Prácticamente todos los fabricantes mencionados previamente disponen de sus propios PLC.

Ilustración 5 - Siemens PLC

Ilustración 6 - ABB PLC

2.2.2.4 Sistemas de control de automatismos programables (PACs)

Aunque cumplen con la misma función que los PLC, los PAC siguen una estructura modular que hace que las expansiones sean más sencillas que con PLC y por lo general tienen mayor capacidad de computación. Mientras que los PLC están mas orientados a una simple ejecución lineal de un programa, los PAC tienen mas capacidad de procesado de señales y permiten hacer una programación mas compleja.

Ilustración 5 - ABB PAC

2.2.2.5 Unidades de terminal remota (RTUs)

Una RTU es un dispositivo basado en microprocesadores, el cual permite obtener señales independientes de los procesos y enviar la información a un sitio remoto donde se procese. Los dispositivos RTU permiten la comunicación de datos de telemetría desde sensores hasta SCADAs o DCS.

Un RTU puede recibir diferentes inputs tanto señales digitales como analógicas y las convierte en el formato de comunicación necesario para que se puedan transmitir hacia el SCADA o DCS correspondiente a través de diferentes protocolos soportados.

Ilustración 6 - Schneider Electric RTU

2.2.2.6 Dispositivos electrónicos inteligentes (IEDs)

Es el término utilizado en la industria de la energía eléctrica para describir equipos de regulación electrónica inmersos en los sistemas eléctricos, por ejemplo, utilizados en interruptores, transformadores y bancos de capacitores.

Los IEDs reciben datos de los sensores y diversos dispositivos eléctricos, y puede informar los comandos de control, tales como interruptores que se disparan cuando se detectan voltajes, corrientes o frecuencias anómalas, cuando se suceden las variaciones por el aumento o niveles de tensión inferior para mantener el nivel deseado.

Ilustración 7 - Siemens IED

2.2.3 Software de los dispositivos IoT en ICS

2.2.3.1 Sistemas operativos

Los dos únicos sistemas de ICS que se montan sobre un sistema operativo estándar son los SCADA y los DCS, para ambos tipos existen soluciones en las dos principales arquitecturas de sistemas operativos, Microsoft Windows y Linux.

2.2.3.2 Software

El proveedor de la solución proporciona el software que implementa las características de los SCADA y DCS. Para PLCs, PACs, RTUs y IEDs al ser programables según las necesidades del usuario, el proveedor de la solución proporciona un framework que facilita la programación de estos dispositivos. Por lo tanto, estos dispositivos tienen tanto una componente del fabricante, que hace las veces de base o núcleo del sistema, como una componente de personalización para cada aplicación de la solución para implementar la funcionalidad deseada.

2.2.3.3 Protocolos de comunicaciones

Existen multitud de protocolos de comunicaciones, que utilizan diferentes capas físicas, lógicas, desarrollados por diferentes empresas y utilizados con diferentes propósitos. Cada sector industrial suele preferir algunos protocolos de comunicaciones sobre otros. Por otra parte, existen protocolos propietarios y cerrados, y alternativas libres.

Algunos de estos protocolos de uso industrial son:

	Propietario	**Capa física**	**Capa Lógica**	**Desarrollador**
OMRON FINS	Cerrado	Ethernet	Sobre TCP/IP	Omron
S7/S7-Plus	Abierto	Ethernet	Sobre TCP/IP	Siemens
EtherNet/IP Siemens	Abierto	Industrial Ethernet	Sobre TCP/IP	Siemens
GE-SRTP	Cerrado	Ethernet	Protocolo propio	General Electrics
GE EDG	Abierto	Ethernet	UDP	General Electrics
Sattbus	Cerrado	Ethernet	Sobre TCP/IP	Kepware
UA/AE/DA OPC	Cerrado	Ethernet	Sobre TCP/IP	Microsoft
Profinet	Abierto	Industrial Ethernet	Sobre TCP/IP	Profibus & Profinet International
Profibus	Abierto	RS-485, Fibra óptica		BMBF
Modbus	Abierto	Ethernet	Sobre TCP/IP	Modicon
RTU Modbus	Abierto	Serie, RS232		Modicon

DNP3	Abierto	Ethernet	Sobre TCP/IP	GE-Harris Canada
MQTT	Abierto	Ethernet	Sobre TCP/IP	IBM / Cirrus
SNMP	Abierto	Ethernet	Sobre TCP/IP	IETF
IEC 60870-5-104	Abierto	Ethernet, PPP	Sobre TCP/IP	IEC TC

Ilustración 8 - Protocolos de comunicaciones

Existen otros muchos protocolos utilizados en el sector industrial. Y también existen adaptaciones de protocolos abiertos, realizadas por diferentes compañías como puede ser Modbus Plus desarrollado por Schneider o Pemex Modbus desarrollado para Pemex.

2.2.3.4 Servicios

En el entorno industrial, en las soluciones que lo permiten, se utilizan los principales servicios TCP/IP, como son:

- http, https: Hypertext Transfer Protocol, protocolo utilizado para transferencia de documentos html
- ftp, sftp: File Transfer Protocol, protocolo utilizado para la transferencia de ficheros
- dns: Domain name system, protocolo utilizado para la resolución de nombres a direcciones IP
- dhcp: Dynamic Host Configuration Protocol, protocol utilizado para la asignación de IP a equipos en red
- arp: Address Resolution Protocol, protocolo utilizado para resolver cual es la dirección física que le corresponde a una IP en una red de comunicaciones.
- telnet: Telecommunication Network, es un protocolo de red que nos permite acceder a otra máquina remotamente.
- Ssh: Secure Shell, es un protocolo que nos proporciona acceso remoto a otra máquina por medio de un canal seguro.

2.2.3.5 Vulnerabilidades de los dispositivos IoT en ICS

Analizando los diferentes sistemas operativos, el software y servicios, y protocolos utilizados en los ICS, podemos hacer un análisis de algunas de las potenciales vulnerabilidades de estos sistemas. Podemos dividir las diferentes vulnerabilidades por tipos. Y así podemos encontrar en este tipo de sistemas vulnerabilidades causadas por:

- Malware
- Bugs / exploits
- Autentificación / Autorización, privilegios incorrectos
- Falta de cifrado en los datos o comunicaciones
- Vulnerabilidades propias del software, sistema operativo o servicio (SQL Injection, XSS, buffer overflow, Command injection, DNS Spoofing, Arp Spoofing)

En el caso de las vulnerabilidades causadas por Malware son aquellas en las que un tercero consigue instalar algún tipo de software que realiza funciones no deseadas. Las vulnerabilidades de Malware pueden ser causadas por Virus, Troyanos, Gusanos, etc. El sistema perjudicado por la explotación de esta vulnerabilidad puede ser usado por el atacante con diferentes fines, desvelar información protegida, utilizar los dispositivitos como parte de una Botnet etc. Esta vulnerabilidad puede ser mitigada teniendo un estricto control del software y permisos instalados en cada uno de los sistemas. Esta vulnerabilidad es muy importante ya que gran parte de las vulnerabilidades de otros tipos se utilizarán para que el atacante pueda disponer de algún software instalado en nuestro sistema.

Los bugs son errores o problemas del sistema que permite realizar acciones no deseadas por el administrador del sistema. Estos bugs suelen ser corregidos en sucesivas versiones del software. En función del tipo de bug que se encuentre y se utilice el atacante podrá realizar diferentes acciones. En algunos casos es posible tener acceso a sistema totalmente e instalar y gestionarlo completamente, con lo que podría apagarlo, encenderlo, modificar las aplicaciones instaladas, o los datos, causando grandes daños. Esta vulnerabilidad puede causar grandes daños a cualquier sistema, y puede ser utilizada para instalar Malware.

Los problemas causados por los fallos de autentificación y autorización son muy similares a los causados por los bugs / exploits. Estos problemas suelen afectar a los sistemas operativos con mayor frecuencia, aunque también pueden afectar a aplicaciones que utilicen este tipo de sistemas. Este problema puede ser normalmente solucionado por parte del administrador del sistema. Como hemos comentado en el caso de los bugs / exploits, este problema podría causar que un atacante infectara nuestros sistemas con algún tipo de Malware.

Los problemas causados por la falta de cifrado en datos o comunicaciones, facilitar mucho a que un atacante lea información protegida, o bien mensajes entre los diferentes dispositivos. Incluso, en algunos protocolos como Modbus que son completamente abiertos, si no se cifra la información el atacante podría inyectar paquetes y corromper la información del sistema.

En caso de las vulnerabilidades causadas por falta de protección en el propio software, sistema operativo o servicio, por fallos de diseño o implementación como son los SQL

Injection que permite ejecutar consultas en la base de datos, los XSS que permite modificar los datos de entrada de una pagina web, o los buffer overflow que pueden permitir la ejecución de código remoto, mediante el acceso a una zona de memoria no protegida. A nivel de sistema operativo uno de los principales problemas es el command injection, que a través de diferentes vulnerabilidades en diferentes protocolos puede permitir la ejecución arbitraria de comandos del sistema operativo, lo que permitiría el acceso a la información o datos del sistema asi como la instalación de Malware. Si comentamos vulnerabilidades en los servicios, el servicio de DNS, o ARP mal configurado permitiría que los mensajes no llegaran a su destino mediante ataques de DNS o ARP spoofing. Esto podría acarrear un mal funcionamiento del sistema, o una extracción de la información contenida en los paquetes enviados.

Los objetivos finales de explotar estas vulnerabilidades pueden ser varios. Muchos de estos ataques tienen un fin económico, en muchos casos pretenden negociar con la información obtenida, ya sea extorsionando al propietario legítimo de la información o bien negociando con esa información que puede ser interesante para la competencia o los medios de comunicación. Otro fin económico puede ser el de perjudicar a la empresa atacada, provocando retrasos, averías en la maquinaria o perdidas de información que requieran un tiempo de estudio. Otro tipo de atacantes potenciales de estas infraestructuras son los creadores de botnets, que intentan obtener dispositivos conectados para, poder lanzar diferentes tipos de ataques como DDoS, o distribución de SPAM. Estos atacantes normalmente reciben algún tipo de retribución económica por estas tareas.

2.3 Análisis de requisitos

Como hemos comentado previamente, los requisitos para este TFM los podemos dividir en 2 partes. Una primera parte de investigación y documentación que se puede denominar “Análisis Tecnológico” y una segunda parte de desarrollo de la infraestructura y del software necesario que denominaremos “Desarrollo”. Analizaremos los requisitos de ambas partes por separado:

2.3.1 Requisitos del Análisis Tecnológico

- La documentación debe recoger información sobre cuales son los dispositivos utilizados en sistemas ICS. Esta información debe incluir datos sobre modelos, software utilizado, tipos, valoración de criticidad.
- La documentación debe recoger información sobre cuales son las vulnerabilidades mas comunes de los dispositivos IoT utilizados en ICS, así como la gravedad de las mismas. Se realizará una valoración de la importancia de las mismas en función de su complejidad de explotación, o si es una vulnerabilidad reciente.
- La documentación debe recoger cuales son los ataques mas frecuentes a dispositivos IoT, y si fuera posible, información sobre los ataques mas frecuentes a dispositivos IoT utilizados en ICS. Cuales han sido los últimos ataques producidos, así como su relevancia y su complejidad.

2.3.2 Requisitos del Desarrollo

- El sistema debe ser capaz de emular un dispositivo IoT
- El sistema debe ser capaz de emular una vulnerabilidad de un dispositivo IoT
- El sistema debe ser capaz de ser reproducido en diferentes entornos hardware / software, es decir, el desarrollo debe poder ser exportado desde mi sistema de desarrollo, hasta otra plataforma donde se le realicen pruebas.
- El sistema debe ser capaz de identificar un ataque a una vulnerabilidad concreta.
- El sistema debe ser capaz de registrar los ataques producidos
- El sistema debe ser escalable. Es decir, se debe poder reproducir tantas instancias del sistema como sean requeridas. Esta escalabilidad estará limitada por el hardware del que se disponga.
- El sistema debe ser gestionable fácilmente, es decir, debe poder ser reiniciado con facilidad, debe poder iniciarse una nueva instancia rápidamente si se requiere y los datos obtenidos deben poder ser recuperados con facilidad.
- Las diferentes partes del desarrollo deben ser documentadas, es decir, el análisis, el diseño, la configuración y desarrollo, y las pruebas.

2.3.3 Infraestructura, software y licencias necesarias

Para el desarrollo de este proyecto se requerirá con total seguridad, de un sistema hardware que permita generar maquinas virtuales o contenedores con relativa facilidad. Existen soluciones open source que no requieren de ninguna licencia para esto.

También se necesitará probablemente servidores web, o de cualquier otro tipo de protocolo por el cual se produzca la vulnerabilidad. En el caso de servidores web existen soluciones open source que no requieren de ninguna licencia de uso.

Para el desarrollo del software necesario se necesitará algún IDE para lo cual también existen soluciones open source sin necesidad de licencia. También se necesitaran lenguajes de programación y marcado, para los cuales también existen soluciones open source, y con licencias libres.

Para la gestión del proyecto, será necesario el uso de alguna herramienta de gestión de proyectos, existen alternativas libres y gratuitas para esto. En este caso, utilizaré Google Docs.

2.4 Diseño del sistema

2.4.1 Introducción

El objetivo del TFM es el de simular los dispositivos que existen en una red de comunicaciones ICS para poder detectar algún ataque, analizando las solicitudes que le llegan a los diferentes dispositivos y como son estas. Debido que se trata de un "honeypot", trataremos de simular / virtualizar / emular diferentes dispositivos comunes en los sistemas de control industrial. Para esto necesitaremos una infraestructura software / hardware que lo permita. En los siguientes epígrafes de este documento se definirán cuales son los principales requisitos que debe cumplir el sistema para ser un "honeypot" funcional para la detección de ataques en sistemas IoT ICS. Así como cual será la infraestructura y arquitectura del sistema, el software y las tecnologías necesarias para ello. En el caso de existir diferentes alternativas se justificará la elección de unas sobre otras.

Por ultimo se explicará cual es el proceso de creación y configuración del entorno y del software necesario para el funcionamiento del "honeypot".

2.4.2 Requisitos

Para que el sistema sea funcional debe cumplir estos requisitos:

- El sistema debe ser capaz de emular algunas características externas de un dispositivo ICS real, como puertos abiertos, interfaces, etc.
- El sistema debe ser capaz de emular algunos de los bugs de dispositivos ICS, como problemas con las versiones de los servidores o similar.
- Un dispositivo emulado por el sistema debe ser capaz de enviar y recibir mensajes del sistema original por los protocolos de comunicaciones emulados.
- El sistema debe ser capaz de recoger información relevante de todos los dispositivos emulados y los eventos generados por estos al recibir un mensaje o detectar algún tipo de anomalía.
- El sistema generado debe poder ser reproducido en diferentes infraestructuras hardware y software
- El sistema debe ser capaz de generar alertas configurables.

2.4.3 Selección de tecnologías

2.4.3.1 Sistemas operativos

Existen sistemas SCADA y DCS para los sistemas operativos mas populares como Windows o Linux. El resto de los dispositivos IoT en ICS no hacen uso de un sistema operativo generalista propio. Teniendo en cuenta el alcance de este TFM, y los costes posibles, trataremos de utilizar tecnologías libres, abiertas y gratuitas, pero manteniendo en la medida de lo posible la compatibilidad para la emulación de otros sistemas.

Por otra parte, al no tener que simular el comportamiento completo de estos sistemas, solo las entradas y salidas de estos, los sistemas operativos basados en Unix pueden simplificarnos la tarea debido a su simplicidad a la hora de gestionar puertos y servicios.

De entre todas las distribuciones Linux, actualmente una de las mas populares es Ubuntu, una distribución basada en Debian, pero con sus repositorios de paquetes mas actualizados. Otras distribuciones de Linux que serian interesantes para este proyecto serian Red Hat o CentOs. Sin embargo, Red Hat requiere de licencias especiales para su uso, y personalmente estoy menos habituado a trabajar con CentOs, por lo que la distribución que mejor se adapta para este TFM es Ubuntu.

2.4.3.2 Virtualización del sistema

Debemos ser capaces de hacer que el sistema sea fácilmente exportable a otro hardware. Para esto, existen alternativas como Vagrant, Virtualbox, VMWare Client, QEmu, VMWare VCenter o Proxmox con sus ventajas e inconvenientes.

Vagrant es un sistema automatizado de creación de maquinas virtuales. Mediante un fichero de descripción permite generar una maquina virtual útil para diferentes sistemas de virtualización. Este sistema aporta grandes ventajas a la portabilidad de la solución sin embargo incrementa la complejidad del sistema. VMWare Client, Virtualbox y QEmu son sistemas de gestión de maquinas virtuales, que permite generar maquinas virtuales fácilmente para sistemas de escritorio. Son buenas alternativas para entornos de desarrollo, sin embargo, consumen muchos recursos y no permiten la gestión de varias maquinas de forma simultanea ni su monitorización de recursos.

VCenter de VMWare ofrece la posibilidad de lanzar múltiples maquinas virtuales desde su interfaz además de gestionar sus recursos, monitorización, o gestión de HA, sin embargo, requiere una licencia comercial para su uso.

Proxmox es una alternativa libre y gratuita para VCenter, por lo que parece la opción mas correcta, ya que permite la creación sencilla de múltiples maquinas virtuales, su gestión y monitorización de forma sencilla. Proxmox también proporciona las

herramientas para exportar las maquinas virtuales creadas por el a otros sistemas como VMWare o Virtualbox.

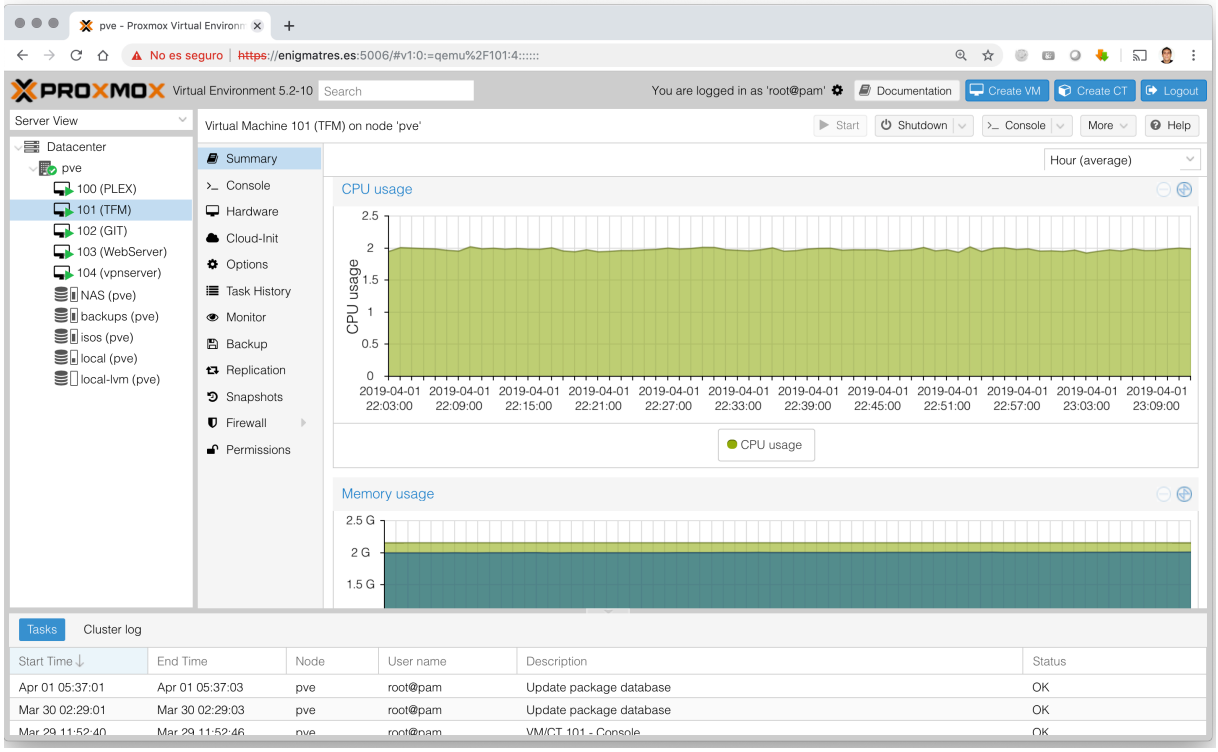

Ilustración 9 - Proxmox interface

2.4.3.3 Virtualización de los dispositivos IoT en ICS

Una vez seleccionada cual será la tecnología para virtualizar todo el sistema, debemos seleccionar cual es la tecnología que utilizaremos para virtual izar los diferentes dispositivos IoT en ICS. También existen diferentes alternativas para la emulación de los dispositivos IoT en ICS, los más populares son Rkt, Singularity y Docker. Lo que necesitamos para este TFM es un sistema que nos permita generar fácilmente nuevas instancias con nuevas configuraciones, así como una gestión y monitorización sencilla de las mismas. Por otra parte, sería deseable que nos permitiera con facilidad añadir un sistema de registro de logs, que permita monitorizar correctamente lo que esta ocurriendo en nuestras instancias.

Después de valorar ambas opciones, me he decantado por Docker, debido a la gran cantidad de información y contenedores prefabricados existentes, que facilitarán mucho la tarea de este TFM.

2.4.3.4 Desarrollo software

En función de las necesidades se desarrollarán las aplicaciones necesarias para la emulación de los dispositivos IoT en ICS. Es posible que sea necesario desarrollar alguna interfaz web que actúe como mockup para detectar ataques de XSS o SQL

injection. También es posible que para implementar alguno de los protocolos de comunicaciones que queremos emular, sea necesario el desarrollo de alguna aplicación que haga uso de estos protocolos. Este software desarrollado se incorporará a un contenedor Docker, y será automáticamente lanzado cada vez que se inicie el contenedor concreto.

2.4.3.5 Emulación de comunicaciones

En algunos casos será necesario simular comunicaciones entre los dispositivos IoT emulados. Para esto, seleccionaremos varios de los protocolos actualmente mas utilizados en el sector industrial como Modbus, DNP3 o el IEC-104. Y dotaremos a algunos de nuestros contenedores de la capacidad de enviar mensajes entre si utilizando estos protocolos.

2.5 Infraestructura

Recopilando y resumiendo la información expuesta anteriormente, virtualizaremos el sistema utilizando el sistema de gestión de máquinas virtuales Proxmox. Dentro de esa maquina virtual instalaremos una distribución de Linux, Ubuntu. Dentro de esta instalación de Linux, instalaremos el sistema de gestión de contenedores Docker. Los distintos contenedores de Docker se encargarán de gestionar las necesidades para cubrir los diferentes requisitos de este proyecto, como la gestión y análisis de logs, la monitorización y gestión de contenedores, y la generación y envío de alarmas.

Para la gestión de los contenedores existen diferentes alternativas como Kubernetes, Docker Swarm. Aunque kubernetes ofrece mayores facilidades para la distribución de la carga y otras ventajas, en este caso me decantaré por Docker Swarm por su mayor facilidad de uso.

Existen multitud de aplicaciones dockerizadas, que permiten una fácil implantación de sistemas. Una de ellas es el stack ELK, que incluye Elasticsearch, Logstash y Kibana. Estas tres herramientas permiten una fácil gestión de los logs, alarmas y visualización de la información. También permiten la centralización de los logs de todos los contenedores de Docker. Existen otros contenedores Docker que pueden resultar interesantes, como Portainer que permite gestionar en tiempo real los contendores y ver su estado.

Debido a la gran cantidad de contenedores prefabricados que existen para el entorno Docker, es posible que durante el proceso de desarrollo del TFM se utilicen otros contenedores ya preconfigurados.

El esquema de la infraestructura por tanto queda de la siguiente forma:

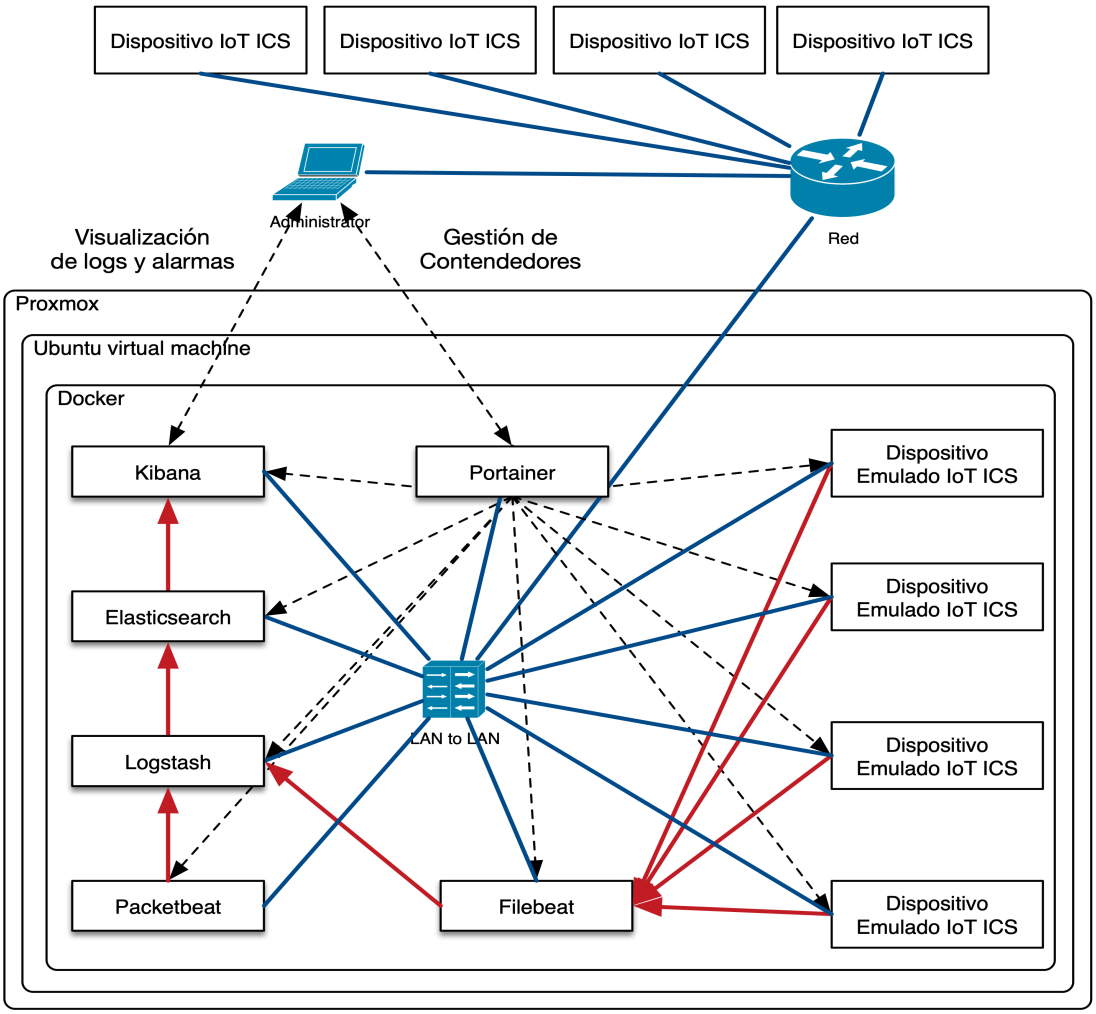

Ilustración 10 - Diagrama del sistema

Ya que el objetivo del sistema es hacer “honeypot” que permita identificar ataques a sistema IoT en ICS, se utilizarán contenedores que simulen ser dispositivos IoT en ICS, y necesitamos herramientas para analizar que esta ocurriendo en esos contenedores, y que tráfico de red esta ocurriendo entre ellos y también desde el exterior. Para analizar estos eventos, y poder detectar posibles ataques, vamos a utilizar software open source desarrollado por Elastic. En concreto vamos a utilizar el stack ELK. Este stack, está compuesto por tres componentes software principalmente: Elasticsearch, Logstash, Kibana, que se describirán mas profundamente posteriormente.

2.6.1 Proxmox

Proxmox Virtual Environment (Proxmox VE o PVE) es un sistema de virtualización open-source. Esta basado en la distribución debian, con un kernel Ubuntu LTS y permite desplegar máquinas virtuales y contenedores. Proxmox incluye una interfaz web y herramientas de línea de comandos, así como un API REST para herramientas de terceros. La primera versión de Proxmox se hizo disponible en abril de 2008. Para este proyecto se hará uso de la versión 5.2-10.

Algunas de las características principales de Proxmox son:

- Es open-source
- Permite configuraciónes para alta disponibilidad
- Permite la migración en vivo entre servidores
- Permite utilizar diferentes sistemas de almacenamiento como NFS, ZFS y otros.

Proxmox permite la virtualización de máquinas bajo diferentes tecnologías, en este caso haremos uso de máquinas virtuales basadas en QEMU.

Proxmox esta disponible bajo la licencia GNU Affero General Public License, en su versión 3. Aunque también dispone de una licencia comercial, que aporta nuevas características, aunque para este proyecto no serán necesarias.

2.5.2 Portainer

Portainer es un software que proporciona una interfaz que permite manejar de forma sencilla diferentes entornos de docker (Docker hosts o clusters Swarm). Portainer esta pensado para ser fácilmente desplegado y usado. Consiste en un solo container que puede desplegarse en cualquier entorno de docker (Linux, Windows y otros). Portainer permite gestionar todos los recursos asociados a tu instalación de docker (contendores, imágenes, volúmenes, redes, etc).

Portainer es desarrollado por Portainer.io y dispone de una versión "community" que es 100% gratuita y open-source, disponible en este repositorio: https://github.com/portainer/portainer. Esta es la versión que se utilizará en este proyecto, a través de un contenedor descargado de dockerhub.com. En concreto para este proyecto utilizaremos la versión 1.20.2.

Utilizaremos portainer para ver el estado de los diferentes containers, y desplegar los contenedores de los dispositivos IoT ICS.

2.5.3 ELK Stack

El paquete ELK es un conjunto de aplicaciones destinadas a la monitorización, procesamiento de datos y visualización de informes. Estas herramientas están desarrolladas por Elastic, y tienen disponible una versión open-source.

2.5.3.1 ElasticSearch

Elasticsearch es un motor de búsqueda distribuido, de tenencia múltiple, basado en la librería Lucene y está desarrollado en Java. Como he comentado previamente, se va a utilizar una versión de Elasticsearch open-source, con la licencia Apache v2. Elasticsearch puede ser utilizado para la búsqueda de cualquier tipo de documentos.

2.5.3.2 Logstash

Logstash es un sistema de procesamiento de datos que permite recibir datos de muchas fuentes diferentes simultáneamente, transformar esos datos, y enviarlos a algún sistema de almacenamiento / búsqueda. En nuestro caso, estos datos serán enviados directamente a Elasticsearch. Esta desarrollada en jRuby, y por tanto requiere de JVM para su funcionamiento. Ya que permite recibir información de múltiples fuentes, desde Elastic han desarrollado diferentes módulos para enviar información a Logstash mas fácilmente. Estos módulos son denominados Beats.

2.5.3.3 Beats

Existen diferentes beats, como Filebeat, Metricbeat, Packetbeat, o Heartbeat. Cada uno de estos está orientado a recoger información de diferentes fuentes y procesarla para ser enviada al sistema de búsqueda o almacenamiento correspondiente. En nuestro caso, actualmente estamos interesados en analizar el trafico de red y también los logs de los diferentes containers con los dispositivos IoT ICS emulados. Para esto hacernos uso de Filebeat y Packetbeat de momento, aunque no se descarta el uso o incluso el desarrollo de algún otro beat en un futuro.

Filebeat

Filebeat es capaz de obtener logs de servidores y contenedores, procesar la información y enviarla al destino seleccionado.

Packetbeat

Packetbeat analiza el trafico de red y comunica todos los eventos que surjan en la misma hasta el destino seleccionado.

2.5.3.4 Kibana

Kibana es un sistema open-source de visualización de datos, diseñado para Elasticsearch. Proporciona diferentes formas de visualización del contenido almacenado en Elasticsearch. Los usuarios pueden crear gráficos de barras, de

puntos, o porciones, basándose en grandes cantidades de datos almacenadas en Elasticsearch. Kibana también permite la configuración de filtros haciendo que la información extraída de los datos sea mas relevante y exacta.

2.5.3.4 Dispositivos Simulados

Para las pruebas con los dispositivos simulados, por el momento vamos a utilizar un servidor Nginx ya que es muy sencillo de desplegar, y proporciona trafico de red y logs. Posteriormente seremos capaces de analizar con éxito ese trafico de red y logs y detectar diferentes ataques sobre este tipo de servidores.

En posteriores fases de proyecto, simularemos otros dispositivos con otras comunicaciones además de la normal TCP/IP http, como por ejemplo modbus, MQTT o DNP3.

2.5.4 Creación de la máquina virtual

El proceso para crear una máquina virtual en Proxmox es bastante sencillo a través de su interfaz web. Es necesario tener Proxmox instalado y configurado, y disponer de un nodo creado. Una vez cumplidos esos requisitos, el proceso de crear una máquina virtual es el siguiente:

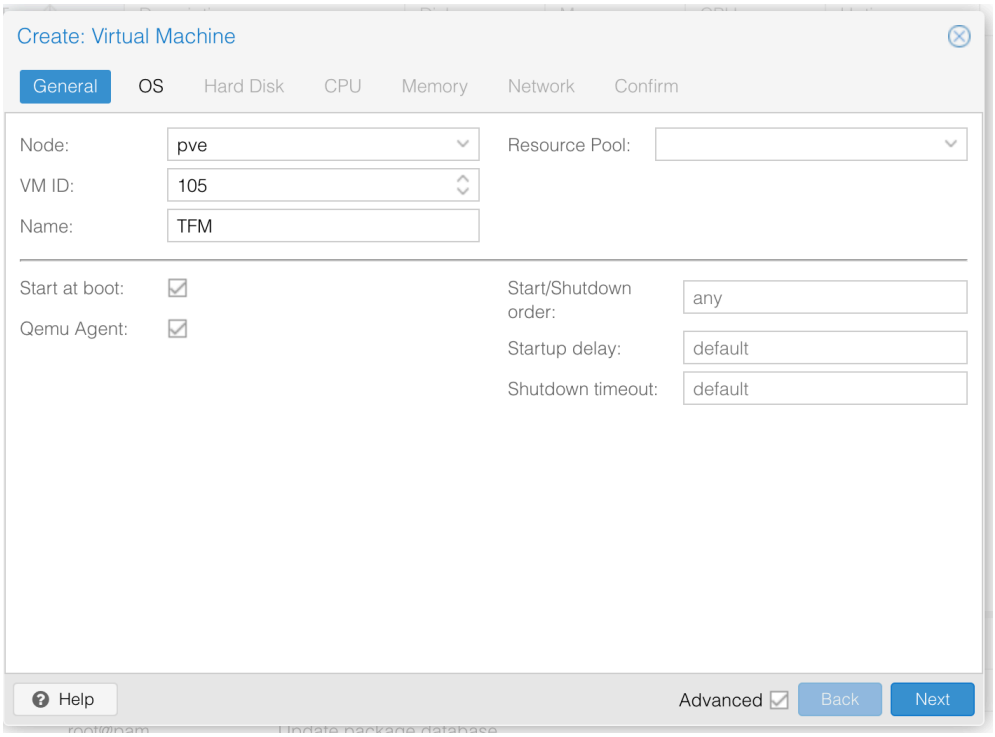

Ilustración 11 - Proxmox, asignación, de nodo y nombre

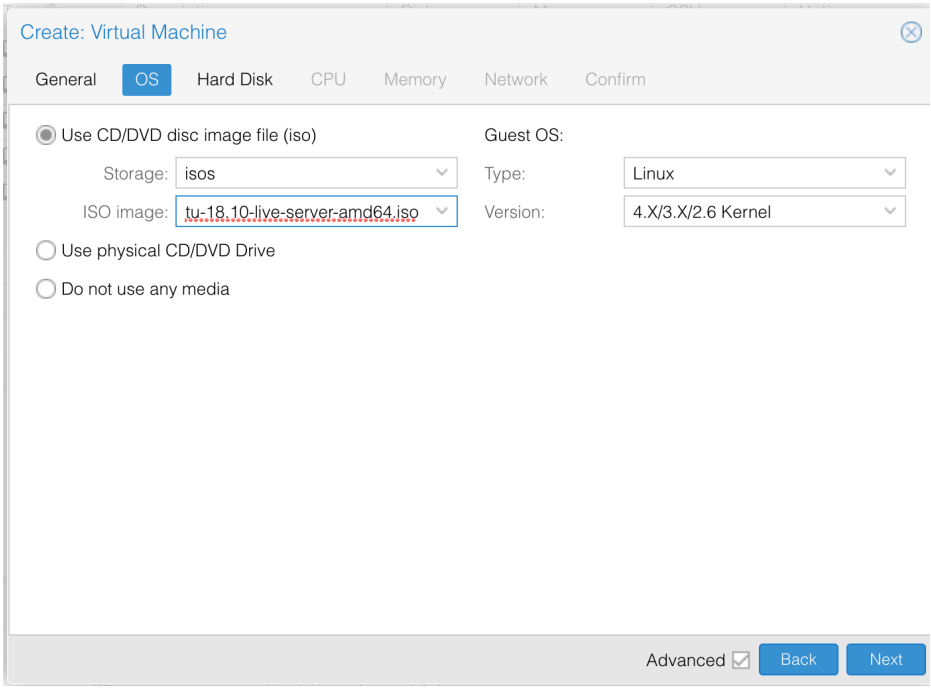

Ilustración 12 - Configuración de imagen de instalación y sistema operativo

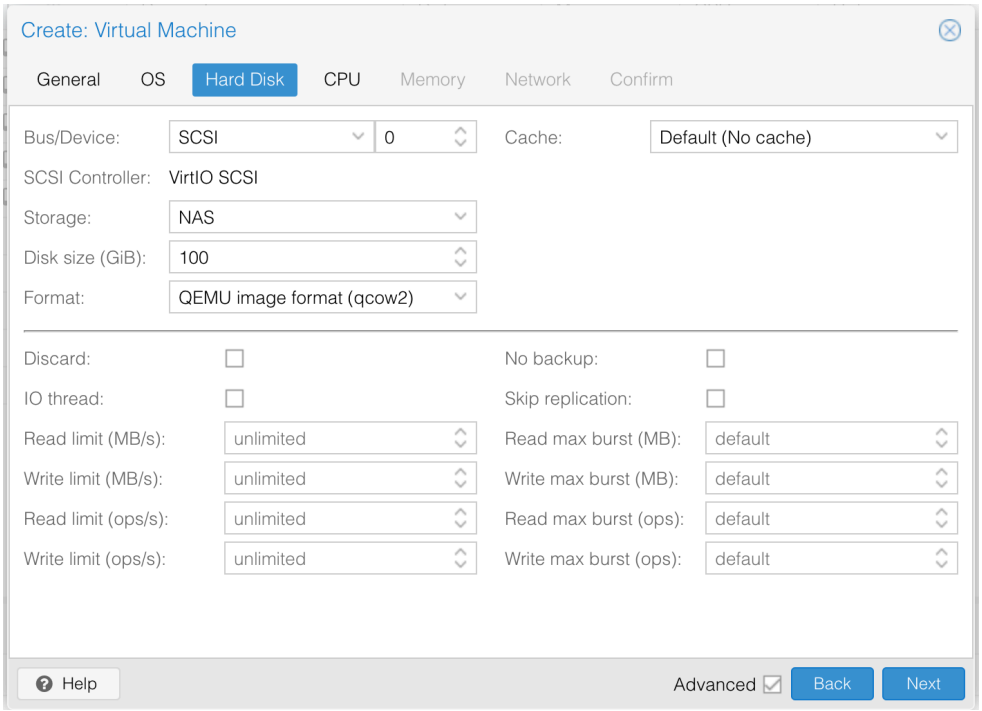

Ilustración 13 - Configuración de almacenamiento

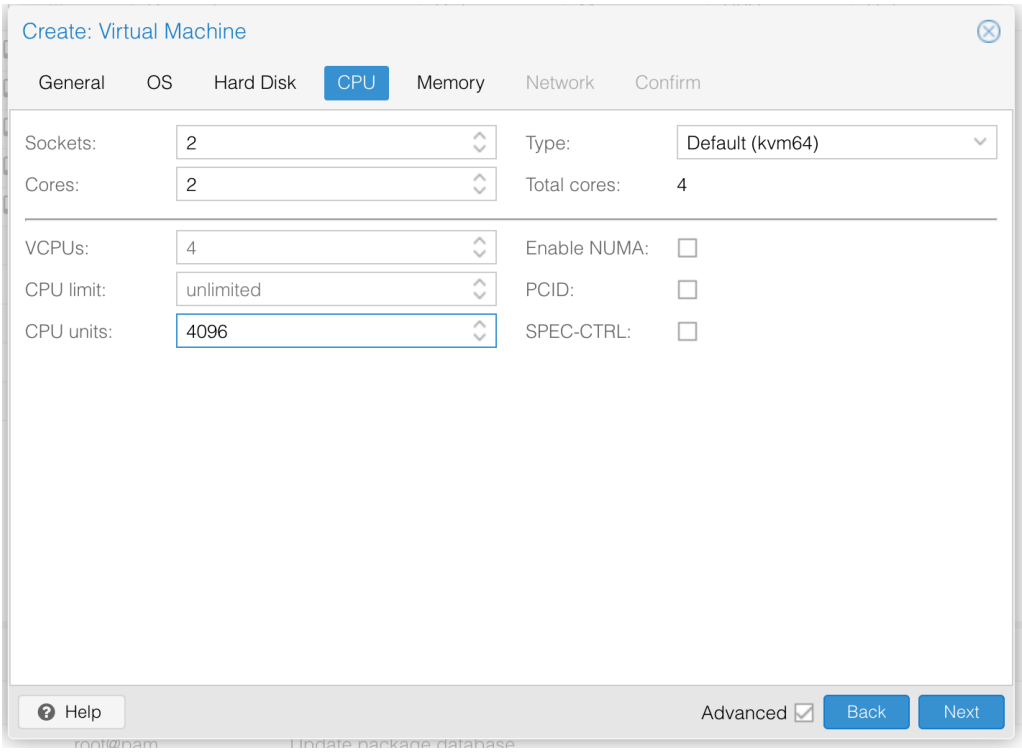

Ilustración 14 - Configuración de procesador

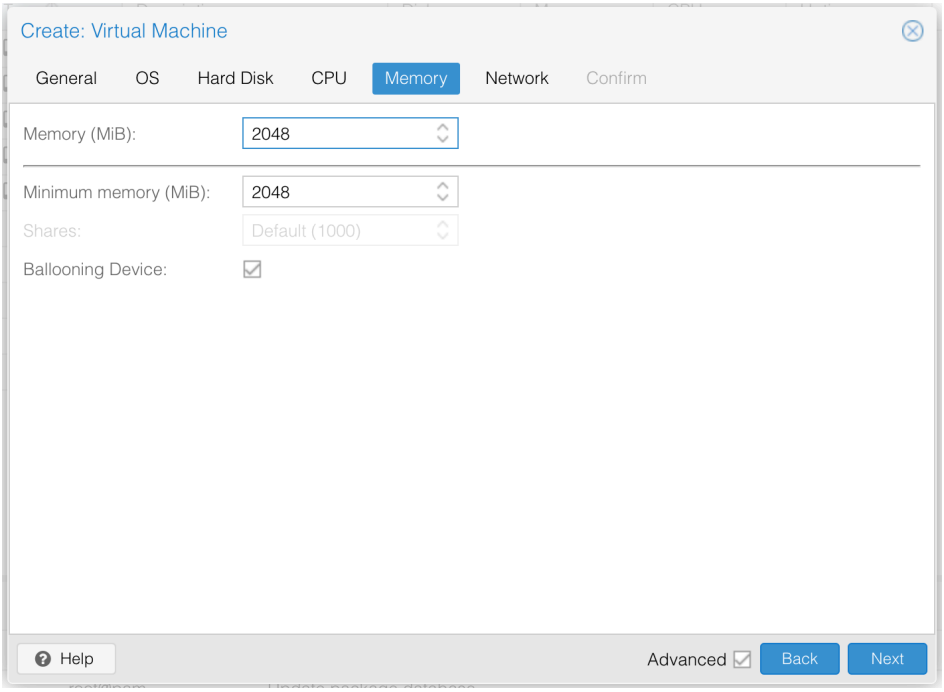

Ilustración 15 - Configuración memoria

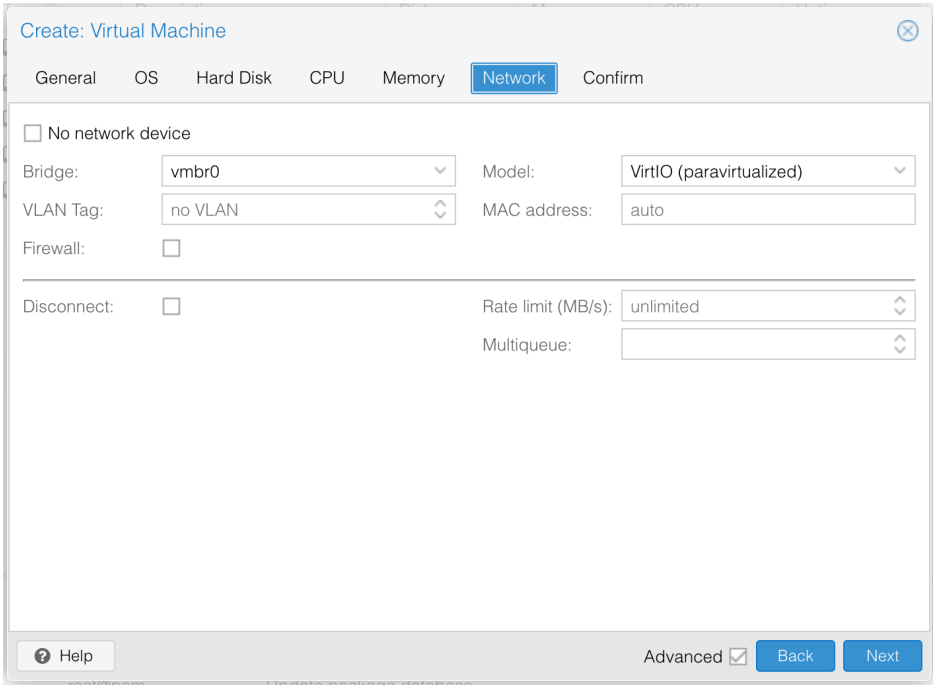

Ilustración 16 - Configuración de red

Después de este proceso, procedemos a la instalación de Ubuntu server, en este caso, la versión 18.10. El proceso de instalación de Ubuntu es muy sencillo y no requiere ninguna configuración especifica, ni ningún paquete adicional que no se recoja posteriormente en este documento.

La única consideración especial, es que en mi caso he configurado la máquina virtual con la dirección IP manual a 192.168.1.107, y como nombre de usuario ubuntu.

2.5.5 Configuración de la máquina virtual

Accedemos mediante la consola de Proxmox a la máquina virtual, o a través de SSH y procedemos a actualizar todos los paquetes del sistema:

```
sudo apt-get update
sudo apt-get upgrade
```

Añadimos el repositorio de docker a Ubuntu:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-
key add –
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Actualizamos los paquetes e instalamos docker, posteriormente lo inicializamos.

```
sudo apt-get update
sudo apt-get install -y docker-ce
```

```
systemctl start docker
systemctl enable docker
sudo usermod -a -G docker $USER
```

Hemos agregado el usuario actual (en este caso ubuntu) al grupo “docker”, de esta forma, podemos utilizar docker sin tener que utilizar privilegios de super-usuario.

Otra de las herramientas necesarias a instalar es docker-compose. Podríamos utilizar la versión actual que se encuentra en los repositorios de Ubuntu, sin embargo, existen versiones posteriores que incluyen nuevas características de las que hacemos uso en este proyecto, por lo que instalamos la ultima versión de la siguiente forma:

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.24.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

2.5.6 Configuración Portainer

Para desplegar container, debemos crear una carpeta en la maquina virtual, donde se almacenaran las configuraciones de Portainer y se harán persistentes. Posteriormente crearemos un volumen de docker.

```
mkdir $HOME/data
sudo chmod -R 777 $HOME/data
docker volume create portainer_data
```

Después de esto, podemos lanzar portainer, descargando la imagen de dockerhub:

```
docker run -d -p 9000:9000 --name portainer --restart always -v
/var/run/docker.sock:/var/run/docker.sock -v
portainer_data:/home/ubuntu/data portainer/portainer
```

Una vez desplegado, podemos acceder a su interfaz web desde la dirección ip de la maquina virtual (192.168.1.107 en mi caso) y el puerto 9000. Crearemos un usuario, y elegiremos gestionar el docker local. Después de seguir estos pasos, accederemos a una interfaz como esta:

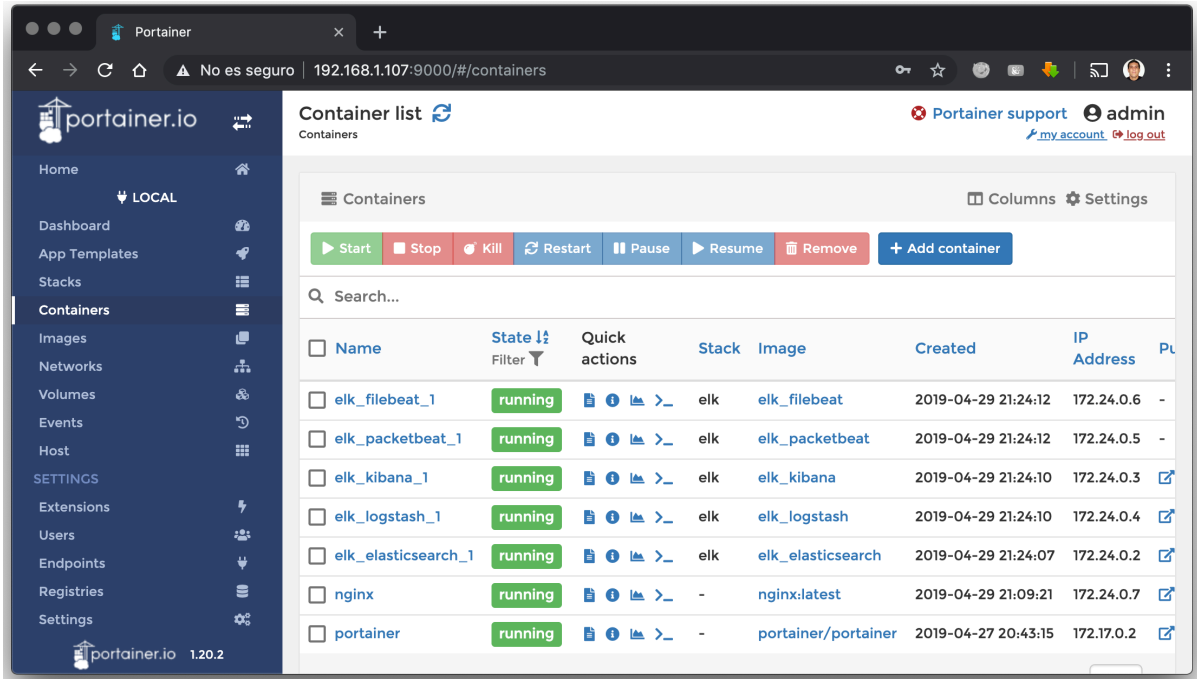

Ilustración 17 - Portainer interface

Desde esta interfaz podemos consultar los logs, el estado, o acceder a una terminal de cada uno de los dockers desplegados en el sistema.

2.5.7 ELK Stack

Para hacer el despliegue de los containers que previamente hemos configurado únicamente hay que acceder a la máquina virtual a través de SSH o de la consola de Proxmox, clonar el repositorio con las configuraciones y ejecutar lo siguiente:

```
cd /path.to.repo/TFM/elk
docker-compose up -d
```

Este comando analizará el archivo /elk/docker-compose.yml y creará los contenedores según esta especificación. La configuración de este archivo y todos los relacionados con el es compleja y se explica en los siguientes apartados.

2.5.7.1 ElasticSearch

La configuración asociada a Elasticsearch en el archivo docker-compose.yml es la siguiente:

```
  elasticsearch:
    environment:
      ES_JAVA_OPTS: "-Xmx256m -Xms256m"
    build:
      context: elasticsearch/
    volumes:
      -
./elasticsearch/config/elasticsearch.yml:/usr/share/elasticsearch/co
nfig/elasticsearch.yml:ro
    ports:
      - "9200:9200"
      - "9300:9300"
    networks:
      - elk
```

Con esto le estamos diciendo a docker, que ejecute el archivo dockerfile que se encuentra en la carpeta elasticsearch, que copie el archivo de configuración elasticsearch.yml que se encuentra en la carpeta config a la carpeta del container "/usr/share/elasticsearch/config/elasticsearch.yml:ro" ya que es la configuración que se utiliza por defecto.

También le decimos que utilice los puertos 9200 y 9300 y que el nombre de la red que se va a utilizar es elk. Además ya que Elastichsearch esta desarrollado en java, es necesario que le pasemos la variable de configuración que le permite saber cuanta memoria destinamos a java.

2.5.7.2 Logstash

La configuración asociada a Logstash en el archivo docker-compose.yml es la siguiente:

```
logstash:
    environment:
      LS_JAVA_OPTS: "-Xmx256m -Xms256m"
    build:
      context: logstash/
    volumes:
      -
./logstash/config/logstash.yml:/usr/share/logstash/config/logstash.y
ml:ro
      - ./logstash/pipeline:/usr/share/logstash/pipeline:ro
    ports:
      - "5000:5000"
      - "5001:5001"
      - "9600:9600"
    expose:
      - "5000"
    networks:
      - elk
    depends_on:
      - elasticsearch
```

Esta configuración es muy similar a la de elasticsearch, con la diferencia de que en este caso se copian dos archivos de configuración y se abren diferentes puertos.

2.5.7.3 Kibana

La configuración asociada a Kibana en el archivo docker-compose.yml es la siguiente:

```
kibana:
    build:
      context: kibana/
    volumes:
      - ./kibana/config/:/usr/share/kibana/config:ro
    ports:
      - "5601:5601"
    networks:
      - elk
    depends_on:
      - elasticsearch
```

Esta configuración también es muy similar a las anteriores.

2.5.7.4 Filebeat

La configuración asociada a Filebeat en el archivo docker-compose.yml es la siguiente:

```
filebeat:
    user: root
    build:
      context: filebeat/
    volumes:
      -
./filebeat/config/filebeat.yml:/usr/share/filebeat/filebeat.yml:ro
      - /var/lib/docker/containers:/var/lib/docker/containers:ro
      - /var/run/docker.sock:/var/run/docker.sock:ro
    command: ["--strict.perms=false"]
    networks:
      - elk
    depends_on:
      - kibana
      - elasticsearch
      - logstash
```

En este caso en particular hay que recalcar que se utiliza “user: root” y command: ["--strict.perms=false"], para ejecutar el container con permisos, ya que al tene que acceder a otros container necesita unos permisos especiales.

2.5.7.5 Packetbeat

La configuración asociada a Packetbeat en el archivo docker-compose.yml es la siguiente:

```
packetbeat:
    user: root
    build:
      context: packetbeat/
    volumes:
      -
./packetbeat/config/packetbeat.yml:/usr/share/packetbeat/config/pack
etbeat.yml:ro
      - /var/lib/docker/containers:/var/lib/docker/containers:ro
    cap_add:
      - NET_RAW
      - NET_ADMIN
    command: ["--strict.perms=false"]
    networks:
      - elk
```

```
    depends_on:
      - elasticsearch
      - logstash
```

En este caso, la configuración incluye algunas diferencias con las anteriores, como por ejemplo “user:root”, que lanza el container con privilegios de usuario y le permite acceder a algunas características del host, al igual que cap_add:NET_RAW y NET_ADMIN. Estas configuraciones son necesarias para poder acceder al trafico de la red interna del docker.

2.6 Puesta en marcha del sistema

Como he comentado previamente, es necesario clonar el repositorio en la maquina virtual con el entorno docker instalado y ejecutar estos comandos:

```
cd /path.to.repo/TFM/elk
docker-compose up -d
```

Una vez ejecutados estos comandos podemos comprobar que Kibana esta funcionando visitando la dirección web http://192.168.1.107:5601. Debemos obtener una web como esta:

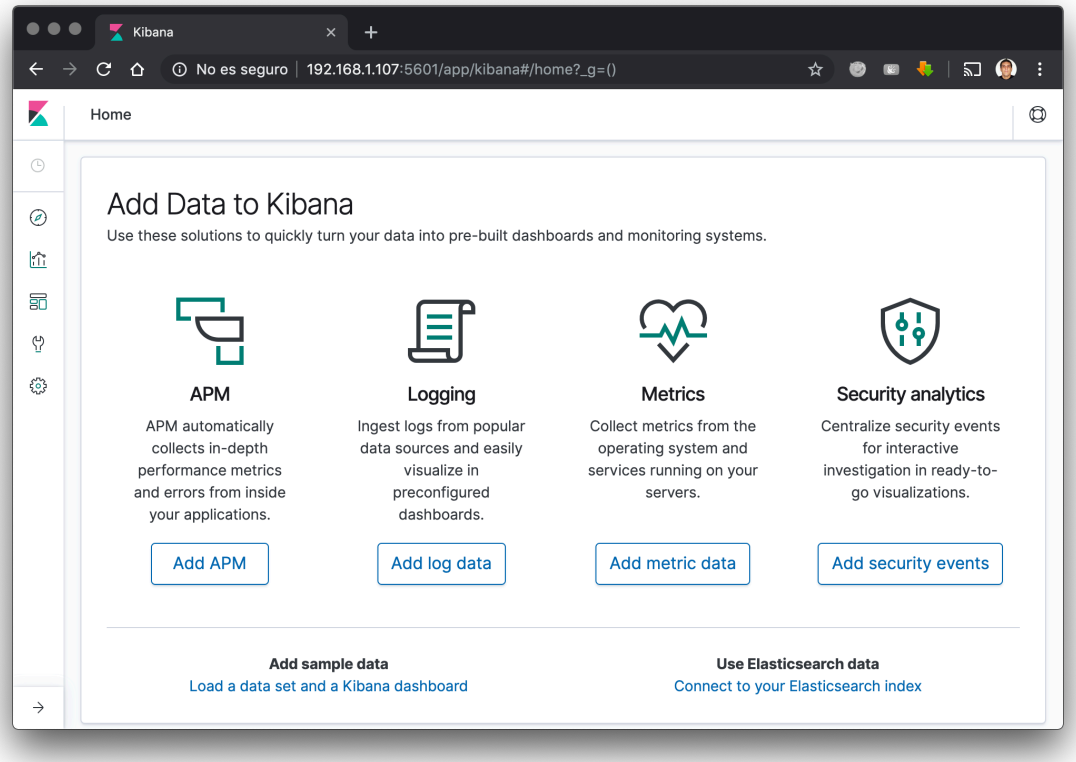

Ilustración 18 - Kibana start

2.7 Funcionamiento

Una vez que tenemos todo desplegado y funcionando, podemos comprobar que recibimos información desde Filebeat y Packetbeat en Kibana. Para esto procedemos a añadir un nuevo índice y podremos ver lo siguiente:

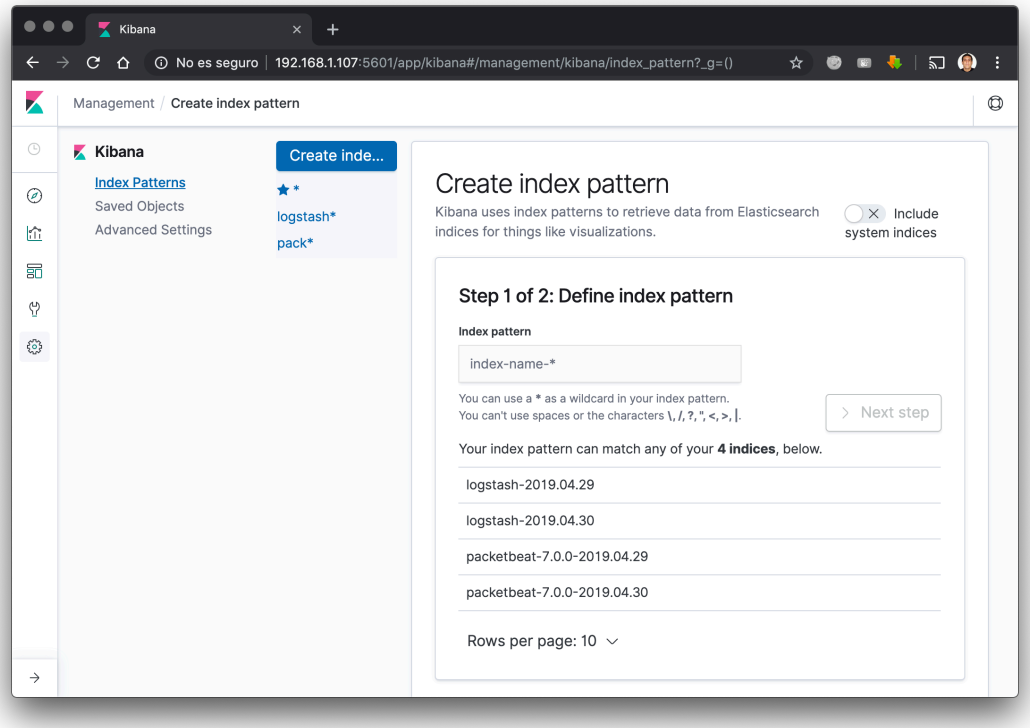

Ilustración 19 - Kibana añadir índice

Como podemos ver, kibana nos sugiere dos tipos de índices diferentes, el primero “logstash….” proviene de filebeat, mientras que “packetbeat-…” son los paquetes que provienen de packetbeat. Si creamos un índice que incluya a los dos, podremos ver los datos que incluyen esos paquetes por el momento:

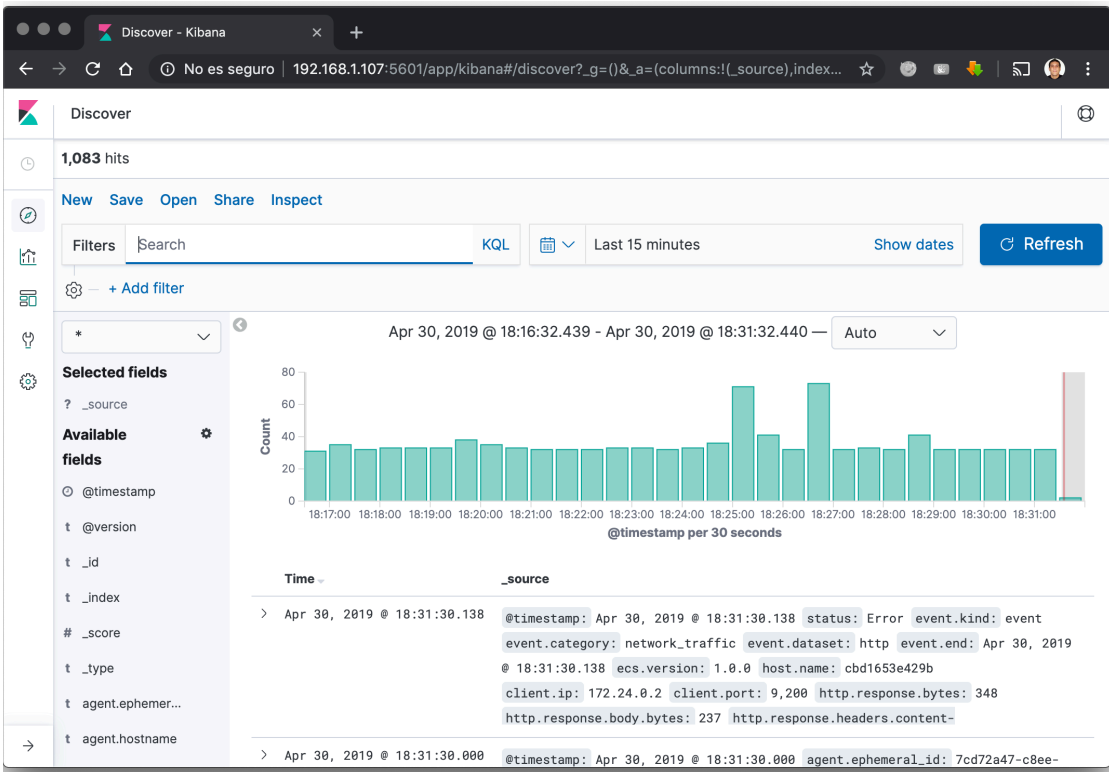

Ilustración 20 - Kibana indice *

Si creamos un índice para filebeat en concreto, podemos ver:

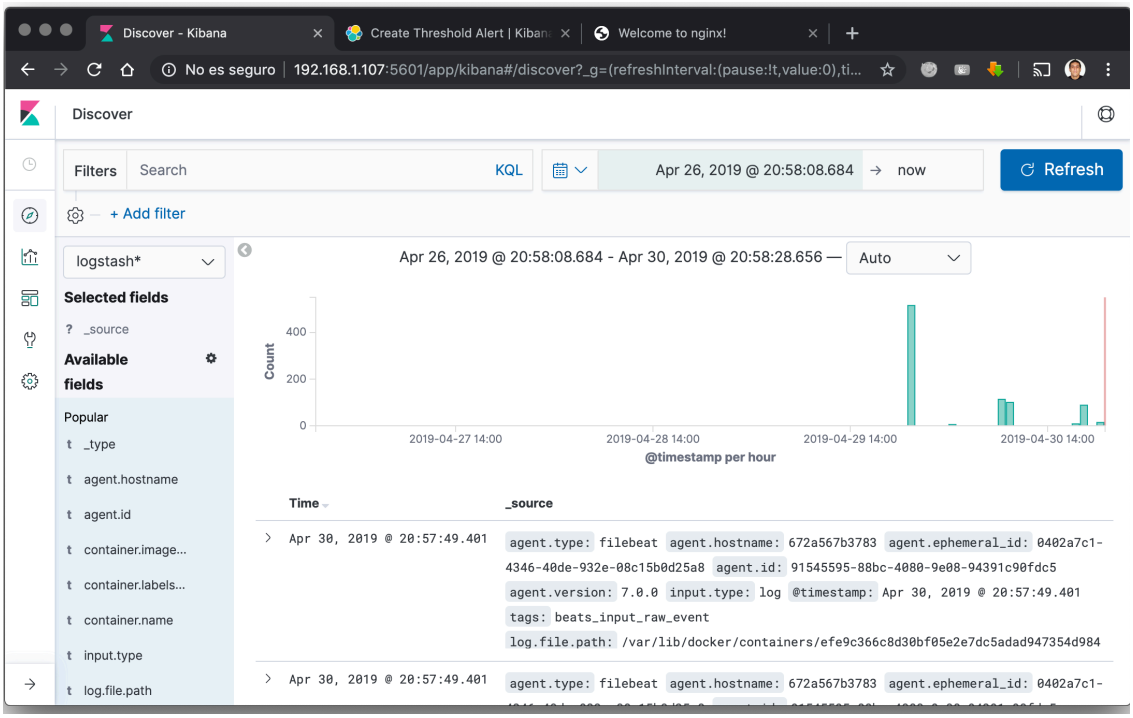

Ilustración 21 - Kibana indice filebeat

Como se puede comprobar, se han obtenido paquetes con información de diferentes logs en diferentes momentos en los últimos días.

Si hacemos un índice para Packetbeat:

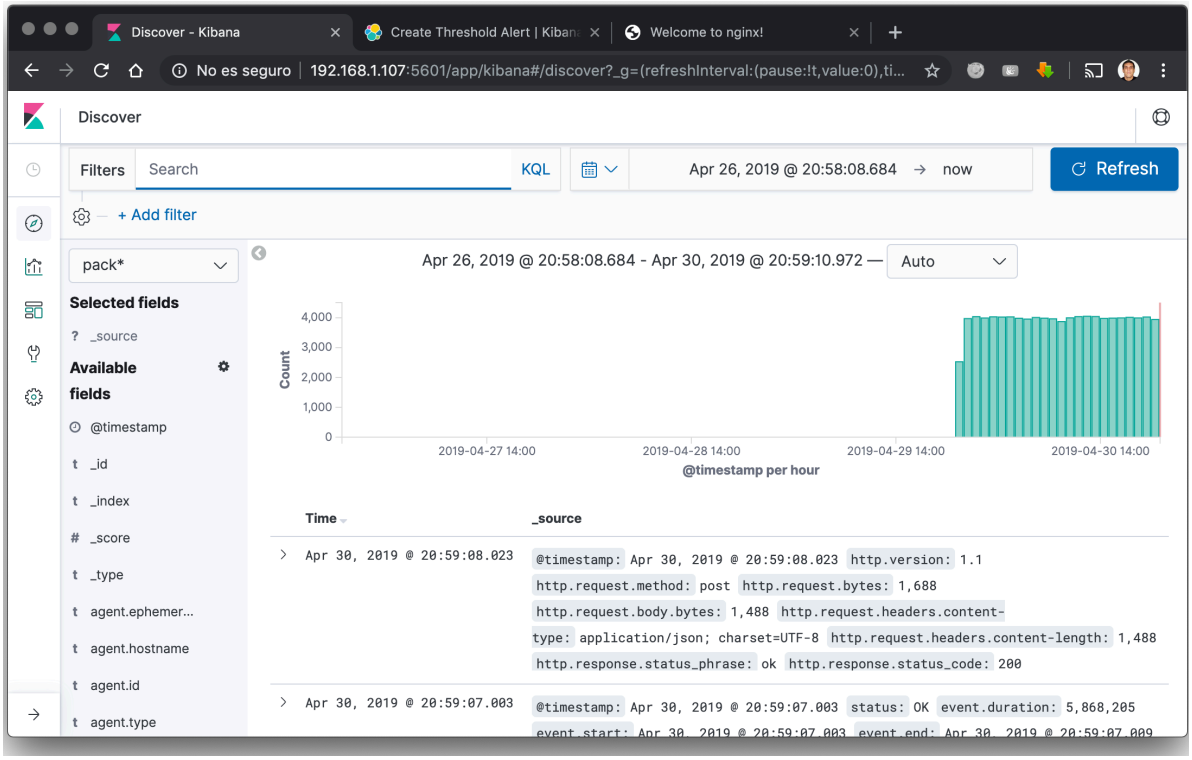

Ilustración 22 - Kibana indice packetbeat

En esta captura también observamos mensajes mandados por packetbeat a elasticsearch.

En esta interfaz podemos analizar los datos, filtrándolos o eligiendo los datos que deseemos ver de cada uno de los mensajes capturados. Kibana también permite la generación de alertas que utilizaremos para ser avisados cada vez que se detecte un posible ataque o un uso anormal de nuestros dispositivos emulados.

3. Conclusiones

3.1 Resultados obtenidos

Se ha conseguido implementar una infraestructura que permite monitorizar en tiempo real dispositivos simulados mediante diferentes sistemas, como es el análisis del trafico de la red simulada y también de los logs de los dispositivos simulados.

A partir de esta monitorización es posible en tiempo real detectar ataques de diferentes tipos en estos sistemas simulados, pudiendo tomar decisiones en tiempo real para evitar ataques, o reducir las consecuencias de estos ataques a nuestros dispositivos reales.

Este sistema de monitorización y vigilancia, es un sistema fácilmente reproducible, ya que hace uso de tecnologías de contenedores. Mediante esta tecnología, es fácil reproducir configuraciones en diferentes sistemas, ya que estas configuraciones son reutilizables. Por otra parte, el uso de estas tecnologías, también permite un fácil mantenimiento, ya que permiten el reinicio prácticamente instantáneo de cada una de las máquinas simuladas, o la creación de nuevas máquinas.

También es un sistema que consume pocos recursos, y que al ser contenedores que virtualizan máquinas, permite la modificación de los recursos hardware utilizados prácticamente en tiempo real, y sin tener que hacer paradas en el sistema.

Las herramientas utilizadas en este TFM son de código abierto, por lo que a nivel de software no existe un coste de licencias asociado al mismo.

Otra de las ventajas de este sistema, es su facilidad desarrollar nuevas características sobre esta infraestructura, como se recogerá en el punto 3.3 Lineas de trabajo futuro.

3.2 Revisión de objetivos

Los objetivos que planteamos al inicio de este trabajo fin de master, incluían principalmente dos áreas, una primera en la que se analizaban la necesidad de un proyecto como este, y cuales serían sus características deseables de un sistema de honeypot para dispositivos ICS. Dentro de estos objetivos, se incluia un análisis de cuales eran los dispositivos mas comunes que podrían ser afectados por este tipo de ataques, cuales eran los protocolos de comunicaciones mas utilizados entre ellos asi como cuales serian los ataques mas comunes.

Durante el desarrollo de este trabajo fin de master, se han analizado cuales son los dispositivos ICS que se podrían ver afectados, y que por tanto, seria deseable simular mediante un honeypot. También se han analizado cuales son los protocolos de comunicación mas comúnmente utilizados en estos protocolos, y cuales son los servicios con potenciales vulnerabilidades de los que hacen usos estos dispositivos simulados.

Debido a los tiempos de los que se disponía para el desarrollo de este trabajo fin de master, no se ha profundizado lo necesario en el análisis de los potenciales ataques, y puede ser un buen campo de avance para líneas posteriores de investigación.

Por otra parte, se incluyeron objetivos técnicos que debía incluir el proyecto para cumplir con la funcionalidad de un honeypot simulando dispositivos ICS. Estos objetivos eran fundamentalmente, construir un sistema que permitiera simular algunas características de dispositivos ICS, simulando sus vulnerabilidades inclusive. Quizás hubiera sido conveniente profundizar mas en este campo, aunque cumple con una funcionalidad básica. Con este sistema es posible simular un dispositivo mediante contenedores Docker y simular sus vulnerabilidades, que era otro de los objetivos establecidos al principio del trabajo fin de master.

Otra de las necesidades que se planteaba, era que el sistema debe ser escalable. Mediante el uso de contenedores es posible escalar fácilmente el sistema, añadiendo y eliminando dispositivos simulados fácilmente. El uso de una aplicación como Portainer para esto facilita mucho la tarea. Y también facilita el mantenimiento de la herramienta, ya que desde ella, podemos visualmente conocer cuales son los contenedores que están funcionando, cuales son los que tienen algún problema, y si existe alguna necesidad de actuación.

3.3 Líneas de trabajo futuro

Algunas de las líneas de trabajo futuro comentadas previamente en las conclusiones son:

Este sistema puede utilizarse como base para el desarrollo de otras funcionalidades. Actualmente hace uso de dos herramientas para capturar el trafico de red y los logs generados. Sin embargo, además de estos dos sistemas, existen otras herramientas que permite obtener mas información en tiempo real del sistema.

Para esto, se pueden utilizar otras tecnologías de monitorización, algunas de estas tecnologías son: Metricbeat o Heartbeat. Metricbeat permite recoger información sobre consumo de CPU, memoria, y otros datos métricos de diferentes contenedores y procesos dentro de los mismos. Heartbeat, permite monitorizar el tiempo que están activos los contenedores detectar y lanzar alarmas en caso de que se produzcan paradas no planificadas.

Otra de las líneas de trabajo futuro, seria el desarrollo mas contenedores con diferentes vulnerabilidades. Durante el desarrollo de este trabajo fin de master, a modo de prueba se ha trabajado con un servidor de nginx, para detectar los accesos y recolectar sus logs generados. De la misma forma, se podría trabajar con otros contenedores o propiciar otros protocolos y servicios.

4. Glosario

Honeypot: Un honeypot, o sistema trampa o señuelo es una herramienta de la seguridad informática dispuesto en una red o sistema informático para ser el objetivo de un posible ataque informático, y así poder detectarlo y obtener información del mismo y del atacante.

ICS: Sistema de control industrial (ICS) es un término general que abarca varios tipos de sistemas de control e instrumentos asociados utilizados para el control de procesos industriales. Tales sistemas pueden abarcar desde unos pocos controladores modulares montados en panel hasta grandes sistemas de control distribuidos interactivos e interconectados con muchos miles de conexiones de campo. Todos los sistemas reciben datos recibidos de sensores remotos que miden las variables del proceso (PV), los comparan con los puntos de ajuste deseados (SP) y derivan funciones de comando que se utilizan para controlar un proceso a través de los elementos de control finales (FCE), como las válvulas de control.

Contenedor: Los Contenedores –Containers– son una solución al problema de cómo hacer que el software se ejecute confiablemente cuando se traslada de un entorno informático a otro. Esto podría ser desde la computadora portátil de un desarrollador a un entorno de prueba, desde un entorno de transición al de producción, y tal vez desde una máquina física en un centro de datos a una máquina virtual en una nube privada o pública. Los Contenedores son un método de virtualización de un sistema operativo que permite ejecutar una aplicación junto con sus elementos dependientes, en un ambiente aislado e independiente. Los Contenedores permiten empaquetar fácilmente el código, las configuraciones y las dependencias de una aplicación en bloques fáciles de usar que ofrecen consistencia ambiental, eficiencia operativa, productividad para el desarrollador y control de versiones.

Docker: es un programa de código abierto que permite que una aplicación Linux y sus dependencias se empaqueten como un contenedor. La virtualización basada en contenedores aísla las aplicaciones entre sí en un sistema operativo (OS) compartido.

ELK Stack: Es un conjunto de herramientas de gran potencial de código abierto que se combinan para crear una herramienta de administración de registros permitiendo la monitorización, consolidación y análisis de logs generados en múltiples servidores, estas herramientas son:ElasticSearch, Logstash y Kibana.

5. Bibliografía

http://cloudingmine.com/how-can-i-centralize-docker-container-logs/
https://github.com/deviantony/docker-elk
https://github.com/namvu80ap/docker-elk-filebeat/blob/master/logstash/pipeline/logstash.conf
https://github.com/proyecto/TFM
https://stackoverflow.com/questions/51849542/filebeat-does-not-send-logs-to-logstash
https://www.adictosaltrabajo.com/2016/05/16/extraccion-de-logs-con-logstash/